Workbook on Python

Lecture No	Content
1.	Introduction to Python Why Python? Features of Python Installation of Python interpreter Interpreted Vs Scripting Language Input and Output in Python Variables in Python Operators in Python
2.	Flow controls in Python If statement If else statement Ladder if else statement Loop controls While loop For loop
3.	Collections in Python List in Python Tuple in Python List Vs. Tuple Set in Python Dictionary in Python
4.	Function in Python Why function? Creation of function Example applications of function Recursion Modules in Python Import module From .. import …. Statement
5.	Object oriented programming Pillars of object oriented programming Class and object Static and non-static variables Constructor in Python Types of constructor in Python
6.	Inheritance in Python Types of inheritance in Python Single Inheritance Hierarchical Inheritance Multiple Inheritance Multi-level Inheritance Hybrid Inheritance
7.	Polymorphism in Python Method Overload Method Overriding Method Overloading Vs. Method Overriding

	Exception handling in Python Use of try, except and finally
8.	Work on String Built-in methods of string Work on upper(), lower(), title(), split(), replace() methods Work on Boolean methods

Lecture – 1 (Introduction to Python)

Long Answer Questions:-

1. **What is Python? Describe its applications.**

Are you aware that websites like YouTube and Dropbox make use of Python Programming in their source code? Python is a commonly used language which one can easily understand and apply. You can make nearly anything using Python. Most systems today (Mac, Linux, UNIX, etc.) have Python installed as a default setting since it is an open source and free language. Upon reading this book, you are going to become fluent in this awesome code language and see it applied to a variety of examples. No type declaration of methodology, parameters, functions, or variables (like in other languages) are found in Python making its code concise and easy. As I said earlier, you can use the language in everything if you want to build a website, make a game, or even create a search engine. The big plus of using Python is, an explicit compiler is not necessary since it's an entirely interpreted language (Perl, Shell, etc.).

Introduction to Python

- Python is powerful and general purpose programming language, developed by Guido Van Rossum (1989).
- Guido Van Rossum developed python language at mathematical research institute called 'CWI'.
- CWI is located at Netherland.
- Guido developed python language by taking the different language features like:-
  - i.) Procedure oriented programming language--→C
  - ii.) Object oriented programming language--→C++, Java
  - iii.) Scripting language--→ Shell script, perl.
  - iv.) Modular programming language--→Modula – 3

Note:- Guido Van Rossum made available python language to the public in 1991.

By using python language we can do the following things:-

1. GUI application development
2. Web application development
3. Data analytics
4. Task automation
5. Test cases implementations
6. Scientific application development
7. Network application development
8. Gaming / animation application development

2. **What are different features of Python?**

Python supports the following features:-

1. **Simple and easy to learn:-**
   - ✓ The syntaxes of the python language are very simple.
   - ✓ Anybody can remember the python language syntaxes, rules and regulations very easily.
   - ✓ By developing the python programs (or) applications programmers need not to focus on the syntaxes.
   - ✓ Instead of focusing on syntaxes they can focus on the business logic implementation.
   - ✓ The elegant syntaxes of the python language make the people to learn python in easiest manner.
   - ✓ Without having any other programming language knowledge directly anybody can learn python language.
   - ✓ The simple and powerful syntax of the python language makes the programmers to express their business logic is less lines of code.
   - ✓ Because of simple feature of python language project development cost, development time and maintenance cost will become less.
2. **Platform independent:-**
   - ✓ The Python applications which are developed on one platform are going to execute on irrespective of platforms without making any changes in the python applications.
   - ✓ To achieve the portability feature with respect to every o.s. separate python software is developed for every version of python.
3. **Free, Open source and Re-distribution:-**
   - ✓ Anybody can use the python software without purchasing licence agreement of python.
   - ✓ Anybody can read the python source code and they can do the modifications in the python source code and we can redistribute that code to others.
4. **High level language:-**
   - ✓ While developing python programmers (or) developers need not to focus on memory management and memory used by program (low level details).
5. **Supporting procedure oriented programming & object oriented programming features:-**
   - ✓ Python language supports both p.o.p and o.o.p language features.
   - ✓ If we develop any application according to the oops principles then that application will get security, flexibility & reusability.
   - ✓ Different oops principles are:

i. Encapsulation
ii. Polymorphism
iii. Inheritance
iv. Abstraction

- ✓ Python supports oops principles because of that reason python applications will get the security, flexibility and reusability.

6. Interpreted language:-

- ✓ Python applications don't require explicit compilation so that compiler is not required for in python software.
- ✓ Directly we can run the python applications without compiling explicit.
- ✓ Python interpreter is responsible for execution of python applications.
- ✓ Whenever we run python applications python interpreter will check the syntax error. If no syntax error python interpreter converts that code in the form of intermediate code in the form of low level format and executes it.
- ✓ The intermediate code of python applications is known as byte code.
- ✓ The extension for the byte code file is .pyc.

7. Extensible:-

- ✓ Python application execution is slower compared to c, c++ programs execution.
- ✓ To overcome the above problem we can implement some logics by using c, c++ language and we can use c, c++ programs into python application.
- ✓ Python source code doesn't contain security i.e., anybody can read python code and they can do the modifications in the source code.
- ✓ If we want to provide security to the some part (logic) or some algorithm of python application then we represent that code on logic (or) algorithm by using c (or) c++ languages and we use that code into python application.

8. Embeddable:-

- ✓ We can use the python code into the other language programs such as c, c++, java.
- ✓ In order to provide the scripting capabilities to other language programs / applications we use python code into those applications.

9. Rich in built-in libraries:-

- ✓ Python language is providing huge built – in libraries.
- ✓ Python developers can use built-in libraries in their applications.
- ✓ By using built-in libraries application development will become faster.
- ✓ Third party people developed libraries, modules we can add to python software and used into python applications.

3. Write steps to installation of Python interpreter.

Installation of Python interpreter:-

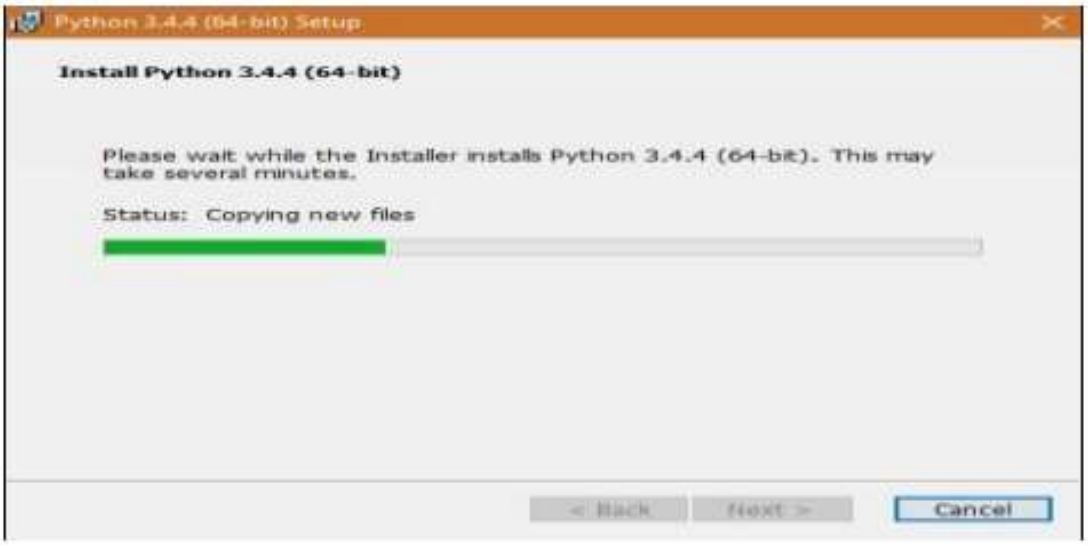
Now we are going to see a step by step guide to download and install the Python language interpreter.
Python programming language is available for all of the three known platforms for Windows, Linux/Unix, and Mac OS. Below are the links from where Python interpreters can be downloaded for these environments.
Steps to install Python on Windows Platform:-
Please follow the below steps:

1. Check for Windows installer if it is 32-bit or 64-bit. Accordingly, download Python version for Windows platform for the given link.
2. Once downloaded, click on the installer. The below screen will be visible which will trigger Python installation on Windows.

3. Choose the first option as “Install for all users” and click on the next button to proceed.
4. Next, the system will ask to select the destination directory. Choose the directory as shown below and click on Next button.

5. Next, the system will ask to customize Python 3.4.4. Keep the default setup and click on the Next button as shown in the below screenshot.

6. Installer will start the installation which will take several minutes and the below screenshot will be visible during this point of time.



7. Once Python interpreter installation is completed, click on the Finish button to complete the installation on Windows platform.

4. **What are different types of operators in Python?**

Operators are used to manipulate the value of operands. Python supports the following types of operators:-

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Special Operators

Arithmetic Operators:- These operators are used to perform the mathematical operations like addition, subtraction, multiplication and so on.

Operator	Meaning
+	Add two operands (or) unary plus
-	Subtract right operand from the left (or) unary minus
*	Multiply two operands
/	Divide left operand by right one (result float)
%	Modulus – remainder of the division of left operand by right.
//	Floor division – division that results into whole number adjusted to the left in number line.
**	Exponent – left operand raised the power of right

Example Application 1:-

```
x=15
y=4
print('x+y=',x+y)
print('x-y=',x-y)
print('x*y=',x*y)
print('x/y=',x/y)
print('x//y=',x//y)
print('x**y=',x**y)

O/P:-
x+y=19
x-y=11
x*y=60
x/y=3
x//y=3
x**y=50625
```

Example Application 2:-

```
x=15.0
y=4
print ('x/y = ',x/y)
print('x//y=',x//y)

O/P:-
x/y=3.75
x//y=3.0
```

Example Application 3:-

```
x='softpro'
y=2
print('x*y=',x*y)
print('y*x=',y*x)

O/P:
x*y=SoftproSoftpro
y*x=SoftproSoftpro
```

Comparison / Relational operator:-

- Comparison operators are used to compare the values.
- Comparison operators return either True/False.

Operator	Meaning
>	Greater than – true if left operand is greater than right one.
<	Less than – true if left operand is less than right one.
==	Equal to – True if both operands are equal.
!=	Not equal to – True if both not equal
>=	Greater than (or) equal to – true if left operand is greater than or equal to right
<=	Less than (or) equal to – True if left operand is less than or equal to right.

Example Application 4:-

```
x=10
y=12
print('x>y is',x>y)
print('x<y is',x<y)
print('x==y is',x==y)
print('x!=y is',x!=y)
print('x>=y is',x>=y)
print('x<=y is',x<=y)
```

```
O/P:-
x>y is False
x<y is True
x==y is False
x!=y is True
x>=y is False
x<=y is True
```

Example Application 5:-

```
x='softpro'
y='softpro'
print('x>y is',x>y)
print('x<y is',x<y)
print('x==y is',x==y)
print('x!=y is',x!=y)
print('x>=y is',x>=y)
print('x<=y is',x<=y)
```

```
O/P:-
x>y is False
x<y is False
x==y is True
x!=y is False
x>=y is True
x<=y is True
```

Logical Operator:-

- They are used to perform the logical operator like and, or, not.
- Logical operators return either True or False values.

Operator	Meaning
And	True if both operands are True
Or	True if either of the operand is True
Not	True if operand is False (complements the operand)

Example Application 6:-

```
x=True
y=False
print ('x and y is',x and y)
print ('x or y is',x or y)
print ('not x is',not x)

O/P:-
x and y is False
x or y is True
not x is False
```

Bitwise operator:-

These operators are used to perform the operations on the values based on the bits.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
^	Bitwise XOR
>>	Bitwise right shift
<<	Bitwise left shift

Example Application 7:-

```
x=10
y=4
print('x&y is : ',x&y)
print('x|y is : ',x|y)
print('~x is : ',~x)
print('x^y is : ',x^y)
print('x>>2 is : ',x>>2)
print('x<<2y is : ',x<<2)

O/P:-
x&y is 0
x|y is 14
~x is -11
x^y is14
x>>2 is 2
x<<y is 40
```

Note:- Bitwise operators internally converts the operands values in the form of binary format and performs the operation and gives the results in the form of decimal format.

Assignment Operators:-

- Assignment operators are used to assign the values to the variables.
- It contains both bitwise and arithmetic operators.

