PYTHON PROGRAMMING

NOTE:

MAKAUT course structure and syllabus of 4th semester has been changed from 2022. **PYTHON PROGRAMMING** has been introduced as a new subject in the present curriculum. Taking special care of this matter we are providing chapterwise model questions & answers, so that students can get an idea about university questions patterns.

INTRODUCTION TO PYTHON

Multiple Choice Type Questions

1. In which of the following languages, the instructions are written in the form of 0s and 1s? [MODEL QUESTION]
   a) Assembly language b) Programming language
   c) High-level language d) Machine language

Answer: (d)

2. Who developed Python Programming Language? [MODEL QUESTION]
   a) Wick van Rossum b) Rasmus Lerdorf
   c) Guido van Rossum d) Niene Stom

Answer: (c)

3. Which type of Programming does Python support? [MODEL QUESTION]
   a) object-oriented programming b) structured programming
   c) functional programming d) all of the mentioned

Answer: (d)

4. Which of the following is the correct extension of the Python file? [MODEL QUESTION]
   a) .python b) .pl c) .py d) .p

Answer: (c)

5. Is Python code compiled or interpreted? [MODEL QUESTION]
   a) Python code is both compiled and interpreted
   b) Python code is neither compiled nor interpreted
   c) Python code is only compiled
   d) Python code is only interpreted

Answer: (b)

6. All keywords in Python are in _____ [MODEL QUESTION]
   a) Capitalized b) lower case
   c) UPPER CASE d) None of these

Answer: (d)

7. What will be the value of the following Python expression? [MODEL QUESTION]
4 + 3 % 5
   a) 7 b) 2 c) 4 d) 1

Answer: (a)

8. Which of the following is used to define a block of code in Python language? [MODEL QUESTION]
   a) Indentation b) Key c) Brackets d) All of these

Answer: (a)

9. Which of the following character is used to give single-line comments in Python? [MODEL QUESTION]
   a) // b) # c) ! d) /*

Answer: (b)

10. What is the order of precedence in python? [MODEL QUESTION]
   a) Exponential, Parentheses, Multiplication, Division, Addition, Subtraction.
   b) Exponential, Parentheses, Division, Multiplication, Addition, Subtraction
   c) Parentheses, Exponential, Multiplication, Division, Subtraction, Addition
   d) Parentheses, Exponential, Multiplication, Division, Addition, Subtraction

Answer: (d)

11. Which of the following functions can help us to find the version of python that we are currently working on? [MODEL QUESTION]
   a) sys.version(1) b) sys.version (0)
   c) sys.version() d) sys.version

Answer: (a)

12. What will be the output of the following Python code? [MODEL QUESTION]

```
i = 1
while True:
    if i%3 == 0:
        break
    print (i)
    i + = 1
```

   a) 1 2 3 b) error c) 1 2 d) none of these

Answer: (b)

13. Which of the following is true for variable names in Python? [MODEL QUESTION]
   a) underscore and ampersand are the only two special characters allowed
   b) unlimited length
   c) all private members must have leading and trailing underscores
   d) none of the mentioned

Answer: (b)

14. Which of the following is the truncation division operator in Python? [MODEL QUESTION]
   a) | b) // c) / d) %

Answer: (b)

15. What will be the output of the following Python code? [MODEL QUESTION]

```
1= [1, 0, 2, 0, 'hello','', []]
list(filter(bool, 1))
```

a) [1, 0, 2, 'hello'," []] b) Error
c) [1, 2, 'hello'] d) [1, 0, 2, 0, 'hello'," []]

Answer: (c)

16. Which keyword is used for function in Python language? [MODEL QUESTION]
a) Function b) Def c) Fun d) Define
Answer: (b)

17. Python supports the creation of anonymous functions at runtime, using a construct called [MODEL QUESTION]
a) pi b) anonymous c) lambda d) none of these
Answer: (c)

18. Which of the following program can work with ____ parameters. [MODEL QUESTION]

```
def f(x):
    def f1(*args, **kwargs):
    print ("Matrix")
    return x(*args, **kwargs)
    return f1
```

a) any number of b) 0 c) 1 d) 2
Answer: (a)

19. What will be the output of the following Python function? [MODEL QUESTION]

```
min(max(False, -3, -4), 2, 7)
```

a) –4 b) –3 c) 2 d) False
Answer: (d)

20. Which of the following function is built-in function in python? [MODEL QUESTION]
a) factorial( ) b) print( ) c) seed( ) d) sqrt( )
Answer: (b)

21. Which function is called when the following Python program is executed? [MODEL QUESTION]

```
f = foo( )
format(f)
```

a) str( ) b) format
c) __str__( ) d) __format__( )
Answer: (c)

22. Which one of the following is the use of function in python? [MODEL QUESTION]
a) Functions don't provide better modularity for your application
b) You can't also create your own functions
c) Functions are reusable pieces of programs
d) All of the mentioned
Answer: (c)

23. What are the two main types of functions in Python? [MODEL QUESTION]
a) System function
b) Custom function
c) Built-in function & user defined function
d) User function
Answer: (c)

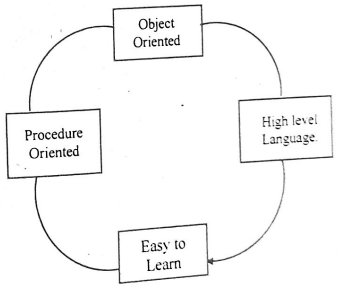
Short Answer Type Questions

1. What is Python Programming language? Explain the features of Python. What are the applications using Python Programming Language? [MODEL QUESTION]

Answer:

1st Part:

- Python was created by Guido Rossumin 1989 and is very easy to learn, read and write.
- Python Programming language is an interpreted, Object oriented, High Level Programming language with dynamic semantics.



2nd Part:

Features of Python Programming Language:

1. Easy to learn. It is loosely typed language.
2. Free and Open source.
3. It contains runtime tracking of data.
4. It is interpreted and dynamic type language.

3rd Part:

Applications of Python Programming Language:

Python is known for its general-purpose nature that makes it applicable in almost every domain of software development. Python makes its presence in every emerging field. It is the fastest-growing programming language and can develop any application. The applications are described in below:

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, beautiful Soup, Feedparser, etc. One of Python web-framework named Django is used on **Instagram**. Python provides many useful frameworks, and these are given below:

- Django and Pyramid framework(Use for heavy applications)
- Flask and Bottle (Micro-framework)
- Plone and Django CMS (Advance Content management)

2) Desktop GUI Applications:

The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a **Tk GUI library** to develop a user interface. Some popular GUI libraries are given below.

- Tkinter or Tk
- wxWidgetM
- Kivy (used for writing multitouch applications)
- PyQt or Pyside

3) Console-based Application:

Console-based applications run from the command-line or shell. These applications are computer program which are used commands to execute. This kind of application was more popular in the old generation of computers. Python can develop this kind of application very effectively. It is famous for having REPL, which means **the Read-Eval-Print Loop** that makes it the most suitable language for the command-line applications. Python provides many free library or module which helps to build the command-line apps. The necessary **IO** libraries are used to read and write. It helps to parse argument and create console help text out-of-the-box. There are also advance libraries that can develop independent console apps.

4) Software Development:

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

- **SCons** is used to build control.
- **Buildbot** and **Apache** Gumps are used for automated continuous compilation and testing.
- **Round** or **Trac** for bug tracking and project management.

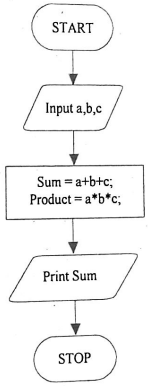
2. What is a Flowchart? Draw a flowchart to display product and sum of three different numbers. **[MODEL QUESTION]**

Answer:

1st Part:

A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction.

2nd Part:



3. What are the Python variables? What is identifier? Define the rules for identify the identifier name. **[MODEL QUESTION]**

Answer:

1st Part:

Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value. In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type. Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore. It is recommended to use lowercase letters for the variable name. Matrix and matrix both are two different variables.

2nd Part:

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program.

3rd part:
The rules to name an identifier are given below.

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case (a-z), upper-case (A-Z), underscore, or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive; for example, my name, and MyName is not the same.
- Examples of valid identifiers: a123, _n, n_9, etc.
- Examples of invalid identifiers: 1a, n%4, n 9, etc.

4. What are the local and global variables? Explain with an example.
[MODEL QUESTION]

Answer:
There are two types of variables in Python - **Local variable and Global variable.**

Local Variable:
Local variables are the variables that declared inside the function and have scope within the function. The following example is shown in below:

Example:
```
# Declaring a function
def add():
# Defining local variables. They has scope only within a function
                a = 20
                b = 30
                c = a + b
 print("The sum is:", c)
# Calling a function
add()
```
Output: The sum is: 50

Global Variables:
Global variables can be used throughout the program, and its scope is in the entire program. We can use global variables inside or outside the function.
A variable declared outside the function is the global variable by default. Python provides the global keyword to use global variable inside the function. If we don't use the global keyword, the function treats it as a local variable. The following example is shown in below:

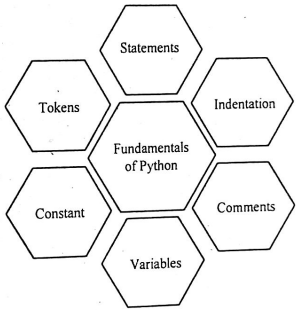
Example:
```
# Declare a variable and initialize it
x = 101
```



```
# Global variable in function
def mainFunction():
 # printing a global variable
global x
print(x)
# modifying a global variable
x = 'Welcome To Matrix'
print(x)
mainFunction()
print(x)
```
Output:
101
Welcome To Matrix
Welcome To Matrix

5. What are the basic fundamentals of Python Programming Languages?
[MODEL QUESTION]

Answer:



- **Statements**

Python statements are nothing but logical instructions that interpreters can read and execute. It can be both single and multiline.
There are two categories of statements in Python:

- Expression Statements
- Assignment Statements

- **Indentation**

Unlike most programming languages python uses indentation to mark a block of code. According to python style guidelines or PEP8, you should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

- **Comments**

Comments are nothing but tagged lines of in codes which increases the readability of the code and make the code self-explanatory. Comments can be of two categories:

- Single-line comments
- Multi-line comments

- **Variables:**

A variable is a memory address that can change and when a memory address cannot change then it is known as constant. Variable is the name of the memory location where data is stored. Once a variable is stored then space is allocated in memory. It defines a variable using a combination of numbers, letters, and the underscore character.

- **Constants:**

Constant is a type of variable that holds values, whose value cannot be changed. In reality, we rarely use constants in python.

- **Tokens:**

The tokens can be defined as a punctuator mark, reserved words, and each word in a statement. The token is the smallest unit inside the given program.
There are following tokens in Python:

- Keywords
- Identifiers
- Literals
- Operators

6. Write a Python program for find the Sum of two numbers. **[MODEL QUESTION]**

Answer:

Program:

```
first_num = int(input("Pehla number do"))
second_num = int(input("Dusra number do"))
print("The sum is",first_num + second_num)
```

Output:

Pehla number do 234
Dusra number do 1234
The sum is 1468.

7. How to delete a variable in Python shell? Explain. **[MODEL QUESTION]**

Answer:

We can delete the variable using the **del** keyword. The syntax is given below.

