



**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
**(Deemed to be University)**

**(Established Under Section 3 of UGC Act, 1956)**

**Coimbatore - 641 021, India**

**FACULTY OF ARTS, SCIENCE AND HUMANITIES (FASH)**

**Department of CS,CA & IT**

**III B.Sc CS**

**V SEMESTER**

**BATCH : 2016 - 2019**

**16CSU504B**

**PROGRAMMING IN PYTHON**

**3H – 3C**

Instruction Hours / week: L: 3 T: 0 P: 0    **Marks: Int : 40 Ext : 60    Total: 100**

**SCOPE**

This programming language is versatile, robust and comprehensive programming language. It has true portability features and can be used across a multitude of platforms.

**COURSE OBJECTIVES**

- To learn how to design and program Python applications.
- To learn how to use indexing and slicing to access data in Python programs.
- To define the structure and components of a Python program.
- To learn how to write loops and decision statements in Python.
- Master the principles of object-oriented programming and the interplay of algorithms and data structures in well-written modular code;
- Solve problems requiring the writing of well-documented programs in the Python language, including use of the logical constructs of that language;
- Demonstrate significant experience with the Python program development environment.

**COURSE OUTCOME**

After the course, students should be able to:

- implement a given algorithm as a computer program (in Python)
- adapt and combine standard algorithms to solve a given problem
- Apply top-down concepts in algorithm design.

- Apply decision and repetition structures in program design.
- Write Python programs to illustrate concise and efficient algorithms
- adequately use standard programming constructs: repetition, selection, functions, composition, modules, aggregated data (arrays, lists, etc.)

#### UNIT-I

**Planning the Computer Program:** Concept of problem solving-Problem definition-Program design-Debugging-Types of errors in programming-Documentation.

#### UNIT-II

**Techniques of Problem Solving:** Flowcharting-decision table-algorithms-Structured programming concepts-Programming methodologies: top-down and bottom-up Programming.

#### UNIT-III

**Overview of Programming:** Structure of a Python Program-Elements of Python.

#### UNIT-IV

**Introduction to Python:** Python Interpreter-Using Python as calculator-Python shell-Indentation. Atoms-Identifiers and keywords-Literals-Strings-Operators(Arithmetic operator, Relational operator, Logical or Boolean operator, Assignment, Operator, Ternary operator, Bit wise operator, Increment or Decrement operator).

#### UNIT-V

**Creating Python Programs:** Input and Output Statements-Control statements(Branching, Looping, Conditional Statement, Exit function, Difference between break, continue and pass.). Defining Functions-Default arguments.

**Suggested Readings**

1. Richard L. Halterman, "FUNDAMENTALS OF PYTHON PROGRAMMIN" 2013
2. Allen Downey, "Think Python How to Think Like a Computer Scientist", 2012, Green Tea Press
3. Andrew Johansen, "Python The Ultimate Beginner's Guide!", 2016
4. Budd, T. (2011). Exploring Python (1st ed.). New Delhi: TMH.
5. Python Tutorial/Documentation [www.python.org](http://www.python.org) 2015.
6. Allen Downey., Jeffrey Elkner., & Chris Meyers. (2012). How to think like a computer scientist : learning with Python. Freely available online.

**WEB SITES**

1. <https://www.slideshare.net>
2. [www.programiz.com](http://www.programiz.com)
3. [www.guru99.com](http://www.guru99.com)
4. <https://www.tutorialspoint.com>
5. <http://docs.python.org/3/tutorial/index.html>.
6. <http://interactivepython.org/courselib/static/pythonds>.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore - 641 021, India

**FACULTY OF ARTS, SCIENCE AND HUMANITIES (FASH)**

Department of CS,CA &amp; IT

**STAFF NAME : A.JEEVARATHINAM****SUBJECT NAME : PROGRAMMING IN PYTHON****SUB.CODE : 16CSU504B****SEMESTER : V****CLASS : III B.Sc (CS)****UNIT I**

<b>S.NO</b>	<b>Lecture Duration (Hours)</b>	<b>Topics To Be Covered</b>	<b>Support Materials/ Pg.No</b>
		<b>Planning the Computer Program</b>	
1	1	Concept of problem solving	W1
2	1	Problem definition	
		Program design	
3	1	Debugging	T2 : 3,4
		Types of errors in programming	T1: 52-56
4	1	Documentation	W4
5	1	<b>Recapitulation and Discussion of Important Questions</b>	
		<b>Total No of Hours Planned for Unit I</b>	<b>5</b>

**UNIT II**

<b>S.NO</b>	<b>Lecture Duration (Hours)</b>	<b>Topics To Be Covered</b>	<b>Support Materials/ Pg.No</b>
		<b>Techniques of Problem Solving</b>	
1	1	Flowcharting	W2
2	1	decision table	W3
3	1	algorithms	W4
4	1	Structured programming concepts	W3
		<b>Programming methodologies</b>	
5	1	top-down Programming	W4
		bottom-up Programming	
6	1	<b>Recapitulation and Discussion of Important Questions</b>	
		<b>Total No of Hours Planned for Unit II</b>	<b>6</b>

**UNIT III**

<b>S.NO</b>	<b>Lecture Duration (Hours)</b>	<b>Topics To Be Covered</b>	<b>Support Materials/ Pg.No</b>
		<b>Overview of Programming</b>	
1	1	Python Introduction	T1:1-4
		Python Features	W4
2	1	Structure of a Python Program	T1:6
		<b>Elements of Python</b>	
3	1	Variables	T1:17
		Data Types	
4	1	Arrays	T2:87-92
		Python List	
5	1	<b>Recapitulation and Discussion of Important Questions</b>	
		<b>Total No of Hours Planned for Unit III</b>	<b>5</b>

## UNIT IV

S.NO	Lecture Duration (Hours)	Topics To Be Covered	Support Materials/ Pg.No
		<b>Introduction to Python</b>	
		<b>Python Interpreter</b>	
1	1	Using Python as calculator	W4
2	1	Python shell Indentation	
3	1	Atoms	W3
4	1	Identifiers and keywords	T1:24-30
		Literals	
5	1	Strings	T2:71-75
		<b>Operators</b>	
6	1	Arithmetic operator	
		Relational operator	
7	1	Logical or Boolean operator	
		Assignment Operator,	T3:40-50
8	1	Ternary operator	
		Bit wise operator	
		Increment or Decrement operator	
9	1	Recapitulation and Discussion of Important Questions	
		<b>Total No of Hours Planned for Unit IV</b>	<b>9</b>

**UNIT V**

<b>S.NO</b>	<b>Lecture Duration (Hours)</b>	<b>Topics To Be Covered</b>	<b>Support Materials/ Pg.No</b>
		<b>Creating Python Programs</b>	
1	1	Input and Output Statements	T3:51-55
		<b>Control statements</b>	
2	1	Branching	W4
3	1	Looping	T1:113-137
4	1	Conditional Statement	T1:67-83
		Exit function	W4
5	1	Difference between break continue and pass	T1:113-137
6	1	Defining Functions	T2:19-24
7	1	Default arguments	
8	1	Recapitulation and Discussion of Important Questions	
9	1	Discussion of previous ESE question papers	
10	1	Discussion of previous ESE question papers	
11	1	Discussion of previous ESE question papers	
		<b>Total No of Hours Planned for Unit V</b>	<b>11</b>
<b>Total Hours</b>			<b>36</b>



<b>S.NO</b>	<b>TEXT BOOKS</b>
<b>T1</b>	Richard L. Halterman, "FUNDAMENTS OF PYTHON PROGRAMMIN" 2013
<b>T2</b>	Allen Downey, "Think Python How to Think Like a Computer Scientist",2012,Green Tea Press
<b>T3</b>	Andrew Johansen, "Python The Ultimate Beginner's Guide!",2016
<b>S.NO</b>	<b>WEB SITES</b>
<b>W1</b>	<a href="https://www.slideshare.net">https://www.slideshare.net</a>
<b>W2</b>	<a href="http://www.programiz.com">www.programiz.com</a>
<b>W3</b>	<a href="http://www.guru99.com">www.guru99.com</a>
<b>W4</b>	<a href="https://www.tutorialspoint.com">https://www.tutorialspoint.com</a>

**CLASS : III B.Sc CS**  
**COURSE NAME : PROGRAMMING IN PYTHON**  
**UNIT I**

**BATCH : 2016 - 2019**  
**COURSE CODE : 16CSU504B**

---

**Planning the Computer Program:** Concept of problem solving-Problem definition- Program design-Debugging-Types of errors in programming-Documentation.

**PLANNING THE COMPUTER PROGRAM:**

**Introduction to Computer Science**

A **computer** is a programmable machine that receives input, stores and manipulates data, and provides output in a useful format.

**Computer science** is the study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.

A **program** is a sequence of instructions that can be executed by a computer to solve some problem or perform a specified task.

**programming language** is an artificial language designed to automate the task of organizing and manipulating information, and to express problem solutions precisely.

A programming language “boils down to” a set of words, rules and tools that are used to explain (or define) what you are trying to accomplish. There are many different programming languages just as there are many different "spoken" languages.

Traditional programming languages were known as structural programming languages (e.g., C, Fortran, Pascal, Cobol, Basic). Since the late 80's however, object-oriented programming languages have become more popular (e.g., JAVA, C++, C#)

There are also other types of programming languages such as functional programming languages and logic programming languages. According to the Tiobe index (i.e., a good site for ranking the popularity of programming languages), as of February 2011 the 10 most actively used programming languages were (in order of popularity):

Java, C, C++, PHP, Python, C#, VisualBasic, Objective-C, Perl, Ruby

The **Computer Sciences Accreditation Board (CSAB)** identifies four general areas that it considers crucial to the discipline of computer science:

- **theory of computation** - investigates how specific computational problems can be solved efficiently

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

- **algorithms and data structures** - investigates efficient ways of storing, organizing and using data
- **programming methodology and languages** - investigates different approaches to describing and expressing problem solutions
- **computer elements and architecture** - investigates the design and operation of computer systems

### CONCEPT OF PROBLEM SOLVING

#### PROBLEM SOLVING

Regardless of the area of study, computer science is all about solving problems with computers. The problems that we want to solve can come from any real-world problem or perhaps even from the abstract world. We need to have a standard systematic approach to solving problems.

**Problem Solving** is the sequential process of analyzing information related to a given situation and generating appropriate response options.

There are 6 steps that you should follow in order to solve a problem:

1. Understand the Problem
2. Formulate a Model
3. Develop an Algorithm
4. Write the Program
5. Test the Program
6. Evaluate the Solution

In order to solve a problem by the computer, one has to pass through certain stages or steps. They are

1. Understanding the problem
2. Analyzing the problem
3. Developing the solution
4. Coding and implementation.

**Understanding the problem:** Here we try to understand the problem to be solved in totality. Before with the next stage or step, we should be absolutely sure about the objectives of the given problem.

- What input data/information is available?
- What does it represent ? o What format is it in ?
- Is anything missing ? o Do I have everything that I need ?
- What output information am I trying to produce ?
- What do I want the result to look like ... text, a picture, a graph ... ?

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

- 
- What am I going to have to compute ?

**Analyzing the problem:** After understanding thoroughly the problem to be solved, we look different ways of solving the problem and evaluate each of these methods. The idea here is to search an appropriate solution to the problem under consideration. The end result of this stage is a broad overview of the sequence of operations that are to be carried out to solve the given problem.

**Developing the solution:** Here the overview of the sequence of operations that was the result of analysis stage is expanded to form a detailed step by step solution to the problem under consideration.

**Coding and implementation:** The last stage of the problem solving is the conversion of the detailed sequence of operations into a language that the computer can understand. Here each step is converted to its equivalent instruction or instructions in the computer language that has been chosen for the implementation.

### Program Development Steps

The various steps involved in Program Development are:

- Defining or Analyzing the problem
- Design (Algorithm)
- Coding
- Documenting the program
- Compiling and Running the Program
- Testing and Debugging
- Maintenance

### Analyzing or Defining the Problem

The problem is defined by doing a preliminary investigation. Defining a problem helps us to understand the problem clearly. It is also known as Program Analysis.

Tasks in defining a problem:

- ❑ Specifying the input requirements
- ❑ Specifying the output requirements
- ❑ Specifying the processing requirements

### Specifying the input requirements

Determine the inputs required and source of the data. The input specification is obtained by answering the following questions:

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

- ❑ What specific values will be provided as input to the program?
- ❑ What format will the values be?
- ❑ For each input item, what is the valid range of values that it may assume?
- ❑ What restrictions are placed on the use of these values?

### Specifying the output requirements

Describe in detail the output that will be produced. The output specification is obtained by answering the following questions:

- ❑ What values will be produced?
- ❑ What is the format of these values?
- ❑ What specific annotation, headings, or titles are required in the report?
- ❑ What is the amount of output that will be produced?

### Specifying the Processing Requirements

Determine the processing requirements for converting the input data to output. The processing requirement specification is obtained by answering the following questions:

- ❑ What is the method (technique) required in producing the desired output?
- ❑ What calculations are needed?
- ❑ What are the validation checks that need to be applied to the input data?

### Example 1.1 Find the factorial of a given number

**Input:** Positive valued integer number

**Output:** Factorial of that number

**Process:** Solution technique which transforms input into output. Factorial of a number can be calculated by the formula  $n! = 1*2*3*4*...*n$

### Problem definition

The art of compiling logic in the form of general flow charts and logic diagrams which clearly explain and present the problem to the programmer in such a way that all requirements involved in the run are presented.

A computational problem is a mathematical object representing a collection of questions that computers might be able to solve. For example, the problem of factoring "Given a positive integer  $n$ , find a nontrivial prime factor of  $n$ ."

What is the mathematical object in the example above?

Commonly encountered mathematical objects include numbers, permutations, partitions, matrices, sets, functions, and relations.

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

---

So how can you 'represent' a collection of questions with numbers or permutations, matrices etc.? What is meant here is probably the following collection of sentences: 'find a nontrivial prime factor of 1', 'find a nontrivial prime factor of 2' and so on... But the thing is - these sentences are not mathematical objects.

A little further in the article it reads:

A computational problem can be viewed as an infinite collection of instances together with a solution for every instance.  
which makes perfect sense, but I don't quite see the relationship with the first definition.

### Program design

#### **Steps to Design**

There are three fundamental steps you should perform when you have a program to write:

- Define the output and data flows.
- Develop the logic to get to that output.
- Write the program.

Program designing begins with deciding the output and framing the program logic. The design is then broken down into modules to facilitate programming. All computer languages have a vocabulary of their own. If a programmer does not strictly follow the syntax of a programming language, the computer will not understand the commands given in the program.

#### **TYPES OF PROGRAM ERRORS**

We distinguish between the following types of errors:

1. **Syntax errors:** errors due to the fact that the syntax of the language is not respected.
2. **Semantic errors:** errors due to an improper use of program statements.
3. **Logical errors:** errors due to the fact that the specification is not respected.

From the point of view of when errors are detected, we distinguish:

1. **Compile time errors:** syntax errors and static semantic errors indicated by the compiler.
2. **Runtime errors:** dynamic semantic errors, and logical errors, that cannot be detected by the compiler (debugging).

#### **Testing and Debugging**

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

*"Program testing can be used to show the presence of bugs, but never to show their absence!" (Dijkstra)*

*"If debugging is the process of removing software bugs, then programming must be the process of putting them in." (Anon)*

*"There are two ways to write error-free programs. Only the third one works." (Anon)*

One of the first things you will discover in writing programs is that a program rarely runs correctly first time. In fact, errors are so common in programming that they have their own special name: **Bugs**. The process of correcting, or getting rid of bugs is called **Debugging**.

### Testing

Testing is the process of executing programs with the intention of finding errors.

*"A good test is one that has a high probability of finding an, as yet, undiscovered error."*

In fact, a good program tester will try to make programs fail.

When considering the kind of tests to apply, a **systematic approach** should be used and **test cases** should be developed which will uncover common classes of errors. Never rely on intuition. The data collected during the testing should be kept as a means of demonstrating the correct operation of the program. This is important for future maintenance.

### Types of Program Error

#### Program errors:

There are three basic categories of program errors:

1. Syntax Errors
2. Run-time Errors
3. Logic Flaws

In the first two cases when an error occurs, the computer displays an '**Error Message**', which describes the error, and its cause. Unfortunately, error messages are often **difficult to interpret**, and are sometimes **misleading**. In the final case, the program will not show an error message but it will not do what the programmer wanted it to do.

### Syntax Errors

These errors are the easiest to find because they are highlighted by the compiler. Error messages are given. This type of error is caused by the failure of the programmer to use the correct **grammatical rules** of the language.

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

Syntax errors are detected, and displayed, by the **compiler** as it attempts to translate your program, ie: the **Source** code into the **Object** code. If a program has a syntax error it cannot be translated, and the program will not be executed.

The compiler tries to highlight syntax errors where there seems to be a problem, however, it is not perfect and sometimes the compiler will indicate the next line of code as having the problem rather than the line of code where the problem actually exists.

un-time errors are detected by the computer and displayed **during** execution of a program. They will halt the program when they occur but they often do not show up for some time. Ex: When you forgot to type a semicolon (;) after the statement, the compiler shows the syntax error and it would point out where the problem occurred.

### Run-time Errors

A run-time error occurs when the user directs the computer to perform an **illegal operation**, eg:

- Dividing a number by zero
- Assigning a variable to the wrong type of variable
- Using a variable in a program before assigning a value to it.
- 

When a run-time error occurs, the computer stops executing your program, and displays a **diagnostic message** that indicates the line where the error occurred.

x: A program error may result from an attempt to divide by zero

```
int a = 5;  
int b = 0;  
int c = a /b;
```

When you compile this program, compiler won't show any Syntax Error. But when you run this code, the compiler show "Attempted to divide by zero".

### Logic Flaws

These are the hardest errors to find as they do not halt the program. They arise from faulty thinking on behalf of the programmer. They can be very troublesome.

These are mistakes in a program's logic.

Programs with logic errors will often compile, execute, and output results. However, at least some of the time the output will be incorrect. Error messages will generally not appear if a logic error occurs, this makes logic errors very difficult to locate and correct.

### Syntax and logical errors in Python

Two types of errors can occur in Python:



**CLASS : III B.Sc CS**  
**COURSE NAME : PROGRAMMING IN PYTHON**  
**UNIT I**

**BATCH : 2016 - 2019**  
**COURSE CODE : 16CSU504B**

**1. Syntax errors** – usually the easiest to spot, syntax errors occur when you make a typo. Not ending an **if** statement with the colon is an example of an syntax error, as is misspelling a Python keyword (e.g. using **whille** instead of **while**). Syntax error usually appear at compile time and are reported by the interpreter. Here is an example of a syntax error:

```
x = int(input('Enter a number: '))

whille x%2 == 0:
    print('You have entered an even number.')
else:
    print ('You have entered an odd number.')
```

Notice that the keyword **while** is misspelled. If we try to run the program, we will get the following error:

```
C:Python34Scripts>python error.py
File "error.py", line 3
    whille x%2 == 0:
      ^
SyntaxError: invalid syntax
```

**2. Logical errors** – also called **semantic errors**, logical errors cause the program to behave incorrectly, but they do not usually crash the program. Unlike a program with syntax errors, a program with logic errors can be run, but it does not operate as intended. Consider the following example of an logical error:

```
x = float(input('Enter a number: '))
y = float(input('Enter a number: '))

z = x+y/2
print ('The average of the two numbers you have entered is:',z)
```

The example above should calculate the average of the two numbers the user enters. But, because of the order of operations in arithmetic (the division is evaluated before addition) the program will not give the right answer:

```
>>>
Enter a number: 3
Enter a number: 4
The average of the two numbers you have entered is: 5.0
>>>
```

To rectify this problem, we will simply add the parentheses: **z = (x+y)/2**  
Now we will get the right result:

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

```
>>>
Enter a number: 3
Enter a number: 4
The average of the two numbers you have entered is: 3.5
>>>
```

## **DEBUGGING**

### **Techniques for detecting errors (debugging)**

If the testing phase signals the presence of logical errors, or if we are not able to detect the cause for a runtime error, it is necessary to **debug** the program.