Operator	Example	Equivalent to
=	x=5	x=5
+=	x+=5	x=x+5
-=	x-=5	x=x-5
*=	x*=5	x=x*5
/=	x/=5	x=x/5
%=	x%=5	x=x%5
//=	x//=5	x=x//5

**=	x**=5	x=x**5
&=	x&=5	x=x&5
|=	x|=5	x=x|5
^=	x^=5	x=x^5
>>=	x>>=5	x=x>>5
<<=	x<<=5	x=x<<5

Example Application 8:-

```
x=10
print('x=10',x)
x+=5
print('x+=5 : ',x)
x-=5
print('x-=5 : ',x)
x*=5
print('x*=5 : ',x)
x**=5
print('x**=5 : ',x)
x/=5
print('x/=5 : ',x)
x%=5
print('x%=5 : ',x)
x//=5
print('x//=5 : ',x)
y=20
y&=10
print('y&=10 : ',y)
y/=30
print('y/=30 : ',y)
y^=10
print('y^=10 : ',y)
y>>=3
print('y>>=3 : ',y)
y<<=3
print('y<<=3 : ',y)
```

```
O/P:-
x=10 : 10
x+=5 : 15
x-=5 : 10
x*=5 : 50
x**=5 : 312500000
x/=5 : 62500000
x%=5 : 40
x//=10 : 0
```

```
y&=10 : 0
y|=30 : 30
y^=10 : 20
y>>=3 : 2
y<<=3 : 16
```

Special Operators:-

Python supports the two types of special operators:-

1. Identity Operators
2. Membership Operators

Identity operators:-

- Identity operators are used to compare the addresses of the memory locations which are pointed by the operands.
- Identity operators' return true (or) false.

Operator	Meaning
Is	True if the operands are identical (refer to the same object)
is not	True if the operands are not identical (do not refer to the same object)

Example Application 9:-

```
x1=5
y1=5
x2='hello'
y2='hello'
print('x1 is y1 : ',x1 is y1)
print('x1 is not y1 : ',x1 is not y1)
print('x2 is y2 : ',x2 is y2)
print('x2 is not y2 : ',x2 is not y2)
```

```
O/P:-
x1 is y1 : True
x1 is not y1 : False
x2 is y2 : True
x2 is not y2 False
```

Membership Operators:-

Membership operators are used to search for a particular element is a string, list, tuple, set and so on. Relationship operators return True or False values.

Operator	Meaning
In	True if value/ variable is found in sequence
not in	True if value/ variable is not found in sequence

Example Application 10:-

```
x='hello world'
print('h' in x)
print('h' not in x)
print('hello' in x)
print('hello' not in x)

O/P:-
True
False
True
False
```

Operator's precedence:-

Operator	Description
**	Exponentiation (raise to the power)
~+-	Complement, unary plus and minus
*/%//	Multiply, divide, modulo and floor division
+-	Addition and subtraction
>> <<	Right shift and left bitwise shift
<= < > >=	Comparison operators
== !=	Equality operator
= %= /= -=	Assignment operator
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Example Application 11:-

```
a=20
b=10
c=15
d=5
e=0
e=(a+b)*c/d
print ('value of (a+b)*c/d is',e)
e=((a+b)*c)/d
print ('value of ((a+b)*c)/d is',e)
e=(a+b)*(c/d)
print ('value of (a+b)*(c/d) is',e)
e=a+(b*c)/d
print ('value of a+(b*c)/d is',e)
```

```
O/P:-
value of (a+b)*c/d is 90
value of ((a+b)*c)/d is 90
value of (a+b)*(c/d) is 90
value of a+(b*c)/d is 50
```

Short Answer Questions:-

1. **What is difference between programming language and scripting language?**

Difference between Programming and Scripting Languages:-

Scripting Language	Programming Language
1. Scripting language are interpreter based languages.	1. Programming language are compiler based languages.
2. Scripting language programs (or) applications explicit compilation is not required.	2. It requires explicit compilation.
3. Scripting language programs (or) applications directly we can run.	3. Programming language programs (or) applications we can't run without compilation.
4. Scripting language programs (or) applications take larger time to execute.	4. Programming language programs (or) applications take less time to execute.
5. Ex. Shell script, perl…	5. C, C++, Java,…..

2. **What are Python identifiers?**

Python Identifiers:-

An identifier in any programming language is the name given to identify a variable, function, class, module or another object. In Python language, an identifier begins with an alphabetic letter A to Z or a to z or an underscore (_) followed by zero or more alphabetic letters, underscores and digits (0 to 9).
Python programming language does not allow special characters such as @, $, /, and % within identifiers. Python is a case sensitive programming language. Therefore, identifiers such as 'Python' and 'python' are two different identifiers in Python programming language.
Below are the naming conventions for identifiers in Python.

- Class name in Python always begins with an uppercase letter and all other Python identifiers starts with a lowercase letter.
- A Python identifier is private when such identifier begins with a single leading underscore.
- A Python identifier is strongly private when such identifier begins with two leading underscores.
- A Python identifier is a language-defined special name when such identifier ends with two trailing underscores.

3. **What are Python reserved words?**

Python Reserve Words:-

Reserve words in any programming language are special commands that compiler or interpreters understands, and these reserve words cannot be used as a constant or variable, or any other identifier names in that programming language.
Python has the following reserve words, and all such keywords contain lowercase letters only.

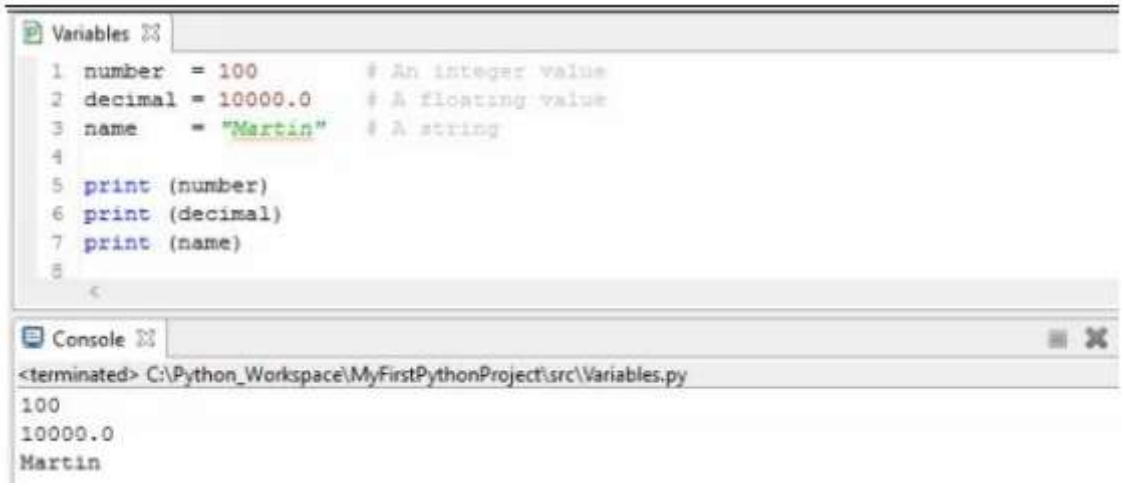
and	def	exec	if	not	return
assert	del	finally	import	or	try
break	elif	for	in	pass	while
class	else	from	is	print	with
continue	except	global	lambda	raise	yield

4. How many types of variables in Python?

Variables in any programming language are the names of the reference to the reserved memory locations which are used to store values. Similarly, when we are creating a variable in Python then we are reserving some space in the memory.
These variables have their own data type. Based on the type, the interpreter allocates memory and decides what kind of data can be stored in these reserved memory locations.
Characters, integers, decimals, etc. are the different data types which can be assigned to such variables.

Assigning Values to Variables

In the Python language, equal sign (=) is used to assign values to the variables. Such variables do not need explicit declaration. When we assign a value to a variable, the declaration or creation happens automatically.
The operand to the left of the equal sign (=) is the name of the variable and the operand to the right of the equal sign (=) is the value stored in the variable. This is demonstrated in the below example.



In the above example, the variable name 'number' has an integer value therefore, it behaves as an integer without any data type declaration. Similarly, the variable name 'decimal' has a floating value and variable name 'name' has a string value. Python is a very flexible language since it automatically determines the data type once the value is assigned to the variable.

Technical Tasks:-

1. **Write a Python program to print a message on screen.**

```
print("Hello Python")

O/P:-
Hello Python
```

2. **Write a Python program to make a simple calculator.**

```
a=int(input("Enter first number : "))
b=int(input("Enter second number : "))
print("Summation=",(a+b))
print("Subtraction=",(a-b))
print("Multiplication=",(a*b))
print("Division=",(a/b))

O/P:-
Enter first number : 10
Enter second number : 2
Summation=12
Subtraction=8
Multiplication=20
```

```
Division=5.0
```

3. **Write a python program to find volume and surface area of cuboid.**

```
l=int(input("Enter length of cuboid : "))
b=int(input("Enter breadth of cuboid : "))
h=int(input("Enter height of cuboid : "))
v=l*b*h
sa=2*(l*b+b*h+h*l)
print("Volume of cuboid=",v)
print("Surface Area of cuboid=",sa)

O/P:-
Enter length of cuboid : 10
Enter breadth of cuboid : 5
Enter height of cuboid : 4
Volume of cuboid=200
Surface Area of cuboid=220
```

4. **Write a program in python to find area and perimeter of circle.**

```
r=float(input("Enter radius of circle : "))
a=3.14*r*r
p=2*3.14*r
print("Area of circle=",a)
print("Perimeter of circle=",p)
O/P:-
Enter radius of circle : 5
```

```
Area of circle= 78.5
Perimeter of circle= 31.400
```

5. **Write a program in python to calculate gross salary. In this program take basic salary as input then calculate HRA and DA. HRA should be 10% of basic salary and DA should be 50% of basic salary. Then find gross salary as sum of basic salary, HRA and DA.**

```
bs=int(input("Enter basic salary : "))
hra=bs*10/100
da=bs*50/100
gs=bs+hra+da
print("Basic Salary=",bs)
print("House Rent Allowances=",hra)
print("Dearness Allowances=",da)
print("Gross Salary=",gs)

O/P:-
Enter basic salary : 5000
Basic salary=5000
House Rent Allowances=500
Dearness Allowances=2500
Gross Salary=8000
```

6. **Write a program in python to calculate simple interest.**

```
p=float(input("Enter value of p : "))
r=float(input("Enter value of r : "))
t=float(input("Enter value of t : "))
si=(p*r*t)/100
print("Simple Interest=",si)

O/P:-
```

```
Enter value of p : 10000
Enter value of r : 5
Enter value of t : 5
Simple Interest=2500.0
```

7. **Write a program in python to find area and perimeter of rectangle.**

```
l=int(input("Enter length of rectangle : "))
b=int(input("Enter breadth of rectangle : "))
a=l*b
p=2*(l+b)
print("Area of rectangle=",a)
print("Perimeter of rectangle=",p)
O/P:-
Enter length of rectangle : 10
Enter breadth of rectangle : 5
Area of rectangle=50
Perimeter of rectangle=30
```

Interview Questions:-

1. **What is python?**

   Python is an interpreted, object oriented, high level programming language with dynamic semantics.

2. **What are different features of Python?**

   Features of Python are given below:-

   i. Python is an interpreted language.

   ii. Python is object oriented language

iii. Python is a multipurpose programming language.

iv. Python supports dynamic semantics.

v. Python is an open source programming language.

vi. Python supports automatic garbage collection.

vii. Write less line of code then other language.

3. What is difference between programming language and scripting language?

Scripting Language	Programming Language
6. Scripting language are interpreter based languages.	6. Programming language are compiler based languages.
7. Scripting language programs (or) applications explicit compilation is not required.	7. It requires explicit compilation.
8. Scripting language programs (or) applications directly we can run.	8. Programming language programs (or) applications we can't run without compilation.
9. Scripting language programs (or) applications take larger time to execute.	9. Programming language programs (or) applications take less time to execute.
10. Ex. Shell script, perl…	10. C, C++, Java,…..

4. What is variable?

Variables are value containers, they create space in memory.

5. What is keyword?

Keywords are reserved word. The meaning of keywords is pre-defined into compiler.

6. What is operator?

Operators are the symbols which are used to perform operations on operands.

7. What is object oriented programming system?

Object oriented programming system is a mechanism of software development.

8. What is open source?

The source code of open source technology is open for all software vendors. A software engineer can make changes in source code of open source technology.

Multiple choice Questions Set - 1:-

1. Python is a language of types:
    a) Interpreted
    b) Object Oriented
    c) High Level
    d) All of the above
2. Python was developed by:
    a) James Gosling
    b) Denis Ritchie
    c) Jack Ma
    d) Guido Van Rossum

3. Is Python case sensitive when dealing with identifiers?
    a) no
    b) yes
    c) machine dependent
    d) none of the mentioned
4. Which of the following is the correct extension of the Python file?
    a) .python
    b) .pl
    c) .py
    d) .p

Answer Key:-

1. D	2. D	3. A	4. c

Multiple choice questions Set - 2:-

1. Select the reserved keyword in python-
    A. else
    B. raise
    C. import
    D. All of the above
2. Which is the special symbol used in python to add comments?
    A. $
    B. //
    C. /*.... */
    D. #
3. Which is not a valid keyword in python?
    A. if
    B. else
    C. switch
    D. elif
4. All keywords in Python are in

A. lower case
B. UPPER CASE
C. Capitalized
D. None of the mentioned

Answer key:-

1. D	2. D	3. C	4. D

Lecture – 2 (Flow Controls In Python)

Long Answer Question:-

1. **Describe decision controls in Python?**

Conditional Statements:-

- Conditional statements are used to decide whether code has to be execute (or) skip based on the evaluation of the condition.
- After evaluating the condition, it should return either True or False value.