Syntax:

del <variable_name>

In the following example, we create a variable x and assign value to it. We deleted variable x, and print it, we get the error **"variable x is not defined"**. The variable x will no longer use in future.

Example:

```
# Assigning a value to x
x = 6
print(x)
# deleting a variable
del x
print(x)
```

Output:

```
6
Traceback (most recent call last):
File       "C:/Users/DEV/PycharmProjects/Hello/multiprocessing.py",
line 389, in
print(x)
NameError: name 'x' is not defined.
```

8. How to print Single and multiple variables in Python? Explain. **[MODEL QUESTION]**

Answer:

We can print multiple variables within the single print statement. Below are the example of single and multiple printing values.

Example - 1 (Printing Single Variable)

```
# printing single value
a = 5
print(a)
print((a))
```

Output:

```
5
5
```

Example - 2 (Printing Multiple Variables)

```
a = 5
b = 6
# printing multiple variables
print(a,b)
# separate the variables by the comma
Print(1, 2, 3, 4, 5, 6, 7, 8)
```

Output:
5 6
1 2 3 4 5 6 7 8

[MODEL QUESTION]

9. Define:
i) Python keywords
ii) Comments in Python.

Answer:

i) Python keywords:

In Python programming language, Keywords are the reserved words in Python. We cannot use a keyword as a variable name, function name or any other identifier. All the Python keywords contain lowercase letters only.

Here's a list of all keywords in Python Programming:

And	Exec	not
Assert	Finally	or
Break	For	pass
Class	From	print
Continue	Global	raise
Def	If	return
Del	Import	try
Elif	In	while
Else	Is	with
Except	Lambda	yield

ii) Comments in Python:

Comments in Python are the lines in the code that are ignored by the compiler during the execution of the program. Comments enhance the readability of the code and help the programmers to understand the code very carefully. There are three types of comments in Python –

- Single line Comments
- Multiline Comments
- Docstring Comments

Example: **Comments in Python**

```
# Python program to demonstrate comments
# sample comment
name = "MatrixEducare"
print(name)
```

Output: MatrixEducare

Comments are generally used for the following purposes:

- Code Readability

- Explanation of the code or Metadata of the project
- Prevent execution of code
- To include resources

There are three main kinds of comments in Python. They are:

1. Single-Line Comments: Python single line comment starts with the hashtag symbol (#) with no white spaces and lasts till the end of the line. If the comment exceeds one line then put a hashtag on the next line and continue the comment. Python's single-line comments are proved useful for supplying short explanations for variables, function declarations, and expressions.

Example:

```
# Print "MatrixEducare !" to console
print("MatrixEducare")
```

Output: MatrixEducare

2. Multi-Line Comments: Python does not provide the option for multiline comments. However, there are different ways through which we can write multiline comments.

Using Multiple Hashtags (#):

We can multiple hashtags (#) to write multiline comments in Python. Each and every line will be considered as a single line comment.

```
# Python program to demonstrate
# multiline comments
print("Multiline comments")
```

Output: Multiline comments

3. Docstring Comments: Python ignores the string literals that are not assigned to a variable so we can use these string literals as a comment.

Python docstring is the string literals with triple quotes that are appeared right after the function. It is used to associate documentation that has been written with Python modules, functions, classes, and methods. It is added right below the functions, modules, or classes to describe what they do. In Python, the docstring is then made available via the __doc__ attribute.

Example:

```
def multiply(a, b):
    """Multiplies the value of a and b"""
    return a*b
# Print the docstring of multiply function
print(multiply.__doc__)
```

Output: Multiplies the value of a and b

10. What is Statement and Expressions in Python Programming language?

[MODEL QUESTION]

Answer:

1st Part:

Any **Instruction** that a python interpreter can execute (carry out) is called a **Statement**. A Statement is the smallest executable unit of code that has an effect, like creating a variable or displaying a value. Each and every line of code that we write in any programming language is called a statement.

Because, all the lines are executable by the interpreter or the compiler of that programming language.

Example:

```
1 | >>> x = 3
2 | print(x)
```

Output:

>>> 3

2nd Part:

An Expression is a sequence or combination of values, variables, operators and function calls that always produces or returns a result value.

Example: x = 5, y = 3, z = x + y

In the above example x, y and z are variables, 5 and 3 are values, = and + are operators. So, the first combination x = 5 is an expression, the second combination y = 3 is an another expression and at last, z = x + y is also an expression. A **value** or a **variable** all by itself is also considered an expression, because these always evaluates to itself.

```
1 | >>> 16
2 | 16
3 | >>> a = 4
4 | a
5 | 4
6 | >>> a+16   #using Arithmetic expression
7 | 20
```

11. Write down the differences between Statement and an Expression.

Answer: **[MODEL QUESTION]**

Expression	Statement
An expression evaluates to a value	A statement executes something
The evaluation of a statement does not changes state	The execution of a statement changes state
Evaluation of an expression always produces or returns a result value	Execution of a statement may or may not produces or displays a result value, it only does whatever the statement says.
Every expression can't be a statement	Every statement can be an expression.

Example: >>> a + 16 >>> 20	Example: >>> x = 3 >>> print(x) Output: 3

12. What is IDLE? **[MODEL QUESTION]**

Answer:

IDLE (Integrated Development Environmentor Integrated Development and Learning Environment) is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit (wrapper functions for Tcl/Tk). IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.

According to the included README, its main features are:

1. Multi-window text editor with syntax highlighting, auto completion, smart indent and other.
2. Python shell with syntax highlighting.
3. Integrated debugger with stepping, persistent breakpoints, and call stack visibility.

13. Define the term: Operator Precedence. **[MODEL QUESTION]**

Answer:

The precedence of the operators is essential to find out since it enables us to know which operator should be evaluated first. The precedence table of the operators in Python is given below:

Operator	Description
**	The exponent operator is given priority over all the others used in the expression.
~ + -	The negation, unary plus, and minus.
* / % //	The multiplication, divide, modules, reminder, and floor division.
+ -	Binary plus, and minus
>> <<	Left shift. and right shift
&	Binary and.
^ |	Binary xor, and or
<= < > >=	Comparison operators (less than, less than equal to, greater than, greater then equal to).
<> == !=	Equality operators.
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators

Operator	Description
not or and	Logical operators

14. In Python how to read the inputs from the keyboard? How the input function works in Python? [MODEL QUESTION]

Answer:

1st Part:

While Python provides us with two inbuilt functions to read the input from the keyboard.

- **input (prompt)**
- **raw_input (prompt)**

input ( ) : This function first takes the input from the user and then evaluates the expression, which means Python automatically identifies whether user entered a string or a number or list. If the input provided is not correct then either syntax error or exception is raised by python. **For example –**

```
# Python program showing
# a use of input()
val = input("Enter your value: ")
print(val)
```

Output:

```
Enter your value: 123
123
>>>
```

2nd Part:

- When input() function executes program flow will be stopped until the user has given an input.
- The text or message display on the output screen to ask a user to enter input value is optional i.e. the prompt, will be printed on the screen is optional.
- Whatever you enter as input, input function convert it into a string. if you enter an integer value still input() function convert it into a string. You need to explicitly convert it into an integer in your code using typecasting.

```
# Program to check input

# type in Python
num = input ("Enter number :")
print(num)
name1 = input("Enter name : ")
print(name1)

# Printing type of input value
```



```
print ("type of number", type(num))
print ("type of name", type(name1))
```

Output:

```
Enter number: 698
698
Enter name: Matrix
Matrix
type of number <class 'str'>
type of name <class 'str'>

>>>
```

15. What do you mean by function? Explain its types and advantages. [MODEL QUESTION]

Answer:

The Function is a piece of code written to carry out a specific task. A function can be defined as the organized block of reusable code, which can be called whenever required. Python allows us to divide a large program into the basic building blocks known as a function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the Python program.

The Function helps to programmer to break the program into the smaller part. It organizes the code very effectively and avoids the repetition of the code. As the program grows, function makes the program more organized.

Types:

There are mainly two types of functions:

- **User-define functions:** The user-defined functions are those define by the **user** to perform the specific task.
- **Built-in functions:** The built-in functions are those functions that are **pre-defined** in Python.

Advantage of Functions in Python:

There are the following advantages of Python functions.

- Using functions, we can avoid rewriting the same code again and again in a program.
- We can call Python functions multiple times in a program and anywhere in a program.
- We can track a large Python program easily when it is divided into multiple functions.
- Reusability is the main achievement of Python functions.

- Function calling is always overhead in a Python program.

16. How to create a function? [MODEL QUESTION]

Answer:
Python provides the def keyword to define the function. The syntax of the define function is given below:

Syntax:
```
def my_function(parameters):
    function_block
return expression
```
The **def** keyword, along with the function name is used to define the function.

- The identifier rule must follow the function name.
- A function accepts the parameter (argument), and they can be optional.
- The function block is started with the colon (:), and block statements must be at the same indentation.
- The **return** statement is used to return the value. A function can have only one **return.**

17. What is Docstring? [MODEL QUESTION]

Answer:
Docstring describe what function does, such as the computations it performs or its return values. These descriptions serve as documentation for function so that anyone reads docstring function without having to trace through all the code in the function definition. Function docstrings are placed in the immediate line after the function header and are placed in between triple quotation marks.
```
def hello():
"""Prints "Hello World",
Returns:
None"""
Print ("Hello World")
Return
```

18. What is return function? [MODEL QUESTION]

Answer:
The return statement is used at the end of the function and returns the result of the function. It terminates the function execution and transfers the result where the function is called. The return statement cannot be used outside of the function. It can contain the expression which gets evaluated and value is returned to the caller function. If the return statement has no expression or does not exist itself in the function then it returns the **None** object.

19. What are the applications of lambda function? [MODEL QUESTION]

Answer:
The lambda function is commonly used with Python built-in functions **filter()** function and **map()** function.

- **Use lambda function with filter():**

The Python built-in **filter() function** accepts a function and a list as an argument. It provides an effective way to filter out all elements of the sequence. It returns the new sequence in which the function evaluates to **True**. Consider the following example where we filter out the only odd number from the given list:
```
#program to filter out the tuple which contains odd numbers
list = (10,22,37,41,100,123,29)
oddlist = tuple(filter(lambda x:(x%3 == 0),lst)) # the tuple cont
ains all the items of the tuple for which the lambda function eva
luates to true
print(oddlist)
```
Output: (37, 41, 123, 29)

- **Using lambda function with map():**

The **map() function** in Python accepts a function and a list. It gives a new list which contains all modified items returned by the function for each item. Consider the following example of **map()** function.
```
#program to filter out the list which contains odd numbers
list = (10,20,30,40,50,60)
square_list = list(map(lambda x:x**2,lst)) # the tuple contains a
ll the items of the list for which the lambda function evaluates
to true
print(square_tuple)
```
Output: (100, 400, 900, 1600, 2500, 3600)

20. What is Type conversion in Python? [MODEL QUESTION]

Answer:
Python defines type conversion functions to directly convert one data type to another.

There are two types of Type Conversion in Python:

1. Implicit Type Conversion:
In Implicit type conversion of data types in Python, the Python interpreter automatically converts one data type to another without any user involvement.