There are two ways in which we can obtain information that is helpful for debugging a program:

1. by inserting output statements in the code;
2. by executing the program by means of a **debugger**.

### **Debugging by inserting output statements**

This debugging technique is based on inserting in suitable positions of the source code statements that print the content of variables that could contain wrong values causing an error.

*Example:*

```
int a, b, x;
a = 5;
b = Integer.parseInt(kb.readLine()); // reading of b
... // statements that do not change b
x = a/b;
```

To *debug* this program statement we can verify, by printing the value of b on the screen, that the error occurs when the variable b has value 0.

```
int a, b, x;
a = 5;
b = Integer.parseInt(kb.readLine()); // reading of b
... // statements that do not change b
System.out.println("b = " + b);
```

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

---

`x = a/b;`

Once the causes for the error have been identified and corrected, the print statements can be removed before closing the debugging session.

*Note:* If it is necessary to explore the content of objects, rather than of simple variables of primitive data types, we can make use of the `toString()` method, which provides information on the content of the object. We could also redefine `toString()` so as to simplify the reading of the state of the object during debugging.

### Execution of the program by means of a debugger

A debugger allows us to:

- execute the statements of a program one at a time,
- execute the program until the execution reaches certain points defined by the user (called **breakpoints**),
- examine the content of variables and objects at any time during the execution.

Debuggers are very useful tools to detect the causes of errors in programs.

#### Example 1:

```
int a, b, x;  
a = 5;  
b = Integer.parseInt(kb.readLine()); // reading of b  
... // statements that do not change b  
x = a/b;
```

By means of a debugger we can verify the value of the variable `b` before executing the statement that generates the error.

#### Example 2:

```
String s, t;  
s = null;  
...  
t = s.concat("a");
```

The assignment statement for `t` generates an exception of type `NullPointerException`. Such an error depends on the fact that, when the assignment statement is executed, the value of `s` is null. To check this error, we can use a debugger and observe the value of the variable `s` before executing the statement that generates the error.

**CLASS : III B.Sc CS**  
**COURSE NAME : PROGRAMMING IN PYTHON**  
**UNIT I**

**BATCH : 2016 - 2019**  
**COURSE CODE : 16CSU504B**

---

### **PROGRAM DOCUMENTATION**

Any written text, illustrations or video that describe a software or program to its users is called **program or software document**. User can be anyone from a programmer, system analyst and administrator to end user. At various stages of development multiple documents may be created for different users. In fact, **software documentation** is a critical process in the overall software development process.

In modular programming documentation becomes even more important because different modules of the software are developed by different teams. If anyone other than the development team wants to or needs to understand a module, good and detailed documentation will make the task easier.

These are some guidelines for creating the documents –

- Documentation should be from the point of view of the reader
- Document should be unambiguous
- There should be no repetition
- Industry standards should be used
- Documents should always be updated
- Any outdated document should be phased out after due recording of the phase out

### **Advantages of Documentation**

These are some of the advantages of providing program documentation –

- Keeps track of all parts of a software or program
- Maintenance is easier
- Programmers other than the developer can understand all aspects of software
- Improves overall quality of the software
- Assists in user training
- Ensures knowledge de-centralization, cutting costs and effort if people leave the system abruptly

CLASS : III B.Sc CS  
COURSE NAME : PROGRAMMING IN PYTHON  
UNIT I

BATCH : 2016 - 2019  
COURSE CODE : 16CSU504B

---

### Example Documents

A software can have many types of documents associated with it. Some of the important ones include –

- **User manual** – It describes instructions and procedures for end users to use the different features of the software.
- **Operational manual** – It lists and describes all the operations being carried out and their inter-dependencies.
- **Design Document** – It gives an overview of the software and describes design elements in detail. It documents details like **data flow diagrams, entity relationship diagrams**, etc.
- **Requirements Document** – It has a list of all the requirements of the system as well as an analysis of viability of the requirements. It can have user cases, reallife scenarios, etc.
- **Technical Documentation** – It is a documentation of actual programming components like algorithms, flowcharts, program codes, functional modules, etc.
- **Testing Document** – It records test plan, test cases, validation plan, verification plan, test results, etc. Testing is one phase of software development that needs intensive documentation.
- **List of Known Bugs** – Every software has bugs or errors that cannot be removed because either they were discovered very late or are harmless or will take more effort and time than necessary to rectify. These bugs are listed with program documentation so that they may be removed at a later date. Also they help the users, implementers and maintenance people if the bug is activated.

**CLASS : III B.Sc CS**  
**COURSE NAME : PROGRAMMING IN PYTHON**  
**UNIT I**

**BATCH : 2016 - 2019**  
**COURSE CODE : 16CSU504B**

---

**Possible Questions:**

**2 Mark Questions**

1. Mention the types of programming language?
2. Assess problem solving method.
3. What is mean by problem solving?
4. What are the properties of algorithm?
5. Define programming language

**6 Mark Questions**

1. Analyze the notations used in algorithmic problem solving
2. Summarize advantage and disadvantage of flow chart
3. How to prepare documentation in programming language?
4. Explain in detail about problem solving techniques?
5. Mention the types of errors in programming. Explain it.
6. Discover the steps of simple strategies for developing algorithms
7. Illustrate with example Programming methodologies in python.



# KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc( CS)

(BATCH 2016-2019)

V SEMESTER

PROGRAMMING IN PYTHON (16CSU504 B )

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

## UNIT I

S.NO	QUESTIONS	OPT 1	OPT 2	OPT 3	OPT 4	ANSWER
1	Last step in process of problem solving is to	design a solution	define a problem	practicing the solution	organizing the data	practicing the solution
2	Second step in problem solving process is to	practicing the solution	organizing the data	design a solution	define a problem	design a solution
3	Thing to keep in mind while solving a problem is	input data	output data	stored data	all of above	all of above
4	First step in process of problem solving is to	design a solution	define a problem	practicing the solution	organizing the data	define a problem
5	Stock maintaining control system in which both hardware and software is present to contact and manage suppliers is called	power system	control system	information processing system	operating system	information processing system
6	All components that work together in the form of unit is called	system	bound	baud	operator	system
7	The main task of a problem-solving agent is	Solve the given problem and reach to goal	To find out which sequence of action will get it to the goal state	Both a and b	Neither a nor b	Both a and b
8	A problem in a search space Is defined by	Initial state	Goal test	Intermediate states	Both a and b	Both a and b

9	The process of removing detail from a given state representation is called_____.	Extraction	Abstraction	Information Retrieval	Mining of data	Abstraction
10	Web Crawler is a/an	Intelligent goal-based agent	Problem-solving agent	Simple reflex agent	Both a and b	Both a and b
11	Error in a program is called	bug	debug	virus	noise	bug
12	Error which occurs when program tried to read from file without opening it is classified as	execution error messages	built in messages	user-defined messages	half messages	execution error messages
13	Error which occurs when user tried to use a device which is not switched ON is classified as	user-defined message	half message	execution error message	built in message	execution error message
14	Types of software programs are	Application programs	Replicate programs	Logical programs	both A and B	both A and B
15	Specialized program that allows the user to utilize in specific application is classified as	relative programs	application programs	relative programs	replicate programs	application programs
16	Program which is used to produce pictures, text and to organize it in newspaper is classified as	text publishing package	desktop publishing package	experimental package	organizing publishing package	desktop publishing package
17	Application program example includes	payroll program	desktop program	publishing program	editing program	payroll program
18	Application program used with all the documentation is considered	applications package	Replicate programs	Logical programs	systems programs	applications package
19	Set of programs which consist of full set of documentations is termed as	database packages	file packages	bus packages	software packages	software packages
20	Program which is readily available for computer users as a part of software package is classified as	library program	program library	software library	directory library	library program



21	Set of software authorized to a specific users is considered as	library program	program library	software library	directory library	program library
22	Programs are fully tested and documented properly before including it into	library	directory	package	database	library
23	To write a program function i.e. program for the sum of four integers, the program refinement first level includes	input four numbers	calculate sum	print the values	display the values	input four numbers
24	Data which is used to test each feature of the program and is carefully selected is classified as	program output	program input	test data	test program	test data
25	Function definition and first level refinement are part of	program design	program statement	program calculation	printing the program	program design
26	There are _____steps to solve the problem	7	4	6	2	6
27	_____is the first step in solving the problem	Understanding the Problem	Identify the Problem	Evaluate the Solution	None of these	Identify the Problem
28	_____is the last step in solving the problem	Understanding the Problem	Identify the Problem	Evaluate the Solution	None of these	Evaluate the Solution
29	Following is true for understanding of a problem	Knowing the knowledgebase	Understanding the subject on which the problem is based	Communication with the client	All of the above	All of the above
30	The six-step solution for the problem can be applied to I. Problems with Algorithmic Solution II. Problems with Heuristic Solution	Only I	Only II	Neither I nor II	Both I and II	Both I and II
31	While solving the problem with computer the most difficult step is _____.	describing the problem	finding out the cost of the software	writing the computer instructions	testing the solution	writing the computer instructions

32	The help menus or user manuals are the part of _____.	Program	Algorithm	Internal Documentation	External Documentation	External Documentation
33	The branch of computer that deals with heuristic types of problem is called _____.	system software	real time software	artificial intelligence	none of these	artificial intelligence
34	The correctness and appropriateness of _____ solution can be checked very easily.	algorithmic solution	heuristic solution	random solution	none of these	algorithmic solution
35	Access in which records are accessed from and inserted into file, is classified as	direct access	sequential access	random access	duplicate access	sequential access
36	Distinct parts of documentation are called	technical documentation	documentation for user	program planning	both a and b	both a and b
37	Hardware and software specifications are part of	computing requirements	statement requirements	system flowchart	decision statement	computing requirements
38	Set of diagrams and notes that accompany program implementation are known as	program execution	program planning	program documentation	program existence	program documentation
39	Program documentation is used by the	programmers	system analyst	modifying the program	all of above	all of above
40	Program background, program functions and the computing requirements are part of	operations detail	predefined programs	decision box	Statement box	operations detail

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT II**

---

**Techniques of Problem Solving:** Flowcharting-decision table-algorithms-Structured programming concepts-Programming methodologies: top-down and bottom-up Programming.

**Techniques of Problem Solving**

**Flowcharting**

A flow chart is a step by step diagrammatic representation of the logic paths to solve a given problem. Or A flowchart is visual or graphical representation of an algorithm.

The flowcharts are pictorial representation of the methods to be used to solve a given problem and help a great deal to analyze the problem and plan its solution in a systematic and orderly manner. A flow chart when translated in to a proper computer language, results in a complete program.

**Advantages of Flowcharts**

1. The flowchart shows the logic of a problem displayed in pictorial fashion which facilitates easier checking of an algorithm.
2. The Flowchart is good means of communication to other users. It is also a compact means of recording an algorithm solution to a problem.
3. The flowchart allows the problem solver to break the problem into parts. These parts can be connected to make master chart.
4. The flowchart is a permanent record of the solution which can be consulted at a later time.

**Symbols used in Flow-Charts**

The symbols that we make use while drawing flowcharts as given below are as per conventions followed by International Standard Organization (ISO).

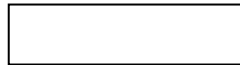
- a. Oval:** Rectangle with rounded sides is used to indicate either START/ STOP of the program. ..



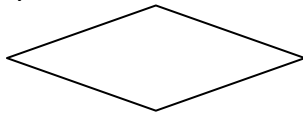
- b. Input and output indicators:** Parallelograms are used to represent input and output operations. Statements like INPUT, READ and PRINT are represented in these Parallelograms.



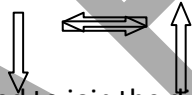
c. **Process Indicators:** - Rectangle is used to indicate any set of processing operation such as for storing arithmetic operations.



d. **Decision Makers:** The diamond is used for indicating the step of decision making and therefore known as decision box. Decision boxes are used to test the conditions or ask questions and depending upon the answers, the appropriate actions are taken by the computer. The decision box symbol is



e. **Flow Lines:** Flow lines indicate the direction being followed in the flowchart. In a Flowchart, every line must have an arrow on it to indicate the direction. The arrows may be in any direction

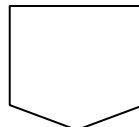


f. **On- Page connectors:** Circles are used to join the different parts of a flowchart and these circles are called on-page connectors. The uses of these connectors give a neat shape to the flowcharts.



In a complicated problems, a flowchart may run in to several pages. The parts of the flowchart on different pages are to be joined with each other. The parts to be joined are indicated by the circle.

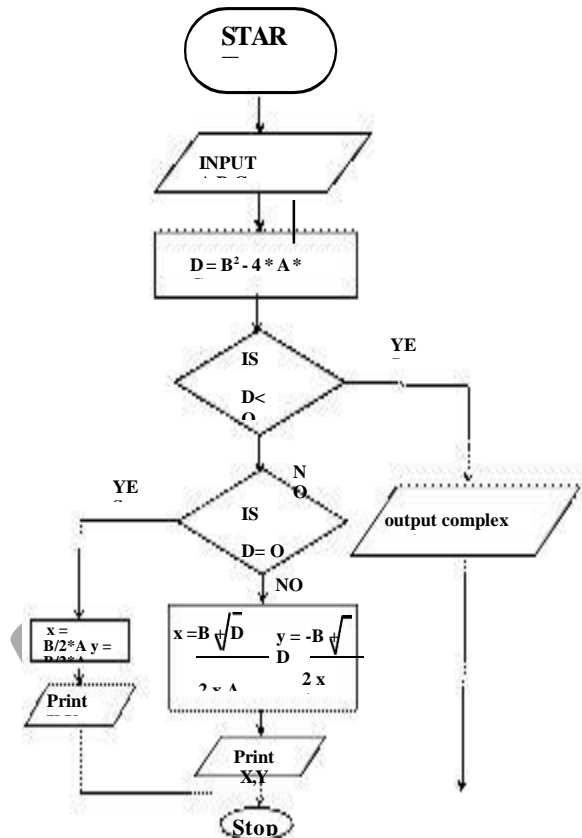
g. **Off-page connectors:** This connector represents a break in the path of flowchart which is too large to fit on a single page. It is similar to on-page connector. The connector symbol marks where the algorithm ends on the first page and where it continues on the second.



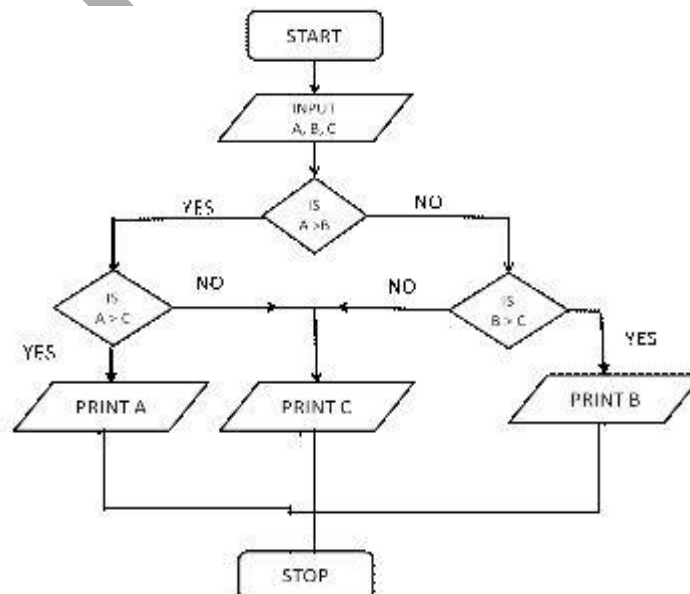
**Simple Problems using Flow Chart Draw the Flowchart for the following**

UNIT II

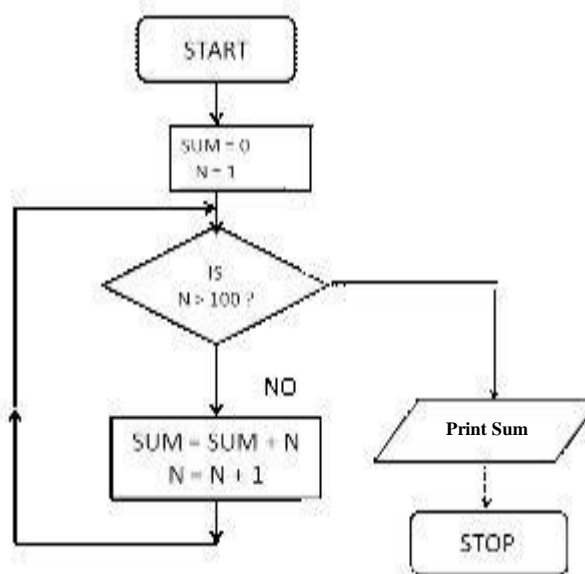
Draw the Flowchart to find Roots of Quadratic equation  $ax^2 + bx + c = 0$ . The coefficients a, b, c are the input data



Draw a flowchart to find out the biggest of the three unequal positive numbers.



Draw a flowchart for adding the integers from 1 to 100 and to print the sum.



### DECISION TABLE

#### What is Decision Table Testing?

Decision table testing is a testing technique used to test system behavior for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect** table where Cause and effects are captured for better test coverage.

A Decision Table is a tabular representation of inputs versus rules/cases/test conditions. Let's learn with an example.

#### **Example 1:** Decision Base Table for Login Screen

Let's create a decision table for a login screen.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT II

Log in

The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

#### Legend:

- T – Correct username/password
- F – Wrong username/password
- E – Error message is displayed

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT II**

---

- H – Home screen is displayed

**Interpretation:**

- Case 1 – Username and password both were wrong. The user is shown an error message.
- Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
- Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
- Case 4 – Username and password both were correct, and the user navigated to homepage

While converting this to test case, we can create 2 scenarios ,

- Enter correct username and correct password and click on login, and the expected result will be the user should be navigated to homepage

And one from the below scenario

- Enter wrong username and wrong password and click on login, and the expected result will be the user should get an error message
- Enter correct username and wrong password and click on login, and the expected result will be the user should get an error message
- Enter wrong username and correct password and click on login, and the expected result will be the user should get an error message

As they essentially test the same rule.

**Example 2:** Decision Base Table for Upload Screen

Now consider a dialogue box which will ask the user to upload photo with certain conditions like –



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT II

1. You can upload only '.jpg' format image
2. file size less than 32kb
3. resolution 137\*177.

If any of the conditions fails the system will throw corresponding error message stating the issue and if all conditions are met photo will be updated successfully

\*upload .jpg file with size not more than 32kb and resolution 137\*177

Let's create the decision table for this case.