Elements of conditional statements:-

In conditional statements we use two elements:-

1. Condition
2. Block

Condition:- Any expression which returns either True (or) False value after evaluating that expression is known as condition.

Block:- All the statements which are following same space indentation is known as "Block".

- Blocks begin with when the indentation increases.
- Blocks can contain other blocks.
- Blocks end when the indentation decreases to zero (or) to a containing blocks indentation.

Python supports three conditional statements, There are,

1. if (or) simple if
2. if else
3. elif

if (or) simple if:-

syntax:-

if condition : statement

(or)

if condition:

stmt 1

stmt 2

Simple if executes the block if condition returns True. Otherwise it will skip the execution of the block.

if – else:-

Syntax:-

```
if condition :
        stmt1
        stmt2
else:
        stmt3
        stmt4
```

- In if-else condition returns True then it will execute if block otherwise it will execute else block.

elif:-

syntax:-

```
if condition:
        stmt1
        stmt2
elif condition:
        stmt3
        stmt4
else:
        stmt5
        stmt6
```

3. **What is Loop Controls? Describe it in brief.**

Looping statements:-

- Looping statements are used to execute the set of statements repeatedly.
- Python supports 2 types of looping statements.
  i.) for loop
  ii.) while loop

for loop:- for loop executes the set of statements (or) blocks with respect to every, element of the string/ collection objects.

syntax:-

for variablename in string / collection variable:

……

……..

range():- range() generate the group of values based on the given range and gives it as a list.

Short Answer Questions:-

1. **What is difference between if and if-else?**

   If is a keyword which works as decision control, we attach a condition with if statement, if given condition is true then if block code will executed and if given condition is false then it do nothing.
   If – else is variation of if statement, we attach a condition with if statement, if given condition is true then if block code will executed and if given condition is false then else block code will executed.

2. **What is ladder if else?**

   If you have many conditions and you want to execute block of code based on those conditions then you can use ladder if – else.

   Syntax of ladder if – else is given below:-

```
if condition1:
   #code1
elif condition2:
   #code2
else:
   #code3
```

3. **Describe while loop in Python?**

   While is a loop control in Python. It is an entry loop control. The syntax of while loop is given below:-

```
Initialization of loop counter
While condition:
   #code
   Updation of loop counter
```

4. **What is for loop in Python?**

   For loop in python is used to traverse elements of a collection. The syntax of for loop is given below:-

```
for variable_name in String / collection_name:
   #code
```

Technical Tasks:-

1. **Write a program in python to demonstrate concept of if.**

```
name=input('enter name : ')
if name == 'brijesh'
        print('Hi',name)
print ('Hello World')

O/P1:-
enter name : brijesh
Hi, brijesh
O/P2:-
enter name : ravi
Hello World
```

2. **Write a program in python to demonstrate concept of if – else.**

```
name=input('enter name : ')
if name=='brijesh':
 print('Hi',name)
else:
 print('Hi, Stranger')
```

```
print('Hello World')

O/P1:-
enter name : brijesh
Hi  brijesh
Hello World

O/P2:-
enter name : 'rohit'
Hi, Stranger
Hello World
```

3. Write a program in python to demonstrate concept of ladder if – else.

```
name=input('enter name : ')
age=int(input('enter age : '))
if age<15:
 print (name,'you are kid')
elif age<40:
 print (name,'you are young')
elif age<100:
 print (name,'you are old')
else:
 print (name,'you are alian')
print ('Hello, World')

O/P1:-
enter name : tom
enter age : 12
tom, you are kid
Hello, World

O/P2:-
enter name : priya
enter age : 20
priya, you are young
Hello, World
```

4. WAP to find biggest number in two numbers.

```
num1=int(input('Enter first no : '))
num2=int(input('Enter second no : '))
if num1>num2:
```

```
 print (num1,'is big')
else:
 print (num2,'is big')

O/P1:-
Enter first no : 100
Enter second no : 50
100 is big

O/P2:-
Enter first no : 100
Enter second no : 200
200 is big
```

5. WAP to find biggest number in three numbers.

```
num1=int(input('Enter first number : '))
num2=int(input('Enter second number : '))
num3=int(input('Enter third number : '))
if num1>num2 and num1>num3:
 print (num1,'is big')
elif num2>num3:
 print (num2,'is big')
else:
 print (num3,'is big')


O/P:-
Enter first number : 100
Enter second number : 500
Enter third number : 200
500 is big.
```

6. WAP to check given number is even or odd.

```
n=int(input("Enter a number : "))
if n%2==0:
      print("Number is even")
else:
     print("Number is odd")

Output 1:-
Enter a number : 4
```

```
Number is even
Output 2:-
Enter a number : 5
Number is odd
```

7. Write a python program to calculate gross salary on following basis.

Basic Salary	HRA	DA
1-4000	10%	50%
4001-8000	20%	60%
8001-12000	25%	75%
More than 12000	30%	80%

```
bs=int(input("Enter basic salary : "))
if bs<=4000:
    hra=bs*10/100
    da=bs*50/100
elif bs>4000 and bs<=8000:
    hra=bs*20/100
    da=bs*60/100
elif bs>8000 and bs<=12000:
    hra=bs*25/100
    da=bs*75/100
else:
    hra=bs*30/100
    da=bs*80/100
gs=bs+hra+da
print("Basic Salary=",bs)
print("HRA=",hra)
print("DA=",da)
print("Gross Salary=",gs)

O/P:-

Output:-
Enter basic salary : 5000
Basic Salary=5000.0
HRA=1000.0
DA=3000.0
Gross Salary=9000.0
```

8. Write a Python program to calculate electricity bill on following basis:-

Unit	Bill/unit
1-150	2.40/unit
For next 151-300	3.00/unit
For next more than 300	3.20/unit

```
unit=int(input("Enter number of units consumed : "))
if unit<=150:
    bill=unit*2.40
elif unit>150 and unit<=300:
    bill=(150*2.40)+(unit-150)*3.00
else:
    bill=(150*2.40)+(150*3.00)+(unit-300)*3.20
print("Your bill=",bill)


Output:-
Enter number of units consumed : 200
Your bill=510.0
```

9. Write a Python program to find factorial of given number.

```
n=int(input("Enter a number to find factorial : "))
f=1
while n>0:
   f=f*n
   n=n-1
print("Factorial=",f)


O/P:-
Enter a number to find factorial : 5
Factorial=120
```

10. Write a Python program to find sum of digits of given number.

```
num=int(input("Enter a number to find sum of digits : "))
s=0
while num>0:
   rem=num%10
   s=s+rem
   num=num//10
print("Sum of digits=",s)

O/P:-
```

```
Enter a number to find sum of digits : 123
Sum of digits=6
```

11. Write a Python program to reverse the digits of given number.

```
num=int(input("Enter a number : "))
rev=0
while num>0:
   rem=rem%10
   rev=rev*10+rem
   num=num//10
print("Reverse No=",rev)

O/P:-
Enter a number : 123
Reverse No=321
```

12. Write a Python program to check given number if prime or not.

```
n=int(input("Enter a number : "))
i=1
c=0
while i<=n:
   if n%i==0:
      c=c+1
   i=i+1
if c==2:
   print(n,"is prime")
else:
   print(n,"is not prime")

O/P:-
Enter a number : 5
5 is prime
```

13. Write a Python program to print table of given number.

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

O/P:-
Enter a number : 5
5*1=5
5*2=10
5*3=15
5*4=20
```

```
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
5*10=50
```

14. Write a Python program to generate Fibonacci sequence up to n terms.

```
n=int(input("How many terms you want in series? "))
n1=0
n2=1
print("Fibonacci Series")
print(n1)
print(n2)
for i in range(1,n-1):
   n3=n1+n2
   n1=n2
   n2=n3

O/P:-
How many terms you want in series? 5
0
1
1
2
3
```

Interview Questions:-

1. **What do you understand by flow controls?**
   If you want to execute code based on some condition then you can use flow controls.
2. **What is decision control?**
   If you want to execute code based on some condition then you can use decision controls. Decision Controls are used for decision making.
3. **What is loop control?**
   If you have a block of code which you want to execute repeatedly up to given condition is true then you can use a loop control.
4. **What is difference between while loop and for loop?**
   While is an entry loop control. For loop in Python is used to traverse elements of a collection.
5. **Describe for loop in Python.**
   For loop in python is used to traverse elements of a collection.

6. **Write syntax of for loop in Python.**
   Syntax of for loop in python:-
   for variable_name in string/collection_name:
       #code
7. **What is range() function in python?**
   range() function in python is used to generate numbers in given range.

Lecture – 3 (Collection in Python)

Long Answer Questions:-

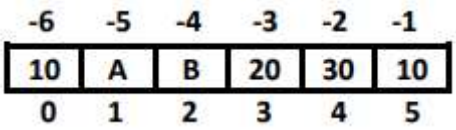
1. **What is list collection? Describe it.**

If we want to represent a group of individual objects as a single entity where insertion order preserved and duplicates are allowed, then we should go for List.

- Insertion order preserved.
- Duplicate objects are allowed
- Heterogeneous objects are allowed.
- List is dynamic because based on our requirement we can increase the size and decrease the size.
- In List the elements will be placed within square brackets and with comma separator.
- We can differentiate duplicate elements by using index and we can preserve insertion order by using index. Hence index will play very important role.

Python supports both positive and negative indexes. +ve index means from left to right whereas negative index means right to left

[10,"A","B", 20, 30, 10]



List objects are mutable. i.e. we can change the content.

2. **What is tuple collection? Describe it.**

1. Tuple is exactly same as List except that it is immutable. i.e once we creates Tuple object, we cannot perform any changes in that object.

Hence Tuple is Read only version of List.

2. If our data is fixed and never changes then we should go for Tuple.

3. Insertion Order is preserved

4. Duplicates are allowed

5. Heterogeneous objects are allowed.

6. We can preserve insertion order and we can differentiate duplicate objects by using index. Hence index will play very important role in Tuple also.

Tuple support both +ve and -ve index. +ve index means forward direction(from left to right) and -ve index means backward direction(from right to left)

7. We can represent Tuple elements within Parenthesis and with comma seperator.

Parenthesis are optional but recommended to use.

3. **What is set collection? Describe it.**

Important points of set collection are given below:-

- If we want to represent a group of unique values as a single entity then we should go for set.
- Duplicates are not allowed.
- Insertion order is not preserved. But we can sort the elements.
- Indexing and slicing not allowed for the set.
- Heterogeneous elements are allowed.
- Set objects are mutable i.e once we creates set object we can perform any changes in
  that object based on our requirement.
- We can represent set elements within curly braces and with comma separation.
- We can apply mathematical operations like union, intersection, difference etc on set objects.

4. **What is dictionary collection? Describe it.**

Import points of dictionary collection are given below:-

- We can use List, Tuple and Set to represent a group of individual objects as a single entity.
- If we want to represent a group of objects as key-value pairs then we should go for Dictionary.

  Eg:

  rollno----name

  phone number--address

ipaddress---domain name

- Duplicate keys are not allowed but values can be duplicated.
- Heterogeneous objects are allowed for both key and values.
- Insertion order is not preserved.
- Dictionaries are mutable.
- Dictionaries are dynamic.
- Indexing and slicing concepts are not applicable.

Note: In C++ and Java Dictionaries are known as "Map" where as in Perl and Ruby it is known as "Hash".

How to create Dictionary?

d={} or d=dict()

We are creating empty dictionary. We can add entries as follows:-

```
d[100]="John"
d[200]="Black"
d[300]="White"
print(d) #{100: 'John', 200: 'Black', 300: 'White'}
```

If we know data in advance then we can create dictionary as follows

```
d={100:'John' ,200:'Black', 300:'White'}
```

d={key:value, key:value}

How to access data from the dictionary?

We can access data by using keys.