Example:
```
x = 10
print("x is of type:",type(x))
y = 10.6
print("y is of type:",type(y))
x = x + y
print(x)
print("x is of type:",type(x))
```

Output:

```
x is of type: <class 'int'>
y is of type: <class 'float'>
20.6
x is of type: <class 'float'>
```

Here we can see the type of 'x' got automatically changed to the "float" type from the "integer" type. this is a simple case of Implicit type conversion in python.

2. Explicit Type Conversion:

In Explicit Type Conversion in Python, the data type is manually changed by the user as per their requirement. Various forms of explicit type conversion are explained below:

1. int(a, base): This function converts any data type to integer. 'Base' specifies the base in which string is if the data type is a string.

2. float(): This function is used to convert any data type to a floating-point number.

3. ord() : This function is used to convert a character to integer.

4.hex() : This function is to convert integer to hexadecimal string.

5. oct() : This function is to convert integer to octal string.

6. tuple() : This function is used to convert to a tuple.

7. set() : This function returns the type after converting to set.

8. list() : This function is used to convert any data type to a list type.

9. dict() : This function is used to convert a tuple of order (key,value) into a dictionary.

10. str() : Used to convert integer into a string.

11. complex(real,imag) : This function converts real numbers to complex(real,imag) number.

Example:

```
# Python code to demonstrate Type conversion
# using  tuple(), set(), list()
# initializing string
s = 'Matrix'
# printing string converting to tuple
c = tuple(s)
print ("After converting string to tuple : ",end="")
print (c)
# printing string converting to set
c = set(s)
print ("After converting string to set : ",end="")
print (c)
# printing string converting to list
c = list(s)
print ("After converting string to list : ",end="")
print (c)
```


Output:

```
After converting string to tuple : ('M', 'a', 't', 'r', 'i', 'x')
After converting string to set : {'t', 'a', 'r', 'i'}
After converting string to list : ['g', 'e', 'e', 'k', 's']
```

Long Answer Type Questions

1. Write a short note on Python Programming concept. **[MODEL QUESTION]**

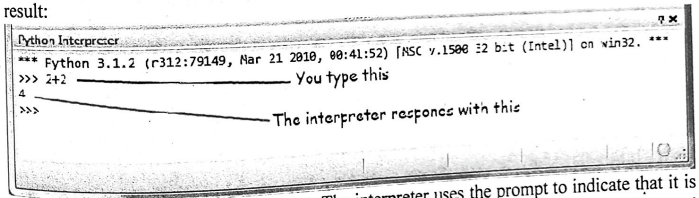
Answer:

Python is the programming language which is an example of a **high-level language**. Other high-level languages are C++, PHP, Pascal, C#, and Java.

As we might infer from the name high-level language, there are also **low-level languages**, sometimes referred to as machine languages or assembly languages. Loosely speaking, computers can only execute programs written in low-level languages. Thus, programs written in a high-level language have to be translated into something more suitable before they can run.

Almost all programs are written in high-level languages because of their advantages. It is much easier to program in a high-level language so programs take less time to write, they are shorter and easier to read, and they are more likely to be correct. Second, high-level languages are **portable**, meaning that they can run on different kinds of computers with few or no modifications.

The engine that translates and runs Python is called the **Python Interpreter**: There are two ways to use it: *immediate mode* and *script mode*. In immediate mode, we type Python expressions into the Python Interpreter window, and the interpreter immediately shows the result:



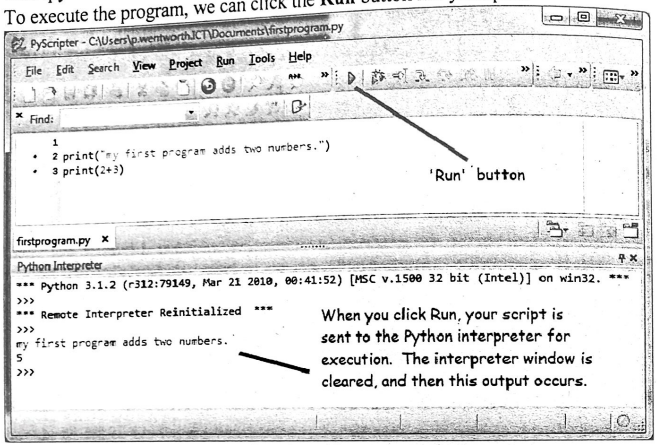
The >>> is called the **Python prompt**. The interpreter uses the prompt to indicate that it is ready for instructions. We typed 2 + 2, and the interpreter evaluated our expression, and replied 4, and on the next line it gave a new prompt, indicating that it is ready for more input.

Alternatively, you can write a program in a file and use the interpreter to execute the contents of the file. Such a file is called a **script**. Scripts have the advantage that they can be saved to disk, printed, and so on.

We use a program development environment called **PyScripter**. There are various other development environments. For example, we created a file named firstprogram.py using

PyScripter. By convention, files that contain Python programs have names that end with .py

To execute the program, we can click the **Run** button in PyScripter:



Most programs are more interesting than this one.

Working directly in the interpreter is convenient for testing short bits of code because we get immediate feedback.

2. Print "Hello World" in console using Python Programming Language. Explain each step how to write and run Python script? [MODEL QUESTION]

Answer:

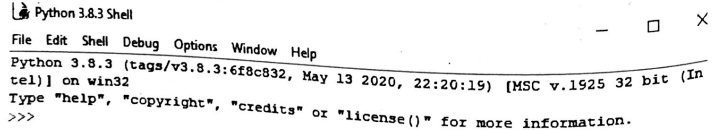
Python provides us the two ways to run a program:

- Using Interactive interpreter prompt
- Using a script file.

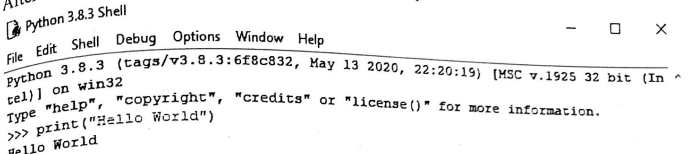
Using Interactive Interpreter prompt:

To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have Python2 and Python3 both installed on your system).

It will open the following prompt where we can execute the Python statement and check their impact on the console.



After writing the print statement, press the **Enter** key.



Here, we get the output message **"Hello World !"** printed on the console. The interpreter prompt is best to run the single-line statements of the code. However, we cannot write the code every-time on the terminal. It is not suitable to write multiple lines of code.

Using the script mode:

We can write multiple lines code into a file which can be executed later. For this purpose, we need to open an editor like notepad, create a file named and save it with **.py** extension, which stands for **"Python".** Now, we will implement the above example using the script mode.

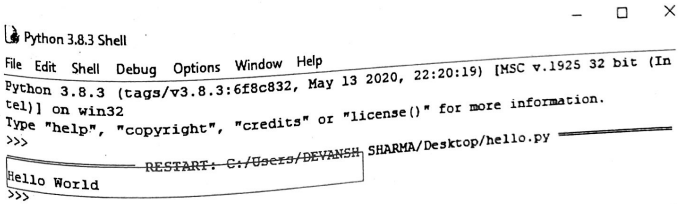
print ("hello world"); #here, we have used print() function to print the message on the co nsole.

To run this file named as first.py, we need to run the following command on the terminal.

Step - 1: Open the Python interactive shell, and click **"File"** then choose **"New",** it will open a new blank script in which we can write our code.

Step -2: Now, write the code and press **"Ctrl+S"** to save the file.

Step - 3: After saving the code, we can run it by clicking "Run" or "Run Module"(Press F5 button). It will display the output to the shell.The output will be shown as follows.



Step - 4: Apart from that, we can also run the file using the operating system terminal. But, we should be aware of the path of the directory where we have saved our file. Open the command line prompt and navigate to the directory.

C:\Users\Filename\Desktop>python hello.py
We need to type the **python** keyword, followed by the file name and hit enter to run the Python file.

3. What are the different types of Operators used in Python? [MODEL QUESTION]

Answer:
The operator can be defined as a symbol which is responsible for a particular operation between two operands. Operators are the pillars of a program on which the logic is built in a specific programming language. Python provides a variety of operators, which are described as follows.

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operators:
Arithmetic operators are used to perform arithmetic operations between two operands. It includes + (addition), - (subtraction), *(multiplication), /(divide), %(reminder), //(floor division), and exponent (**) operators.
Consider the following table for a detailed explanation of arithmetic operators.

Operator	Description
+ (Addition)	It is used to add two operands. For example, if a = 20, b = 10 => a+b = 30
- (Subtraction)	It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if a = 20, b = 10 => a - b = 10
/ (divide)	It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a/b = 2.0
* (Multiplication)	It is used to multiply one operand with the other. For example, if a = 20, b = 10 => a * b = 200
% (reminder)	It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a%b = 0
** (Exponent)	It is an exponent operator represented as it calculates the first operand power to the second operand.

Comparison Operators:
Comparison operators are used to comparing the value of the two operands and returns Boolean true or false accordingly. The comparison operators are described in the following table.

Operator	Description
==	If the value of two operands is equal, then the condition becomes true.
!=	If the value of two operands is not equal, then the condition becomes true.
<=	If the first operand is less than or equal to the second operand, then the condition becomes true.
>=	If the first operand is greater than or equal to the second operand, then the condition becomes true.
>	If the first operand is greater than the second operand, then the condition becomes true.
<	If the first operand is less than the second operand, then the condition becomes true.

Assignment Operators:
The assignment operators are used to assign the value of the right expression to the left operand. The assignment operators are described in the following table.

Operator	Description
=	It assigns the value of the right expression to the left operand.
+=	It increases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if a = 10, b = 20 => a+ = b will be equal to a = a+ b and therefore, a = 30.
-=	It decreases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if a = 20, b = 10 => a- = b will be equal to a = a- b and therefore, a = 10.
*=	It multiplies the value of the left operand by the value of the right operand and assigns the modified value back to then the left operand. For example, if a = 10, b = 20 => a* = b will be equal to a = a* b and therefore, a = 200.
%=	It divides the value of the left operand by the value of the right operand and assigns the reminder back to the left operand. For example, if a = 20, b = 10 => a % = b will be equal to a = a % b and therefore, a = 0.
**=	a**=b will be equal to a=a**b, for example, if a = 4, b =2, a**=b will assign 4**2 = 16 to a.
//=	A//=b will be equal to a = a// b, for example, if a = 4, b = 3, a//=b will assign 4//3 = 1 to a.

Bitwise Operators:
The bitwise operators perform bit by bit operation on the values of the two operands. Consider the following example.

Example:

```
if a = 7
 b = 6
then, binary (a) = 0111
binary (b) = 0011
hence, a & b = 0011
a | b = 0111
 a ^ b = 0100
~ a = 1000
```

Operator	Description
& (binary and)	If both the bits at the same place in two operands are 1, then 1 is copied to the result. Otherwise, 0 is copied.
| (binary or)	The resulting bit will be 0 if both the bits are zero; otherwise, the resulting bit will be 1.
^ (binary xor)	The resulting bit will be 1 if both the bits are different; otherwise, the resulting bit will be 0.
~ (negation)	It calculates the negation of each bit of the operand, i.e., if the bit is 0, the resulting bit will be 1 and vice versa.
<< (left shift)	The left operand value is moved left by the number of bits present in the right operand.
>> (right shift)	The left operand is moved right by the number of bits present in the right operand.