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Format	.jpg	.jpg	.jpg	.jpg	Not .jpg	Not .jpg	Not .jpg	Not .jpg
Size	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb
resolution	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177
Output	Photo uploaded	Error message resolution mismatch	Error message size mismatch	Error message size and resolution mismatch	Error message for format mismatch	Error message for format and resolution mismatch	Error message for format and size mismatch	Error message for format, size, and resolution

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019


COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT II

				h		h	h	mismatch h
--	--	--	--	---	--	---	---	---------------

For this condition, we can create 8 different test cases and ensure complete coverage based on the above table.

- 
1. Upload a photo with format '.jpg', size less than 32kb and resolution 137\*177 and click on upload. Expected result is Photo should upload successfully
  2. Upload a photo with format '.jpg', size less than 32kb and resolution not 137\*177 and click on upload. Expected result is Error message resolution mismatch should be displayed
  3. Upload a photo with format '.jpg', size more than 32kb and resolution 137\*177 and click on upload. Expected result is Error message size mismatch should be displayed
  4. Upload a photo with format '.jpg', size less than 32kb and resolution not 137\*177 and click on upload. Expected result is Error message size and resolution mismatch should be displayed
  5. Upload a photo with format other than '.jpg', size less than 32kb and resolution 137\*177 and click on upload. Expected result is Error message for format mismatch should be displayed
  6. Upload a photo with format other than '.jpg', size less than 32kb and resolution not 137\*177 and click on upload. Expected result is Error message format and resolution mismatch should be displayed
  7. Upload a photo with format other than '.jpg', size more than 32kb and resolution 137\*177 and click on upload. Expected result is Error message for format and size mismatch should be displayed
  8. Upload a photo with format other than '.jpg', size more than 32kb and resolution not 137\*177 and click on upload. Expected result is Error message for format, size and resolution mismatch should be displayed

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT II

---

**Why is Decision Table Testing is important?**

This testing technique becomes important when it is required to test different combination. It also helps in better test coverage for complex business logic.

Boundary value and equivalent partition are other similar techniques used to ensure better coverage. They are used if the system shows the **same** behavior for a large set of inputs. However, in a system where for each set of input values the system behavior is **different**, boundary value and equivalent partitioning technique are not effective in ensuring good test coverage.

In this case, decision table testing is a good option. This technique can make sure of good coverage, and the representation is simple so that it is easy to interpret and use.

This table can be used as the reference for the requirement and for the functionality development since it is easy to understand and cover all the combinations.

**The** significance of this technique becomes immediately clear as the number of inputs increases. Number of possible Combinations is given by  $2^n$ , where  $n$  is the number of Inputs. For  $n = 10$ , **which is very common in the web based testing, having big input forms, the number of combinations will be 1024. Obviously, you cannot test all but** you will choose a rich sub-set of the possible combinations using decision based testing technique

**Advantages of decision table testing**

When the system behavior is different for different input and not same for a range of inputs, both equivalent partitioning, and boundary value analysis won't help, but decision table can be used.

The representation is simple so that it can be easily interpreted and is used for development and business as well.

This table will help to make effective combinations and can ensure a better coverage for testing

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT II

---

Any complex business conditions can be easily turned into decision tables

In a case we are going for 100% coverage typically when the input combinations are low, this technique can ensure the coverage.

**Disadvantages of decision table testing**

The main disadvantage is that when the number of input increases the table will become more complex

**ALGORITHMS**

**Definition**

An algorithm is a precise sequence of instructions for solving a problem. A set of sequential steps usually written in Ordinary Language to solve a given problem is called Algorithm.

It may be possible to solve the problem in more than one way, resulting in more than one algorithm. The choice of various algorithms depends on the factors like reliability, accuracy and easy to modify. The most important factor in the choice of algorithm is the time requirement to execute it, after writing code in High-level language with the help of a computer. The algorithm which will need the least time when executed is considered the best.

**Steps involved in algorithm development**

An algorithm can be defined as “a complete, unambiguous, finite number of logical steps for solving a specific problem “

**Step1. Identification of input:** For an algorithm, there are quantities to be supplied called input and these are fed externally. The input is to be identified first for any specified problem.

**Step2: Identification of output:** From an algorithm, at least one quantity is produced, called for any specified problem.

**Step3 : Identification the processing operations :** All the calculations to be performed in order to lead to output from the input are to be identified in an orderly manner.

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT II**

---

**Step4 : Processing Definiteness** : The instructions composing the algorithm must be clear and there should not be any ambiguity in them.

**Step5 : Processing Finiteness** : If we go through the algorithm, then for all cases, the algorithm should terminate after a finite number of steps.

**Step6 : Possessing Effectiveness** : The instructions in the algorithm must be sufficiently basic and in practice they can be carried out easily.

**An algorithm must possess the following properties**

- 1.Finiteness:**An algorithm must terminate in a finite number of steps
- 2.Definiteness:** Each step of the algorithm must be precisely and unambiguously stated
- 3.Effectiveness:** Each step must be effective, in the sense that it should be primitive easily convertible into program statement) can be performed exactly in a finite amount of time.
- 4.Generality:** The algorithm must be complete in itself so that it can be used to solve problems of a specific type for any input data.
- 5.Input/output:** Each algorithm must take zero, one or more quantities as input data produce one or more output values. An algorithm can be written in English like sentences or in any standard representation sometimes, algorithm written in English like languages are called Pseudo Code

**Example**

1. Suppose we want to find the average of three numbers, the algorithm is as follows

**Step 1** Read the numbers a, b, c

**Step 2** Compute the sum of a, b and c

**Step 3** Divide the sum by 3

**Step 4** Store the result in variable d

**Step 5** Print the value of d

**Step 6** End of the program

**Algorithms for Simple Problem**

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT II

**Write an algorithm for the following**

Write an algorithm to calculate the simple interest using the formula. Simple interest =  $P \times N \times R / 100$ .

Where P is principle Amount, N is the number of years and R is the rate of interest.

Step 1: Read the three input quantities' P, N and R.

Step 2 : Calculate simple interest as

Simple interest =  $P \times N \times R / 100$

Step 3: Print simple interest.

Step 4: Stop.

**Pseudo code**

**Writing a program** is often called "writing code" or "implementing an algorithm". So the code (or source code) is actually the program itself. **pseudocode** is a simple and concise sequence of English-like instructions to solve a problem

The **Pseudo code** is neither an algorithm nor a program. It is an abstract form of a program. It consists of English like statements which perform the specific operations. It is defined for an algorithm. It does not use any graphical representation. In pseudo code, the program is represented in terms of words and phrases, but the syntax of program is not strictly followed.

**Advantages:** \* Easy to read, \* Easy to understand, \* Easy to modify.

**Example:** Write a pseudo code to perform the basic arithmetic operations.

Read n1, n2

Sum =  $n1 + n2$

Diff =  $n1 - n2$

Mult =  $n1 * n2$

Quot =  $n1 / n2$

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT II

Print sum, diff, mult, quot

End.

**Compiling** is the process of converting a program into instructions that can be understood by the computer.

Structured Programming

One of the most important features of structured programming is that all program structures have only one entry point and one exit point.

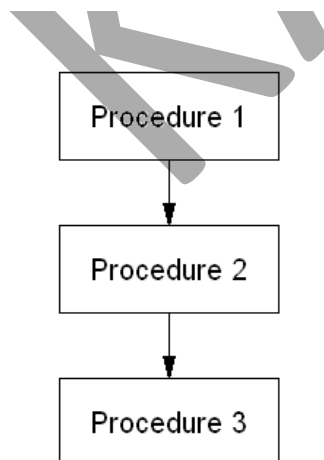
There are three main types of control structures

1. **Sequence** (see figure 2)
2. **Selection** (IF-THEN), (IF-THEN-ELSE), see figure 3; IF-THEN-ELSEIF-THEN-ELSE, see figure 4)
3. **Iteration** (DO-WHILE, see figure 5; DO-UNTIL, see figure 6)

A sequence of instructions in a sequence structure must all be carried out.

In a selection structure certain instructions will only be carried out **IF** a certain condition is found to be true.

In an iteration structure certain instructions will be repeatedly carried out until a condition is no longer true (DO-WHILE) or alternatively until a condition is no longer false (DO-UNTIL). Such structures consist of a loop which takes the program back to the top of the instructions to be repeated.



**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT II**

---

Figure 2: An example of a sequence control structure.

**PROGRAMMING METHODOLOGIES**

When programs are developed to solve real-life problems like inventory management, payroll processing, student admissions, examination result processing, etc. they tend to be huge and complex. The approach to analyzing such complex problems, planning for software development and controlling the development process is called programming methodology.

**Types of Programming Methodologies**

There are many types of programming methodologies prevalent among software developers –

**Procedural Programming**

Problem is broken down into procedures, or blocks of code that perform one task each. All procedures taken together form the whole program. It is suitable only for small programs that have low level of complexity.

Example – For a calculator program that does addition, subtraction, multiplication, division, square root and comparison, each of these operations can be developed as separate procedures. In the main program each procedure would be invoked on the basis of user's choice.

**Object-oriented Programming**

Here the solution revolves around entities or objects that are part of problem. The solution deals with how to store data related to the entities, how the entities behave and how they interact with each other to give a cohesive solution.

Example – If we have to develop a payroll management system, we will have entities like employees, salary structure, leave rules, etc. around which the solution must be built.

**Functional Programming**

Here the problem, or the desired solution, is broken down into functional units. Each unit performs its own task and is self-sufficient. These units are then stitched together to form the complete solution.



**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT II**

---

Example – A payroll processing can have functional units like employee data maintenance, basic salary calculation, gross salary calculation, leave processing, loan repayment processing, etc.

**Logical Programming**

Here the problem is broken down into logical units rather than functional units. Example: In a school management system, users have very defined roles like class teacher, subject teacher, lab assistant, coordinator, academic in-charge, etc. So the software can be divided into units depending on user roles. Each user can have different interface, permissions, etc.

Software developers may choose one or a combination of more than one of these methodologies to develop a software. Note that in each of the methodologies discussed, problem has to be broken down into smaller units. To do this, developers use any of the following two approaches –

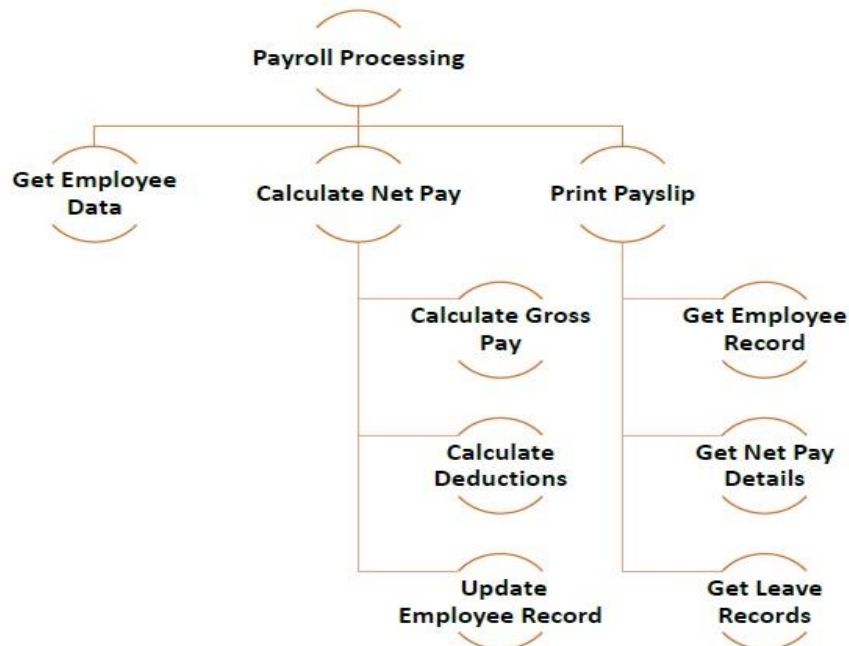
Top-down approach

Bottom-up approach

**Top-down or Modular Approach**

The problem is broken down into smaller units, which may be further broken down into even smaller units. Each unit is called a module. Each module is a self-sufficient unit that has everything necessary to perform its task.

The following illustration shows an example of how you can follow modular approach to create different modules while developing a payroll processing program.

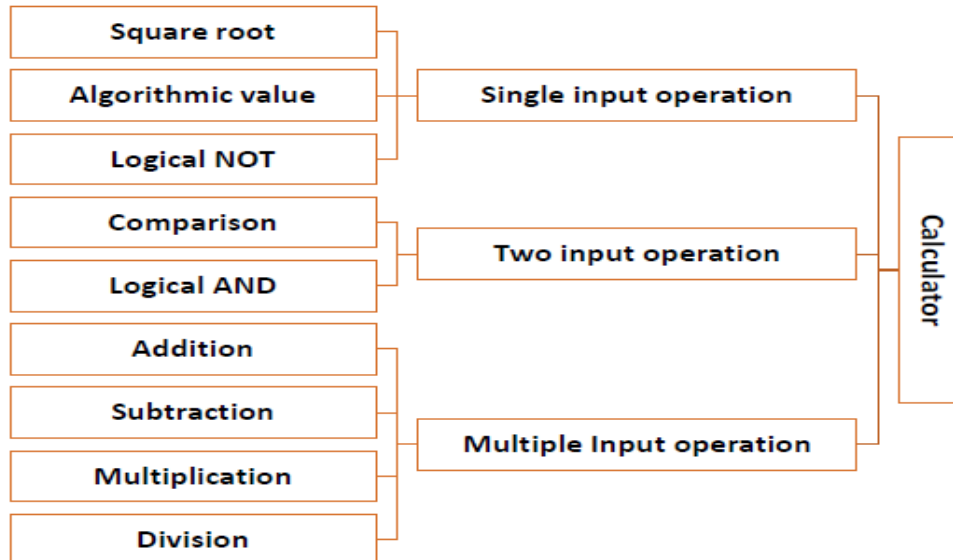


### Bottom-up Approach

In bottom-up approach, system design starts with the lowest level of components, which are then interconnected to get higher level components. This process continues till a hierarchy of all system components is generated. However, in real-life scenario it is very difficult to know all lowest level components at the outset. So bottom up approach is used only for very simple problems.

Let us look at the components of a calculator program.

UNIT II



**STRUCTURED PROGRAMMING CONCEPTS**

**What is Structured Programming Language**

**Definition** – It is a programming method which aimed at improving quality, clarity and access time of computer program by the use of block structures, subroutines, for and while loops. This programming features will be helpful when concept of exception handling is needed in the program. It uses various control structures, sub routines, blocks and theorem. The theorems involved in structure programming are Sequence, Selection, Iteration and Recursion. Most of the programming language uses structured programming language features such as ALGOL, Pascal, PL/I, Ada, C, etc. The structure programming enforces a logical structure on the program being written to make it more efficient and easy to modify and understand. What is Structured Programming Language is explained in simple and precise manner.

**Why C – Language is called as Structured Programming language?**

In order to accomplish any task, C-language divide the problem into smaller modules called functions or procedure each of which handles a particular job. That is why C-language is also called as the structured programming language. The program which solves the entire problem is a collection of such functions.

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT II**

---

**What are the main features of Structural Programming language?**

1. Division of Complex problems into small procedures and functions.
2. No presence of GOTO Statement
3. The main statement include – If-then-else, Call and Case statements.
4. Large set of operators like arithmetic, relational, logical, bit manipulation, shift and part word operators.
5. Inclusion of facilities for implementing entry points and external references in program.

**What is difference between Structured and Unstructured Programming Language?**

1. The main difference between structured and unstructured programming language is that a structured programming language allows a programmer to dividing the whole program into smaller units or modules. But in unstructured programming language, the whole program must be written in single continuous way; there is no stop or broken block.
2. Structured Programming language is a subset of Procedural Programming language. But in unstructured Programming language no subset exists.
3. Structured Programming language is a precursor to Object Oriented Programming (OOP) language. But another one is not.
4. Structured Programming language produces readable code while Unstructured Programming language produces hardly readable code “spaghetti”.
5. Structured Programming language has some limitations while unstructured Programming language offers freedom to program as they want.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT II

6. Structured Programming language is easy to modify and debug, while unstructured Programming language is very difficult to modify and debug.
7. Examples of Structured Programming language are C, C+, C++, C#, Java, PERL, Ruby, PHP, ALGOL, Pascal, PL/I and Ada; and example of unstructured Programming language are BASIC (early version), JOSS, FOCAL, MUMPS, TELCOMP, COBOL.

#### Possible Questions:

#### 2 Mark Questions

1. Distinguish between pseudo code and flowchart.
2. Define an algorithm
3. Define control flow statement with an eg:
4. Draw a flow chart for factorial given number (3\*3)
5. Draw the flow chart sum of n numbers
6. What is pseudo code?

#### 6 Mark Questions

1. Explain the basic symbols for constructing a flowchart & discuss about the basic guidelines for preparing flowchart.
2. Develop algorithm for i) Prime number or not (ii) odd or even
3. Develop algorithm for Celsius to Fahrenheit and vice versa
4. Discuss building blocks of algorithm
5. Summarize the symbol used in flow chart
6. Develop algorithm and flow chart for Celsius to Fahrenheit and vice versa.
7. Summarize decision table with neat sketch.
8. Write algorithm, pseudo code and flow chart for any example?



# KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc( CS)

(BATCH 2016-2019)

V SEMESTER

PROGRAMMING IN PYTHON (16CSU504 B )

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

## UNIT II

S.NO	QUESTIONS	OPT 1	OPT 2	OPT 3	OPT 4	ANSWER
1	A search algorithm takes _____ as an input and returns _____ as an output.	Input, output	Problem, solution	Solution, problem	Parameters, sequence of actions	Parameters, sequence of actions
2	In flow chart, diamond shaped symbol is used to represent	decision box	statement box	error box	if-statement box	decision box
3	The repetition of statement in the Jackson method is represented by	drawing sign of equal	drawing sign of subtraction	drawing an asterisk	drawing an arrow	drawing an asterisk
4	Symbol used in flowchart such as rectangle with the horizontal lines on two sides is used for	defined statement	predefined process	error fix	variables defined	predefined process
5	Program link with other parts of the program or connectors in flowchart are represented by	rhombus	parallelogram	circle	trapezoid	circle
6	Part of algorithm which is repeated for the fixed number of times is classified as	iteration	selection	sequence	reverse action	iteration
7	Defining data requirements such as input and output is classified as	process definition	function definition	print definition	writing purpose	function definition
8	Method used in writing and designing of a program is termed as	bottom-up method	top-down method	split method	binary states method	top-down method

9	The PAC stands for	Program Analysis Chart	Problem Algorithm Code	Problem Access Code	Problem Analysis Chart	Problem Analysis Chart
10	In interactivity chart the darkened circle indicates _____.	duplicate module	loop	decision	no special meaning	loop
11	. In interactivity chart the diamond indicates _____.	duplicate module	loop	decision	no special meaning	decision
12	The interactivity chart is also known as _____.	IPO Chart	Problem Analysis Chart	flow chart	structure chart	structure chart
13	The IPO stands for	Input Programming Option	Input Programming Output	Input Processing Output	Input Operating Operation	Input Processing Output
14	The main measure for efficiency algorithm are-	Processor and Memory	Complexity and Capacity	Data and Space	Time and space	Time and space
15	The time factor when determining the efficiency of algorithm is measured by	Counting microseconds	Counting the number of key operations	Counting the number of statements	Counting the kilobytes of algorithm	Counting the number of key operations
16	For retail shopping software which table would be example of Decision Table?	A table containing rules of discount.	A table containing rules for interfaces between components.	A table containing rule of employee behavior.	A table containing rules for combination of input	A table containing rules of discount.
17	When different combination of input requires different combination of actions,Which of the following technique is used in such situation?	Boundary Value Analysis	Equivalence Partition	Decision Table	Decision Coverage	Decision Table
18	Which of the following Use Cases are useful?	Performance Testing	Business Scenarios	Static Testing	Unit Testing	Business Scenarios
19	A decision table is	a truth table	a table which facilitates taking decisions	a table listing conditions and actions to be taken based on the testing	a table in a Decision Support System	a table listing conditions and actions to be taken based on the

20	A decision table	has a structured English equivalent representation	cannot be represented using structured English	does not have an equivalent algorithmic representation	cannot be used to represent processes in a DFD	has a structured English equivalent representation
21	A decision table is preferable when the number of	conditions to be checked in a procedure is small	conditions to be checked in a procedure is large	actions to be carried out are large	actions to be carried out are small	conditions to be checked in a procedure is large
22	Decision table description of data processing is	non-procedural specification	procedural specification	purely descriptive specification	very imprecise specification	non-procedural specification
23	A rule in a limited entry decision table is a	row of the table consisting of condition entries	row of the table consisting of action entries	column of the table consisting of condition entries and the corresponding action	columns of the tables consisting of conditions of the stub	column of the table consisting of condition entries and the corresponding
24	In a limited entry decision table the condition entries may be	Y or N only	Y, N or –	A binary digit	Any integer	Y, N or –
25	In a limited entry decision table the action entries	list X or – corresponding to actions to be executed	list the conditions to be tested	have Y or N or – entries	list the actions to be taken	list X or – corresponding to actions to be executed
26	Decision Tables are preferred when	Too many conditions need to be tested	Sequencing of testing conditions is important	When there are many loops to be performed	When too many actions are to be taken	Too many conditions need to be tested
27	The data type for numbers such as 3.14159 is:	double	int	real	string	double
28	To write a program function i.e. program for the sum of four integers, the program refinement first level includes	input four numbers	calculate sum	print the values	display the values	input four numbers



29	Data which is used to test each feature of the program and is carefully selected is classified as	program output	program input	test data	test program	test data
30	Object oriented programming employs_____ programming approach.	top-down	procedural	bottom-up	all of these.	bottom-up
31	_____are used for generic programming.	Inheritance	Virtual Functions	Templates	None of these	Templates
32	Higher-order functions are not built into the	structural language	object oriented programming	JAVA	C++	structural language
33	In structural language, we can't add a new sort of	loop	function	variable	constant	loop
34	No matter how well the structured programming is implemented, large programs become excessively	complex	simpler	formal	network	complex
35	Unrelated functions and data in procedural paradigm provides real world	poor model	simplest model	formal model	good model	poor model
36	Function has unrestricted access to	global data	local data	local variable	global name	global data
37	The _____ provides pictorial representation of given problem.	Algorithm	Flowchart	Pseudocode	All of these	Flowchart
38	_____ is a procedure or step by step process for solving a problem.	Algorithm	Flowchart	Pseudocode	All of these	Algorithm
39	The _____ symbol is used at the beginning of a flow chart.	Circle	Rectangle	Diamond	None of these	Circle
40	The _____ symbol is used to represent decision in flowchart.	Circle	Rectangle	Diamond	None of these	Diamond

41	The _____ symbol is used to represent process in flowchart.	Circle	Rectangle	Diamond	None of these	Rectangle
42	_____ symbol is used to represent input and output operation in flowchart.	Circle	Rectangle	Diamond	Parallelogram	Parallelogram
43	_____ is a symbol used connects two symbols of flowchart.	Circle	Rectangle	Diamond	Arrow	Arrow

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

UNIT-III

**Overview of Programming:** Structure of a Python Program-Elements of Python.

**OVERVIEW OF PROGRAMMING:**

**PYTHON INTRODUCTION**

**Python** is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is *easy to learn* yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and *dynamic typing* with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports *multiple programming pattern*, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as *multipurpose* because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is *dynamically typed* so we can write `a=10` to assign an integer value in an integer variable.

Python makes the development and debugging *fast* because there is no compilation step included in python development and edit-test-debug cycle is very fast.

**Python Features**

Python provides lots of features that are listed below.

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT III**

---

**1) Easy to Learn and Use**

Python is easy to learn and use. It is developer-friendly and high level programming language.

**2) Expressive Language**

Python language is more expressive means that it is more understandable and readable.

**3) Interpreted Language**

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

**4) Cross-platform Language**

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

**5) Free and Open Source**

Python language is freely available at [official web address](#). The source-code is also available. Therefore it is open source.

**6) Object-Oriented Language**

Python supports object oriented language and concepts of classes and objects come into existence.

**7) Extensible**

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

### 8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

### 9) GUI Programming Support

Graphical user interfaces can be developed using Python.

### 10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

### Python History



- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:
  - ABC language.
  - Modula-3

## KARPAGAM ACADEMY OF HIGHER EDUCATION

---

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT III

---

#### Python Version

Python programming language is being updated regularly with new features and supports. There are lots of updations in python versions, started from 1994 to current release.

A list of python versions with its released date is given below.

Python Version	Released Date
Python 1.0	January 1994
Python 1.5	December 31, 1997
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
Python 2.1	April 17, 2001
Python 2.2	December 21, 2001
Python 2.3	July 29, 2003
Python 2.4	November 30, 2004
Python 2.5	September 19, 2006
Python 2.6	October 1, 2008
Python 2.7	July 3, 2010
Python 3.0	December 3, 2008
Python 3.1	June 27, 2009
Python 3.2	February 20, 2011

## KARPAGAM ACADEMY OF HIGHER EDUCATION

---

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT III

---

Python 3.3	September 29, 2012
Python 3.4	March 16, 2014
Python 3.5	September 13, 2015
Python 3.6	December 23, 2016
Python 3.6.4	December 19, 2017

#### Python Applications Area

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifying applications areas where python can be applied.

#### **1) Web Applications**

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, BeautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: PythonWikiEngines, Pycoco, PythonBlogSoftware etc.

#### **2) Desktop GUI Applications**

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, PyQt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

#### **3) Software Development**

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT III**

---

#### **4) Scientific and Numeric**

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

#### **5) Business Applications**

Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high level application platform.

#### **6) Console Based Application**

We can use Python to develop console based applications. For example: **IPython**.

#### **7) Audio or Video based Applications**

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

#### **8) 3D CAD Applications**

To create CAD application Fandango is a real application which provides full features of CAD.

#### **9) Enterprise Applications**

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

#### **10) Applications for Images**

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python



CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

**Python Example**

Python is easy to learn and code and can be execute with python interpreter. We can also use Python interactive shell to test python code immediately.

A simple hello world example is given below. Write below code in a file and save with **.py** extension. Python source file has **.py** extension.

**hello.py**

```
print("hello world by python!")
```

Execute this example by using following command.

Python3 hello.py

After executing, it produces the following output to the screen.

**Output**

```
hello world by python!
```

***Python Example using Interactive Shell***

Python interactive shell is used to test the code immediately and does not require to write and save code in file.

Python code is simple and easy to run. Here is a simple Python code that will print "Welcome to Python".

**A simple python example is given below.**

```
>>> a="Welcome To Python"
```

```
>>> print a
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

## UNIT III

---

Welcome To Python

```
>>>
```

### *Python 3.4 Example*

In python 3.4 version, you need to add parenthesis () in a string code to print it.

```
>>> a=("Welcome To Python Example")
```

```
>>> print a
```

Welcome To Python Example

```
>>>
```

### Structure of a Python Program

Python is very flexible in terms of program structure. That is, the syntax does not require a specific ordering to functions or classes within a module or methods within a class. (Note, however, that functions and classes must be defined before they can be used.) But large programs can quickly become unmanageable and difficult to read. This section outlines several requirements in terms of program structure.

- The Main Driver
- Modules
- Classes
- Storage Classes

### The Main Driver

---

The Python source file that contains the starting point or the first instruction to be executed is known as the *driver* module. There are two styles that can be used for organizing the driver module, depending on whether functions are used. In no case should a class be defined within the driver module.

### Driver Without Functions

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT III**

---

The first statement to be executed is the first statement at file level (outside all functions and class definitions). For simple drivers that do not need to be organized into multiple functions, all statements can be at file level as illustrated by the sample program below. For this style, which does not include any function definitions, you are to use the following structure:

- All import statements that are needed by code within the driver should be specified immediately following the file header.
- Constant variables, if any, should be specified after the import statements.
- The executable statements follow the constant variables.

```
x = float( input("Enter a real value:") )  
y = math.sqrt( x )  
print( "The square root of", x, "is", y )
```

**Driver With Functions**

Large driver modules should be organized into multiple functions and include a main() routine. The order in which functions are listed within a file is not important, but the order of executable statements at file level is. When subdividing a program into various functions using a top-down design, it is good programming practice to place all executable statements within functions and to specify one function as the main routine, as illustrated in diceroll.py program below. With this approach:



- The main function, which is commonly named main() due to that name being used in other languages, should be defined first.
- All remaining functions definitions must follow, in any order.
- At the bottom of the file, a single file-level statement (not including constant variables) is used to call the driver function.
- All function definitions must follow any import statements and constant variables that are defined after the file header.

```
from random import *
```

```
# Minimum number of sides on a die.  
MIN_SIDES = 4
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

*# Our very own main routine for a top-down design.*

```
def main():  
    print( "Dice roll simulation." )  
    numSides = int( input("How many sides should the die have? ") )  
    if numSides < MIN_SIDES :  
        numSides = MIN_SIDES  
    value = rollDice( numSides )  
    print( "You rolled a", value )
```

*# Simulate the rollowing of two nSided dice.*

```
def rollDice( nSides ):  
    die1 = randint( 1, nSides + 1 )  
    die2 = randint( 1, nSides + 1 )  
    return die1 + die2
```

*# Call the main routine which we defined first.*

```
main()
```

## Modules

---

Modules can contain any combination of function and class definitions, constant and non-constant variable declarations and executable statements. In this course, all non-driver modules will be limited to class definitions and constant variable declarations and should be organized as follows:

- Specify all import statements that are needed by code within the driver immediately following the file header. Note, if a module is not directly needed by this module, it should not be imported.
- Constant variables, if any, should be defined after the import statements.
- The class definitions should follow the constant variables, if any.

**One Public Class.** If a module contains a single public class:

- The public class should be listed first.
- Followed by any private or storage classes.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT III

---

For example, the following code segment illustrates the definition of two classes, one public and the other private.

```
class Bag :  
    # Constructs an empty bag.  
    def __init__( self ):  
        self._theltems = list()  
        .....  
  
    def __iter__( self ):  
        return _BagIterator( self._theltems )  
  
# Defines the iterator used with a Bag.  
class _BagIterator :  
    .....
```

**Multiple Public Classes.** If a module contains multiple public class definitions:

- Any storage or other private classes used by the public class can be defined immediately after the public class.
- Each additional public class, along with its private or storage classes will follow the first group.

The following code segment illustrates a module containing several public classes, all of which are related to array definitions.

```
# Defines a 1-D array implemented using hardware supported arrays.  
class Array :  
    .....  
  
# Defines a 2-D array implemented as an array of arrays.  
class Array2D :  
    .....  
  
# Defines a multi-dimensional array using a single 1-D array and
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

## UNIT III

---

*# row-major ordering.*

**class** MultiArray :

.....

### Classes

---

To aide in readability and understanding of a class definition, the class should be well structured and documented. The comments required for a class and its methods are described in the “Comments” section.

**Organization.** The methods defined for a class can technically occur in any order. To provide a well structured implementation and to aide the reader, you are to use the following structure:

- Class constants, if any, should be defined first.
- The constructor should be the first method defined.
- The public methods should follow the constructor.
- The special `__iter__()` method should be listed after all other public methods.
- Helper methods (private methods), if any, should be at the bottom.

### Storage Classes

---

A storage class is used to create objects for simply storing structured or multi-component data items. Typically, they only contain a constructor used to create and initialize the various data fields. Any class or code segment that has access to the storage class can directly reference the various data attributes.

**class** MapElement :

**def** `__init__`( `self`, key, value ):

`self.key` = key

`self.value` = value

- Storage classes should be defined within the same module in which they will be used.
- The data attributes of a storage class should be defined as public and thus not include the preceding underscore as used with other classes.
- The visibility status (public or private) of the storage class should be clearly stated in the class description. A private storage class should not be imported into any other module.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

### Python Variables

Variable is a name which is used to refer memory location. Variable also known as identifier and used to hold value.

In Python, we don't need to specify the type of variable because Python is a type infer language and smart enough to get variable type.

Variable names can be a group of both letters and digits, but they have to begin with a letter or an underscore.

It is recommended to use lowercase letters for variable name. Rahul and rahul both are two different variables.

#### **Assigning value to variable:**

Value should be given on the right side of assignment operator(=) and variable on left side.

```
>>>counter =45  
print(counter)
```

Assigning a single value to several variables simultaneously:

```
>>> a=b=c=100
```

Assigning multiple values to multiple variables:

```
>>> a,b,c=2,4,"ram"
```

### ELEMENTS OF PYTHON

#### **Python List**

Python list is a data structure which is used to store various types of data.

In Python, lists are mutable i.e., Python will not create a new list if we modify an element of the list.

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT III**

---

It works as a container that holds other objects in a given order. We can perform various operations like insertion and deletion on list.

A list can be composed by storing a sequence of different type of values separated by commas.

Python list is enclosed between square([]) brackets and elements are stored in the index basis with starting index 0.

### **Python List Example**

```
data1=[1,2,3,4];  
data2=['x','y','z'];  
data3=[12.5,11.6];  
data4=['raman','rahul'];  
data5=[];  
data6=['abhinav',10,56.4,'a'];
```

A list can be created by putting the value inside the square bracket and separated by comma.

### **Python List Syntax**

```
<list_name>=[value1,value2,value3,...,valuen];
```

### **Syntax to Access Python List**

```
<list_name>[index]
```

Python allows us to access value from the list by various ways.

### **Python Accessing List Elements Example**

```
data1=[1,2,3,4];  
data2=['x','y','z'];
```



```
print data1[0]
print data1[0:2]
print data2[-3:-1]
print data1[0:]
print data2[:2]
```

Output:

```
1
[1, 2]
['x', 'y']
[1, 2, 3, 4]
['x', 'y']
>>>
```

### *Elements in a Lists:*

Following are the pictorial representation of a list. We can see that it allows to access elements from both end (forward and backward).

Data=[1,2,3,4,5];

Forward indexing

→ 0    1    2    3    4

1	2	3	4	5
---	---	---	---	---

-5    -4    -3    -2    -1

← Backward indexing

[javatpoint.com](http://javatpoint.com)

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT III

---

Data[0]=1=Data[-5] , Data[1]=2=Data[-4] , Data[2]=3=Data[-3] ,  
=4=Data[-2] , Data[4]=5=Data[-1].

***Python List Operations***

Apart from creating and accessing elements from the list, Python allows us to perform various other operations on the list. Some common operations are given below

**a) Adding Python Lists**

In Python, lists can be added by using the concatenation operator(+) to join two lists.

**Add lists Example 1**

```
list1=[10,20]
list2=[30,40]
list3=list1+list2
print list3
```

**Output:** [10, 20, 30, 40]

**Add lists Example 2**

```
list1=[10,20]
list1+30
print list1
```

**Output:**

Traceback (most recent call last):

File "C:/Python27/lis.py", line 2, in <module>

list1+30

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

**b) Python Replicating lists**

Replicating means repeating, It can be performed by using '\*' operator by a specific number of time.

**Python list Replication Example**

```
list1=[10,20]
print list1*1
```

**Output:**[10, 20]

**c)Python List Slicing**

A subpart of a list can be retrieved on the basis of index. This subpart is known as list slice. This feature allows us to get sub-list of specified start and end index.

**Python List Slicing Example**

```
list1=[1,2,4,5,7]
print list1[0:2]
print list1[4]
list1[1]=9
print list1
```

**Output:**

```
[1, 2]
7
[1, 9, 4, 5, 7]
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT III

---

### *Python List Other Operations*

Apart from above operations various other functions can also be performed on List such as Updating, Appending and Deleting elements from a List.

### **Python Updating List**

To update or change the value of particular index of a list, assign the value to that particular index of the List.

### **Python Updating List Example**

```
data1=[5,10,15,20,25]
print "Values of list are: "
print data1
data1[2]="Multiple of 5"
print "Values of list are: "
print data1
```

Output:

```
Values of list are:
[5, 10, 15, 20, 25]
Values of list are:
[5, 10, 'Multiple of 5', 20, 25]
```

### **Appending Python List**

Python provides, append() method which is used to append i.e., add an element at the end of the existing elements.

### **Python Append List Example**

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

```
list1=[10,"rahul",'z']  
print "Elements of List are: "  
print list1  
list1.append(10.45)  
print "List after appending: "  
print list1
```

Output:

```
Elements of List are:  
[10, 'rahul', 'z']  
List after appending:  
[10, 'rahul', 'z', 10.45]
```

### Deleting Elements

In Python, **del** statement can be used to delete an element from the list. It can also be used to delete all items from startIndex to endIndex.

### Python delete List Example

```
list1=[10,'rahul',50.8,'a',20,30]  
print list1  
del list1[0]  
print list1  
del list1[0:3]  
print list1
```

Output:

```
[10, 'rahul', 50.8, 'a', 20, 30]
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT III**

---

```
['rahul', 50.8, 'a', 20, 30]
[20, 30]
```

***Python lists Method***

Python provides various Built-in functions and methods for Lists that we can apply on the list.

**Following are the common list functions.**

Function	Description
min(list)	It returns the minimum value from the list given.
max(list)	It returns the largest value from the given list.
len(list)	It returns number of elements in a list.
cmp(list1,list2)	It compares the two list.
list(sequence)	It takes sequence types and converts them to lists.

**Python List min() method Example**

This method is used to get min value from the list.

```
list1=[101,981,'abcd','xyz','m']
list2=['aman','shekhar',100.45,98.2]
print "Minimum value in List1: ",min(list1)
print "Minimum value in List2: ",min(list2)
```

**Output:**

Minimum value in List1: 101

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT III

---

Minimum value in List2: 98.2

### Python List max() method Example

This method is used to get max value from the list.

```
list1=[101,981,'abcd','xyz','m']
list2=['aman','shekhar',100.45,98.2]
print "Maximum value in List : ",max(list1)
print "Maximum value in List : ",max(list2)
```

#### Output:

```
Maximum value in List : xyz
Maximum value in List : shekhar
```

### Python List len() method Example

This method is used to get length of the the list.

```
list1=[101,981,'abcd','xyz','m']
list2=['aman','shekhar',100.45,98.2]
print "No. of elements in List1: ",len(list1)
print "No. of elements in List2: ",len(list2)
```

#### Output:

```
No. of elements in List1 : 5
No. of elements in List2 : 4
```

### Python List cmp() method Example

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT III**

---

Explanation: If elements are of the same type, perform the comparison and return the result. If elements are different types, check whether they are numbers.

- If numbers, perform comparison.
- If either element is a number, then the other element is returned.
- Otherwise, types are sorted alphabetically .

If we reached the end of one of the lists, the longer list is "larger." If both list are same it returns 0.

**Python List cmp() method Example**

```
list1=[101,981,'abcd','xyz','m']
list2=['aman','shekhar',100.45,98.2]
list3=[101,981,'abcd','xyz','m']
print cmp(list1,list2)
print cmp(list2,list1)
print cmp(list3,list1)
```

**Output:**

```
-1
1
0
```

**Python List list(sequence) method Example**

This method is used to form a list from the given sequence of elements.