```
d={100: 'John', 200: 'Black', 300: 'White'}
print(d[100]) #John
print(d[300]) #White
```

If the specified key is not available then we will get KeyError

```
print(d[400]) # KeyError: 400
```

Short Answer Questions:-

1. **What is difference between list and tuple?**

List	Tuple
1) List is a Group of Comma separated Values within Square Brackets and Square Brackets are mandatory. Eg: i = [10, 20, 30, 40]	1) Tuple is a Group of Comma separated Values within Parenthesis and Parenthesis are optional. Eg: t = (10, 20, 30, 40) t = 10, 20, 30, 40
2) List Objects are Mutable i.e. once we creates List Object we can perform any changes in that Object. Eg: i[1] = 70	2) Tuple Objects are Immutable i.e. once we creates Tuple Object we cannot change its content. t[1] = 70 ==> ValueError: tuple object does not support item assignment.
3) If the Content is not fixed and keep on changing then we should go for List.	3) If the content is fixed and never changes then we should go for Tuple.
4) List Objects can not used as Keys for	4) Tuple Objects can be used as Keys for

Dictionaries because Keys should be Hashable and Immutable.	Dictionaries because Keys should be Hashable and Immutable.

2. **How to create empty list object?**

```
list=[]
print(list)
print(type(list))
```

Output:-

```
[]
<class 'list'>
```

3. **How to create a list with dynamic input?**

```
list=eval(input("Enter List:"))
print(list)
print(type(list))
```

Output:-

```
Enter List:[10,20,30,40]
[10, 20, 30, 40]
<class 'list'>
```

4. **How to create list with split function?**

```
s="Learning Python is very very easy !!!"
l=s.split()
print(l)
print(type(l))
```

Output:-

```
['Learning', 'Python', 'is', 'very', 'very', 'easy', '!!!']
<class 'list'>
```

5. **How to access list elements?**

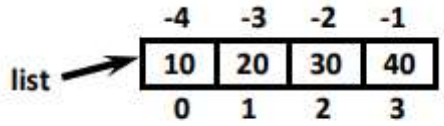
We can access elements of the list either by using index or by using slice operator(:)

1. By using index:

- List follows zero based index. ie index of first element is zero.
- List supports both +ve and -ve indexes.

- +ve index meant for Left to Right
- -ve index meant for Right to Left

list=[10,20,30,40]



print(list[0]) ==>10

print(list[-1]) ==>40

print(list[10]) ==>IndexError: list index out of range

2. By using slice operator:

Syntax:

list2= list1[start:stop:step]

start ==>it indicates the index where slice has to start default value is 0

stop ===>It indicates the index where slice has to end default value is max allowed index of list i.e. length of the list

step ==>increment value default value is 1

6. How to traverse list elements?

The sequential access of each element in the list is called traversal.

By using while loop.

```
n=[0,1,2,3,4,5,6,7,8,9,10]
i=0
while i<len(n):
    print(n[i],end=" ")
    i=i+1
```

Output:-

0 1 2 3 4 5 6 7 8 9 10

By using for loop

```
n=[0,1,2,3,4,5,6,7,8,9,10]
for n1 in n:
    print(n1, end=" ")
```

Output:-

0 1 2 3 4 5 6 7 8 9 10

7. How to get information about list?

1. len():- Returns the number of elements present in the list
Eg: n=[10,20,30,40]
print(len(n))==>4
2. count():- It returns the number of occurrences of specified item in the list.

```
n=[1,2,2,2,2,3,3]
print(n.count(1))
print(n.count(2))
print(n.count(3))
print(n.count(4))
```

Output:-

1
4
2
0

3. index():- Returns the index of first occurrence of the specified item.
Eg:

```
n=[1,2,2,2,2,3,3]
print(n.index(1)) ==>0
print(n.index(2)) ==>1
print(n.index(3)) ==>5
print(n.index(4)) ==>ValueError: 4 is not in list
```

8. How to manipulate list?

1. append():- We can use append() function to add item at the end of the list.

```
list=[]
list.append("A")
list.append("B")
list.append("C")
print(list)
```

Output:-

['A', 'B', 'C']

To add all elements to list upto 100 which are divisible by 10.

```
list=[]
for i in range(101):
    if i%10==0:
        list.append(i)
print(list)
```

Output:-
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

2. insert():-To insert item at specified index position.

```
n=[1,2,3,4,5]
n.insert(1,888)
print(n)
```

Output:-
[1, 888, 2, 3, 4, 5]

Differences between append() and insert():-

append()	insert()
In List when we add any element it will come in last i.e. it will be last element.	In List we can insert any element in particular index number

3. extend() function:- To add all items of one list to another list
l1.extend(l2)
All items present in l2 will be added to l1

```
order1=["Dosa","Pizza","Burger"]
order2=["RC","KF","FO"]
order1.extend(order2)
print(order1)
```

Output:-
['Dosa', 'Pizza', 'Burger', 'RC', 'KF', 'FO']

4. remove() function:- We can use this function to remove specified item from the list. If the item present multiple times then only first occurrence will be removed.

```
n=[10,20,10,30]
n.remove(10)
print(n)
```

Output:-
[20, 10, 30]
If the specified item not present in list then we will get ValueError

```
n=[10,20,10,30]
n.remove(40)
print(n)
```

ValueError: list.remove(x): x not in list
Note: Hence before using remove() method first we have to check specified element present in the list or not by using in operator.

5. pop() function:- It removes and returns the last element of the list. This is only function which manipulates list and returns some element.

```
n=[10,20,30,40]
print(n.pop())
print(n.pop())
print(n)
```

Output:-

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

If the list is empty then pop() function raises IndexError

Eg:

```
n=[]
print(n.pop()) ==> IndexError: pop from empty list
```

Note:

1. pop() is the only function which manipulates the list and returns some value.
2. In general we can use append() and pop() functions to implement stack data structure by using list, which follows LIFO(Last In First Out) order.

In general we can use pop() function to remove last element of the list. But we can use to remove elements based on index.

n.pop(index)===>To remove and return element present at specified index.

n.pop()==>To remove and return last element of the list.

```
n=[10,20,30,40,50,60]
print(n.pop()) #60
print(n.pop(1)) #20
print(n.pop(10)) ==>IndexError: pop index out of range
```

Differences between remove() and pop():-

remove()	pop()
1) We can use to remove special element from the List.	1) We can use to remove last element from the List.
2) It can't return any value.	2) It returned removed element.
3) If special element not available then we get VALUE ERROR.	3) If List is empty then we get Error.

Note:

List objects are dynamic. i.e based on our requirement we can increase and decrease the size.

append(),insert() ,extend() ===>for increasing the size/growable nature

remove(), pop() ======>for decreasing the size /shrinking nature

9. How to make ordering of list?

1. reverse():- We can use to reverse() order of elements of list.

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

2. sort():- In list by default insertion order is preserved. If want to sort the elements of list according to default natural sorting order then we should go for sort() method.

For numbers ==>default natural sorting order is Ascending Order

For Strings ==> default natural sorting order is Alphabetical Order

```
n=[20,5,15,10,0]
n.sort()
print(n) #[0,5,10,15,20]
s=["Dog","Banana","Cat","Apple"]
s.sort()
print(s) #['Apple','Banana','Cat','Dog']
```

Note: To use sort() function, compulsory list should contain only homogeneous elements. otherwise we will get TypeError.

```
n=[20,10,"A","B"]
n.sort()
print(n)
TypeError: '<' not supported between instances of 'str' and 'int'
```

10. How to access elements of a tuple?

We can access either by index or by slice operator

1. By using index:-

```
t=(10,20,30,40,50,60)
print(t[0]) #10
print(t[-1]) #60
print(t[100]) IndexError: tuple index out of range
```

2.By using slice operator

```
t=(10,20,30,40,50,60)
print(t[2:5])
print(t[2:100])
print(t[::2])
```

Output:-

```
(30, 40, 50)
(30, 40, 50, 60)
(10, 30, 50)
```

11. How to create set object?

```
s={10,20,30,40}
print(s)
```

```
print(type(s))
```

Output:-

```
{40, 10, 20, 30}
<class 'set'>
```

We can create set objects by using set() function

```
s=set(any sequence)
l = [10,20,30,40,10,20,10]
s=set(l)
print(s) # {40, 10, 20, 30}

s=set(range(5))
print(s) #{0, 1, 2, 3, 4}
```

Note: While creating empty set we have to take special care. Compulsory we should use set() function.

s={} ==>It is treated as dictionary but not empty set.

12. What are import functions of set?

1. add(x): Adds item x to the set

```
s={10,20,30}
s.add(40)
print(s) #{40, 10, 20, 30}
```

2. update(x,y,z):

- To add multiple items to the set.
- Arguments are not individual elements and these are iterable objects like List, range etc.
- All elements present in the given iterable objects will be added to the set.

```
s={10,20,30}
l=[40,50,60,10]
s.update(l, range(5))
print(s)
```

Output:

```
{0, 1, 2, 3, 4, 40, 10, 50, 20, 60, 30}
```

3. copy():-Returns copy of the set. It is cloned object.

```
s={10,20,30}
s1=s.copy()
print(s1)
```

4. pop():- It removes and returns some random element from the set.

```
s={40,10,30,20}
print(s)
print(s.pop())
print(s)
```

Output

```
{40, 10, 20, 30}
40
{10, 20, 30}
```

5. remove(x):- It removes specified element from the set. If the specified element not present in the Set then we will get KeyError.

```
s={40,10,30,20}
s.remove(30)
print(s) # {40, 10, 20}
s.remove(50) ==>KeyError: 50
```

6. discard(x):- It removes the specified element from the set. If the specified element not present in the set then we won't get any error.

```
s={10,20,30}
s.discard(10)
print(s) ===>{20, 30}
s.discard(50)
print(s) ==>{20, 30}
```

7.clear():- To remove all elements from the Set.

```
s={10,20,30}
print(s)
s.clear()
print(s)
```

Output

```
{10, 20, 30}
set()
```

13. What is the difference between add() and update() functions in set?

We can use add() to add individual item to the Set, where as we can use update() function to add multiple items to Set.
add() function can take only one argument where as update() function can take any number of arguments but all arguments should be iterable objects.

14. How to create dictionary in Python?

d={} or d=dict()

We are creating empty dictionary. We can add entries as follows:-

d[100]="John"
d[200]="Black"
d[300]="White"
print(d) #{100: 'John', 200: 'Black', 300: 'White'}
If we know data in advance then we can create dictionary as follows
d={100:'John' ,200:'Black', 300:'White'}
d={key:value, key:value}

15. How to access data from dictionary?

We can access data by using keys.
d={100: 'John', 200: 'Black', 300: 'White'}
print(d[100]) #John
print(d[300]) #White
If the specified key is not available then we will get KeyError
print(d[400]) # KeyError: 400
We can prevent this by checking whether key is already available or not by using has_key() function or by using in operator.
d.has_key(400) ==> returns 1 if key is available otherwise returns 0
But has_key() function is available only in Python 2 but not in Python 3. Hence compulsory we have to use in operator.

```
if 400 in d:
    print(d[400])
```

Technical Tasks:-

1. **WAP to create a list by taking input from user and display list elements.**

```
list1=[]
n=int(input("How many elements you want in list? "))
print("Enter",n,"elements")
for i in range(n):
   e=eval(input())
   list1.insert(i,e)
print("You have entered following elements")
for e in list1:
   print(e)
```

O/P:-
How many elements you want in list? 5
Enter 5 elements
10
20
30

```
"A"
"B"
You have entered following elements
10
20
30
A
B
```

2. **WAP to create a list of five numbers and find maximum and minimum number.**

```
list1=[10,20,30,40,50]
print("Maximum Number=",max(list1))
print("Minimum Number=",min(list1))

O/P:-
Maximum Number=50
Minimum Number=10
```

3. **WAP to create a list with five names now display names in ascending and descending order.**

```
list1=[]
print("Enter five names to the list")
for c in range(5):
   name=input()
   list1.insert(c,name)
list1.sort()
print("Names in ascending order")
for n in list1:
   print(n)
list1.reverse()
print("Names in descending order")
for n in list1:
   print(n)
```

4. **WAP to demonstrate concept of tuple.**

```
tup1=(10,20,30,40,50)
print(tup1)
print("Length=",len(tup1))
print("Sum=",sum(tup1))
print("Maximum value=",max(tup1))
print("Minimum value=",min(tup1))
```

5. **Write a program to enter name and percentage marks in a dictionary and display information on the screen.**

```
rec={}
n=int(input("Enter number of students: "))
i=1
while i <=n:
name=input("Enter Student Name: ")
marks=input("Enter % of Marks of Student: ")
rec[name]=marks
i=i+1
print("Name of Student","\t","% of marks")
for x in rec:
print("\t",x,"\t\t",rec[x])
```

Output:

```
Enter number of students: 3
Enter Student Name: durga
Enter % of Marks of Student: 60%
Enter Student Name: ravi
Enter % of Marks of Student: 70%
Enter Student Name: shiva
Enter % of Marks of Student: 80%
Name of Student % of marks
durga                60%
ravi                 70 %
shiva                80%
```

6. **Write a program to find number of occurrences of each vowel present in the given string?**

```
word=input("Enter any word: ")
vowels={'a','e','i','o','u'}
d={}
for x in word:
        if x in vowels:
                d[x]=d.get(x,0)+1
for k,v in sorted(d.items()):
        print(k,"occurred ",v," times")
```

Output:

```
Enter any word: doganimaldoganimal
a occurred 4 times
i occurred 2 times
o occurred 2 times
```

Interview Questions:-

1. **What is list in python?**
   List is a collection in python which is used to store multiple values of different data types.
2. **What is tuple in python?**
   Tuple is also a collection in python which is used to store multiple values of different data types. Tuple is immutable whereas list is mutable.
3. **What is dictionary in python?**
   Dictionary is a collection in python which stores elements in key and value pair.
4. **What is difference between list and tuple?**

List	Tuple
1) List is a Group of Comma separated Values within Square Brackets and Square Brackets are mandatory. Eg: i = [10, 20, 30, 40]	1) Tuple is a Group of Comma separated Values within Parenthesis and Parenthesis are optional. Eg: t = (10, 20, 30, 40) t = 10, 20, 30, 40
2) List Objects are Mutable i.e. once we creates List Object we can perform any changes in that Object. Eg: i[1] = 70	2) Tuple Objects are Immutable i.e. once we creates Tuple Object we cannot change its content. t[1] = 70 ==> ValueError: tuple object does not support item assignment.
3) If the Content is not fixed and keep on changing then we should go for List.	3) If the content is fixed and never changes then we should go for Tuple.
4) List Objects can not used as Keys for Dictionaries because Keys should be Hashable and Immutable.	4) Tuple Objects can be used as Keys for Dictionaries because Keys should be Hashable and Immutable.

5. **What is difference between insert() and append() functions?**

   Differences between append() and insert():-

append()	insert()
In List when we add any element it will come in last i.e. it will be last element.	In List we can insert any element in particular index number

6. W**hat is difference between pop() and remove() function?**

Differences between remove() and pop():-

remove()	pop()
1) We can use to remove special element from the List.	1) We can use to remove last element from the List.
2) It can't return any value.	2) It returned removed element.
3) If special element not available then we get VALUE ERROR.	3) If List is empty then we get Error.

Multiple choice questions Set - 1:-

1. Which of the following commands will create a list?

a) list1 = list()
b) list1 = []
c) list1 = list([1, 2, 3])
d) all of the mentioned

2. Suppose listExample is ['h','e','l','l','o'], what is len(listExample)?

a) 5
b) 4
c) None
d) Error

3. Suppose list1 is [2445,133,12454,123], what is max(list1)?

a) 2445
b) 133
c) 12454
d) 123

4. Suppose list1 is [3, 5, 25, 1, 3], what is min(list1)?

a) 3
b) 5
c) 25
d) 1

5. Suppose list1 is [1, 5, 9], what is sum(list1)?

a) 1
b) 9
c) 15
d) Error

6. Which of the following is a Python tuple?

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

7. What is the data type of (1)?

a) Tuple
b) Integer
c) List
d) Both tuple and integer

Answer Key:-

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

Multiple choice questions Set - 2:-

1. Which of the following is a Python tuple?

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

2. Suppose t = (1, 2, 4, 3), which of the following is incorrect?

a) print(t[3])
b) t[3] = 45
c) print(max(t))
d) print(len(t))

3. What is the data type of (1)?

a) Tuple
b) Integer
c) List
d) Both tuple and integer

4. If a=(1,2,3,4), a[1:-1] is _____

a) Error, tuple slicing doesn't exist
b) [2,3]
c) (2,3,4)
d) (2,3)

5. What type of data is: a=[(1,1),(2,4),(3,9)]?

a) Array of tuples
b) List of tuples
c) Tuples of lists
d) Invalid type

Answer key:-

1. B	2. b	3. b	4. d	5. b

Multiple choice questions Set - 3:-

1. Which of these about a set is not true?

a) Mutable data type
b) Allows duplicate values
c) Data type with unordered values
d) Immutable data type

2. Which of the following is not the correct syntax for creating a set?

a) set([[1,2],[3,4]])
b) set([1,2,2,3,4])
c) set((1,2,3,4))
d) {1,2,3,4}

3. Which of the following statements is used to create an empty set?

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

4. If a={5,6,7,8}, which of the following statements is false?

a) print(len(a))
b) print(min(a))
c) a.remove(5)

d) a[2]=45

5. If a={5,6,7}, what happens when a.add(5) is executed?

a) a={5,5,6,7}
b) a={5,6,7}
c) Error as there is no add function for set data type
d) Error as 5 already exists in the set

6. What will be the output of the following Python code?

s=set()
type(s)
a) <'set'>
b) <class 'set'>
c) set
d) class set

7. Which of the following functions cannot be used on heterogeneous sets?

a) pop
b) remove
c) update
d) sum

Answer key:-

1. d	2. a	3. b	4. d	5. b	6. b	7. d

Multiple choice questions Set - 4:-

1. Which of the following statements create a dictionary?
a) d = {}
b) d = {"john":40, "peter":45}
c) d = {40:"john", 45:"peter"}
d) All of the mentioned

2. Suppose d = {"john":40, "peter":45}, to delete the entry for "john" what command do we use?
a) d.delete("john":40)
b) d.delete("john")
c) del d["john"]
d) del d("john":40)

3. Suppose d = {"john":40, "peter":45}. To obtain the number of entries in dictionary which command do we use?

a) d.size()
b) len(d)
c) size(d)
d) d.len()

4. Suppose d = {"john":40, "peter":45}, what happens when we try to retrieve a value using the expression d["susan"]?
a) Since "susan" is not a value in the set, Python raises a KeyError exception
b) It is executed fine and no exception is raised, and it returns None
c) Since "susan" is not a key in the set, Python raises a KeyError exception
d) Since "susan" is not a key in the set, Python raises a syntax error

5. Which of these about a dictionary is false?
a) The values of a dictionary can be accessed using keys
b) The keys of a dictionary can be accessed using values
c) Dictionaries aren't ordered
d) Dictionaries are mutable

6. Which of the following is not a declaration of the dictionary?
a) {1: 'A', 2: 'B'}
b) dict([[1,"A"],[2,"B"]])
c) {1,"A",2"B"}
d) { }

7. Which of the following isn't true about dictionary keys?
a) More than one key isn't allowed
b) Keys must be immutable
c) Keys must be integers
d) When duplicate keys encountered, the last assignment wins

8. Which of the statements about dictionary values if false?
a) More than one key can have the same value
b) The values of the dictionary can be accessed as dict[key]
c) Values of a dictionary must be unique
d) Values of a dictionary can be a mixture of letters and numbers

Answer Key:-

1. d	2. c	3. b	4. c	5. b	6. c	7. c	8. c

Lecture – 4 (Function and Module in Python)

Long Answer Questions:-

1. **What is function in Python? Describe it.**

If a group of statements is repeatedly required then it is not recommended to write these statements every time separately. We have to define these statements as a single unit and we can call that unit any number of times based on our requirement without rewriting.
This unit is nothing but function.
The main advantage of functions is code Reusability.
Note: In other languages functions are known as methods, procedures, subroutines etc.
Python supports 2 types of functions:-

1. Built in Functions
2. User Defined Functions

1. Built in Functions:

The functions which are coming along with Python software automatically, are called built in functions or pre defined functions
Eg:
id()
type()
input()

eval()
etc..

2. User Defined Functions:

The functions which are developed by programmer explicitly according to business requirements ,are called user defined functions.
Syntax to create user defined functions:

```
def function_name(parameters) :
        """ doc string"""
        ----
        -----
        return value
```

Note: While creating functions we can use 2 keywords
1. def (mandatory)
2. return (optional)

2. **What is module in Python? Describe it.**

A group of functions, variables and classes saved to a file, which is nothing but module.
Every Python file (.py) acts as a module.

Example Application : mymath.py

```
x=888
def add(a,b):
print("The Sum:",a+b)
def product(a,b):
print("The Product:",a*b)
```

mymath module contains one variable and 2 functions.
If we want to use members of module in our program then we should import that module.
import module_name

We can access members by using module name.
module_name.variable
module_name.function()

test.py:

```
import mymath
print(mymath.x)
mymath.add(10,20)
```

```
mymath.product(10,20)
```

Output:

```
888
The Sum: 30
The Product: 200
```

Note:
Whenever we are using a module in our program, for that module compiled file will be generated and stored in the hard disk permanently.

Renaming a module at the time of import (module aliasing):

Example Application :

```
import mymath as m
```
Here mymath is original module name and m is alias name.
We can access members by using alias name m.

test.py:

```
import mymath as m
print(m.x)
m.add(10,20)
m.product(10,20)
```

from ... import:

We can import particular members of module by using from ... import .
The main advantage of this is we can access members directly without using module name.

Example Application :

```
from mymath import x,add
print(x)
add(10,20)
product(10,20)==> NameError: name 'product' is not defined
```

We can import all members of a module as follows

```
from mymath import *
```

test.py:

```
from mymath import *
print(x)
add(10,20)
product(10,20)
```

Various possibilities of import:

```
import modulename
import module1,module2,module3
import module1 as m
import module1 as m1,module2 as m2,module3
from module import member
from module import member1,member2,member3
from module import member1 as x
from module import *
```

Member aliasing:

```
from mymath import x as y, add as sum
print(y)
sum(10,20)
```

Once we defined as alias name, we should use alias name only and we should not use original name.

Eg:

```
from mymath import x as y
print(x)==>NameError: name 'x' is not defined
```

Short Answer Questions:-

1. **What is parameter in python function?**

Parameters are inputs to the function. If a function contains parameters, then at the time of calling, compulsory we should provide values otherwise we will get error.

Example Application: Write a function to take name of the student as input and print wish message by name.

```
def wish(name):
        print("Hello",name," Good Morning")
```

```
wish("John")
wish("Kim")
```

Output:

```
Hello John Good Morning
Hello Kim Good Morning
```

2. **What about return statement in python?**

Function can take input values as parameters and executes business logic, and returns output to the caller with return statement.

Example Application: Write a function to accept 2 numbers as input and return sum.

```
def add(x,y):
    return x+y
result=add(10,20)
print("The sum is",result)
print("The sum is",add(100,200))
```

```
The sum is 30
The sum is 300
```

3. **How to return multiple values by a function?**

In other languages like C, C++ and Java, function can return at most one value. But in Python, a function can return any number of values.

Example Application:

```
def sum_sub(a,b):
        sum=a+b
        sub=a-b
        return sum,sub
x,y=sum_sub(100,50)
print("The Sum is :",x)
print("The Subtraction is :",y)
```

Output:

```
The Sum is : 150
The Subtraction is : 50
```

Technical Tasks:-

1. **WAP to find area and perimeter of rectangle using user defined function.**

```
def area(x,y):
  return (x+y)
def perimeter(x,y):
```

```
    return 2*(x+y)
l=int(input("Enter length of rectangle : "))
b=int(input("Enter breadth of rectangle : "))
print("Area of rectangle=",area(l,b))
print("Perimeter of rectangle=",perimeter(l,b))

O/P:-
Enter length of rectangle : 10
Enter breadth of rectangle : 5
Area of rectangle=50
Perimeter of rectangle=30
```

2. **WAP to find factorial of given number using 'Recursion'.**

```
def fact(n):
    if n==0 or n==1:
        return 1
    else:
        return n*fact(n-1)
x=int(input("Enter a number to find factorial : "))
print("Factorial=",fact(x))
O/P:- Enter a number to find factorial : 5
Factorial=120
```

3. **WAP to make a simple calculator using single function named calc().**

```
def calc(x,y):
    return [x+y,x-y,x*y,x/y]
a=int(input("Enter first no : "))
b=int(input("Enter second no : "))
res=calc(a,b)
print("Summation=",res[0])
print("Subtraction=",res[1])
print("Multiplication=",res[2])
print("Division=",res[3])

O/P:-
Enter first no : 10
Enter second no : 2
Summation=12
Subtraction=8
Multiplication=20
Division=5.0
```

4. **WAP to create a module named myutil in this module create two functions add() and greatest(). In add() function write code to return sum of two numbers and in**

greatest() function write code to return greatest number in two numbers. Now test module myutil.