Logical Operators:

The logical operators are used primarily in the expression evaluation to make a decision. Python supports the following logical operators.

Operator	Description
and	If both the expression are true, then the condition will be true. If a and b are the two expressions, a → true, b → true => a and b → true.
or	If one of the expressions is true, then the condition will be true. If a and b are the two expressions, a → true, b → false => a or b → true.
not	If an expression **a** is true, then not (a) will be false and vice versa.

Membership Operators:

Python membership operators are used to check the membership of value inside a Python data structure. If the value is present in the data structure, then the resulting value is true otherwise it returns false.

Operator	Description
in	It is evaluated to be true if the first operand is found in the second operand (list, tuple, or dictionary).
not in	It is evaluated to be true if the first operand is not found in the second operand (list, tuple, or dictionary).

Identity Operators:

The identity operators are used to decide whether an element certain class or type.

Operator	Description
is	It is evaluated to be true if the reference present at both sides point to the same object.
is not	It is evaluated to be true if the reference present at both sides do not point to the same object.

4. What are the types of function arguments in Python? Explain. [MODEL QUESTION]

Answer:

There may be several types of arguments which can be passed at the time of function call.

a. Required arguments
b. Default arguments
c. Variable-length arguments
d. Keyword arguments

a. Required Arguments:

We can provide the arguments at the time of the function call. As far as the required arguments are concerned, these are the arguments which are required to be passed at the time of function calling. If either of the arguments is not provided in the function call, or the position of the arguments is changed, the Python interpreter will show the error.

Example:

```
def func(name):
    message = "Hi "+name
    return message
name = input("Enter the name:")
print(func(name))
```

Output:

```
Enter the name: John
Hi John
```

b. Default Arguments:

Python allows us to initialize the arguments at the function definition. If the value of any of the arguments is not provided at the time of function call, then that argument can be initialized with the value given in the definition even if the argument is not specified at the function call.

Example:

```
def printme(name,age=22):
    print("My name is",name,"and age is",age)
printme(name = "john")
```

Output:

```
My name is John and age is 22
```

c. Variable-length Arguments (*args):

In large projects, sometimes we may not know the number of arguments to be passed in advance. In such cases, Python provides us the flexibility to offer the comma-separated values which are internally treated as tuples at the function call. By using the variable-length arguments, we can pass any number of arguments.

However, at the function definition, we define the variable-length argument using the ***args** (star) as *<variable - name >.

Example:

```
def printme(*names):
    print("type of passed argument is ",type(names))
    print("printing the passed arguments...")
    for name in names:
        print(name)
printme("john","David","smith","nick")
```

Output:

```
type of passed argument is  <class 'tuple'>
printing the passed arguments...
john
David
smith
nick
```

In the above code, we passed ***names** as variable-length argument. We called the function and passed values which are treated as tuple internally. The tuple is an iterable sequence the same as the list. To print the given values, we iterated ***arg names** using for loop.

d. Keyword arguments(kwargs):**

Python allows us to call the function with the keyword arguments. This kind of function call will enable us to pass the arguments in the random order.

The name of the arguments is treated as the keywords and matched in the function calling and definition. If the same match is found, the values of the arguments are copied in the function definition.

Example:

```
#function func is called with the name and message as the keyword
 arguments
def func(name,message):
    print("printing the message with",name,"and ",message)

    #name and message is copied with the values John and hello re
spectively
    func(name = "John",message="hello")
```

Output: printing the message with John and hello

5. What is Lambda function? Illustrate with example. [MODEL QUESTION]

Answer:

Python Lambda function is known as the anonymous function that is defined without a name. Python allows us to not declare the function in the standard manner, i.e., by using the **def** keyword. Rather, the anonymous functions are declared by using the **lambda** keyword. However, Lambda functions can accept any number of arguments, but they can return only one value in the form of expression.

The anonymous function contains a small piece of code. It simulates inline functions of C and C++, but it is not exactly an inline function. It can accept any number of arguments and has only one expression. It is useful when the function objects are required.The syntax to define an anonymous function is given below:

Syntax:

lambda arguments: expression

Example: Multiple arguments to Lambda function

```
# a and b are the arguments and a*b is the expression which gets
evaluated and returned.
x = lambda a,b: a*b
print("mul = ", x(20,10))
<p><strong>Output:</strong></p>
<div class="codeblock3"><pre>
mul =  200
</pre></div>
<h2 class="h2">Why use lambda function?</h2>
<p>The main role of the lambda function is better described in th
e scenarios when we use them anonymously inside another function.
 In Python, the lambda function can be used as an argument to the
 <strong>higher-
order functions</strong> which accepts other functions as argumen
ts.</p>
<p>Consider the following example:</p>
<h3 class="h3">Example 1:</h3>
<div class="codeblock"><textarea name="code" class="python">
#the function table(n) prints the table of n
def table(n):
    return lambda a:a*n # a will contain the iteration variable i
 and a multiple of n is returned at each function call
n = int(input("Enter the number:"))
b = table(n) #the entered number is passed into the function tabl
e. b will contain a lambda function which is called again and aga
in with the iteration variable i
for i in range(1,11):
    print(n,"X",i,"=",b(i)) #the lambda function b is called with
 the iteration variable i
```

Output:

```
Enter the number:10
10 X 1 = 10
10 X 2 = 20
10 X 3 = 30
10 X 4 = 40
10 X 5 = 50
10 X 6 = 60
10 X 7 = 70
10 X 8 = 80
10 X 9 = 90
10 X 10 = 100
```

6. Write a short note on: Function composition in Python. **[MODEL QUESTION]**

Answer:

Function composition is the way of combining two or more functions in such a way that the output of one function becomes the input of the second function and so on. For example, let there be two functions "F" and "G" and their composition can be represented as F(G(x)) where "x" is the argument and output of G(x) function will become the input of F() function.

Code Example:

```
# Function to add 2
# to a number
def add(x):
    return x + 2

# Function to multiply
# 2 to a number
def multiply(x):
    return x * 2

# Printing the result of
# composition of add and
# multiply to add 2 to a number
# and then multiply by 2
print("Adding 2 to 5 and multiplying the result with 2: ",
      multiply(add(5)))
```

Output:

Adding 2 to 5 and multiplying the result with 2: 14.

CONDITIONS & ITERATIONS

Multiple Choice Type Questions

1. What will be the output of the following Python expression if x = 56.236? **[MODEL QUESTION]**

```
print("%.2f"%x)
```

a) 56.236 **b) 56.23** c) 56.000 d) 56.24

Answer: (d)

2. Which module in the python standard library parses option received from the command line? **[MODEL QUESTION]**

a) getarg **b) getopt** c) main d) os

Answer: (b)

3. What will be the output of the following Python code? **[MODEL QUESTION]**

```
print("abc. DEF" .capitalize())
```

a) Abc.def **b) abc.def** c) Abc.Def d) ABC.DEF

Answer: (a)

4. What will be the output of the following Python code? **[MODEL QUESTION]**

```
x=['ab', 'cd']
for i in x:
    i.upper()
print(x)
```

a) ['ab', 'cd'] b) ['AB', 'CD']
c) [None, None] d) none of these

Answer: (b)

5. What will be the output of the following Python code? **[MODEL QUESTION]**

```
i=1
while True:
    if i%3==0:
        break
    print(i)
    i+=1
```

a) 1 2 **b) 1 2 3** c) error d) none of these

Answer: (c)

6. What will be the output of the following Python code? **[MODEL QUESTION]**

```
i=5
while True:
    if i%0011==0:
        break
    print(i)
    i+=1
```

a) 5 6 7 8 9 10 b) 5 6 7 8 c) 5 6 d) error

Answer: (b)

7. What will be the output of the following Python code? [MODEL QUESTION]

```
i=1
while True:
    if i%2==0:
     break
    print(i)
    i+=2
```

a) 1 b) 1 2 c) 1 2 4 5 6 ... d) 1 3 5 7 9 11 ...

Answer: (d)

8. What will be the output of the following Python code? [MODEL QUESTION]

```
True = False
while True:
    print(True)
    break
```

a) True b) False c) None d) none of these

Answer: (d)

Short Answer Type Questions

1. Explain If-else Statement used in Python Programming language. [MODEL QUESTION]

Answer:

Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the particular conditions. Condition checking is the backbone of decision making.

In python, decision making is performed by the following statements.

Statement	Description
If Statement	The if statement is used to test a specific condition. If the condition is true, a block of code (if-block) will be executed.
If - else Statement	The if-else statement is similar to if statement except the fact that, it also provides the block of the code for the false case of the condition to be checked. If the condition provided in the if statement is false, then the else statement will be executed.
Nested if Statement	Nested if statements enable us to use if ? else statement inside an outer if statement.

2. Explain the statements with flowchart and examples: [MODEL QUESTION]

i) if Statement

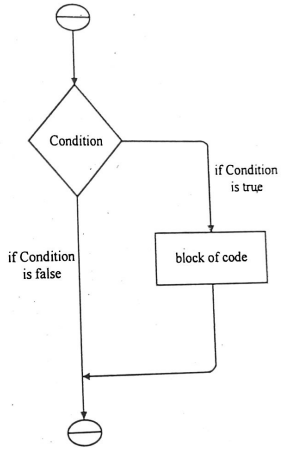
ii) if–else Statement

iii) elif Statement

Answer:

i) if Statement:

The if statement is used to test a particular condition and if the condition is true, it executes a block of code known as if-block. The condition of if statement can be any valid logical expression which can be either evaluated to true or false.



The syntax of the if-statement is given below:

```
if expression:
statement
```

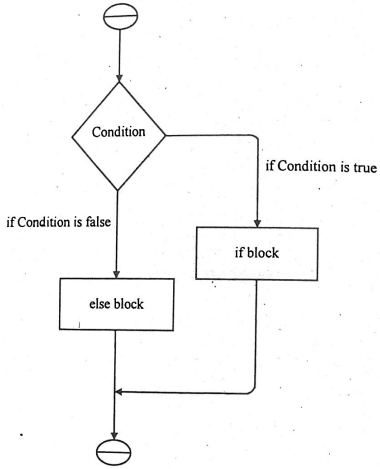
Example: Program to print the largest of the three numbers.

```
a = int(input("Enter a? "))
b = int(input("Enter b? "))
c = int(input("Enter c? "))
if a>b and a>c:
print("a is largest")
if b>a and b>c:
print("b is largest")
if c>a and c>b:
print("c is largest")
```

Output:

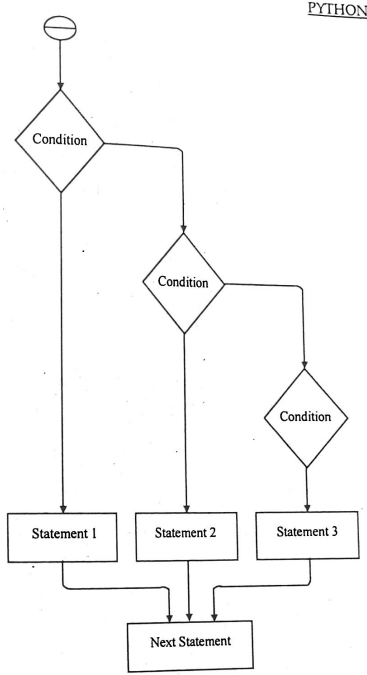
```
Enter a? 100
Enter b? 120
Enter c? 130
c is largest
```

ii) if-else Statement:
The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition. If the condition is true, then the if-block is executed. Otherwise, the else-block is executed.