```
seq=(145,"abcd",'a')
```



**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT III**

---

```
data=list(seq)
print "List formed is :",data
```

**Output:**

List formed is : [145, 'abcd', 'a']

There are following built-in methods of List

Methods	Description
index(object)	It returns the index value of the object.
count(object)	It returns the number of times an object is repeated in list.
pop()/pop(index)	It returns the last object or the specified indexed object. It removes the popped object.
insert(index,object)	It inserts an object at the given index.
extend(sequence)	It adds the sequence to existing list.
remove(object)	It removes the object from the given List.
reverse()	It reverses the position of all the elements of a list.
sort()	It is used to sort the elements of the List.

**Python List index() Method Example**

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

```
data = [786,'abc','a',123.5]
print "Index of 123.5:", data.index(123.5)
print "Index of a is", data.index('a')
```

Output:

```
Index of 123.5 : 3
Index of a is 2
```

Python List count(object) Method Example

```
data = [786,'abc','a',123.5,786,'rahul','b',786]
print "Number of times 123.5 occurred is", data.count(123.5)
print "Number of times 786 occurred is", data.count(786)
```

Output:

```
Number of times 123.5 occurred is 1
Number of times 786 occurred is 3
```

Python List pop()/pop(int) Method Example

```
data = [786,'abc','a',123.5,786]
print "Last element is", data.pop()
print "2nd position element:", data.pop(1)
print data
```

Output:

```
Last element is 786
2nd position element:abc
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

```
[786, 'a', 123.5]
```

**Python List insert(index,object) Method Example**

```
data=['abc',123,10.5,'a']
data.insert(2,'hello')
print data
```

**Output:**

```
['abc', 123, 'hello', 10.5, 'a']
```

**Python List extend(sequence) Method Example**

```
data1=['abc',123,10.5,'a']
data2=['ram',541]
data1.extend(data2)
print data1
print data2
```

**Output:**

```
['abc', 123, 10.5, 'a', 'ram', 541]
['ram', 541]
```

**Python List remove(object) Method Example**

```
data1=['abc',123,10.5,'a','xyz']
data2=['ram',541]
print data1
data1.remove('xyz')
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

```
print data1
print data2
data2.remove('ram')
print data2
```

Output:

```
['abc', 123, 10.5, 'a', 'xyz']
['abc', 123, 10.5, 'a']
['ram', 541]
[541]
```

Python List reverse() Method Example

```
list1=[10,20,30,40,50]
list1.reverse()
print list1
```

Output:

```
[50, 40, 30, 20, 10]
```

Python List sort() Method Example

```
list1=[10,50,13,'rahul','aakash']
list1.sort()
print list1
```

Output:

```
[10, 13, 50, 'aakash', 'rahul']
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT III

---

Python program to swap two variables

**Variable swapping:**

In computer programming, swapping two variables specifies the mutual exchange of values of the variables. It is generally done by using a temporary variable.

For example:

```
data_item x := 1
```

```
data_item y := 0
```

```
swap (x, y)
```

**After swapping:**

```
data_item x := 0
```

```
data_item y := 1
```

**See this example:**

```
# Python swap program
```

```
x = input('Enter value of x: ')
```

```
y = input('Enter value of y: ')
```

```
# create a temporary variable and swap the values
```

```
temp = x
```

```
x = y
```

```
y = temp
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

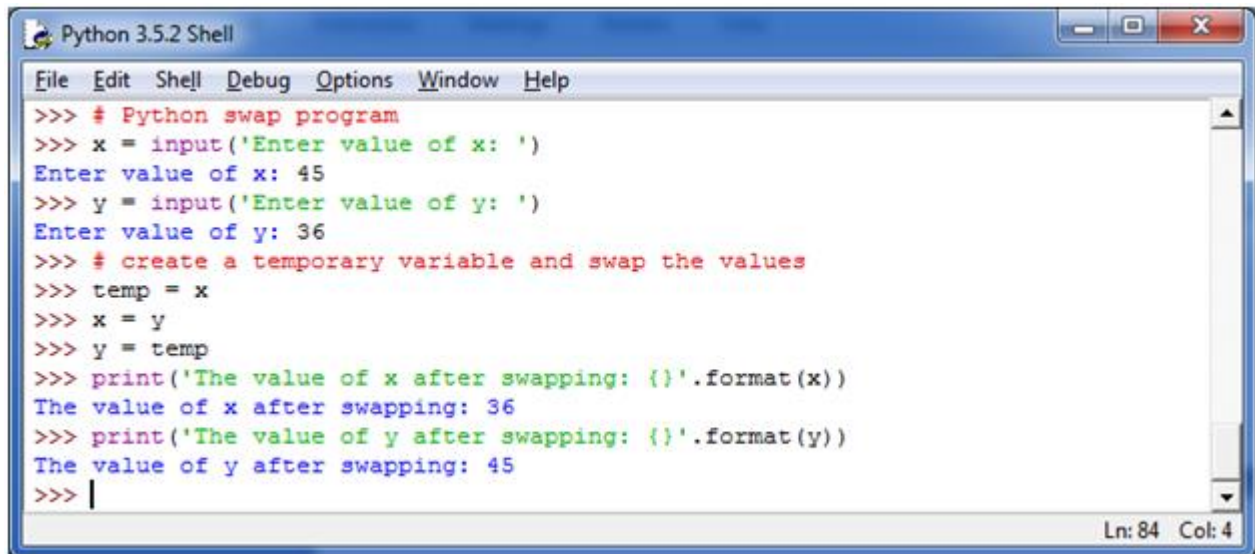
COURSE CODE : 16CSU504B

UNIT III

```
print('The value of x after swapping: {}'.format(x))
```

```
print('The value of y after swapping: {}'.format(y))
```

Output:

A screenshot of a Python 3.5.2 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following code and its output:

```
>>> # Python swap program
>>> x = input('Enter value of x: ')
Enter value of x: 45
>>> y = input('Enter value of y: ')
Enter value of y: 36
>>> # create a temporary variable and swap the values
>>> temp = x
>>> x = y
>>> y = temp
>>> print('The value of x after swapping: {}'.format(x))
The value of x after swapping: 36
>>> print('The value of y after swapping: {}'.format(y))
The value of y after swapping: 45
>>> |
```

The status bar at the bottom right shows 'Ln: 84 Col: 4'.

Possible Questions:

2 Mark Questions

1. What is Python?
2. Define Namespace & mention its type.
3. How does a computer run a python program?
4. What is Python list?
5. Define array.
6. List the features of python.
7. What is necessary to execute a Python program?
8. What is a statement in a Python program?

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT III**

---

**6 Mark Questions**

1. What is Python list? Explain the basic list operations with suitable examples.
2. Briefly discuss about the fundamental of Python.
3. Discuss the structure of python program with suitable example program.
4. What is Python list? Explain python elements.
5. Discuss the methods to manipulate the arrays in python
6. Write a Python program to multiply two matrices.

KAHE



# KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc( CS)

(BATCH 2016-2019)

V SEMESTER

PROGRAMMING IN PYTHON (16CSU504 B )

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

## UNIT III

S.NO	QUESTIONS	OPT 1	OPT 2	OPT 3	OPT 4	ANSWER
1	Python is said to be easily	readable language	writable language	bug-able language	script-able language	readable language
2	Extensible programming language that can be extended through classes and programming interfaces is	Python	Perl	PHP	Ada	Python
3	Python was released publicly in	1941	1971	1981	1991	1991
4	What is the output when following code is executed ? print r"\nhello" The output is	a new line and hello	\nhello	the letter r and then hello	Error	\nhello
5	What is the output of the following code ? example = "snow world" example[3] = 's' print example	snow	snow world	Error	snos world	Error
6	Is Python case sensitive when dealing with identifiers?	yes	no	machine dependent	none	yes
7	What is the maximum possible length of an identifier?	31 characters	63 characters	79 characters	none of the mentioned	none of the mentioned
8	Which of the following is not a keyword?	eval	assert	nonlocal	pass	eval



9	All keywords in Python are in	lower case	UPPER CASE	Capitalized	None of the mentioned	None of the mentioned
10	Which of the following is true for variable names in Python?	unlimited length	all private members must have leading and trailing underscores	underscore and ampersand are the only two special characters allowed	none of the mentioned	unlimited length
11	Which of the following is an invalid statement?	abc = 1,000,000	a b c = 1000 2000 3000	a,b,c = 1000, 2000, 3000	a_b_c = 1,000,000	a b c = 1000 2000 3000
12	Which of the following cannot be a variable?	__init__	in	it	on	in
13	What is the output of print 0.1 + 0.2 == 0.3?	TRUE	FALSE	machine dependent	Error	FALSE
14	Which of the following is not a complex number?	k = 2 + 3j	k = complex(2, 3)	k = 2 + 3l	k = 2 + 3J	k = 2 + 3I
15	Which of the following data types is not supported in python?	number	list	string	slice	slice
16	Which of these is not a core data type?	Lists	Dictionary	Tuples	Class	Class
17	What data type is the object below ? L = [1, 23, 'hello', 1]	Lists	Dictionary	Tuples	Array	Lists
18	Which of the following function convert a string to a float in python?	int(x [,base])	long(x [,base] )	float(x)	str(x)	float(x)
19	The sequence \n does what?	Makes a link	Prints a backslash followed by a n	Adds 5 spaces	Starts a new line	Starts a new line
20	Python is a _____ programming language	higher-level	lowe level	mid level	first level	higher-level

21	An _____ translates a source file into machine language as the program executes	complier	interpreter	editor	none	interpreter
22	A _____ translates a source file into an executable file	compiler	interpreter	editor	none	compiler
23	Messages can be printed in the output window by using Python's _____ function.	scan	edit	print	none	print
24	Python is a _____ language	case insensitive	case sensitive	character	none	case sensitive
25	The _____ function enables a Python program to display textual information to the user	scan	edit	print	input	print
26	Programs may use the _____ function to obtain information from the user.	scan	edit	print	input	input
27	Python does not permit _____ to be used when expressing numeric literals	commas	quote	questionmark	arrow	commas
28	The statement a = b copies the value stored in _____	variable a into variable b	variable b into variable a	error	none	variable b into variable a
29	The _____ function accepts an optional prompt string.	scan	edit	print	input	input
30	The _____ function can be used to convert a string representing a numeric expression into its evaluated numeric value.	scan	eval	print	input	eval
31	Python was created by__	James Gosling	Bill Gates	Steve Jobs	Guido van Rossum	Guido van Rossum
32	To start Python from the command prompt, use the command _____.	execute python	run proram	pyhton	go pyhton	pyhton

33	To run python script file named t.py, use the command _____.	execute python t.py	run python t.py	python t.py	go python t.py	python t.py
34	A Python line comment begins with _____.	//	/*	#	@	#
35	A Python paragraph comment uses the style _____.	// comments //	/* comments */	" comments "	/# comments #/	" comments "
36	In Python, a syntax error is detected by the _____ at _____.	compiler/at compile time	interpreter/at runtime	compiler/at runtime	interpreter/at compile time	interpreter/at runtime
37	_____ is the code in natural language mixed with some program code.	Python program	A Python statement	Pseudocode	A flowchart diagram	Pseudocode
38	2 ** 3 evaluates to _____.	9	8	9.1	8.1	8
39	The _____ function immediately terminates the program.	sys.terminate()	sys.halt()	sys.exit()	sys.stop()	sys.exit()
40	The following is NOT an example of a data type.	int	public	double	void	public

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

**UNIT-IV**

**Introduction to Python:** Python Interpreter-Using Python as calculator-Python shell-Indentation. Atoms-Identifiers and keywords-Literals-Strings-Operators(Arithmetic operator, Relational operator, Logical or Boolean operator, Assignment, Operator, Ternary operator, Bit wise operator, Increment or Decrement operator).

**Using the Python Interpreter**

**Interpreter:** To execute a program in a high-level language by translating it one line at a time.

**Compiler:** To translate a program written in a high-level language into a low-level language all at once, in preparation for later execution.

<b>Compiler</b>	<b>Interpreter</b>
Compiler Takes <b>Entire</b> program as input	Interpreter Takes <b>Single</b> instruction as input
Intermediate Object Code is <b>Generated</b>	<b>No</b> Intermediate Object Code is <b>Generated</b>
Conditional Control Statements are Executes <b>faster</b>	Conditional Control Statements are Executes <b>slower</b>
<b>Memory Requirement</b> is <b>More</b> (Since Object Code is Generated)	<b>Memory Requirement</b> is <b>Less</b>
Program need not be <b>compiled</b> every time	Every time higher level program is converted into lower level program
<b>Errors</b> are displayed after <b>entire program</b> is checked	<b>Errors</b> are displayed for <b>every instruction</b> interpreted (if any)
<b>Example</b> : C Compiler	<b>Example</b> : PYTHON

**MODES OF PYTHON INTERPRETER:**

Python Interpreter is a program that reads and executes Python code. It uses 2 modes of Execution.

1. Interactive mode
2. Script mode

**Interactive mode:**

❖ Interactive Mode, as the name suggests, allows us to interact with OS.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

❖ When we type Python statement, interpreter displays the result(s) immediately.

**Advantages:**

- ❖ Python, in interactive mode, is good enough to learn, experiment or explore.
- ❖ Working in interactive mode is convenient for beginners and for testing small pieces of code.

**Drawback:**

- ❖ We cannot save the statements and have to retype all the statements once again to re-run them.

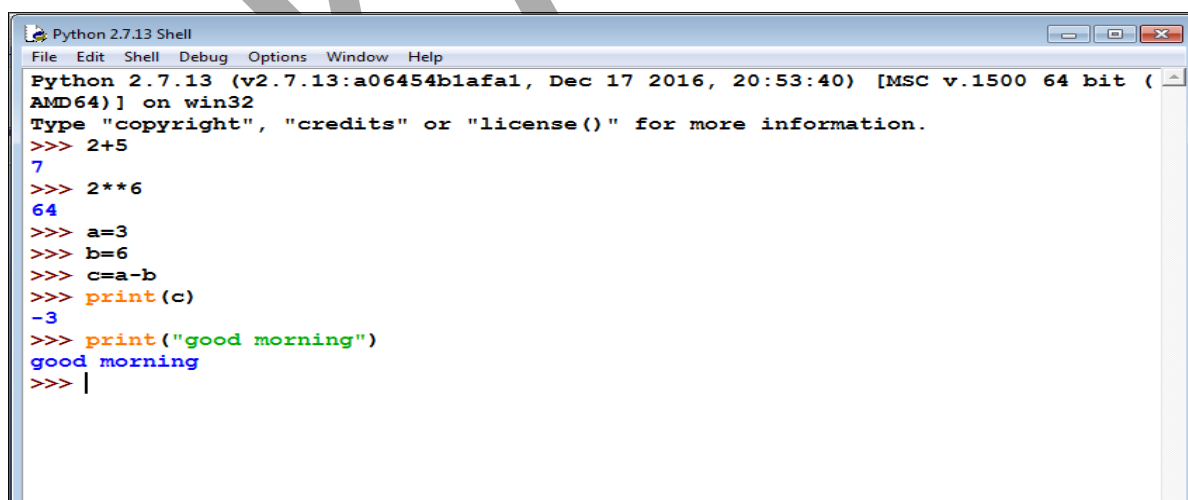
In interactive mode, you type Python programs and the interpreter displays the result:

```
>>> 1 + 1
2
```

The chevron, >>>, is the prompt the interpreter uses to indicate that it is ready for you to enter code. If you type 1 + 1, the interpreter replies 2.

```
>>> print('Hello, World!')
Hello, World!
```

**This is an example of a print statement. It displays a result on the screen. In this case, the result is the words**



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+5
7
>>> 2**6
64
>>> a=3
>>> b=6
>>> c=a-b
>>> print(c)
-3
>>> print("good morning")
good morning
>>> |
```

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT IV

#### Script mode:

- In script mode, we type python program in a file and then use interpreter to execute the content of the file.
- Scripts can be saved to disk for future use. **Python scripts have the extension .py**, meaning that the filename ends with .py
- Save the code with **filename.py** and run the interpreter in script mode to execute the script.

#### Example1:

```
print(1)
x = 2
print(x)
```

#### Output:

```
>>>1
2
```

Interactive mode	Script mode
A way of using the Python interpreter by typing commands and expressions at the prompt.	A way of using the Python interpreter to read and execute statements in a script.
Cant save and edit the code	Can save and edit the code
If we want to experiment with the code, we can use interactive mode.	If we are very clear about the code, we can use script mode.
we cannot save the statements for further use and we have to retype all the statements to re-run them.	we can save the statements for further use and we no need to retype all the statements to re-run them.
We can see the results immediately.	We cant see the code immediately.

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

### **Python shell**

A shell in biology is a calcium carbonate "wall" which protects snails or mussels from its environment or its enemies. Similarly, a shell in operating systems lies between the kernel of the operating system and the user. It's a "protection" in both direction. The user doesn't have to use the complicated basic functions of the OS but is capable of using simple and easier to understand shell commands. The kernel is protected from unintended incorrect usages of system function.

Python offers a comfortable command line interface with the Python shell, which is also known as the "Python interactive shell".

It looks like the term "interactive Shell" is a tautology, because "Shell" is interactive on its own, at least the kind of shells we have described in the previous paragraphs.

### **Using the Python interactive Shell**

With the Python interactive interpreter it is easy to check Python commands. The Python interpreter can be invoked by typing the command "python" without any parameter followed by the "return" key at the shell prompt:

```
python
```

Python comes back with the following information:

```
$ python
```

```
Python 2.7.11+ (default, Apr 17 2016, 14:00:29)
```

```
[GCC 5.3.1 20160413] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT IV**

---

**Python Program to Make a Simple Calculator**

```
''' Program make a simple calculator that can add, subtract, multiply and divide using functions
'''
```

```
# This function adds two numbers
```

```
def add(x, y):
```

```
    return x + y
```

```
# This function subtracts two numbers
```

```
def subtract(x, y):
```

```
    return x - y
```

```
# This function multiplies two numbers
```

```
def multiply(x, y):
```

```
    return x * y
```

```
# This function divides two numbers
```

```
def divide(x, y):
```

```
    return x / y
```

```
print("Select operation.")
```

```
print("1.Add")
```

```
print("2.Subtract")
```

```
print("3.Multiply")
```

```
print("4.Divide")
```



**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT IV**

---

```
# Take input from the user

choice = input("Enter choice(1/2/3/4):")

num1 = int(input("Enter first number: "))

num2 = int(input("Enter second number: "))

if choice == '1':

    print(num1,"+",num2,"=", add(num1,num2))

elif choice == '2':

    print(num1,"-",num2,"=", subtract(num1,num2))

elif choice == '3':

    print(num1,"*",num2,"=", multiply(num1,num2))

elif choice == '4':

    print(num1,"/",num2,"=", divide(num1,num2))

else:

    print("Invalid input")
```

**Output**

Select operation.

1.Add

2.Subtract

3.Multiply

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

4.Divide

Enter choice(1/2/3/4): 3

Enter first number: 15

Enter second number: 14

15 \* 14 = 210

### Indentation

Leading whitespace (spaces and tabs) at the beginning of a logical line is used to compute the indentation level of the line, which in turn is used to determine the grouping of statements.

Most of the programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation.

One typical way to do it in other languages is this:

```
if len(name) > 0 {  
    new_message = 'Yay! Name is not empty'  
    print new_message  
}
```

Python relies on indentation for this. To know which lines or block of code is contained under the if condition.

A code block (body of a [function](#), [loop](#) etc.) starts with indentation and ends with the first unindented line. The amount of indentation is up to you, but it must be consistent throughout that block.

Generally four whitespaces are used for indentation and is preferred over tabs. Here is an example.