```
myutil.py

def add(x,y):
   return x+y
def greatest(x,y):
   if x>y:
      return x
   else:
      return y

test.py
import myutil
a=int(input("Enter first no : "))
b=int(input("Enter second no : "))
print("Sum=",myutil.add(a,b))
print("Greatest=",myutil.greatest(a,b))


O/P:-
Enter first no : 10
Enter second no : 20
Sum=30
Greatest=20
```

Interview Questions:-

1. **What is function?**
   Function is a named block of code which perform specific task.
2. **What are advantages of functions?**
   By using function you can avoid to write same code over and over.
3. **What is Recursion?**
   When a function call itself, then it is called 'Recursion'.
4. **How to create function in python?**
   In python you can create function by using def keyword followed by function name.
   The syntax of function creation is given below:-
   def function_name(parameters):
      #code
5. **What is module in python?**
   Module is a collection of functions , classes and submodules.

Multiple choice questions Set -1 :-

1. Which of the following is the use of function in python?
a) Functions are reusable pieces of programs

b) Functions don’t provide better modularity for your application
c) you can’t also create your own functions
d) All of the mentioned

2. Which keyword is used for function?
a) fun
b) define
c) def
d) function

3. Which are the advantages of functions in python?
a) Reducing duplication of code
b) Decomposing complex problems into simpler pieces
c) Improving clarity of the code
d) All of the mentioned

4. What are the two main types of functions?
a) Custom function
b) Built-in function & User defined function
c) User function
d) System function

5. Where is function defined?
a) Module
b) Class
c) Another function
d) All of the mentioned

6. What is called when a function is defined inside a class?
a) Module
b) Class
c) Another function
d) Method

7. What is a variable defined outside a function referred to as?
a) A static variable
b) A global variable
c) A local variable
d) An automatic variable

Answer Key:-

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

Multiple choice questions Set - 2:-

1. Which of these definitions correctly describes a module?
a) Denoted by triple quotes for providing the specification of certain program elements
b) Design and implementation of specific functionality to be incorporated into a program
c) Defines the specification of how it is to be used
d) Any program that reuses code

2. Which of the following is not an advantage of using modules?
a) Provides a means of reuse of program code
b) Provides a means of dividing up tasks
c) Provides a means of reducing the size of the program
d) Provides a means of testing individual parts of the program

3. Program code making use of a given module is called a _____ of the module.
a) Client
b) Docstring
c) Interface
d) Modularity

4. _____ is a string literal denoted by triple quotes for providing the specifications of certain program elements.
a) Interface
b) Modularity
c) Client
d) Docstring

5. Which of the following is true about top-down design process?
a) The details of a program design are addressed before the overall design
b) Only the details of the program are addressed
c) The overall design of the program is addressed before the details
d) Only the design of the program is addressed

6. Which of the following isn't true about main modules?
a) When a python file is directly executed, it is considered main module of a program
b) Main modules may import any number of modules
c) Special name given to main modules is: __main__
d) Other main modules can import main modules

7. Which of the following is false about "import modulename" form of import?
a) The namespace of imported module becomes part of importing module
b) This form of import prevents name clash
c) The namespace of imported module becomes available to importing module
d) The identifiers in module are accessed as: modulename.identifier

8. Which of the following is false about "from-import" form of import?
a) The syntax is: from module name import identifier
b) This form of import prevents name clash
c) The namespace of imported module becomes part of importing module

d) The identifiers in module are accessed directly as: identifier

Answer Key:-

1. b	2. c	3. a	4. d	5. c
6. d	7. a	8. b		

Lecture – 5 (Object Oriented Programming)

Long Answer Questions:-

1. **What is object oriented programming system? Describe its pillars.**

Object oriented programming system is a mechanism of software development.

OOPS principles:-

- OOPS are the rules (or) guidelines which are supposed to be satisfied by any programming language in order to call that programming language as OOPL.
- Different oops principles are:-
  1. Encapsulation
  2. Polymorphism
  3. Inheritance
  4. Abstraction

Encapsulation: - Encapsulation is the wrapping of properties and functionalities in a single unit. That single unit is called object.

Polymorphism: - Term Polymorphism means one thing many forms. If you use a thing in many ways it is called Polymorphism.

Inheritance:- Inheritance is a feature of object oriented programming. In Inheritance you can create a new product by using existing product.

Abstraction:- Abstraction is a mechanism to show essential functionalities and hide all other functionalities.

2. **What is constructor? What is difference between method and constructor?**

Constructor Concept:-

- Constructor is a special method in python.
- The name of the constructor should be __init__(self)
- Constructor will be executed automatically at the time of object creation.
- The main purpose of constructor is to declare and initialize instance variables.
- Per object constructor will be executed only once.
- Constructor can take at least one argument(at least self)
- Constructor is optional and if we are not providing any constructor then python will provide default constructor.

Differences between Methods and Constructors:

Method	Constructor
1. Name of method can be any name.	1. Constructor name should be always __init__.
2. Method will be executed if we call that method.	2. Constructor will be executed automatically at the time of object creation.
3. Per object, method can be called any number of times.	3. Per object, Constructor will be executed only once.
4. Inside method we can write business logic.	4. Inside Constructor we have to declare and initialize instance variables.

3. **Describe different types of variables in python.**

Types of Variables:

Inside Python class 3 types of variables are allowed.

1. Instance Variables (Object Level Variables)
2. Static Variables (Class Level Variables)
3. Local variables (Method Level Variables)

1. Instance Variables:- If the value of a variable is varied from object to object, then such type of variables are called instance variables.

For every object a separate copy of instance variables will be created.

Where we can declare Instance variables:

1. Inside Constructor by using self variable.
2. Inside Instance Method by using self variable.
3. Outside of the class by using object reference variable.

1. Inside Constructor by using self variable:
We can declare instance variables inside a constructor by using self keyword. Once we creates object, automatically these variables will be added to the object.

Example Application:

```
class Employee:
        def __init__(self):
                self.eno=100
                self.ename='John'
                self.esal=10000
e=Employee()
print(e.__dict__)
```

Output:

```
{'eno': 100, 'ename': 'John', 'esal': 10000}
```

2. Inside Instance Method by using self variable:
We can also declare instance variables inside instance method by using self variable. If any instance variable declared inside instance method, that instance variable will be added once we call that method.

Example Application:

```
class Test:
        def __init__(self):
                self.a=10
                self.b=20
        def m1(self):
                self.c=30
t=Test()
t.m1()
print(t.__dict__)
```

Output:

{'a': 10, 'b': 20, 'c': 30}

3. Outside of the class by using object reference variable:

We can also add instance variables outside of a class to a particular object.

Example Application:

```
class Test:
        def __init__(self):
                self.a=10
                self.b=20
        def m1(self):
                self.c=30
t=Test()
t.m1()
t.d=40
print(t.__dict__)
```

Output {'a': 10, 'b': 20, 'c': 30, 'd': 40}

Short Answer Questions:-

1. **What is class?**

- In Python everything is an object. To create objects we required some Model or Plan or Blue print, which is nothing but class.
- We can write a class to represent properties (attributes) and actions (behaviour) of object.
- Properties can be represented by variables.
- Actions can be represented by methods.
- Hence class contains both variables and methods.

2. **How to create class?**

We can define a class by using class keyword.

Syntax:

```
class className:
 ''' documentation string '''
 variables: instance variables, static and local variables
 methods: instance methods, static methods, class methods
```

Documentation string represents description of the class. Within the class doc string is always
optional. We can get doc string by using the following 2 ways.
1. print (classname.__doc__)
2. help (classname)

3. What is object?

Physical existence of a class is nothing but object. We can create any number of objects for a class.
Syntax to create object: referencevariable = classname()
Example: s = Student()

4. What is reference variable?

The variable which can be used to refer object is called reference variable. By using reference variable, we can access properties and methods of object.

5. What is self variable?

Self variable:- self is the default variable which is always pointing to current object (like this keyword in Java) By using self we can access instance variables and instance methods of object.

Note:
1. self should be first parameter inside constructor
 def __init__(self):
2. self should be first parameter inside instance methods
 def talk(self):

Technical Tasks:-

Example Application:-

```
class Student:
        '''Developed by Brijesh for python demo'''
        def __init__(self):
                self.name='Brijesh'
                self.age=40
                self.marks=80
        def talk(self):
                print("Hello I am :",self.name)
                print("My Age is:",self.age)
                print("My Marks are:",self.marks)
st=Student()
st.talk()
```

```
O/P:-
Hello I am : Brijesh
My Age is : 40
My Marks are: 80
```

Example Application:- Write a Python program to create a Student class and Creates an object to it. Call the method talk() to display student details.

```
class Student:
        def __init__(self,name,rollno,marks):
                self.name=name
                self.rollno=rollno
                self.marks=marks
        def talk(self):
                print("Hello My Name is:",self.name)
                print("My Rollno is:",self.rollno)
                print("My Marks are:",self.marks)
s1=Student("Brijesh",101,80)
s1.talk()
```

Output:

```
Hello My Name is: Brijesh
My Rollno is: 101
My Marks are: 80
```

Example Application:- Program to demonstrate constructor will execute only once per object.

```
class Test:
        def __init__(self):
                print("Constructor execution...")
        def m1(self):
                print("Method execution...")
t1=Test()
t2=Test()
t3=Test()
t1.m1()
```

Output:

```
Constructor execution...
Constructor execution...
Constructor execution...
Method execution...
```

Example Application:-

```
class Student:
        ''''' This is student class with required data'''
```

```
        def __init__(self,x,y,z):
                self.name=x
                self.rollno=y
                self.marks=z
        def display(self):
                print("StudentName:{}\nRollno:{}\nMarks:{}".format(self.name,self.rollno,self.marks))
s1=Student("John",101,80)
s1.display()
s2=Student("Brown",102,100)
s2.display()
```

Output:

```
Student Name:John
Rollno:101
Marks:80
Student Name:Brown
Rollno:102
Marks:100
```

Example Application:-

```
class Employee:
        def __init__(self):
                self.eno=100
                self.ename='John'
                self.esal=10000
e=Employee()
print(e.__dict__)
```

Output:

```
{'eno': 100, 'ename': 'John', 'esal': 10000}
```

Example Application:-

```
class Test:
        def __init__(self):
                self.a=10
                self.b=20
        def m1(self):
                self.c=30
t=Test()
t.m1()
```

```
print(t.__dict__)
```
Output:

```
{'a': 10, 'b': 20, 'c': 30}
```

Interview Questions:-

1. **What is Object oriented programming system?**
   Object oriented programming system is a mechanism of software development.
2. **What are pillars of object oriented programming system?**
   There are four pillars of object oriented programming system. These are given below:-
   i. Abstraction
   ii. Encapsulation
   iii. Inheritance
   iv. Polymorphism
3. **What is static variable?**
   Static variables are also known as class variables, these are accessible by using class name.
4. **What is non-static variable?**
   Non-static variables are also called instance variables, these are accessible by using object name.

5. **What is constructor?**
   Constructor is a special method which is used to initialize variables.
6. **What is need of constructor?**
   Constructor call only once with one object.
7. **What is class?**
   Class is a blueprint of object. Class is the collection of variables and methods.
8. **What is object?**
   Object is a real world entity, which has its properties and functionalities.
9. **What is difference between method and constructor?**

Differences between Methods and Constructors:

Method	Constructor
1. Name of method can be any name.	1. Constructor name should be always __init__.
2. Method will be executed if we call that method.	2. Constructor will be executed automatically at the time of object creation.
3. Per object, method can be called any number of times.	3. Per object, Constructor will be executed only once.
4. Inside method we can write business logic.	4. Inside Constructor we have to declare and initialize instance variables.

10. **What is purpose of self?**

Self is the default parameters , it is used to specify instance variables.

Multiple Choice Questions:-

1. _____ represents an entity in the real world with its identity and behaviour.
a) A method
b) An object
c) A class
d) An operator

2. _____ is used to create an object.
a) class
b) constructor
c) User-defined functions
d) In-built functions

3. Which of the following is not a class method?
a) Non-static
b) Static
c) Bounded
d) Unbounded

4. What are the methods which begin and end with two underscore characters called?
a) Special methods
b) In-built methods
c) User-defined methods
d) Additional methods

5. Which of the following is not OOPS concept in Python?
a) Inheritance
b) Encapsulation
c) Polymorphism
d) Compilation

Answer Key:-

1. B	2. b	3. a	4. a	5. d

Lecture – 6 (Inheritance in Python)

Long Answer Questions:-

1. **What is Inheritance? Write its importance.**

Inheritance:-

The concept of using properties of one class into another class without creating object of that class explicitly is known as inheritance.

- A class which is extended by another class is known as 'super' class.
- A class which is extending another class is known as 'sub' class.
- Both super class properties and sub class properties can be accessed through subclass reference variable.
- Super class properties directly we can use within the subclass.

Example Application:-

```
class x:
   def m1(self):
      self.i=1000
class y(x):
```

```
    def m2(self):
        self.j=2000
y1=y()
y1.m1()
y1.m2()
print (y1.i)
print (y1.j)
```

Output:-

```
1000
2000
```

Example Application:-

```
class x:
    def m1(self):
        self.i=1000
class y(x):
    def m2(self):
        self.j=2000
    def display(self):
        print self.i
        print self.j
y1=y()
y1.m1()
y1.m2()
y1.display()
```

Output:-

```
1000
2000
```

Example Application:-

```
class parent:
    parentAttr=100
    def __init__(self):
        print ('Calling parent constructor')
    def parent_method(self):
        print ('Calling parent method')
    def setAttr(self,attr):
        parent.parentAttr=attr
    def getAttr(self):
```

```
        print ('parent Attribute',parent.parentAttr)
class child(parent):
    def __init__(self):
        print ('Calling child constructor')
    def child_method(self):
        print ('Calling child method')
p=parent()
p.parent_method()
p.setAttr(200)
p.getAttr()
c=child()
c.child_method()
c.parent_method()
c.setAttr(300)
c.getAttr()
```

Output:-

```
Calling parent constructor
Calling parent method
parent Attribute 200
Calling child constructor
Calling child method
Calling parent method
parent Attribute 300
```

Example Application:-

```
class x:
    def __init__(self):
        print 'in const of x'
class y(x):
    def m1(self):
        print 'in m1 of y'
y1=y()
y1.m1()
```

Output:-

```
in const of x
in m1 of y
```

2. **Describe types of inheritance in python.**

Types of Inheritance:-

Single Inheritance:- The concept of inheriting properties from only one class into another class is known as single inheritance.

Example Application:-