The syntax of the if-else statement is given below:

```
if condition:
#block of statements
else:
#another block of statements (else-block)
```

Example: Program to check whether a person is eligible to vote or not.

```
age = int (input("Enter your age? "))
if age>=18:
 print("You are eligible to vote !!");
else:
print("Sorry! you have to wait !!");
```

Output:

```
Enter your age? 90
You are eligible to vote !!
```

iii) elif Statement:
The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them. We can have any number

of elif statements in our program depending upon our need. However, using elif is optional.
The elif statement works like an if-else-if ladder statement in C. It must be succeeded by an if statement.

The syntax of the elif statement is given below:

```
if expression 1:
# block of statements
elif expression 2:
# block of statements
elif expression 3:
# block of statements
else:
# block of statements
```

Example: Program to check the number which is divisible by 10,50 and 100.

```
number = int(input("Enter the number?"))
if number==10:
print("number is equals to 10");
elif number==50:
print("number is equal to 50");
elif number==100:
print("number is equal to 100");
else:
print("number is not equal to 10, 50 or 100");
```

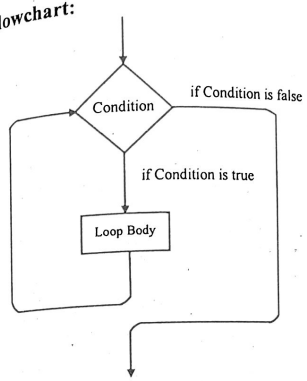
Output:

```
Enter the number? 15
number is not equal to 10, 50 or 100
```

3. Explain the following loops with flowchart and examples: **[MODEL QUESTION]**
i) While loop
ii) for loop
Answer:
i) While loop:
The Python while loop allows a part of the code to be executed until the given condition returns false. It is also known as a pre-tested loop.It can be viewed as a repeating if statement. When we don't know the number of iterations then the while loop is most effective to use.
The syntax is given below:

```
while expression:
 statements
```


Flowchart:



Example: Program to print 1 to 10 using while loop

```
i=1
#The while loop will iterate until condition becomes false.
While(i<=10):
 print(i)
 i=i+1
```

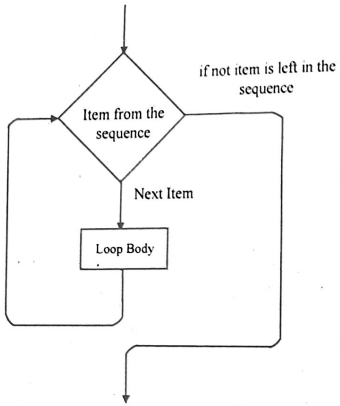
Output:

```
1 2 3 4 5 6 7 8 9 10
```

ii) For loop:
The for loop in Python is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.
The syntax of for loop in python is given below:

```
for iterating_var in sequence:
statement(s)
```

The For loop flowchart:



Example: Program to print table of given number.

```
n = int(input("Enter the number "))
for i in range(1,11):
c = n*i
print(n,"*",i,"=",c)
```

Output:

```
Enter the number 10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
```

4. What is range() function?

Answer: **[MODEL QUESTION]**

The **range()** function is used to generate the sequence of the numbers. If we pass the range(10), it will generate the numbers from 0 to 9. The syntax of the range() function is given below:

Syntax: range(start,stop,step size)

- The start represents the beginning of the iteration.
- The stop represents that the loop will iterate till stop-1. The **range(1,5)** will generate numbers 1 to 4 iterations. It is optional.
- The step size is used to skip the specific numbers from the iteration. It is optional to use. By default, the step size is 1. It is optional.

Example: Program to print numbers in sequence.

```
for i in range(10):
print(i,end = ' ')
```

Output: 0 1 2 3 4 5 6 7 8 9

5. Define 'Pass statements in loop'.

Answer: **[MODEL QUESTION]**

The pass statement is used to declare the empty loop. It is also used to define empty class, function, and control statement.

Example:

```
# An empty loop
str1 = 'javatpoint'
i = 0
while i < len(str1):
i += 1
pass
print('Value of i :', i)
```

Output: Value of i: 10

6. Define: Using else in while loop. Write a program to print Fibonacci numbers in given limit. **[MODEL QUESTION]**

Answer:

1st Part:

Python allows us to use the else statement with the while loop also. The else block is executed when the condition given in the while statement becomes false. Like for loop, if the while loop is broken using break statement, then the else block will not be executed, and the statement present after else block will be executed. The else statement is optional to use with the while loop.

Example:

```
i=1
while(i<=5):
print(i)
 i=i+1
else:
print("The while loop exhausted")
```

Output: 1 2

2nd Part:

A program to print Fibonacci numbers in given limit:

```
terms = int(input("Enter the terms "))
# first two initial terms
a = 0
b = 1
count = 0
```

```
# check if the number of terms is Zero or negative
if (terms <= 0):
   print("Please enter a valid integer")
elif (terms == 1):
   print("Fibonacci sequence upto",limit,":")
   print(a)
else:
   print("Fibonacci sequence:")
   while (count < terms) :
       print(a, end = ' ')
       c = a + b
       # updating values
       a = b
       b = c
    count += 1
```

7. Write a program to get the remainder of two negative number using while loop and modulus (%) operator in Python. **[MODEL QUESTION]**

Answer:

```
while True:
   x = input(' Do you want to continue (Y / N)? ')
  if x.upper() != 'Y':
      break
# input two integer number and store it into x and y
 x = int(input (' First number: '))
 y = int(input (' Second number: '))
print("Modulus of negative number is: ", x, "%", y, " = ", x % y,
 sep = " ")
print("Modulus of negative number is: ", y, "%", x, " = ", y % x,
 sep = " ")
```

8. What is modulus operator? **[MODEL QUESTION]**

Answer:

Python Modulus Operator is an inbuilt operator that returns the remaining numbers by dividing the first number from the second. It is also known as the **Python modulo**. In Python, the modulus symbol is represented as the percentage (%) symbol. Hence, it is called the remainder operator.

Rem = X % Y

Syntax:

Here X and Y are two integer numbers, and the modulus (%) is used in between to get the remainder where the first number (X) is divided by the second number (Y).

For example, we have two numbers, 24 and 5. And we can get the remainder by using the modulus or modulo operator between the numbers 24 % 5.

Example:

Write a program to get the remainder of two numbers using the while loop and modulus (%) operator in Python.

```
while True: # if the while condition is true if block is executed
  a = input ('Do you want to continue or not (Y / N)? ')
 if a.upper() != 'Y':  # If the user pass 'Y', the following stat
ement is executebreak
  a = int(input (' First number is: ')) # first number
  b = int(input (' Second number is: ')) # second number
print(a, ' % ', b, ' = ', a % b) # perform a % b
print(b, ' % ', a, ' = ', b % a) # perform b % a
```

Output:

```
Do you want to continue or not (Y/N)? Y
First number is: 37
Second number is: 5
37 % 5 = 2
5 % 37 = 5

Do you want to continue or not (Y/N)? Y
First number is: 37
Second number is: 5
24 % 5 = 4
5 % 24 = 5

Do you want to continue or not (Y/N)? Y
First number is: 37
Second number is: 5
28 % 5 = 4
5 % 28 = 5
```

Long Answer Type Questions

1. Briefly explain Conditional statement used in Python Programming language. **[MODEL QUESTION]**

Answer:

In python, there are three conditional loops. They are:

i) if Statement:

The if statement is used to test a particular condition and if the condition is true, it executes a block of code known as if-block. The condition of if statement can be any valid logical expression which can be either evaluated to true or false.

The syntax of the if-statement is given below:

```
if expression:
statement
```

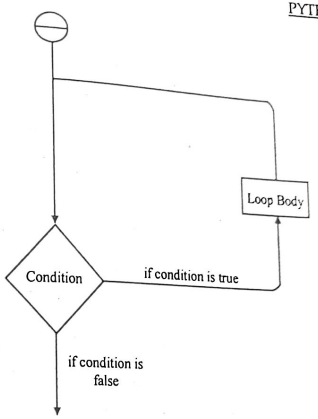
ii) if-else Statement:
The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition. If the condition is true, then the if-block is executed. Otherwise, the else-block is executed.
The syntax of the if-else statement is given below:

```
if condition:
#block of statements
else:
#another block of statements (else-block)
```

iii) elif Statement:
The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them. We can have any number of elif statements in our program depending upon our need. However, using elif is optional.
The elif statement works like an if-else-if ladder statement in C. It must be succeeded by an if statement.
The syntax of the elif statement is given below:

```
if expression 1:
# block of statements
elif expression 2:
# block of statements
elif expression 3:
# block of statements
else:
# block of statements
```

2. Briefly explain the loops used in Python Programming language. Why we use loops in Python? Explain the advantages of loops. [MODEL QUESTION]
Answer:
The flow of the programs written in any programming language is sequential by default. Sometimes we may need to alter the flow of the program. The execution of a specific code may need to be repeated several numbers of times.
For this purpose, The programming languages provide various types of loops which are capable of repeating some specific code several numbers of times. Consider the following diagram to understand the working of a loop statement.



There are the following loop statements in Python.

Loop	Description
for loop	The for loop is used in the case where we need to execute some part of the code until the given condition is satisfied. The for loop is also called as a per-tested loop. It is better to use for loop if the number of iteration is known in advance.
while loop	The while loop is to be used in the scenario where we don't know the number of iterations in advance. The block of statements is executed in the while loop until the condition specified in the while loop is satisfied. It is also called a pre-tested loop.
do-while loop	The do-while loop continues until a given condition satisfies. It is also called post tested loop. It is used when it is necessary to execute the loop at least once (mostly menu driven programs).

2nd Part:
The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times. For example, if we need to print the first 10 natural numbers then, instead of using the print statement 10 times, we can print inside a loop which runs up to 10 iterations.

3rd Part:
There are the following advantages of loops in Python.
1. It provides code re-usability.

2. Using loops, we do not need to write the same code again and again.
3. Using loops, we can traverse over the elements of data structures (array or linked lists).

3. Briefly explain the nested loops used in Python Programming Languages. [MODEL QUESTION]

Answer:
Python allows us to nest any number of for loops inside a **for** loop. The inner loop is executed n number of times for every iteration of the outer loop. The syntax is given below.

Syntax:
```
for iterating_var1 in sequence:  #outer loop
for iterating_var2 in sequence:  #inner loop
#block of statements
#Other statements
```

Program:
```
# User input for number of rows
rows = int(input("Enter the rows:"))
# Outer loop will print number of rows
for i in range(0,rows+1):
# Inner loop will print number of Astrisk
  for j in range(i):
  print("*",end = '')
   print()
```

Output:
```
Enter the rows:5
*
**
***
****
*****
```

4. Write a Python Program for guessing the number using while loop. [MODEL QUESTION]

Answer:
```
Guessing the number using while loop:
Import random
counter = 1
correct_num = random.randint(1,100)
User_input = int(input("Enter the number between 1 and 100"))
print(correct_num)
while user_input! = correct_num:
if user_input<correct_num:
print("guess a higher number")
```



```
counter = counter + 1
user_input = int(input("Enter the number between 1 and 100"))
else:
print("lower number")
counter = counter + 1
user_input = int(input("Enter the number between 1 and 100"))
print("Correct guess, the number of attempt is," counter)
```

5. What are the loops control statements used in Python Programming Languages? [MODEL QUESTION]

Answer:
We can change the normal sequence of **while** loop's execution using the loop control statement. When the while loop's execution is completed, all automatic objects defined in that scope are demolished. Python offers the following control statement to use within the while loop.