```
for i in range(1,11):  
    print(i)  
    if i == 5:
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

break

The enforcement of indentation in Python makes the code look neat and clean. This results into Python programs that look similar and consistent.

Indentation can be ignored in line continuation. But it's a good idea to always indent. It makes the code more readable. For example:

```
if True:
    print('Hello')
    a = 5
```

and

```
if True: print('Hello'); a = 5
```

both are valid and do the same thing. But the former style is clearer.

Incorrect indentation will result into IndentationError.

## Atoms

Atoms are the most basic elements of expressions. The simplest atoms are identifiers or literals. Forms enclosed in reverse quotes or in parentheses, brackets or braces are also categorized syntactically as atoms. The syntax for atoms is:

atom: identifier | literal | enclosure  
enclosure: parenth\_form | list\_display | dict\_display | string\_conversion

## **Identifiers (Names)**

An identifier occurring as an atom is a reference to a local, global or built-in name binding. If a name is assigned to anywhere in a code block (even in unreachable code), and is not mentioned in a global statement in that code block, then it refers to a local name throughout that code block. When it is not assigned to anywhere in the block, or when it is assigned to but also

UNIT IV

explicitly listed in a global statement, it refers to a global name if one exists, else to a built-in name (and this binding may dynamically change).

When the name is bound to an object, evaluation of the atom yields that object. When a name is not bound, an attempt to evaluate it raises a NameError exception.

**VALUES AND DATA TYPES**

**Value:**

**Value** can be any letter ,number or string.

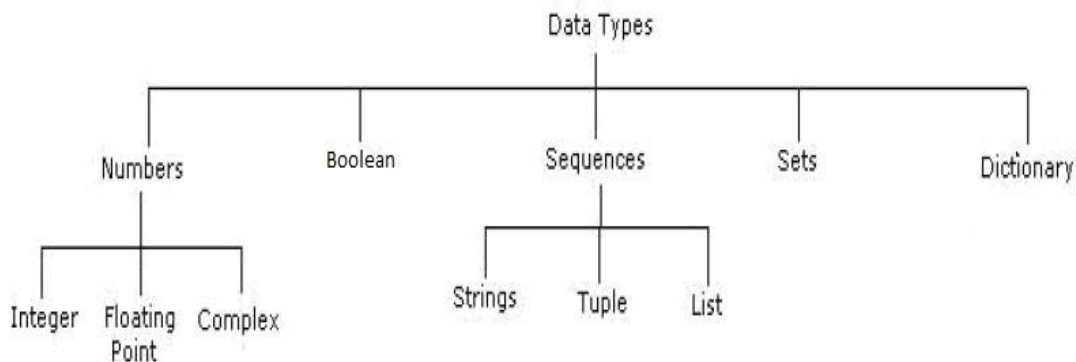
**Eg,** Values are 2, 42.0, and 'Hello, World!'. (These values belong to different datatypes.)

**Data type:**

Every value in Python has a data type.

It is a set of values, and the allowable operations on those values.

**Python has four standard data types:**



**Python Keywords**

Python Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword have a special meaning and a specific operation. These keywords can't be used as variable. Following is the List of Python Keywords.

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

True	False	None	and	as
assert	def	class	continue	break
else	finally	elif	del	except
global	for	if	from	import
raise	try	or	return	pass
nonlocal	in	not	is	lambda

**Identifiers**

Identifiers are the names given to the fundamental building blocks in a program.

These can be variables ,class ,object ,functions , lists , dictionaries etc.

There are certain rules defined for naming i.e., Identifiers.

I. An identifier is a long sequence of characters and numbers.

II.No special character except underscore ( \_ ) can be used as an identifier.

III.Keyword should not be used as an identifier name.

IV.Python is case sensitive. So using case is significant.

V.First character of an identifier can be character, underscore ( \_ ) but not digit.

**Python Literals**

Literals can be defined as a data that is given in a variable or constant.

Python support the following literals:

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

**I. String literals:**

String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes for a String.

Eg: "Aman" , '12345'

**Types of Strings:**

There are two types of Strings supported in Python:

a).**Single line String**- Strings that are terminated within a single line are known as Single line Strings.


Eg: text1='hello'

b).**Multi line String**- A piece of text that is spread along multiple lines is known as Multiple line String.

There are two ways to create Multiline Strings:

**1). Adding black slash at the end of each line.**

Eg:



```
>>> text1='hello\  
user'  
>>> text1  
'hellouser'  
>>>
```

**2).Using triple quotation marks:-**

Eg:

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
>>> str2=""welcome
to
SSSIT""
>>> print str2
welcome
to
SSSIT
>>>
```

**II.Numeric literals:**

Numeric Literals are immutable. Numeric literals can belong to following four different numerical types.

Int(signed integers)	Long(long integers)	float(floating point)	Complex(complex)
Numbers( can be both positive and negative) with no fractional part.eg: 100	Integers of unlimited size followed by lowercase or uppercase L eg: 87032845L	Real numbers with both integer and fractional part eg: -26.2	In the form of a+bj where a forms the real part and b forms the imaginary part of complex number. eg: 3.14j

**III. Boolean literals:**

A Boolean literal can have any of the two values: True or False.

**IV. Special literals.**

Python contains one special literal i.e., None.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

None is used to specify to that field that is not created. It is also used for end of lists in Python.

Eg:

```
>>> val1=10
>>> val2=None
>>> val1
10
>>> val2
>>> print val2
None
>>>
```

**V.Literal Collections.**

Collections such as tuples, lists and Dictionary are used in Python.

List:

- List contain items of different data types. Lists are mutable i.e., modifiable.
- The values stored in List are separated by commas(,) and enclosed within a square brackets([]). We can store different type of data in a List.
- Value stored in a List can be retrieved using the slice operator([] and [:]).
- The plus sign (+) is the list concatenation and asterisk(\*) is the repetition operator.

Eg:

```
>>> list=['aman',678,20.4,'saurav']
>>> list1=[456,'rahul']
>>> list
['aman', 678, 20.4, 'saurav']
>>> list[1:3]
[678, 20.4]
```



CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

```
>>> list+list1
['aman', 678, 20.4, 'saurav', 456, 'rahul']
>>> list1*2
[456, 'rahul', 456, 'rahul']
>>>
```

### **Python Operators**

Operators are particular symbols that are used to perform operations on operands. It returns result that can be used in application.

**Example**                      4 + 5 = 9

Here 4 and 5 are Operands and (+) , (=) signs are the operators. This expression produces the output 9.

### **Types of Operators**

Python supports the following operators

1. Arithmetic Operators.
2. Relational Operators.
3. Assignment Operators.
4. Logical Operators.
5. Membership Operators.
6. Identity Operators.
7. Bitwise Operators.

### **Arithmetic Operators**

The following table contains the arithmetic operators that are used to perform arithmetic operations.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT IV

Operators	Description
//	Perform Floor division(gives integer value after division)
+	To perform addition
-	To perform subtraction
*	To perform multiplication
/	To perform division
%	To return remainder after division(Modulus)
**	Perform exponent(raise to power)

#### Example

```
>>> 10+20
30
>>> 20-10
10
>>> 10*2
20
>>> 10/2
5
>>> 10%3
1
>>> 2**3
8
>>> 10//3
3
>>>
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

**Relational Operators**

The following table contains the relational operators that are used to check relations.

Operators	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to
<>	Not equal to(similar to !=)

**eg:**

```
>>> 10<20
True
>>> 10>20
False
>>> 10<=10
True
>>> 20>=15
True
>>> 5==6
False
>>> 5!=6
True
```

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT IV

```
>>> 10<>2
```

```
True
```

```
>>>
```

#### Assignment Operators

The following table contains the assignment operators that are used to assign values to the variables.

Operators	Description
=	Assignment
/=	Divide and Assign
+=	Add and assign
-=	Subtract and Assign
*=	Multiply and assign
%=	Modulus and assign
**=	Exponent and assign
//=	Floor division and assign

#### Example

```
>>> c=10
```

```
>>> c
```

```
10
```

```
>>> c+=5
```

```
>>> c
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
15
>>> c-=5
>>> c
10
>>> c*=2
>>> c
20
>>> c/=2
>>> c
10
>>> c%=3
>>> c
1
>>> c=5
>>> c**=2
>>> c
25
>>> c//=2
>>> c
12
>>>
```

**Logical Operators**

The following table contains the arithmetic operators that are used to perform arithmetic operations.

Operators	Description
and	Logical AND(When both conditions are true output will be true)
or	Logical OR (If any one condition is true output will be true)

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT IV

not

Logical NOT(Compliment the condition i.e., reverse)

#### Example

```
a=5>4 and 3>2
```

```
print a
```

```
b=5>4 or 3<2
```

```
print b
```

```
c=not(5>4)
```

```
print c
```

#### Output:

```
>>>
```

```
True
```

```
True
```

```
False
```

```
>>>
```

#### Membership Operators

The following table contains the membership operators.

Operators	Description
in	Returns true if a variable is in sequence of another variable, else false.
not in	Returns true if a variable is not in sequence of another variable, else false.

#### Example

```
a=10
```

```
b=20
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
list=[10,20,30,40,50];  
if (a in list):  
    print "a is in given list"  
else:  
    print "a is not in given list"  
if(b not in list):  
    print "b is not given in list"  
else:  
    print "b is given in list"
```

**Output:**

```
>>>  
a is in given list  
b is given in list  
>>>
```

**Identity Operators**

The following table contains the identity operators.

Operators	Description
is	Returns true if identity of two operands are same, else false
is not	Returns true if identity of two operands are not same, else false.

**Example**

```
a=20  
b=20  
if( a is b):  
    print a,b have same identity
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

#### UNIT IV

---

```
else:
    print a, b are different
b=10
if( a is not b):
    print a,b have different identity
else:
    print a,b have same identity
```

#### Output

```
>>>
a,b have same identity
a,b have different identity
>>>
```

#### Python ternary operator

- Python ternary operator was introduced in Python 2.5.
- If used properly, ternary operator can reduce code size and increase readability of the code.
- There is no special keyword for ternary operator, it's the way of writing if-else statement that creates a ternary statement or conditional expression.

#### Python ternary operator syntax

Python ternary operator is written with simple syntax using if else statement.

[when\_true] if [condition] else [when\_false]

Python ternary operator example

On the above syntax, let us quickly build up an **example**:

```
is_fast = True
car = "Ferrari" if is_fast else "Sedan"
```

Clearly, the presented example is a lot more readable than the usual if statement, as follows:



**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
if is_fast:
    car = "Ferrari"
else:
    car="Sedan"
```

Of course, python ternary operator made above code very small.

**Simple Method to use ternary operator:**

```
# Program to demonstrate conditional operator
a, b = 10, 20

# Copy value of a in min if a < b else copy b
min = a if a < b else b
print(min)
```

**Output:**  
10

**Bitwise Operators**

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
Binary OR	It copies a bit if it exists in either operand.	(a   b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT IV

<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	$a \ll 2 = 240$ (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	$a \gg 2 = 15$ (means 0000 1111)

#### Example

```
#!/usr/bin/python
a = 60      # 60 = 0011 1100
b = 13      # 13 = 0000 1101
c = 0
c = a & b;   # 12 = 0000 1100
print "Line 1 - Value of c is ", c

c = a | b;   # 61 = 0011 1101
print "Line 2 - Value of c is ", c

c = a ^ b;   # 49 = 0011 0001
print "Line 3 - Value of c is ", c

c = ~a;      # -61 = 1100 0011
print "Line 4 - Value of c is ", c

c = a << 2;   # 240 = 1111 0000
print "Line 5 - Value of c is ", c

c = a >> 2;   # 15 = 0000 1111
print "Line 6 - Value of c is ", c
```

When you execute the above program it produces the following result –

```
Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
Line 4 - Value of c is -61
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

Line 5 - Value of c is 240

Line 6 - Value of c is 15

### **Increment or Decrement operator**

Increment and Decrement operators ( both pre and post) are not allowed in it.

Python is designed to be consistent and readable. One common error by a novice programmer in languages with ++ and -- operators is mixing up the differences (both in precedence and in return value) between pre and post increment/decrement operators. Simple increment and decrement operators aren't needed as much as in other languages.

**You don't write things like :**

```
for (int i = 0; i < 5; ++i)
```

**In Python, instead we write it like**

# A Sample Python program to show loop (unlike many other languages, it doesn't use ++)

```
for i in range(0, 5):
```

```
    print(i)
```

**Output:**

0

1

2

3

4

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

We can almost always avoid use of ++ and --. For example, x++ can be written as x += 1 and x-- can be written as x -= 1.

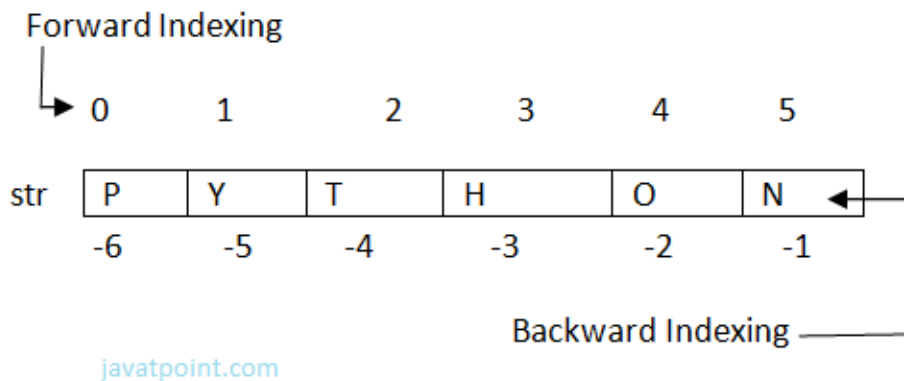
**PYTHON STRINGS**

Python string is a built-in type text sequence. It is used to handle textual data in python. Python Strings are immutable sequences of Unicode points. Creating Strings are simplest and easy to use in Python.

We can simply create Python String by enclosing a text in single as well as double quotes. Python treat both single and double quotes statements same.

**Accessing Python Strings**

- In Python, Strings are stored as individual characters in a contiguous memory location.
- The benefit of using String is that it can be accessed from both the directions (forward and backward).
- Both forward as well as backward indexing are provided using Strings in Python.
  - Forward indexing starts with 0,1,2,3,....
  - Backward indexing starts with -1,-2,-3,-4,....

**Example**

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
str[0]='P'=str[-6] , str[1]='Y' = str[-5] , str[2] = 'T' = str[-4] , str[3] = 'H' = str[-3]  
str[4] = 'O' = str[-2] , str[5] = 'N' = str[-1].
```

**Python String Example**

Here, we are creating a simple program to retrieve String in reverse as well as normal form.

```
name="Rajat"  
length=len(name)  
i=0  
for n in range(-1,(-length-1),-1):  
    print name[i],"\\t",name[n]  
    i+=1
```

**Output:**

```
>>>  
R      t  
a      a  
j      j  
a      a  
t      R  
>>>
```

**Python Strings Operators**

To perform operation on string, Python provides basically 3 types of Operators that are given below.

1. Basic Operators.
2. Membership Operators.
3. Relational Operators.

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

**Python String Basic Operators**

There are two types of basic operators in String "+" and "\*".

**String Concatenation Operator (+)**

The concatenation operator (+) concatenates two Strings and creates a new String.

Python String Concatenation Example

```
>>> "ratan" + "jaiswal"
```

**Output:**

```
'ratanjaiswal'
```

```
>>>
```

Expression	Output
'10' + '20'	'1020'
"s" + "007"	's007'
'abcd123' + 'xyz4'	'abcd123xyz4'

**Eg:**

```
'abc' + 3
```

```
>>>
```

**output:**

```
Traceback (most recent call last):
```

```
File "", line 1, in
```

```
'abc' + 3
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
TypeError: cannot concatenate 'str' and 'int' objects
>>>
```

**Python String Replication Operator (\*)**

Replication operator uses two parameters for operation, One is the integer value and the other one is the String argument.

The Replication operator is used to repeat a string number of times. The string will be repeated the number of times which is given by the integer value.

**Python String Replication Example**

```
>>> 5*"Vimal"
```

**Output:**

```
'VimalVimalVimalVimalVimal'
```

Expression	Output
"soono"*2	'soonosoono'
3*'1'	'111'
'\$'*5	'\$\$\$\$\$'

**Python String Membership Operators**

Membership Operators are already discussed in the Operators section. Let see with context of String.

**There are two types of Membership operators**

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

**1) in:** "in" operator returns true if a character or the entire substring is present in the specified string, otherwise false.

**2) not in:** "not in" operator returns true if a character or entire substring does not exist in the specified string, otherwise false.

Python String membership operator Example

```
>>> str1="javatpoint"
>>> str2='sssit'
>>> str3="seomount"
>>> str4='java'
>>> st5="it"
>>> str6="seo"
>>> str4 in str1
True
>>> str5 in str2
>>> st5 in str2
True
>>> str6 in str3
True
>>> str4 not in str1
False
>>> str1 not in str4
True
```

### Python Relational Operators

All the comparison (relational) operators i.e., (<,>,<=,>=,==,!=,<>) are also applicable for strings. The Strings are compared based on the ASCII value or Unicode(i.e., dictionary Order).

Python Relational Operators Example

```
>>> "RAJAT"=="RAJAT"
```



**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
True
>>> "afsha">='Afsha'
True
>>> "Z"<>"z"
True
```

**Explanation:**

The ASCII value of a is 97, b is 98, c is 99 and so on. The ASCII value of A is 65, B is 66, C is 67 and so on. The comparison between strings are done on the basis on ASCII value.

**Python String Slice Notation**

Python String slice can be defined as a substring which is the part of the string. Therefore further substring can be obtained from a string.

There can be many forms to slice a string, as string can be accessed or indexed from both the direction and hence string can also be sliced from both the directions.

**Python String Slice Syntax**

```
<string_name>[startIndex:endIndex],
<string_name>[:endIndex],
<string_name>[startIndex:]
```

**Python String Slice Example 1**

```
>>> str="Nikhil"
>>> str[0:6]
'Nikhil'
>>> str[0:3]
'Nik'
>>> str[2:5]
'khi'
>>> str[:6]
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

```
'Nikhil'  
>>> str[3:]  
'hil'
```

String slice can also be used with Concatenation operator to get whole string.

**Python String Slice Example 2**

```
>>> str="Mahesh"  
>>> str[:6]+str[6:]  
'Mahesh'
```

//here 6 is the length of the string.