```
class x:
   def m1(self):
      print('m1 of x')
class y(x):
   def m2(self):
      print('m2 of y')
y1=y()
y1.m1()
y1.m2()
```

Output:-

```
m1 of x
m2 of y
```

Multi-level Inheritance:- The concept of inheriting properties from multiple classes into single class with the concept of 'one after another' is known as a multilevel inheritance.

Example Application:-

```
class x:
   def m1(self):
      print ('m1 of x')
class y(x):
   def m2(self):
      print ('m2 of y')
class z(y):
   def m3(self):
      print ('m3 of z')
z1=z()
z1.m1()
z1.m2()
z1.m3()
```

Output:-

```
m1 of x
m2 of y
```

m3 of z

Hierarchical Inheritance:- The concept of inheriting properties from one class into multiple classes is known as a hierarchical inheritance.

Example Application:-

```
class x:
   def m1(self):
      print ('in m1 of x')
class y(x):
   def m2(self):
      print ('in m2 of y')
class z(x):
   def m3(self):
      print ('in m3 of z')
y1=y()
y1.m1()
y1.m2()
z1=z()
z1.m1()
z1.m3()
```

Output:-

```
in m1 of x
in m2 of y
in m1 of x
in m3 of z
```

Multiple Inheritance:- The concept of inheriting properties from multiple classes into single class at a time is known as multiple inheritance.

Example Application:-

```
class x:
   def m1(self):
      print ('in m1 of x')
class y:
   def m2(self):
      print ('in m2 of y')
class z(x,y):
   def m3(self):
      print ('in m3 of z')
z1=z()
```

```
z1.m1()
z1.m2()
z1.m3()
y1=y()
y1.m2()
x1=x()
x1.m1()
```

Output:-

```
in m1 of x
in m2 of y
in m3 of z
in m2 of y
in m1 of x
```

Cyclic Inheritance:- The concept of inheriting the properties from subclass to superclass is known as a cyclic inheritance. Python does not support cyclic inheritance.

Technical Tasks:-

1. **WAP to demonstrate single inheritance.**

```
class Rundog:
    def bark(self):
        print("Sheru……")
        print("Bho…bho……")
class Bulldog(Rundog):
    def growl(self):
        print("Tommy……")
        print("Gurr.. gurr……")
dog=Bulldog()
dog.bark()
dog.growl()
```

2. **WAP to demonstrate hierarchical inheritance.**

```
class Shape:
    def setValue(self, s):
        self.s=s
class Square(Shape):
    def area(self):
```

```
        return self.s*self.s
class Cube(Shape):
    def volume(self):
        return self.s*self.s*self.s
sq=Square()
x=int(input("Enter side of square : "))
a=sq.area()
print("Area of square=",a)
cu=Cube()
x=int(input("Enter side of cube : "))
v=cu.volume()
print("Volume of cube=",v)
```

3. WAP to demonstrate hybrid inheritance.

```
class Employee:
    def setEmployee(self,empid,empname):
        self.empid=empid
        self.empname=empname
    def getEmployee(self):
        print("Employee Id=",self.empid)
        print("Employee Name=",self.empname)
class Payroll(Employee):
    def setPayroll(self,bs,hra,da):
        self.bs=bs
        self.hra=hra
        self.da=da
    def getPayroll(self):
        print("Basic Salary=",self.bs)
        print("House Rent Allowances=",self.hra)
        print("Dearness Allowances=",self.da)
class Loan:
    def setLoan(self,amt):
        self.amt=amt
class Payslip(Payroll,Loan):
    def netSalary(self):
        print("Net Salary=",self.bs+self.hra+self.da)
        print("Salary on hand=",self.bs+self.hra+self.da-self.amt)
ps=Payslip()
empid=int(input("Enter Employee Id : "))
empname=input("Enter Employee Name : ")
ps.setEmployee(empid,empname)
bs=int(input("Enter basic salary : "))
hra=int(input("Enter house rent allowances : "))
da=int(input("Enter dearness allowances : "))
ps.setPayroll(bs,hra,da)
amt=int(input("Enter monthly loan amt : "))
```

```
ps.setLoan(amt)
print("**********PAY SLIP**********")
ps.getEmployee()
ps.getPayroll()
ps.netSalary()
```

Interview Questions:-

1. **What is inheritance? Write its importance.**
   Inheritance is a feature of object oriented programming. In Inheritance you can create a new class by using existing class. The concept of inheritance is also called 'Reusability'.
2. **How many types of inheritance?**
   In Python there are following types of Inheritance:-
   i. Single Inheritance
   ii. Hierarchical Inheritance
   iii. Multiple Inheritance
   iv. Multilevel Inheritance
   v. Hybrid Inheritance

Multiple Choice Questions:-

1. Which of the following best describes inheritance?
a) Ability of a class to derive members of another class as a part of its own definition
b) Means of bundling instance variables and methods in order to restrict access to certain class members
c) Focuses on variables and passing of variables to functions
d) Allows for implementation of elegant software that is well designed and easily modified

2. Which of the following is not a type of inheritance?
a) Double-level
b) Multi-level
c) Single-level
d) Multiple

3. What does single-level inheritance mean?
a) A subclass derives from a class which in turn derives from another class
b) A single superclass inherits from multiple subclasses
c) A single subclass derives from a single superclass
d) Multiple base classes inherit a single derived class

4. Which of the following represents a distinctly identifiable entity in the real world?
A. A class
B. An object
C. A method
D. A data field

5. Which of the following represents a template, blueprint, or contract that defines objects of the same type?
A. A class
B. An object
C. A method
D. A data field

6. Which of the following is required to create a new instance of the class?
A. A constructor
B. A class
C. A value-returning method
D. A None method

7. Which of the following statements is most accurate for the declaration x = Circle()?
A. x contains an int value.
B. x contains an object of the Circle type.
C. x contains a reference to a Circle object.
D. You can assign an int value to x.

8. Which of the following statements can be used to check, whether an object “obj” is an instance of class A or not?
A. obj.isinstance(A)
B. A.isinstance(obj)
C. isinstance(obj, A)
D. isinstance(A, obj)

Answer Key:-

1. A	2. a	3. c	4. b
5. a	6. a	7. c	8. c

Lecture – 7(Concept of Polymorphism and Exception Handling)

Long Answer Questions:-

1. **What is Polymorphism? Describe it.**

Polymorphism:-

- Poly means many and morphs means forms.
- Forms mean functionalities or logics.
- The concept of defining multiple logics to perform some operation is known as a polymorphism.
- Polymorphism can be implemented in python by using method overriding.

Note:- Python does not support method overloading.

Method overriding:- The concept of defining multiple methods with the same name with the same no. of parameters, one is in superclass and another in subclass is known as method overriding.

Example Application:-

```
class parent:
    def myMethod(self):
        print ('Calling parent method')
class child(parent):
    def myMethod(self):
        print ('Calling child method')
c=child()
c.myMethod()
p=parent()
p.myMethod()
```

Output:-

Calling child method
Calling parent method

super():- when method overriding.

Example Application:-

```
class parent(object):
    def altered(self):
        print ('parent altered')
class child(parent):
    def altered(self):
        print ('child before parent altered')
        super(child,self).altered()
        print ('child after parent altered')
dad=parent()
son=child()
dad.altered()
son.altered()
```

Output:-

parent altered
child before parent altered
parent altered
child after parent altered

Abstraction (or) Data hiding:-

The concept of hiding the properties of one class from the other classes or other programs directly is known as data hiding or Abstraction.(__is prefix)

Example Application:-

```
class x:
   __p=1000
   def m1(self):
      print ('in m1 of x')
x1=x()
x1.m1()
print x.p
```

Output:-

```
in m1 of x
AttributeError: class x has no attribute 'p'
```

Example Application:-

```
class JustCounter:
   __secretcount=0
   def count(self):
      self.__secretcount+=1
      print self.__secretcount
counter=JustCounter()
counter.count()
counter.count()
```

Output:-

```
1
2
```

2. What is Exception handling? Describe it.

In any programming language there are 2 types of errors are possible.

1. Syntax Errors
2. Runtime Errors

1. Syntax Errors:

The errors which occurs because of invalid syntax are called syntax errors.

Eg 1:

```
x=10
if x==10
        print("Hello")
SyntaxError: invalid syntax
```

Eg 2:

```
print "Hello"
SyntaxError: Missing parentheses in call to 'print'
```

Note:
Programmer is responsible to correct these syntax errors. Once all syntax errors are corrected then only program execution will be started.

2. Runtime Errors:- Runtime Errors are also known as exceptions. While executing the program if something goes wrong because of end user input or programming logic or memory problems etc then we will get Runtime Errors.

Eg: print(10/0) ==>ZeroDivisionError: division by zero
print(10/"ten") ==>TypeError: unsupported operand type(s) for /: 'int' and 'str'

What is Exception?
An unwanted and unexpected event that disturbs normal flow of program is called exception.
Eg:
ZeroDivisionError
TypeError
ValueError
FileNotFoundError
EOFError
SleepingError
TyrePuncturedError

It is highly recommended to handle exceptions. The main objective of exception handling is Graceful Termination of the program(i.e we should not block our resources and we should not miss anything) Exception handling does not mean repairing exception. We have to define alternative way to continue rest of the program normally.
Eg:
For example our programming requirement is reading data from remote file locating at London. At runtime if london file is not available then the program should not be terminated abnormally. We have to provide local file to continue rest of the program normally. This way of defining alternative is nothing but exception handling.

Default Exception Handing in Python:

Every exception in Python is an object. For every exception type the corresponding classes are available. Whenever an exception occurs PVM will create the corresponding exception object and will check for handling code. If handling code is not available then Python interpreter terminates the program abnormally and prints corresponding exception information to the console. The rest of the program won't be executed.

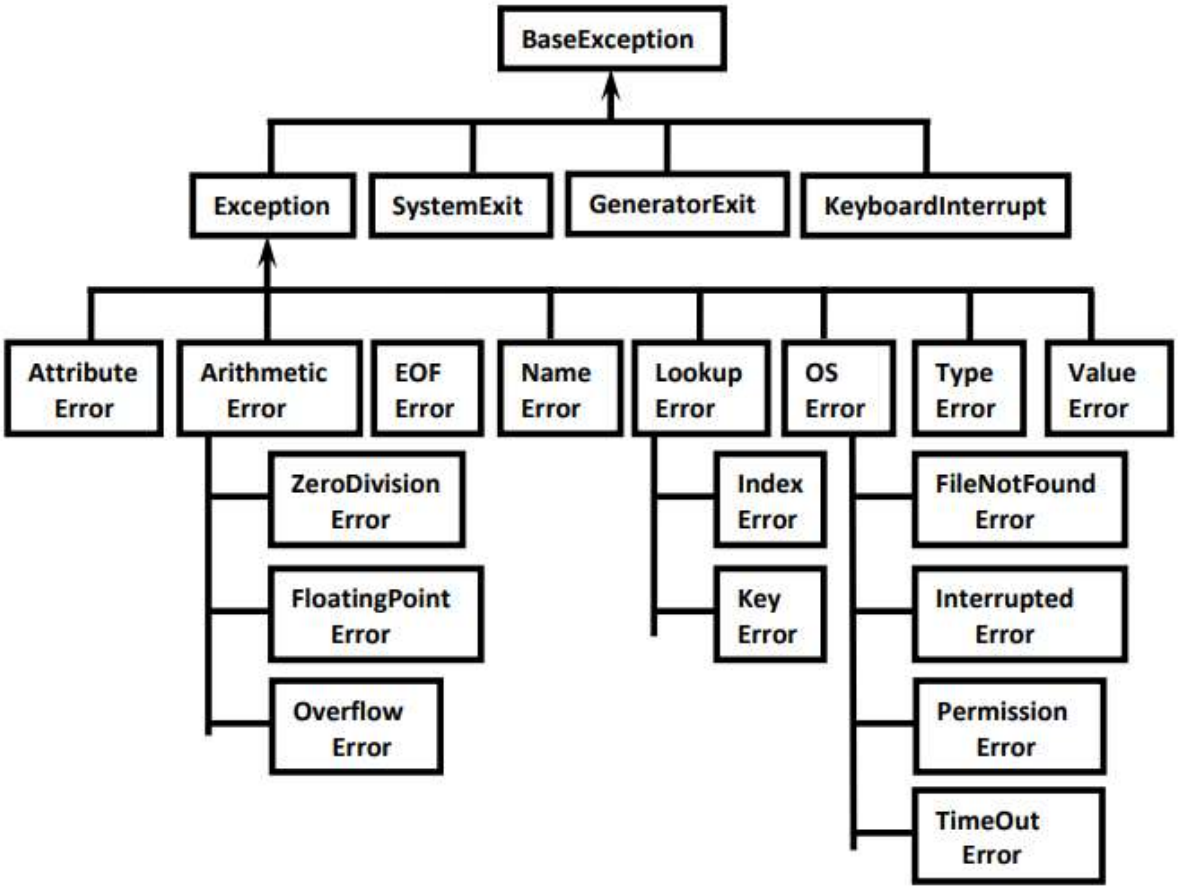
Eg:
```
print("Hello")
print(10/0)
```

```
print("Hi")
```

Output:-

```
Hello
Traceback (most recent call last):
File "test.py", line 2, in <module>
print(10/0)
ZeroDivisionError: division by zero
```

Python Exception Hierarchy:-



- Every Exception in Python is a class.
- All exception classes are child classes of BaseException .i.e every exception class extends BaseException either directly or indirectly. Hence BaseException acts as root for Python Exception Hierarchy.
- Most of the times being a programmer we have to concentrate Exception and its child
  classes.

Customized Exception handling by using try-except:

- It is highly recommended to handle exceptions.
- The code which may raise exception is called risky code and we have to take risky code inside try block. The corresponding handling code we have to take inside except block.