1. Continue Statement - When the continue statement is encountered, the control transfer to the beginning of the loop.

Example:
```
# prints all letters except 'a' and 't'
i = 0
str1 = 'javatpoint'
while i < len(str1):
if str1[i] == 'a' or str1[i] == 't':
 i += 1
 continue
print('Current Letter :', a[i])
  i += 1
```

Output:
```
Current Letter: j
Current Letter: v
Current Letter: p
Current Letter: o
Current Letter: i
Current Letter: n
```

2. Break Statement - When the break statement is encountered, it brings control out of the loop.

Example:
```
# The control transfer is transferred
# when break statement soon it sees t
i = 0
str1 = 'EDUCATION'
while i < len(str1):
    if str1[i] == 'N':
        i += 1
```

```
        break
    print('Current Letter :', str1[i])
    i += 1
```

Output:

```
Current Letter: E
Current Letter: D
Current Letter: U
Current Letter: C
```

3. Pass Statement - The pass statement is used to declare the empty loop. It is also used to define empty class, function, and control statement. Let's understand the following example.

Example:

```
# An empty loop
str1 = 'POPULARPUB'
i = 0
while i < len(str1):
    i += 1
    pass
print('Value of i :', i)
```

Output: `Value of i: 10`

6. Explain the following statements: **[MODEL QUESTION]**

i) Break Statement

ii) Continue Statement

Answer:

i) Break Statement:

The break is a keyword in python which is used to bring the program control out of the loop. The break statement breaks the loops one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. In other words, we can say that break is used to abort the current execution of the program and the control goes to the next line after the loop.

The break is commonly used in the cases where we need to break the loop for a given condition. **The syntax of the break is given below:**

```
1. #loop statements
2. break;
```

Example: break statement with while loop

```
i = 0;
while 1:
    print(i," ",end=""),
   i=i+1;
        if i == 10:
```



```
    break;
print("came out of while loop");
```

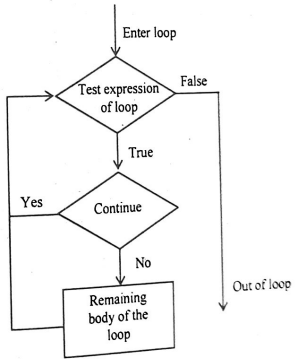
ii) Continue Statement:

The continue statement in Python is used to bring the program control to the beginning of the loop. The continue statement skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition. Python is used to bring the program control to the beginning of the loop. The continue statement in Python skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition.

Syntax:

```
3. #loop statements
   continue
          #the code to be skipped
```

Flow diagram:



Example:

```
str = "POPULARPUB"
 for i in str:
        if(i == 'T'):
           continue
        print(i)
```

RECURSION, STRINGS, LIST, DICTIONARIES & TUPLES

Multiple Choice Type Questions

1. Which of the following is not a core data type in Python programming? [MODEL QUESTION]

a) Tuples b) Lists c) Class d) Dictionary

Answer: (c)

2. Which one of the following is not a keyword in Python language? [MODEL QUESTION]

a) pass b) eval c) assert d) nonlocal

Answer: (b)

3. What will be the output of the following Python program? [MODEL QUESTION]

```
z=set('abc')
z.add('san')
z.update(set(['p', 'q']))
z
```

a) {'a', 'c', 'c', 'p', 'q', 's', 'a', 'n'}
b) {'abc', 'p', 'q', 'san'}
c) {'a', 'b', 'c', 'p', 'q', 'san'}
d) {'a', 'b', 'c', ['p', 'q'], 'san'}

Answer: (c)

4. Which of the following statements is used to create an empty set in Python? [MODEL QUESTION]

a) ( ) b) [ ] c) { } d) set()

Answer: (d)

5. To add a new element to a list we use which Python command? [MODEL QUESTION]

a) `list1.addEnd(5)` b) `list1.addLast(5)`
c) `list1.append(5)` d) `list1.add(5)`

Answer: (c)

6. What will be the value of 'result' in following Python program? [MODEL QUESTION]

```
list1=[1, 2, 3, 4]
list2=[2, 4, 5, 6]
list3=[2, 6, 7, 8]
result=list()
result.extend(i for i in list1 if i not in (list2+list3) and i
not in result)
result.extend(i for i in list2 if i not in (list1+list3) and i
not in result)
result.extend(i for i in list3 if i not in (list1+list2) and i
not in result)
```

a) [1, 3, 5, 7, 8] b) [1, 7, 8] c) [1, 2, 4, 7, 8] d) error

Answer: (a)

7. What will be the output of the following Python code? [MODEL QUESTION]

```
print ('*', "abcde".center(6), '*', sep=' ')
```

a) * abcde * b) *abcde * c) * abcde* d) * abced *

Answer: (b)

8. What will be the output of the following Python code? [MODEL QUESTION]

```
1. >>>list1 = [1, 3]
2. >>>list2 = list1
3. >>>list1[0] = 4
4. >>>print(list2)
```

a) [1, 4] b) [1, 3, 4] c) [4, 3] d) [1, 3]

Answer: (c)

9. Which of the following Python statements will result in the output: 6?

```
A=[[1, 2, 3],
   [4, 5, 6],
   [7, 8, 9]]
```

[MODEL QUESTION]

a) A[2][1] b) A[1][2] c) A[3][2] d) A[2][3]

Answer: (b)

10. Which of the following is a Python tuple? [MODEL QUESTION]

a) {1, 2, 3} b) { } c) [1, 2, 3] d) (1, 2, 3)

Answer: (d)

Short Answer Type Questions

1. What do you mean by python list? Explain its characteristics. [MODEL QUESTION]

Answer:
A list in Python is used to store the sequence of various types of data. Python lists are mutable type its mean we can modify its element after it created. However, Python consists of six data-types that are capable to store the sequences, but the most common and reliable type is the list.
A list can be defined as a collection of values or items of different types. The items in the list are separated with the comma (,) and enclosed with the square brackets [].

Creating a List:
L1 = ["John", 102, "USA"]
L2 = [1, 2, 3, 4, 5, 6]

Characteristics of Lists:
The list has the following characteristics:

- The lists are ordered.
- The element of the list can access by index.
- The lists are the mutable type.
- The lists are mutable types.
- A list can store the number of various elements.

2. Explain the operations of Python lists. [MODEL QUESTION]

Answer:
The concatenation (+) and repetition (*) operators work in the same way as they were working with the strings.

Operator	Description	Example
Repetition	The repetition operator enables the list elements to be repeated multiple times.	L1*2 = [1, 2, 3, 4, 1, 2, 3, 4]
Concatenation	It concatenates the list mentioned on either side of the operator.	l1+l2 = [1, 2, 3, 4, 5, 6, 7, 8]
Membership	It returns true if a particular item exists in a particular list otherwise false.	print(2 in l1) prints True.
Iteration	The for loop is used to iterate over the list elements.	for i in l1: print(i) **Output:** 1 2 3 4 ...
Length	It is used to get the length of the list	len(l1) = 4

3. Explain: Iterating the list & Append function. [MODEL QUESTION]

Answer:
1st Part:
A list can be iterated by using a for - in loop. A simple list containing four strings, which can be iterated as follows:

```
list = ["John", "David", "James", "Jonathan"]
for i in list:
# The i variable will iterate over the elements of the List and
contains each element in each iteration.
print(i)
```

Output:

```
John
David
James
Jonathan
```

2nd Part:
Python provides append() function which is used to add an element to the list. However, the append() function can only add value to the end of the list.
Consider the following example in which, we are taking the elements of the list from the user and printing the list on the console.

```
#Declaring the empty list
l =[]
#Number of elements will be entered by the user
n = int(input("Enter the number of elements in the list:"))
# for loop to take the input
for i in range(0,n):
    # The input is taken from the user and added to the list as t
he item
    l.append(input("Enter the item:"))
print("printing the list items..")
# traversal loop to print the list items
for i in l:
    print(i, end = "  ")
```

Output:

```
Enter the number of elements in the list:5
Enter the item:25
Enter the item:46
Enter the item:12
Enter the item:75
Enter the item:42
```

```
printing the list items
25  46  12  75  42
```

4. Define the term: List indexing & splitting. [MODEL QUESTION]

Answer:

List indexing: The indexing is processed in the same way as it happens with the strings. The elements of the list can be accessed by using the slice operator []. The index starts from 0 and goes to length - 1. The first element of the list is stored at the 0th index, the second element of the list is stored at the 1st index, and so on.

List = [0,1,2,3,4,5]

0	1	2	3	4	5

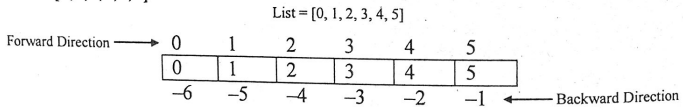
Here,

List[0] = 0
List[1] = 1
List[2] = 2
List[3] = 3
List[4] = 4

List[0:] = [0,1,2,3,4,5]
List[2:4] = [2,3]

Python provides the flexibility to use the negative indexing also. The negative indices are counted from the right. The last element (rightmost) of the list has the index -1; its adjacent left element is present at the index -2 and so on until the left-most elements are encountered.

List = [0,1,2,3,4,5]



5. Explain the escape sequence in Strings. [MODEL QUESTION]

Answer:

Python provides the escape sequence. The backslash(/) symbol denotes the escape sequence. The backslash can be followed by a special character and it interpreted differently. The single quotes inside the string must be escaped. We can apply the same as in the double quotes.

Example:

```
# using triple quotes
print('''''Hii, "What's there?"''')
# escaping single quotes
print('Hello, "What\'s going on?"')
# escaping double quotes
print("Hello, \"What's going on?\"")
```


The list of an escape sequence is given below:

Escape Sequence	Description
\newline	It ignores the new line.
\\	Backslash
\'	Single Quotes
\\"	Double Quotes

6. What is Python Set? How to create Set? [MODEL QUESTION]

Answer:

A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

Creating a Set:

The set can be created by enclosing the comma-separated immutable items with the curly braces {}. Python also provides the set() method, which can be used to create the set by the passed sequence.

7. What is Tuple? How to create the tuple? [MODEL QUESTION]

Answer:

Python Tuple is used to store the sequence of immutable Python objects. The Tuple is similar to lists since the value of the items stored in the list can be changed, whereas the tuple is immutable, and the value of the items stored in the tuple cannot be changed.

Creating a Tuple:

A tuple can be written as the collection of comma-separated (,) values enclosed with the small () brackets. The parentheses are optional but it is good practice to use. A tuple can be defined as follows:

```
T1 = (101, "Peter", 22)
T2 = ("Apple", "Banana", "Orange")
T3 = 10,20,30,40,50
print(type(T1))
print(type(T2))
print(type(T3))
```

8. What is Python Dictionary? [MODEL QUESTION]

Answer:
Python Dictionary is used to store the data in a key-value pair format. The dictionary is the data type in Python, which can simulate the real-life data arrangement where some specific value exists for some particular key. It is the mutable data-structure. The dictionary is defined into element Keys and values.