**Python String Functions and Methods**

Python provides various predefined or built-in string functions. They are as follows:

capitalize()	It capitalizes the first character of the String.
count(string,begin,end)	It Counts number of times substring occurs in a String between begin and end index.
endswith(suffix ,begin=0,end=n)	It returns a Boolean value if the string terminates with given suffix between begin and end.
find(substring ,beginIndex, endIndex)	It returns the index value of the string where substring is found between begin index and end index.
index(subsring, beginIndex, endIndex)	It throws an exception if string is not found and works same as find() method.
isalnum()	It returns True if characters in the string are alphanumeric i.e., alphabets or numbers and there is at least 1 character.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT IV

	Otherwise it returns False.
isalpha()	It returns True when all the characters are alphabets and there is at least one character, otherwise False.
isdigit()	It returns True if all the characters are digit and there is at least one character, otherwise False.
islower()	It returns True if the characters of a string are in lower case, otherwise False.
isupper()	It returns False if characters of a string are in Upper case, otherwise False.
isspace()	It returns True if the characters of a string are whitespace, otherwise false.
len(string)	It returns the length of a string.
lower()	It converts all the characters of a string to Lower case.
upper()	It converts all the characters of a string to Upper Case.
startswith(str, begin=0, end=n)	It returns a Boolean value if the string starts with given str between begin and end.
swapcase()	It inverts case of all characters in a string.
lstrip()	It removes all leading whitespace of a string and can also be used to remove particular character from leading.
rstrip()	It removes all trailing whitespace of a string and can also be used to remove particular character from trailing.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

**Python String capitalize() Method Example**

This method capitalizes the first character of the String.

```
>>> 'abc'.capitalize()
```

**Output:** 'Abc'

**Python String count(string) Method Example**

This method counts number of times substring occurs in a String between begin and end index.

```
msg = "welcome to sssit";  
substr1 = "o";  
print msg.count(substr1, 4, 16)  
substr2 = "t";  
print msg.count(substr2)
```

**Output:**

```
>>>  
2  
2  
>>>
```

**Python String endswith(string) Method Example**

This method returns a Boolean value if the string terminates with given suffix between begin and end.

```
string1="Welcome to SSSIT";  
substring1="SSSIT";  
substring2="to";  
substring3="of";  
print string1.endswith(substring1);
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

```
print string1.endswith(substring2,2,16);  
print string1.endswith(substring3,2,19);  
print string1.endswith(substring3);
```

**Output:**

```
>>>  
True  
False  
False  
False  
>>>
```

**Python String find(string) Method Example**

This method returns the index value of the string where substring is found between begin index and end index.

```
str="Welcome to SSSIT";  
substr1="come";  
substr2="to";  
print str.find(substr1);  
print str.find(substr2);  
print str.find(substr1,3,10);  
print str.find(substr2,19);
```

**Output:**

```
>>>  
3  
8  
3  
-1  
>>>
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

**Python String index() Method Example**

This method returns the index value of the string where substring is found between begin index and end index.

```
str="Welcome to world of SSSIT";
substr1="come";
substr2="of";
print str.index(substr1);
print str.index(substr2);
print str.index(substr1,3,10);
print str.index(substr2,19);
```

**Output:**

```
>>>
3
17
3
Traceback (most recent call last):
  File "C:/Python27/fin.py", line 7, in
    print str.index(substr2,19);
ValueError: substring not found
>>>
```

**Python String isalnum() Method Example**

This method returns True if characters in the string are alphanumeric i.e., alphabets or numbers and there is at least 1 character. Otherwise it returns False.

```
str="Welcome to sssit";
print str.isalnum();
str1="Python47";
print str1.isalnum();
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

**Output:**

```
>>>
False
True
>>>
```

**Python String isalpha() Method Example**

It returns True when all the characters are alphabets and there is at least one character, otherwise False.

```
string1="HelloPython"; # Even space is not allowed
print string1.isalpha();
string2="This is Python2.7.4"
print string2.isalpha();
```

**Output:**

```
>>>
True
False
>>>
```

**Python String isdigit() Method Example**

This method returns True if all the characters are digit and there is at least one character, otherwise False.

```
string1="HelloPython";
print string1.isdigit();
string2="98564738"
print string2.isdigit();
```

**Output:**

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT IV

```
>>>
False
True
>>>
```

**Python String islower() Method Example**

This method returns True if the characters of a string are in lower case, otherwise False.

```
string1="Hello Python";
print string1.islower();
string2="welcome to "
print string2.islower();
```

**Output:**

```
>>>
False
True
>>>
```

**Python String isupper() Method Example**

This method returns False if characters of a string are in Upper case, otherwise False.

```
string1="Hello Python";
print string1.isupper();
string2="WELCOME TO"
print string2.isupper();
```

**Output:**

```
>>>
False
True
```



CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

```
>>>
```

### Python String isspace() Method Example

This method returns True if the characters of a string are whitespace, otherwise false.

```
string1=" ";  
print string1.isspace();  
string2="WELCOME TO WORLD OF PYT"  
print string2.isspace();
```

**Output:**

```
>>>  
True  
False  
>>>
```

### Python String len(string) Method Example

This method returns the length of a string.

```
string1=" ";  
print len(string1);  
string2="WELCOME TO SSSIT"  
print len(string2);
```

**Output:**

```
>>>  
4  
16  
>>>
```

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

---

**UNIT IV**

---

### **Python String lower() Method Example**

It converts all the characters of a string to Lower case.

```
string1="Hello Python";  
print string1.lower();  
string2="WELCOME TO SSSIT"  
print string2.lower();
```

**Output:**

```
>>>  
hello python  
welcome to sssit  
>>>
```

### **Python String upper() Method Example**

This method converts all the characters of a string to upper case.

```
string1="Hello Python";  
print string1.upper();  
string2="welcome to SSSIT"  
print string2.upper();
```

**Output:**

```
>>>  
HELLO PYTHON  
WELCOME TO SSSIT  
>>>
```

### **Python String startswith(string) Method Example**

This method returns a Boolean value if the string starts with given str between begin and end.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

#### UNIT IV

---

```
string1="Hello Python";  
print string1.startswith('Hello');  
string2="welcome to SSSIT"  
print string2.startswith('come',3,7);
```

**Output:**

```
>>>  
True  
True  
>>>
```

#### Python String swapcase() Method Example

It inverts case of all characters in a string.

```
string1="Hello Python";  
print string1.swapcase();  
string2="welcome to SSSIT"  
print string2.swapcase();
```

**Output:**

```
>>>  
hELLO pYTHON  
WELCOME TO sssit  
>>>
```

#### Python String lstrip() Method Example

It removes all leading whitespace of a string and can also be used to remove particular character from leading.

```
string1=" Hello Python";  
print string1.lstrip();
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

---

UNIT IV

---

```
string2="@@@@@@welcome to SSSIT"  
print string2.lstrip('@');
```

**Output:**

```
>>>  
Hello Python  
welcome to world to SSSIT  
>>>
```

**Python String rstrip() Method Example**

It removes all trailing whitespace of a string and can also be used to remove particular character from trailing.

```
string1=" Hello Python ";  
print string1.rstrip();  
string2="@welcome to SSSIT!!!"  
print string2.rstrip('!');
```

**Output:**

```
>>>  
Hello Python  
@welcome to SSSIT  
>>>
```

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT IV**

---

**Possible Questions:**

**2 Mark Questions**

1. Define string. How to get a string at run time?
2. List the features of python.
3. What is interpreter?
4. Define operator and operand?
5. Differentiate for loop and while loop.

**6 Mark Questions**

1. Explain the different types of operators in python. Explain any two with example.
2. Define string. How to get a string at run time? Explain with example.
3. Demonstrate the various operators in python with suitable examples
4. How to make a calculator program in python? Explain it.
5. Define String. Create a program to reverse a string without using recursion.
6. Explain Ternary operator and Bit wise operator with suitable program.
7. Define methods in a string with an example program using at least five methods.
8. Illustrate with example. i) Relational operator ii) Logical operator



# KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc( CS)

(BATCH 2016-2019)

V SEMESTER

PROGRAMMING IN PYTHON (16CSU504 B )

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

## UNIT IV

S.NO	QUESTIONS	OPT 1	OPT 2	OPT 3	OPT 4	ANSWER
1	The following is NOT an example of a data type.	int	public	double	void	public
2	Identifiers must contain at least ____character	one	two	three	four	one
3	A _____ word cannot be used as an identifier	variable	reserved	string	token	reserved
4	_____ are identifiers that have predefined meanings in Python	Keywords	string	variable	token	Keywords
5	A _____ operator performs an operation using one operand	binary	unary	trinary	none	unary
6	_____ expression, sometimes called a predicate	bitwise	assignemnt	Boolean	arithmetic	Boolean
7	In Python, a string literal is enclosed in _____.	parentheses	brackets	single-quotes	braces	single-quotes
8	Suppose x is a char variable with a value 'b'. What will be displayed by the statement print(chr(ord(x) + 1))?	a	b	c	d	c
9	What is chr(ord('B')))?	A	B	C	D	B

10	Which of the following statement prints smith\exam1\test.txt?	print("smith\exam1\test.txt")	print("smith\\exam1\\test.txt")	print("smith\"exam1\"test.txt")	print("smith"\exam1"\test.txt")	print("smith\\exam1\\test.txt")
11	The Unicode of 'a' is 97. What is the Unicode for 'c'?	96	97	98	99	99
12	Suppose s = "Welcome", what is type(s)?	int	float	string	str	str
13	The format function returns _____.	an int	a float	a str	a chat	a str
14	Which of the following operators are right-associative.	=	*	+	–	=
15	Assume x = 4 and y = 5, Which of the following is true?	not (x == 4)	x != 4	x == 5	x != 5	x != 5
16	Which operator is overloaded by the or() function?			//	/	
17	What is the output of the following program : i = 0 while i < 3: print i i++ print i+1	0 2 1 3 2 4	0 1 2 3 4 5	Error	1 0 2 4 3 5	Error
18	Which function overloads the >> operator?	more()	gt()	ge()	None of the above	None of the above
19	_____ creates a list.	list1 = list()	list1 = []	list1 = list([12, 4, 4])	list1 = [12, 4, 4]	All
20	Which of the following statements is used to create an empty set?	{ }	set()	[ ]	( )	set()

21	What is the output of the following piece of code when executed in the python shell? a={1,2,3} a.intersection_update({2,3,4,5}) a	{2,3}	Error, duplicate item present in list	Error, no method called intersection_update for set data type	{1,4,5}	{2,3}
22	Which of the following lines of code will result in an error?	s={abs}	s={4, 'abc', (1,2)}	s={2, 2.2, 3, 'xyz'}	s={san}	s={san}
23	What is the output of the line of code shown below, if s1= {1, 2, 3}? s1.issubset(s1)	TRUE	Error	No output	FALSE	TRUE
24	What is the output of the code shown below? s=set([1, 2, 3]) s.union([4, 5]) s ([4, 5])	{1, 2, 3, 4, 5}{1, 2, 3, 4, 5}	Error{1, 2, 3, 4, 5}	{1, 2, 3, 4, 5}Error	ErrorError	{1, 2, 3, 4, 5}Error
25	Which of the following function capitalizes first letter of string?	shuffle(lst)	capitalize()	isalnum()	isdigit()	capitalize()
26	Which of the following function checks in a string that all characters are digits?	shuffle(lst)	capitalize()	isalnum()	isdigit()	isdigit()
27	Which of the following function convert an integer to octal string in python?	unichr(x)	ord(x)	hex()	oct(x)	oct(x)
28	What is the name of data type for character in python ?	char	python do not have any data type for characters	charcter	chr	python do not have any data type for characters
29	In python 3 what does // operator do ?	Float division	Integer division	returns remainder	same as a**b	Integer division



30	What is "Programming is fun"[4: 6]?	ram	ra	r	pr	ra
31	What is "Programming is fun"[-1]?	pr	ram	ra	n	n
32	What is "Programming is fun"[1:1]?	pr	p	r	' '	' '
33	Given a string s = "Welcome", which of the following code is incorrect?	print(s[0])	print(s.lower())	s[1] = 'r'	print(s.strip())	s[1] = 'r'
34	Given a string s = "Programming is fun", what is s.find('ram')?	1	2	3	4	4
35	Given a string s = "Programming is fun", what is s.startswith('Program')?	0	1	TRUE	FALSE	TRUE
36	A ____ is an associative array of key-value pairs	dictionary	list	tuple	sequence	dictionary
37	A _____ is though similar to a list, but it's immutable.	dictionary	list	tuple	sequence	tuple
38	A ____, in Python, stores a sequence of objects in a defined order.	dictionary	list	tuple	sequence	list
39	What Will Be The Output Of The Following Code Snippet? a=[1,2,3,4,5,6,7,8,9] print(a[::-2])	[1,2]	[8,9]	[1,3,5,7,9]	[1,2,3]	[1,3,5,7,9]
40	What is the output of the expression? round(4.5676,2)?	4.5	4.6	4.57	4.56	4.57



CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

UNIT-V

**Creating Python Programs:** Input and Output Statements-Control statements(Branching, Looping, Conditional Statement, Exit function, Difference between break, continue and pass.). Defining Functions-Default arguments.

**Input and Output Statements**

Python provides numerous [built-in functions](#) that are readily available to us at the Python prompt. Some of the functions like input() and print() are widely used for standard input and output operations respectively

**OUTPUT**

**Python Output Using print() function**

**OUTPUT:** Output can be displayed to the user using Print statement . We use the print() function to output data to the standard output device (screen).

**Syntax:**

print (expression/constant/variable)

**Example : 1**

```
>>> print ("Hello")  
Hello
```

**Example : 2**

```
print('This sentence is output to the screen')
```

# Output: This sentence is output to the screen

**Example : 3**

```
a = 5  
print('The value of a is', a)
```

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**

---

# Output: The value of a is 5

**Output formatting**

Sometimes we would like to format our output to make it look attractive. This can be done by using the `str.format()` method. This method is visible to any string object.

```
>>> x = 5; y = 10
>>> print('The value of x is {} and y is {}'.format(x,y))
The value of x is 5 and y is 10
```

Here the curly braces {} are used as placeholders.

```
print('I love {} and {}'.format('bread','butter'))
# Output: I love bread and butter
```

```
print('I love {} and {}'.format('bread','butter'))
# Output: I love butter and bread
```

**INPUT**

**INPUT:** Input is data entered by user (end user) in the program. In python, **input () function** is available for input. To allow flexibility we might want to take the input from the user. In Python, we have the `input()` function to allow this. The syntax for `input()` is

**Syntax for input() is:**

variable = input ("data")      OR      input([prompt])

where `prompt` is the string we wish to display on the screen. It is optional.

```
>>> num = input('Enter a number: ')
Enter a number: 10
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

```
>>> num  
'10'
```

Here, we can see that the entered value 10 is a string, not a number. To convert this into a number we can use `int()` or `float()` functions.

```
>>> int('10')  
10  
>>> float('10')  
10.0
```

### Raw Input and Input

There are two functions in Python that you can use to read data from the user:

`raw_input` and `input`

You can store the results from them into a variable.

### Raw Input

`raw_input` is used to read text (strings) from the user:

```
name = raw_input("What is your name? ")
```

```
type(name)
```

```
>>> output
```

```
What is your name? spilcm
```

```
type 'str'>
```

## KARPAGAM ACADEMY OF HIGHER EDUCATION

---

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

### UNIT V

---

#### Input

```
input is used to read integers
age = input("What is your age? ")
print "Your age is: ", age
type(age)
```

```
>>>output
What is your age? 100
Your age is: 100
type 'int'>
```

#### **Example:**

```
>>> x=input("enter the name:")
enter the name: george

>>>y=int(input("enter the number"))
enter the number 3
```

#python accepts string as default data type. conversion is required for type.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

## Control statements

### Branching

#### Python If Statements

The Python if statement is a statement which is used to test specified condition. We can use if statement to perform conditional operations in our Python application. The if statement executes only when specified condition is **true**. We can pass any valid expression into the if parentheses.

There are various types of if statements in Python.

- if statement
- if-else statement
- nested if statement

#### Python If Statement Syntax

```
if(condition):  
    statements
```

#### Python If Statement Example

```
a=10  
if a==10:  
    print "Welcome to javatpoint"
```

**Output:** Hello User

#### Python If Else Statements

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

The If statement is used to test specified condition and if the condition is true, if block executes, otherwise else block executes. The else statement executes when the if statement is false.

### Python If Else Syntax

```
if(condition): False
    statements
else: True
    statements
```

#### Example-

```
year=2000
if year%4==0:
    print "Year is Leap"
else:
    print "Year is not Leap"
```

Output: Year is Leap

### Python Nested If Else Statement

In python, we can use nested If Else to check multiple conditions. Python provides **elif** keyword to make nested If statement. This statement is like executing a if statement inside a else statement.

### Python Nested If Else Syntax

If statement:



CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

Body

**elif** statement:

Body

**else:**

Body

### Python Nested If Else Example

```
a=10
if a>=20:
    print "Condition is True"
else:
    if a>=15:
        print "Checking second value"
    else:
        print "All Conditions are false"
```

**Output:** All Conditions are false.

### EXAMPLE:

Python Program to get a number num and check whether num is odd or even?

```
num1=int(input("Enter your number:"))
if(num1%2==0):
    print("{} is even".format(num1))
else:
    print("{} is odd".format(num1))
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

## Looping

### For Loop

Python **for loop** is used to iterate the elements of a collection in the order that they appear. This collection can be a sequence(list or string).

#### Python For Loop Syntax

**for** <variable> **in** <sequence>:

#### Python For Loop Simple Example

```
num=2
for a in range (1,6):
    print num * a
```

Output:

```
2
4
6
8
10
```

#### Python Example to Find Sum of 10 Numbers

```
sum=0
for n in range(1,11):
    sum+=n
print sum
```

Output: 55

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**

---

**Python Nested For Loops**

Loops defined within another Loop are called Nested Loops. Nested loops are used to iterate matrix elements or to perform complex computation. When an outer loop contains an inner loop in its body it is called Nested Looping.

**Python Nested For Loop Syntax**

```
for <expression>:  
    for <expression>:  
        Body
```

**Python Nested For Loop Example**

```
for i in range(1,6):  
    for j in range (1,i+1):  
        print i,  
        print
```

**Output:**

```
>>>  
1  
2 2  
3 3 3  
4 4 4 4  
5 5 5 5 5  
>>>
```

**Explanation:**

For each value of Outer loop the whole inner loop is executed.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

For each value of inner loop the Body is executed each time.

### Python Nested Loop Example 2

```
for i in range (1,6):  
    for j in range (5,i-1,-1):  
        print "*",  
    print
```

Output:

```
>>>  
* * * * *  
* * * *  
* * *  
* *  
*  
*
```

### Python While Loop

In Python, while loop is used to execute number of statements or body till the specified condition is true. Once the condition is false, the control will come out of the loop.

### Python While Loop Syntax

```
while <expression>:  
    Body
```

Here, loop Body will execute till the expression passed is true. The Body may be a single statement or multiple statement.