```
try:
        Risky Code
except XXX:
        Handling code/Alternative Code
```

Without try-except:

```
print("stmt-1")
print(10/0)
print("stmt-3")
```

Output:

```
stmt-1
ZeroDivisionError: division by zero
Abnormal termination/Non-Graceful Termination
```

With try-except:

```
print("stmt-1")
try:
        print(10/0)
except ZeroDivisionError:
        print(10/2)
print("stmt-3")
```

Output:

```
stmt-1
5.0
stmt-3
Normal termination/Graceful Termination
```

Control Flow in try-except:

```
try:
 stmt-1
 stmt-2
 stmt-3
except XXX:
 stmt-4
stmt-5
```

case-1: If there is no exception
1,2,3,5 and Normal Termination
case-2: If an exception raised at stmt-2 and corresponding except block matched
1,4,5 Normal Termination
case-3: If an exception raised at stmt-2 and corresponding except block not matched
1, Abnormal Termination
case-4: If an exception raised at stmt-4 or at stmt-5 then it is always abnormal termination.

Conclusions:
1. Within the try block if anywhere exception raised then rest of the try block wont be executed even though we handled that exception. Hence we have to take only risky code inside try block and length of the try block should be as less as possible.
2. In addition to try block, there may be a chance of raising exceptions inside except and finally blocks also.
3. If any statement which is not part of try block raises an exception then it is always abnormal termination.

How to print exception information:

```
try:
        print(10/0)
except ZeroDivisionError as msg:
        print("exception raised and its description is:",msg)
```

Output: exception raised and its description is: division by zero

try with multiple except blocks:
The way of handling exception is varied from exception to exception. Hence for every exception type a separate except block we have to provide. i.e try with multiple except blocks is possible and recommended to use.
Eg:

```
try:
        -------
         -------
        -------
except ZeroDivisionError:
        perform alternative
        arithmetic operations
except FileNotFoundError:
        use local file instead of remote file
```

If try with multiple except blocks available then based on raised exception the corresponding except block will be executed.

Eg:

```
try:
        x=int(input("Enter First Number: "))
        y=int(input("Enter Second Number: "))
        print(x/y)
except ZeroDivisionError :
        print("Can't Divide with Zero")
except ValueError:
        print("please provide int value only")
```

Example Application:-

```
try:
        x=int(input("Enter First Number: "))
        y=int(input("Enter Second Number: "))
        print(x/y)
except ZeroDivisionError :
        print("Can't Divide with Zero")
except ValueError:
        print("please provide int value only")
```

Output 1:-

```
Enter First Number: 10
Enter Second Number: 2
5.0
```

Output 2:-

```
Enter First Number: 10
Enter Second Number: 0
Can't Divide with Zero
```

Output 3:-

```
Enter First Number: 10
Enter Second Number: ten
please provide int value only
```

If try with multiple except blocks available then the order of these except blocks is important. Python interpreter will always consider from top to bottom until matched except block identified.

Example Application:-

```
try:
        x=int(input("Enter First Number: "))
        y=int(input("Enter Second Number: "))
        print(x/y)
except ArithmeticError :
        print("ArithmeticError")
except ZeroDivisionError:
        print("ZeroDivisionError")
```

Output:-

Enter First Number : 10
Enter Second Number : 0
ArithmeticError

Single except block that can handle multiple exceptions:

We can write a single except block that can handle multiple different types of exceptions.
except (Exception1,Exception2,exception3,..): or
except (Exception1,Exception2,exception3,..) as msg :
Parenthesis are mandatory and this group of exceptions internally considered as tuple.

Example Application:-

```
try:
        x=int(input("Enter First Number: "))
        y=int(input("Enter Second Number: "))
        print(x/y)
except (ZeroDivisionError,ValueError) as msg:
        print("Plz Provide valid numbers only and problem is: ",msg)
```

Output 1:-

Enter First Number: 10
Enter Second Number: 0
Plz Provide valid numbers only and problem is: division by zero

Output 2:-

Enter First Number: 10
Enter Second Number: ten
Plz Provide valid numbers only and problem is: invalid literal for int() with base 10: 'ten'

Default except block:-

We can use default except block to handle any type of exceptions. In default except block generally we can print normal error messages.

Syntax:-

```
 except:
 statements
```

Example Application:-

```
try:
        x=int(input("Enter First Number: "))
        y=int(input("Enter Second Number: "))
        print(x/y)
except ZeroDivisionError:
        print("ZeroDivisionError:Can't divide with zero")
except:
        print("Default Except:Plz provide valid input only")
```

Output 1:-

```
Enter First Number: 10
Enter Second Number: 0
ZeroDivisionError:Can't divide with zero
```

Output 2:-

```
Enter First Number: 10
Enter Second Number: ten
Default Except:Plz provide valid input only
```

Short Answer Questions:-

1. **What is method overriding?**

Re-writing of base class method into derived class is called method overriding.

2. **What is exception?**

The dictionary meaning of exception is abnormal termination. When exception is occurred then program is terminated abnormally and rest of code is not executed.

3. **What is exception handling?**

Exception handling is a mechanism to handle exception to achieve normal execution of program. For exception handling in python we use try, except and finally keywords.

Syntax for exception handling:-

```
try:
    # Code which you want to protect
except ExceptionName:
    # Code to handle exception
finally:
    # Code which you want to execute always
```

Technical Tasks:-

1. **WAP to demonstrate concept of method overriding.**

```
class parent:
        def myMethod(self):
                print('Calling parent method‘)
class child(parent):
        def myMethod(self):
                print('Calling child method‘)
c=child()
c.myMethod()
p=parent()
p.myMethod()
```

2. **WAP to demonstrate method overriding with super().**

```
# super():- when method overriding.
class parent(object):
        def altered(self):
                print('parent altered')
class child(parent):
        def altered(self):
                print('child before parent altered')
                super(child,self).altered()
                print('child after parent altered')
dad=parent()
son=child()
dad.altered()
son.altered()
```

3. **WAP to find division of two numbers.**

```
try:
        x=int(input("Enter first no : "))
        y=int(input("Enter second no : "))
        z=x/y
        print("Result=",z)
except ValueError:
        print("Enter numbers only")
except ZeroDivisionError:
        print("Are you trying to / by zero?")
finally: #Optional
        print("Bye...Bye...")
```

Interview Questions:-

1. **What is Polymorphism?**
   The term Polymorphism means one thing many forms.
2. **What is method overriding?**
   Rewriting of base class method into derived class is called method overriding.
3. **What is exception?**
   The dictionary meaning of exception is abnormal termination. When exception is occurred program is terminated abnormally and rest of code is not executed.
4. **What is exception handling?**
   Exception handling is a mechanism to handle exception to achieve normal execution of program.
5. **Explain try, except and finally blocks.**

```
try:
    # Code which you want to protect
except:
   # Code which is used to handle exception
finally:
   # Code which you want to execute always
```

Multiple Choice Questions Set - 1:-

1. Which of the following best describes polymorphism?
a) Ability of a class to derive members of another class as a part of its own definition
b) Means of bundling instance variables and methods in order to restrict access to certain class members
c) Focuses on variables and passing of variables to functions
d) Allows for objects of different types and behaviour to be treated as the same general type

2.What is the biggest reason for the use of polymorphism?
a) It allows the programmer to think at a more abstract level
b) There is less program code to write
c) The program will have a more elegant design and will be easier to maintain and update
d) Program code takes up less space

3. Overriding means changing behaviour of methods of derived class methods in the base class.
a) True
b) False

4. Which of the following statements is true?

a) A non-private method in a superclass can be overridden
b) A subclass method can be overridden by the superclass
c) A private method in a superclass can be overridden
d) Overriding isn't possible in Python

Answer Key:-

1. d	2. c	3. b	4. a

Multiple Choice Questions Set - 2:-

1. When will the else part of try-except-else be executed?
a) always
b) when an exception occurs
c) when no exception occurs
d) when an exception occurs in to except block

2. Which of the following is not an exception handling keyword in Python?
a) try
b) except
c) accept
d) finally

3. Which of the following is not a standard exception in Python?
a) NameError
b) IOError
c) AssignmentError
d) ValueError

4. Which of these keywords is not a part of exception handling?
a) try
b) finally
c) thrown
d) except

Answer Key:-

1. c	2. c	3. c	4. c

Lecture – 8 (String Manipulation in Python)

Long Answer Questions:-

1. **What is string? Describe it.**

String is a sequence of characters. This is a widely used data type in projects. Python has several built-in functions associated with the string data type. These functions let us easily modify and manipulate strings. We can think of functions as being actions that we perform on elements of our code. Built-in functions are those that are defined in the Python programming language and are readily available for us to use.

Making String Upper And Lower Case:-

The functions str.upper() and str.lower() will return a string with all the letters of an original string converted to upper- or lower-case letters. Because strings are immutable data types, the returned string will be a new string. Any characters in the string that are not letters will not be changed.
For Example:-
ss="Softpro India"
print(ss.upper())
O/P:-
SOFTPRO INDIA

Example Application:-

```
"""
Develop a program in python to take a string as input now display string in upper case and lowercase also find the length of string.
"""
st=input("Enter a string : ")
print("String in upper case : ",st.upper())
print("String in lower case : ",st.lower())
print("The length of string : ",len(st)) # The len() method find the length of string
```

join(), split(), and replace() Methods

The str.join(), str.split(), and str.replace() methods are a few additional ways to manipulate strings in Python.

- The str.join() method will concatenate two strings, but in a way that passes one string through another.
- The str.split() method returns a list of strings that are separated by whitespace if no other parameter is given.
- The str.replace() method can take an original string and return an updated string with some replacement.

Technical Tasks:-

1. **Develop a program in python to check given string is palindrome or not.**

```
string=input("Enter a string : ")
reverse_string="".join(reversed(string))
print(reverse_string)
if string==reverse_string:
        print("String is palindrome")
else:
        print("String is non-palindrome")
```

2. **Develop a program in python to take user full name as input and display short name.**

```
name=input("Enter your full name : ")
shortname=name.split(" ")
print("Your short name :",end="")
for n in range(len(shortname)-1):
        print(shortname[n][0]+".",end="")
print(shortname[len(shortname)-1])
```

3. **Develop a program in python to take a sentence now search a word in sentence and replace that word with another word.**

```
sentence=input("Enter a sentence : ")
fw=input("Find what? ")
rw=input("Replace with : ")
print("Modified sentence : "+sentence.replace(fw,rw))
```

4. **Develop a program in python to take a decimal no. as input and display its binary, octal and hexa-decimal equivalent.**

```
n=int(input("Enter a number : "))
print("Binary format : "+bin(n).replace("0b",""))
print("Octal format : "+oct(n).replace("0o",""))
print("Hexa-decimal format : "+hex(n).replace("0x",""))
```

Multiple choice Questions:-

1. What will be the output of the following Python code?

```
print('{0:.2}'.format(1/3))
```

a) 0.333333
b) 0.33
c) 0.333333:.2
d) Error

2. What will be the output of the following Python code?

```
print('{0:.2%}'.format(1/3))
```

a) 0.33
b) 0.33%
c) 33.33%
d) 33%

3. What will be the output of the following Python code?

```
print('ab12'.isalnum())
```

a) True
b) False
c) None
d) Error

4. What will be the output of the following Python code?

```
print('ab,12'.isalnum())
```

a) True
b) False
c) None
d) Error

5. What will be the output of the following Python code?

```
print('ab'.isalpha())
```

a) True
b) False
c) None
d) Error

6. What will be the output of the following Python code?

```
print('a B'.isalpha())
```

a) True
b) False

c) None
d) Error

7. What will be the output of the following Python code snippet?

print('0xa'.isdigit())
a) True
b) False
c) None
d) Error

8. What will be the output of the following Python code snippet?

print(''.isdigit())
a) True
b) False
c) None
d) Error

9.What will be the output of the following Python code snippet?

print('my_string'.isidentifier())
a) True
b) False
c) None
d) Error

10.What will be the output of the following Python code snippet?

print('abc'.islower())
a) True
b) False
c) None
d) Error

Answer Key:-

1. B	2. c	3. a	4. b	5. a
6. b	7. b	8. b	9. a	10. a