- Keys must be a single element
- Value can be any type such as list, tuple, integer, etc.

In other words, A dictionary is the collection of key-value pairs where the value can be any Python object. In contrast, the keys are the immutable Python object, i.e., Numbers, string, or tuple.

Creating the dictionary:
The dictionary can be created by using multiple key-value pairs enclosed with the curly brackets {}, and each key is separated from its value by the colon (:).The syntax to define the dictionary is given below.

Syntax:

```
Dict = {"Name": "Tom", "Age": 22}
In the above dictionary Dict, The keys Name and Age are the string that is an immutable object.
Let's see an example to create a dictionary and print its content.
Employee = {"Name": "John", "Age": 25, "salary":25000,"Company":"MATRIX"}
print(type(Employee))
print("printing Employee data .... ")
print(Employee)
```

Output:

```
<class 'dict'>
Printing Employee data ....
{'Name': 'John', 'Age': 25, 'salary': 25000, 'Company': 'MATRIX'}
```

Python provides the built-in function dict() method which is also used to create dictionary. The empty curly braces {} is used to create empty dictionary.

9. Write down the advantages and disadvantages of Recursion in Python. **[MODEL QUESTION]**

Answer:
Advantages of Python Recursion:
i) A complicated function can be split down into smaller sub-problems utilizing recursion.
ii) Sequence creation is simpler through recursion than utilizing any nested iteration.
iii) Recursive functions render the code look simple and effective.

Disadvantages of Python Recursion:
i) Slow.
ii) Logical but difficult to trace and debug.
iii) Requires extra storage space. For every recursive calls separate memory is allocated for the variables.
iv) Recursive functions often throw a Stack Overflow Exception when processing or operations are too large.

10. What are the differences between List and Tuple? **[MODEL QUESTION]**

Answer:

	List	Tuple
1.	The literal syntax of list is shown by the [].	The literal syntax of the tuple is shown by the ().
2.	The List is mutable.	The tuple is immutable.
3.	The List has the a variable length.	The tuple has the fixed length.
4.	The list provides more functionality than a tuple.	The tuple provides less functionality than the list.
5.	The list is used in the scenario in which we need to store the simple collections with no constraints where the value of the items can be changed.	The tuple is used in the cases where we need to store the read-only collections i.e., the value of the items cannot be changed. It can be used as the key inside the dictionary.
6.	The lists are less memory efficient than a tuple.	The tuples are more memory efficient because of its immutability.

11. What is recursive function in Python? Write down the python code to find a factorial of 3 using recursion. **[MODEL QUESTION]**

Answer:
1st Part:
A recursive function (DEF) is a function which either calls itself or is in a potential cycle of function calls. The term Recursion can be defined as the process of defining something in terms of itself. In simple words, it is a process in which a function calls itself directly or indirectly.

Syntax:

```
def func(): <--
              |
              | (recursive call)
              |
    func() ----
```

2nd Part:

```
#recursive function to calculate factorial
def fact(n):
  """ Function to find factorial """
  if n == 1:
    return 1
  else:
    return (n * fact(n-1))
print ("3! = ",fact(3))
```

Output:

```
3! = 6
```

12. How to creating a string in Python? What is slicing strings? **[MODEL QUESTION]**

Answer:

1st Part:

We can create a string by enclosing the characters in single-quotes or double- quotes. Python also provides triple-quotes to represent the string, but it is generally used for multiline string or **docstrings**.

Example:

```
#Using single quotes
str1 = 'Hello Python'
print(str1)
#Using double quotes
str2 = "Hello Python"
print(str2)
#Using triple quotes
str3 = '''''Triple quotes are generally used for
  represent the multiline or
   docstring'''
print(str3)
```


2nd Part:

Python slicing is about obtaining a sub-string from the given string by slicing it respectively from start to end.

Python slicing can be done in two ways.

- slice() Constructor
- Extending Indexing

slice() Constructor:

The slice() constructor creates a slice object representing the set of indices specified by range(start, stop, step).

Syntax:

- slice(stop)
- slice(start, stop, step)

Parameters:

start: Starting index where the slicing of object starts.

stop: Ending index where the slicing of object stops.

step: It is an optional argument that determines the increment between each index for slicing.

Return Type: Returns a sliced object containing elements in the given range only.

13. Write down the difference between List and String. **[MODEL QUESTION]**

Answer:

List	String
Mutable	Immutable
Element can be assigned at a specified index	Element/character cannot be assigned at specified index.
Example: >>>L=[7,4,8,9] >>>L[2]=6 #valid	Example: >>>str= "python" >>>str[2]= 'p' #error

Long Answer Type Questions

1. What are the build-in functions available in Python list? Write the program to remove the duplicate element of the list. **[MODEL QUESTION]**

Answer:

Python provides the following built-in functions, which can be used with the lists.

Function	Description	Example
cmp(list1, list2)	It compares the elements of both the lists.	This method is not used in the Python 3 and the above versions.
len(list)	It is used to calculate the length of the list.	L1 = [1,2,3,4,5,6,7,8] print(len(L1)) Output: 8
max(list)	It returns the maximum element of the list.	L1 = [12,34,26,48,72] print(max(L1)) Output: 72
min(list)	It returns the minimum element of the list.	L1 = [12,34,26,48,72] print(min(L1)) Output: 12
list(seq)	It converts any sequence to the list.	str = "Johnson" s = list(str) print(type(s)) <class list>

A Program to remove the duplicate element of the list:

```
list1 = [1,2,2,3,55,98,65,65,13,29]
# Declare an empty list that will store unique values
list2 = []
for i in list1:
    if i not in list2:
        list2.append(i)
print(list2)
```

2. What are the String functions used in Python? **[MODEL QUESTION]**

Answer:

Python provides various in-built functions that are used for string handling.

Method	Description
capitalize()	It capitalizes the first character of the String. This function is deprecated in python3
casefold()	It returns a version of s suitable for case-less comparisons.
center(width ,fillchar)	It returns a space padded string with the original string centred with equal number of left and right spaces.
count(string,begin,end)	It counts the number of occurrences of a substring in a String between begin and end index.
decode(encoding = 'UTF8', errors = 'strict')	Decodes the string using codec registered for encoding.

Method	Description
encode()	Encode S using the codec registered for encoding. Default encoding is 'utf-8'.
endswith(suffix ,begin=0,end=len(string))	It returns a Boolean value if the string terminates with given suffix between begin and end.
expandtabs(tabsize = 8)	It defines tabs in string to multiple spaces. The default space value is 8.
find(substring ,beginIndex, endIndex)	It returns the index value of the string where substring is found between begin index and end index.
format(value)	It returns a formatted version of S, using the passed value.
index(subsring, beginIndex, endIndex)	It throws an exception if string is not found. It works same as find() method.
isalnum()	It returns true if the characters in the string are alphanumeric i.e., alphabets or numbers and there is at least 1 character. Otherwise, it returns false.
isalpha()	It returns true if all the characters are alphabets and there is at least one character, otherwise False.
isdecimal()	It returns true if all the characters of the string are decimals.
isdigit()	It returns true if all the characters are digits and there is at least one character, otherwise False.
isidentifier()	It returns true if the string is the valid identifier.
islower()	It returns true if the characters of a string are in lower case, otherwise false.
isnumeric()	It returns true if the string contains only numeric characters.
isprintable()	It returns true if all the characters of s are printable or s is empty, false otherwise.
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 white-space, otherwise false.
istitle()	It returns true if the string is titled properly and false otherwise. A title string is the one in which the first character is upper-case whereas the other characters are lower-case.
isupper()	It returns true if all the characters of the string(if exists) is true otherwise it returns false.
join(seq)	It merges the strings representation of the given sequence.
len(string)	It returns the length of a string.

Method	Description
ljust(width[,fillchar])	It returns the space padded strings with the original string left justified to the given width.
lower()	It converts all the characters of a string to Lower case.
lstrip()	It removes all leading whitespaces of a string and can also be used to remove particular character from leading.
partition()	It searches for the separator sep in S, and returns the part before it, the separator itself, and the part after it. If the separator is not found, return S and two empty strings.
maketrans()	It returns a translation table to be used in translate function.
replace(old,new[,count])	It replaces the old sequence of characters with the new sequence. The max characters are replaced if max is given.

3. What are the basic tuple functions used in Python? **[MODEL QUESTION]**

Answer:

Operator	Description	Example
Repetition	The repetition operator enables the tuple elements to be repeated multiple times.	T1*2 = (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
Concatenation	It concatenates the tuple mentioned on either side of the operator.	T1+T2 = (1, 2, 3, 4, 5, 6, 7, 8, 9)
Membership	It returns true if a particular item exists in the tuple otherwise false	print (2 in T1) prints True.
Iteration	The for loop is used to iterate over the tuple elements.	for i in T1: print(i) **Output: 1 2 3 4..**
Length	It is used to get the length of the tuple.	len(T1) = 5

4. Write down the Python code for Bubble sort. **[MODEL QUESTION]**

Answer:

The bubble sort uses a straightforward logic that works by repeating swapping the adjacent elements if they are not in the right order. It compares one pair at a time and swaps if the first element is greater than the second element; otherwise, move further to the next pair of elements for comparison.

Implementation:

```
# Creating a bubble sort function
def bubble_sort(list1):
    # Outer loop for traverse the entire list
    for i in range(0,len(list1)-1):
        for j in range(len(list1)-1):
```



```
            if(list1[j]>list1[j+1]):
                temp = list1[j]
                list1[j] = list1[j+1]
                list1[j+1] = temp
    return list1
list1 = [5, 3, 8, 6, 7, 2]
print("The unsorted list is: ", list1)
# Calling the bubble sort function
print("The sorted list is: ", bubble_sort(list1))
```

Output:

```
The unsorted list is:  [5, 3, 8, 6, 7, 2]
The sorted list is:  [2, 3, 5, 6, 7, 8]
```

5. Write down the Python code for Selection Sort. **[MODEL QUESTION]**

Answer:

The array spilled virtually in the two parts in the insertion sort - An **unsorted part** and **sorted** part.

The sorted part contains the first element of the array and other unsorted subpart contains the rest of the array. The first element in the unsorted array is compared to the sorted array so that we can place it into a proper sub-array.

It focuses on inserting the elements by moving all elements if the right-side value is smaller than the left side.

It will repeatedly happen until the all element is inserted at correct place. To sort the array using insertion sort below is the algorithm of insertion sort.

- Spilt a list in two parts - sorted and unsorted.
- Iterate from arr[1] to arr[n] over the given array.
- Compare the current element to the next element.
- If the current element is smaller than the next element, compare to the element before, Move to the greater elements one position up to make space for the swapped element.