### Python While Loop Example 1

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

```
a=10
```

```
while a>0:
```

```
    print "Value of a is",a
```

```
    a=a-2
```

```
print "Loop is Completed"
```

Output:

```
>>>
```

```
Value of a is 10
```

```
Value of a is 8
```

```
Value of a is 6
```

```
Value of a is 4
```

```
Value of a is 2
```

```
Loop is Completed
```

```
>>>
```

### Python While Loop Example 2

```
n=153
```

```
sum=0
```

```
while n>0:
```

```
    r=n%10
```

```
    sum+=r
```

```
    n=n/10
```

```
print sum
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

Output:

```
>>>
9
>>>
```

### Python Break

Break statement is a jump statement which is used to transfer execution control. It breaks the current execution and in case of inner loop, inner loop terminates immediately. When break statement is applied the control points to the line following the body of the loop, hence applying break statement makes the loop to terminate and controls goes to next line pointing after loop body.

#### Python Break Example 1

```
for i in [1,2,3,4,5]:
    if i==4:
        print "Element found"
        break
    print i,
```

Output:

```
>>>
1 2 3 Element found
>>>
```

#### Python Break Example 2

```
for letter in 'Python3':
    if letter == 'o':
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

```
break
print (letter)
```

Output:

```
P
y
t
h
```

### Python Break Example 3

```
number = 0
for number in range(10):
    number = number + 1

    if number == 5:
        break # break here

print('Number is ' + str(number))

print('Out of loop')
```

Output:

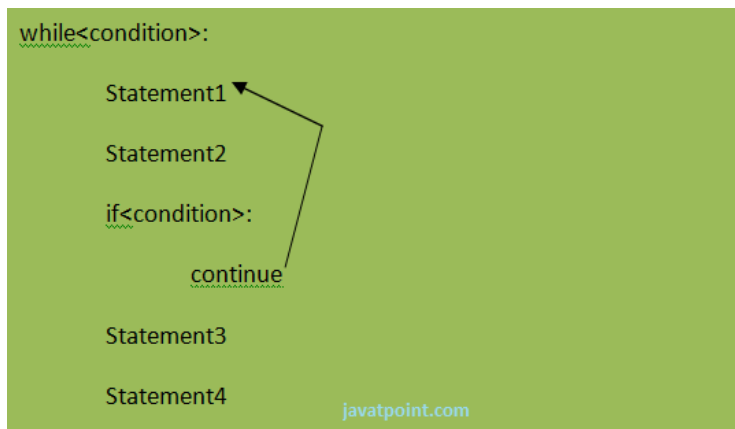
```
Number is 1
Number is 2
Number is 3
Number is 4
Out of loop
```

### Python Continue Statement

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**

---

Python Continue Statement is a jump statement which is used to skip execution of current iteration. After skipping, loop continue with next iteration. We can use continue statement with for as well as while loop in Python.

**Python Continue Statement Example1**

```
a=0  
while a<=5:  
    a=a+1  
    if a%2==0:  
        continue  
    print a  
print "End of Loop"
```

**Output:**

```
>>>  
1  
3
```



CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

```
5
End of Loop
>>>
```

### Python Continue Statement Example2

```
number = 0

for number in range(10):
    number = number + 1

    if number == 5:
        continue # continue here

    print('Number is ' + str(number))

print('Out of loop')
```

### Output:

```
Number is 1
Number is 2
Number is 3
Number is 4
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
Out of loop
```

### Python Pass

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**

---

In Python, pass keyword is used to execute nothing; it means, when we don't want to execute code, the pass can be used to execute empty. It is same as the name refers to. It just makes the control to pass by without executing any code. If we want to bypass any code pass statement can be used.

**Python Pass Syntax**

**pass**

**Python Pass Example1**

```
for i in [1,2,3,4,5]:  
    if i==3:  
        pass  
    print "Pass when value is",i  
print i,
```

**Output:**

```
>>>  
1 2 Pass when value is 3  
3 4 5  
>>>
```

**Python Pass Example2**

number = 0

```
for number in range(10):  
    number = number + 1
```

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

```
if number == 5:  
    pass # pass here  
  
print('Number is ' + str(number))  
  
print('Out of loop')
```

The pass statement occurring after the if conditional statement is telling the program to continue to run the loop and ignore the fact that the variable number evaluates as equivalent to 5 during one of its iterations.

**Output:**

```
Number is 1  
Number is 2  
Number is 3  
Number is 4  
Number is 5  
Number is 6  
Number is 7  
Number is 8  
Number is 9  
Number is 10  
Out of loop
```

**Python Functions**

A Function is a self block of code which is used to organize the functional code. Function can be called as a section of a program that is written once and can be executed whenever required in the program, thus making code reusability. Function is a subprogram that works on data and produces some output.

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

### Types of Functions:

There are two types of Functions.

- a) Built-in Functions: Functions that are predefined and organized into a library. We have used many predefined functions in Python.
- b) User- Defined: Functions that are created by the programmer to meet the requirements.

### Defining a Function

A Function defined in Python should follow the following format:

- 1) Keyword **def** is used to start and declare a function. Def specifies the starting of function block.
- 2) def is followed by function-name followed by parenthesis.
- 3) Parameters are passed inside the parenthesis. At the end a colon is marked.

### Python Function Syntax

```
def <function_name>(parameters):  
</function_name>
```

Example **def** sum(a,b):

- 4) Python code requires indentation (space) of code to keep it associate to the declared block.
- 5) The first statement of the function is optional. It is ?Documentation string? of function.
- 6) Following is the statement to be executed.

### Syntax:

CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

```
def <function_name>([parameters]):  
    "function docstring"  
    Statement1  
    Statement2  
    ...  
    ....
```

### Invoking a Python Function

To execute a function it needs to be called. This is called function calling.

Function Definition provides the information about function name, parameters and the definition what operation is to be performed. In order to execute the function definition, we need to call the function.

### Python Function Syntax

```
<function_name>(parameters)  
</function_name>
```

### Python Function Example

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**

---

sum(a,b)

Here, sum is the function and a, b are the parameters passed to the function definition.

Let's have a look over an example.

**Python Function Example 2**

```
#Providing Function Definition
def sum(x,y):
    "Going to add x and y"
    s=x+y
    print "Sum of two numbers is"
    print s
#Calling the sum Function
sum(10,20)
sum(20,30)
```

**Output:**

```
>>>
Sum of two numbers is
30
Sum of two numbers is
50
>>>
```

**Python Function return Statement**

return[expression] is used to return response to the caller function. We can use expression with the return keyword. send back the control to the caller with the expression. In case no

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**

---

expression is given after return it will return None. In other words return statement is used to exit the function definition.

**Python Function return Example**

```
def sum(a,b):  
    "Adding the two values"  
    print "Printing within Function"  
    print a+b  
    return a+b  
def msg():  
    print "Hello"  
    return
```

```
total=sum(10,20)  
print "Printing Outside: ",total  
msg()  
print "Rest of code"
```

**Output:**

```
>>>  
Printing within Function  
30  
Printing outside: 30  
Hello  
Rest of code  
>>>
```

**Python Function Argument and Parameter**

There can be two types of data passed in the function.

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**

---

1) The First type of data is the data passed in the function call. This data is called ?arguments?.

2) The second type of data is the data received in the function definition. This data is called ?parameters?.

Arguments can be literals, variables and expressions. Parameters must be variable to hold incoming values. Alternatively, arguments can be called as actual parameters or actual arguments and parameters can be called as formal parameters or formal arguments.

**Python Function Example**

```
def addition(x,y):  
    print x+y  
x=15  
addition(x ,10)  
addition(x,x)  
y=20  
addition(x,y)
```

**Output:**

```
>>>  
25  
30  
35  
>>>
```

**Passing Parameters**

Apart from matching the parameters, there are other ways of matching the parameters.

Python supports following types of formal argument:

1) Positional argument (Required argument).



CLASS : III B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : PROGRAMMING IN PYTHON

COURSE CODE : 16CSU504B

UNIT V

---

2) Default argument.

3) Keyword argument (Named argument)

**Positional/Required Arguments:**

When the function call statement must match the number and order of arguments as defined in the function definition. It is Positional Argument matching.

**Python Function Positional Argument Example**

```
#Function definition of sum
def sum(a,b):
    "Function having two parameters"
    c=a+b
    print c
```

```
sum(10,20)
sum(20)
```

**Output:**

```
>>>
30
```

Traceback (most recent call last):

File "C:/Python27/su.py", line 8, in <module>

sum(20)

TypeError: sum() takes exactly 2 arguments (1 given)

```
>>>
```

```
</module>
```

**Explanation:**

**CLASS : III B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : PROGRAMMING IN PYTHON****COURSE CODE : 16CSU504B****UNIT V**


---

1) In the first case, when sum() function is called passing two values i.e., 10 and 20 it matches with function definition parameter and hence 10 and 20 is assigned to a and b respectively. The sum is calculated and printed.


2) In the second case, when sum() function is called passing a single value i.e., 20 , it is passed to function definition. Function definition accepts two parameters whereas only one value is being passed, hence it will show an error.

**Python Function Default Arguments**

Default Argument is the argument which provides the default values to the parameters passed in the function definition, in case value is not provided in the function call default value is used.

**Python Function Default Argument Example**

```
#Function Definition
def msg(Id,Name,Age=21):
    "Printing the passed value"
    print Id
    print Name
    print Age
    return
#Function call
msg(Id=100,Name='Ravi',Age=20)
msg(Id=101,Name='Ratan')
```

**Output:**

```
>>>
100
Ravi
20
101
```

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT V**

---

```
Ratan
21
>>>
```

**Explanation:**

1) In first case, when msg() function is called passing three different values i.e., 100 , Ravi and 20, these values will be assigned to respective parameters and thus respective values will be printed.

2) In second case, when msg() function is called passing two values i.e., 101 and Ratan, these values will be assigned to Id and Name respectively. No value is assigned for third argument via function call and hence it will retain its default value i.e, 21.

**Possible Questions:**

**2 Mark Questions**

1. What is interpreter?
2. Select and assign how an input operation was done in python.
3. Differentiate for loop and while loop.
4. Differentiate break and continue
5. Mention the use of pass.
6. Write the syntax for while loop with flowchart.

**6 Mark Questions**

1. Discuss the need and importance of function in python.
2. Show how an input and output function is performed in python with an example

**CLASS : III B.Sc CS**

**BATCH : 2016 - 2019**

**COURSE NAME : PROGRAMMING IN PYTHON**

**COURSE CODE : 16CSU504B**

**UNIT V**

---

3. Formulate with an example program to pass the list arguments to a function
4. Explain conditional statements in detail with example
5. What are the different loop control statements available in python? Explain with suitable examples.
6. Summarize the difference between break, continue and pass.
7. Briefly discuss about the types of decision making statement
8. Write a Python program using function to check given number is odd or even.

KAHE



# KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc( CS)

(BATCH 2016-2019)

V SEMESTER

PROGRAMMING IN PYTHON (16CSU504 B )

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

## UNIT V

S.NO	QUESTIONS	OPT 1	OPT 2	OPT 3	OPT 4	ANSWER
1	A compound statement is:	A collection of one or more statements enclosed in braces	A statement involving if and else	A way of declaring variables	a way of setting the value of a variable	A collection of one or more statements enclosed in braces
2	What is an infinite loop?	A loop that functions infinitely well	A loop that runs forever	A loop that never starts	A loop that will never function	A loop that runs forever
3	A function is:	An entity that receives inputs and outputs	A way of storing values	A sequence of characters enclosed by quotes	A kind of computer	An entity that receives inputs and outputs
4	What is the value of k after the following code fragment? int k = 0; int n = 12 while (k < n) { k = k + 1; }	0	11	12	13	12

5	What is the output of the following? i = 2 while True: if i%3 == 0: break print(i)	2 4 6 8 10 ...	2 4	2 3	error	2 4
6	_____ is interpreted.	python	c++	ada	c	python
7	The _____ function immediately terminates the program.	sys.terminate()	sys.halt()	sys.exit()	sys.stop()	sys.exit()
8	What will be displayed by the following code? ch = 'F' if ch >= 'A' and ch <= 'Z': print(ch)	F	f	Ff	none	F
9	If a function does not return a value, by default, it returns _____.	int	float	double	none	none
10	The header of a function consists of _____.	function name	function name and parameter list	parameter list	function argument	function name and parameter list
11	A function _____.	must have at least one parameter	may have no parameters	must always have a return statement to return a value	must always have a return statement to return multiple values	may have no parameters
12	Arguments to functions always appear within _____.	brackets	parentheses	curly braces	quotation marks	parentheses
13	A function with no return statement returns _____.	void	nothing	0	none	none

14	A variable defined inside a function is referred to as _____.	a global variable	a function variable	a block variable	a local variable	a local variable
15	Whenever possible, you should avoid using _____.	global variables	function parameters	global constants	local variables	global variables
16	_____ is to implement one function in the structure chart at a time from the top to the bottom.	Bottom-up approach	Top-down approach	Bottom-up and top-down approach	Stepwise refinement	Top-down approach
17	_____ is a simple but incomplete version of a function.	A stub	A function	A function developed using bottom-up approach	A function developed using top-down approach	A stub
18	The keyword _____ is required to define a class.	def	return	class	all	class
19	_____ terminates the process normally.	abort()	exit()	assert()	all	exit()
20	The _____ statement terminates the loop containing it.	break	continue	pass	stop	break
21	The _____ statement is used to skip the rest of the code inside a loop for the current iteration only	break	continue	pass	stop	continue
22	Which of the following keyword is a valid placeholder for body of the function ?	break	continue	pass	body	pass
23	Let a = [ 1,2,3,4,5 ] then which of the following is correct ?	print(a[:]) => [1,2,3,4]	print(a[0:]) => [2,3,4,5]	print(a[:100]) => [1,2,3,4,5]	print(a[-1:]) => [1,2]	print(a[:100]) => [1,2,3,4,5]
24	In python which is the correct method to load a module ?	include math	import math	#include<math.h>	using math	import math

25	What is the need of if __name__ == "__main__": somemethod()	Create new module	Define generators	Run python module as main program	Create new objects	Run python module as main program
26	In python which keyword is used to start function ?	function	def	try	import	def
27	What is the number of iterations in the following loop:  for i in range(1, n): # iteration	2*n	n	n-1	n+1	n-1
28	Whihc function do you use to write data to perform binary output?	write	output	dump	send	dump
29	Whihc function do you use to read data using binary input?	output	read	input	load	load
30	Invoking the _____ method converts raw byte data to a string.	encode()	decode()	convert()	toString()	decode()
31	The readlines() method returns a _____.	str	a list of lines	a list of single characters	a list of integers	a list of lines
32	____ separates the header of the compound statement from the body.	colon (:)	semicolon(;) )	Comma(,)	Singlequot(')	colon (:) )
33	What is the output of the following? x = ['ab', 'cd'] for i in x: i.upper() print(x)	['ab', 'cd'].	['AB', 'CD'].	[None, None].	none of the mentioned	['ab', 'cd'].



34	What is the output of the following?  i = 1 while True: if i%2 == 0: break print(i) i += 2	1	1 2	1 2 3 4 5 6 ...	1 3 5 7 9 11 ...	1 3 5 7 9 11 ...
35	_____ asks the user for a string of data and simply returns the string.	input()	raw_input()	read()	write()	raw_input()
36	Command-line arguments passed to a Python program are stored in _____ list.	sys.argv	sys.argS	sys.argc	sys.argw	sys.argv
37	Python's ____function takes a single parameter that is a string.	input	output	read	load	input
38	The _____ operator is a string operator called the format operator	&	*	%	#	%
39	_____ function you can iterate through the sequence and retrieve the index position and its corresponding value at the same time.	enumerate()	enum()	e()	eindex()	enumerate()
40	_____ finds all the occurrences of match and return them as an iterator.	find()	finditer()	replace()	check()	finditer()

Reg .No.....  
[16CSU504 B]

# KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University )

(Established Under Section 3 of UGC Act, 1956)

**COIMBATORE – 641 021**

(For the candidates admitted from 2016 onwards)

## B.Sc DEGREE EXAMINATION

Fifth Semester

First Internal Examination – July 2018

### COMPUTER SCIENCE

### PROGRAMMING IN PYTHON

**Date** : 14 –07 – 18 ( AN )

**Class** : III.B.Sc CS (A & B)

**Time** : 2:00 Hrs

**Maximum** : 50

---

#### SECTION A – (20\*1 = 20 Marks)

**Answer All Questions**

1. Second step in problem solving process is to
  - a) practicing the solution
  - b) organizing the data
  - c) design a solution
  - d) define a problem
2. Thing to keep in mind while solving a problem is
  - a) input data
  - b) output data
  - c) stored data
  - d) all of above
3. First step in process of problem solving is to
  - a) design a solution
  - b) define a problem
  - c) practicing the solution
  - d) organizing the data
4. Stock maintaining control system in which both hardware and software is present to contact and manage suppliers is called
  - a) power system
  - b) control system
  - c) information processing system
  - d) operating system
5. All components that work together in the form of unit is called
  - a) system
  - b) bound
  - c) baud
  - d) operator

6. The main task of a problem-solving agent is
  - a) Solve the given problem and reach to goal
  - b) To find out which sequence of action will get it to the goal state
  - c) Both a and b
  - d) Neither a nor b
7. Error in a program is called
  - a) bug
  - b) debug
  - c) virus
  - d) noise
8. Error which occurs when program tried to read from file without opening it is classified as
  - a) execution error messages
  - b) built in messages
  - c) user-defined messages
  - d) half messages
9. Specialized program that allows the user to utilize in specific application is classified as
  - a) relative programs
  - b) application programs
  - c) relative programs
  - d) replicate programs
10. Program which is used to produce pictures, text and to organize it in newspaper is classified as
  - a) text publishing package
  - b) desktop publishing package
  - c) experimental package
  - d) organizing publishing package
11. Set of programs which consist of full set of documentations is termed as
  - a) database packages
  - b) file packages
  - c) bus packages
  - d) software packages
12. To write a program function i.e. program for the sum of four integers, the program refinement first level includes
  - a) input four numbers
  - b) calculate sum
  - c) print the values
  - d) display the values
13. Data which is used to test each feature of the program and is carefully selected is classified as
  - a) program output
  - b) program input
  - c) test data
  - d) test program
14. Function definition and first level refinement are part of
  - a) program design
  - b) program statement
  - c) program calculation
  - d) printing the program
15. In flow chart, diamond shaped symbol is used to represent
  - a) decision box
  - b) statement box
  - c) error box
  - d) if-statement box
16. Symbol used in flowchart such as rectangle with the horizontal lines on two sides is used for
  - a) defined statement
  - b) predefined process
  - c) error fix
  - d) variables defined
17. Program link with other parts of the program or connectors in flowchart are represented by
  - a) rhombus
  - b) parallelogram
  - c) circle
  - d) trapezoid

18. In interactivity chart the darkened circle indicates \_\_\_\_\_.  
a) duplicate module                      b) loop                      c) decision                      d) no special meaning
19. The IPO stands for  
a) Input Programming Option                      b) Input Programming Output  
c) Input Processing Output                      d) Input Operating Operation
20. For many problems such as sorting, there are many choices of algorithms to use, some of which are extremely \_\_\_\_\_.  
a) Time efficient                      b) Space efficient                      c) Both (a) and (b)                      d) None of the above

**SECTION B – (3\*2 = 6 Marks)**

**Answer All Questions**

21. How to define problem in computer programming?
22. Write the algorithm of Factorial of given numbers.
23. What is debugging?

**SECTION C – (3\*8 = 24 Marks)**

**Answer the Questions**

24. a). What are the types of errors in programming? Explain it.  
[OR]  
b). Illustrate the Concept of problem solving in computer programming.
25. a) Explain decision table with example  
[OR]  
b) What is the purpose of flow charts in problem solving? Describe symbols used in flow chart.
26. a) Write the algorithm and flow chart to find greatest among three numbers.  
[OR]  
b). Elaborate Documentation in computer program.