Python Program:

```
# creating a function for insertion
def insertion_sort(list1):

        # Outer loop to traverse through 1 to len(list1)
        for i in range(1, len(list1)):

            value = list1[i]

            # Move elements of list1[0..i-1], that are
```

```
            # greater than value, to one position ahead
            # of their current position
            j = i - 1
            while j >= 0 and value < list1[j]:
                list1[j + 1] = list1[j]
                j -= 1
            list1[j + 1] = value
        return list1
            # Driver code to test above

list1 = [10, 5, 13, 8, 2]
print("The unsorted list is:", list1)

print("The sorted list1 is:", insertion_sort(list1))
```

Output:

```
The unsorted list is: [10, 5, 13, 8, 2]
The sorted list1 is: [2, 5, 8, 10, 13]
```

6. Write down the differences between Lists, Sets, Dictionary & Tuple.

[MODEL QUESTION]

Answer:

List	Sets	Dictionaries	Tuples
List = [10, 12, 15]	Set={1, 23, 34} print(set)->{1, 23, 24} Set={1, 1} print(set)->{1}	Dict={"Ram": 26, "mary":24}	Words=("spam", "egss") Or Words="spam", "eggs"
Access: print(list[0])	Print(set) Set elements can't be indexed	print(dict["ram"])	Print(words[0])
Can contains duplicate elements	Can't contain duplicate elements. Faster compared to lists	Can't contain duplicate keys, but can contain duplicate values	Can contains duplicate elements. Faster compared to lists.
List[0]=100	set.add(7)	Dict["Ram"]=27	Words[0]="care"->TypeError
Mutable	Mutable	Mutable	Immutable – values can't bechanged once assigned
List=[]	Set=set()	Dict={}	Words=()

List	Sets	Dictionaries	Tuples
Slicing can be done print(list[1:2)->[12]	Slicing: Not done	Slicing: Not done	Slicing can also be done on tuples
Usage: Use lists if you have a collection of data that doesn't need random access. Use lists when you need a simple, iterable collection that is modified frequently	Usage: - Membership testing and the elimination of duplicate entries. - when you need uniqueness for the elements.	Usage: - When you need a logical association b/w key: value pair. -when you need fast lookup for your data, based on a custom key. -when your data is being constantly modified.	Usage: Use tuples when your data cannot change. A tuple is used in combination with a dictionary, for example, a tuple might represent a key, because its immutable.

CLASSES & OBJECTS

Multiple Choice Type Questions

1. The assignment of more than one function to a particular operator is _____ [MODEL QUESTION]

a) Operator over-assignment b) Operator overriding
c) Operator overloading d) Operator instance

Answer: (c)

2. Which of the following is not a class method? [MODEL QUESTION]

a) Non-static b) Static c) Bounded d) Unbounded

Answer: (a)

3. Special methods need to be explicitly called during object creation. [MODEL QUESTION]

a) True b) False

Answer: (b)

4. What is hasattr(obj, name) used for? [MODEL QUESTION]

a) To access the attribute of the object b) To delete an attribute
c) To check if an attribute exists or not d) To set an attribute

Answer: (c)

5. What is delattr(obj, name) used for? [MODEL QUESTION]

a) To print deleted attribute b) To delete an attribute
c) To check if an attribute is deleted or not d) To set an attribute

Answer: (b)

Explanation: delattr(obj, name) deletes an attribute in a class.

6.__del__ method is used to destroy instances of a class. [MODEL QUESTION]

a) True b) False

Answer: (a)

Short Answer Type Questions

1. What is Class? How to create a class in Python? [MODEL QUESTION]

Answer:

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. A class is a virtual entity and blueprints of an object. The class came into existence when it instantiated.

Creating classes in Python:

In Python, a class can be created by using the keyword class, followed by the class name. The syntax to create a class is given below:

Syntax:

```
class ClassName:
#statement_suite
```

In Python, we must notice that each class is associated with a documentation string which can be accessed by using **<class-name>.__doc__**. A class contains a statement suite including fields, constructor, function, etc. definition.

2. What is Self parameter? [MODEL QUESTION]

Answer:

The self-parameter refers to the current instance of the class and accesses the class variables. We can use anything instead of self, but it must be the first parameter of any function which belongs to the class.

3. What is an Object? How to delete the object? [MODEL QUESTION]

Answer:

A unique instance of the class defined as Object. An object comprises both data and members and methods. The object is an entity that has state and behavior. We can delete the properties of the object or object itself by using the del keyword.

Example:

```
class Employee:
    id = 10
    name = "John"

    def display(self):
        print("ID: %d \nName: %s" % (self.id, self.name))
    # Creating a emp instance of Employee class

emp = Employee()
```

```
# Deleting the property of object
del emp.id
# Deleting the object itself
del emp
emp.display()
```

4. What is data abstraction? What is Polymorphism? **[MODEL QUESTION]**

Answer:

1st Part:

Data abstraction refers to providing only essential information to the outside world and hiding their background details i.e. represent the needed information in program without presenting details.

2nd Part:

It is a technique when one method can assume different forms as per the need. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

5. What is inheritance?

Answer: **[MODEL QUESTION]**

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object. It enables new classes to receive or inherit the properties and methods of existing classes. It is a way to express a relationship between classes.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class. It provides the re-usability of the code.

6. Explain: Built-in class functions.

Answer: **[MODEL QUESTION]**

SN	Function	Description
1.	getattr(obj,name,default)	It is used to access the attribute of the object
2.	setattr(obj, name, value)	It is used to set a particular value to the specific attribute of an object.
3.	delattr(obj, name)	It is used to delete a specific attribute.

4.	hasattr(obj, name)	It returns true if the object contains some specific attribute.

Example:

```
class Student:
def __init__(self, name, id, age):
        self.name = name
        self.id = id
        self.age = age

    # creates the object of the class Student
    s = Student("John", 101, 22)
    # prints the attribute name of the object s
    print(getattr(s, 'name'))
    # reset the value of attribute age to 23
    setattr(s, "age", 23)
    # prints the modified value of age
    print(getattr(s, 'age'))
    # prints true if the student contains the attribute with na
    me id
    print(hasattr(s, 'id'))
    # deletes the attribute age
    delattr(s, 'age')
    # this will give an error since the attribute age has been
    deleted
    print(s.age)
```

7. Explain: Built-in class attributes. **[MODEL QUESTION]**

Answer:

A Python class also contains some built-in class attributes which provide information about the class.

The built-in class attributes are given in the below table.

SN	Attribute	Description
1	__dict__	It provides the dictionary containing the information about the class namespace.
2	__doc__	It contains a string which has the class documentation
3	__name__	It is used to access the class name.
4	__module__	It is used to access the module in which, this class is defined.
5	__bases__	It contains a tuple including all base classes.

8. Write a short note on Method Overriding. **[MODEL QUESTION]**

Answer:

When the parent class method is defined in the child class with some specific implementation, then the concept is called method overriding. We may need to perform method overriding in the scenario where the different definition of a parent class method is needed in the child class.

Consider the following example to perform method overriding in python.

```
class Bank:
def getroi(self):
     return 10;
class SBI(Bank):
 def getroi(self):
      return 7;
class ICICI(Bank):
 def getroi(self):
    return 8;
b1 = Bank()
b2 = SBI()
b3 = ICICI()
print("Bank Rate of interest:",b1.getroi());
print("SBI Rate of interest:",b2.getroi());
print("ICICI Rate of interest:",b3.getroi());
```

Output:

```
Bank Rate of interest: 10
SBI Rate of interest: 7
ICICI Rate of interest: 8
```

Long Answer Type Questions

1. What is difference between Object oriented Programming and procedural programming. **[MODEL QUESTION]**

Answer:

The difference between object-oriented and procedure-oriented programming is given below:

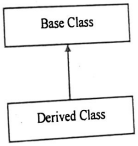
SN	Object-oriented Programming	Procedural Programming
1.	Object-oriented programming is the problem-solving approach and used where computation is done by using objects.	Procedural programming uses a list of instructions to do computation step by step.
2.	It makes the development and maintenance easier.	In procedural programming, It is not easy to maintain the codes when the project becomes lengthy.

SN	Object-oriented Programming	Procedural Programming
3.	It simulates the real world entity. So real-world problems can be easily solved through oops.	It doesn't simulate the real world. It works on step by step instructions divided into small parts called functions.
4.	It provides data hiding. So it is more secure than procedural languages. You cannot access private data from anywhere.	Procedural language doesn't provide any proper way for data binding, so it is less secure.
5.	Example of object-oriented programming languages is C++, Java, .Net, Python, C#, etc.	Example of procedural languages are: C, Fortran, Pascal, VB etc.

2. What are the different types of inheritance in Python? **[MODEL QUESTION]**

Answer:

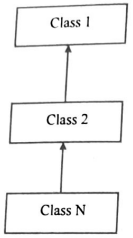
Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch.



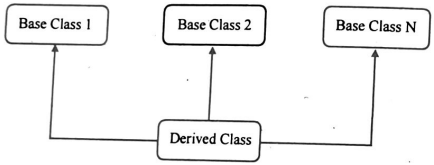
In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class.

Multi-Level inheritance:

Multi-Level inheritance is possible in python like other object-oriented languages. Multi-level inheritance is archived when a derived class inherits another derived class. There is no limit on the number of levels up to which, the multi-level inheritance is archived in python.



Multiple inheritance:
Python provides us the flexibility to inherit multiple base classes in the child class.



3. What are the different types of math module used in Python.

[MODEL QUESTION]

Answer:
There are different types of math modules used in Python. They are-

i) math.log():
This method returns the natural logarithm of a given number. It is calculated to the base e.

Example:
```
import math
number = 2e-7  # small value of of x
print('log(fabs(x), base) is :', math.log(math.fabs(number), 10))
```
Output: log(fabs(x), base) is : -6.698970004336019

ii) math.log10():
This method returns base 10 logarithm of the given number and called the standard logarithm.

Example:
```
import math
x=13  # small value of of x
print('log10(x) is :', math.log10(x))
```


Output: log10(x) is : 1.1139433523068367

iii) math.exp():
This method returns a floating-point number after raising e to the given number.

Example:
```
import math
number = 5e-2  # small value of of x
print('The given number (x) is :', number)
print('e^x (using exp() function) is :', math.exp(number)-1)
```
Output: The given number (x) is : 0.05
e^x (using exp() function) is : 0.05127109637602412

iv) math.pow(x,y):
This method returns the power of the x corresponding to the value of y. If value of x is negative or y is not integer value than it raises a ValueError.

Example:
```
import math
number = math.pow(10,2)
print("The power of number:",number)
```
Output: `The power of number: 100.0`

v) math.floor(x) :
This method returns the floor value of the x. It returns the less than or equal value to x.

Example:
```
import math
number = math.floor(10.25201)
print("The floor value is:",number)
```
Output: The floor value is: 10

vi) math.ceil(x):
This method returns the ceil value of the x. It returns the greater than or equal value to x.

Example:
```
import math
number = math.ceil(10.25201)
print("The floor value is:",number)
```
Output: The floor value is: 11

vii) math.fabs(x):
This method returns the absolute value of x.

Example:
```
import math
```

```
number = math.fabs(10.001)
print("The floor absolute is:",number)
```

Output: The absolute value is: 10.001

viii) math.factorial():

This method returns the factorial of the given number x. If x is not integral, it raises a **ValueError**.

Example:

```
import math
number = math.factorial(7)
print("The factorial of number:",number)
```

Output: The factorial of number: 5040

ix) math.modf(x):

This method returns the fractional and integer parts of x. It carries the sign of x is float.

Python provides the several math modules which can perform the complex task in single-line of code. In this tutorial, we have discussed a few important math modules.