Roll No. | | | |

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC
CELEBRATING 25 YEARS

Internal Assessment Test III – September 2023

Sub:	Introduction to Python Programming				Sub Code:	BPLCK205B	Branch:	Chemistry Cycle
Date:	06-09-2023	Duration:	90 min's	Max Marks:	50	Sem / Sec:	II / A,B,C,E & F	OBE

	Answer any FIVE FULL QUESTIONS	MARKS	CO	RBT
1 (a)	**Explain how to copy, move, rename files in Python using Shutil Module.**	6	CO3	L2

Explanation – [3 Marks]

Example code– [3 Marks]

Solution:

The shutil module in Python provides functions for working with files and directories. You can use shutil to copy, move, and rename files

i**) Copying files**: To copy a file from one location to another, use the shutil.copy(src, dst) method. The src argument is the path of the source file, and the dst argument is the path of the destination file. If dst is a directory, the file is copied with the same name to that directory

Code Snippet

```
import shutil
# Copy a file from src to dst
src = 'path/to/source/file.txt'
dst = 'path/to/destination/file.txt'
shutil.copy(src, dst)
```

ii) **Moving files**: To move a file from one location to another, use the shutil.move(src, dst) method. The src argument is the path of the source file, and the dst argument is the path of the destination file. If dst is a directory, the file is moved with the same name to that directory

Code Snippet

```
import shutil
# Move a file from src to dst
src = 'path/to/source/file.txt'
dst = 'path/to/destination/file.txt'
shutil.move(src, dst)
```

Both shutil.copy() and shutil.move() can handle both files and directories.

iii) Renaming files: To rename a file, use the os.rename(src, dst) method. The src argument is the path of the source file, and the dst argument is the new name of the fileimport os

	Code Snippet
	# Rename a file
	src = 'path/to/source/file.txt'
	dst = 'path/to/source/new_file.txt'
	os.rename(src, dst)			

(b)	**Write a python program to extracts all the files and folders from a ZIP file**
	
	Full Program- [2 Marks]
	Logic-[2 Marks]
	
	Solution:
	Python can be used to extract all files and folders within a zip file. This can be done through the zipfile module.
	
	The zipfile module allows us to work with zip files, doing things such as creating zip files, reading zip files, extracting all files and folders from the zip file, and adding files to zip files.
	
	So, if you want an example of a zip file if you don't have your own, you can use this basic zip file here: documents.zip.
	
	This is a simple zip file composed of 4 files: document1.txt, document2.txt, document3.txt, and document4.txt
	
	This is a zip file, meaning it is compressed.
	
	Code Snippet
	>>>import zipfile
	>>> import os
	
	>>> os.getcwd()
	'C:\\Users\\dlhyl\\AppData\\Local\\Programs\\Python\\Python38-32'
	
	>>> myzipfile= zipfile.ZipFile('documents.zip')
	>>> myzipfile.extractall()
	>>> myzipfile.close()
		4	CO3	L3

2 (a)	**Explain Object-oriented features.** **Explanation- [2 Marks]** **Concepts- Marks]** **Solution:** In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data. **OOPs Concepts in Python** ○ Class ○ Objects ○ Polymorphism ○ Encapsulation ○ Inheritance ○ Data Abstraction	5	CO4	L3

	Python Class A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods. **Python Objects** The object is an entity that has a state and behavior associated with it. It may be any real-world object like a mouse, keyboard, chair, table, pen, etc. Integers, strings, floating-point numbers, even arrays, and dictionaries, are all objects. **Python Inheritance** Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class **Python Polymorphism** Polymorphism simply means having many forms. For example, we need to determine if the given species of birds fly or not, using polymorphism we can do this using a single function. **Python Encapsulation** Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. **Data Abstraction** It hides unnecessary code details from the user. Also, when we do not want to give out sensitive parts of our code implementation and this is where data abstraction came. Data Abstraction in Python can be achieved by creating abstract classes.			

(b)	**Explain __init__ () and __str__ () Methods with example.**
	Explanation – [2 Marks]
	Example code– [3 Marks]
	
	Solution:
	
	In Python, __init__() and __str__() are two special methods that are commonly used in classes.
	__init__() is a constructor method in Python that is used to initialize the object's attributes. It is called when an instance of the class is created. The self parameter in __init__() refers to the instance of the class that is being initialized, and can be used to set attributes for that instance.
	__str__() is a special method in Python that is used to define a string representation of an object. It is called when the str() function is called on an object. The self parameter in __str__() refers to the instance of the class, and can be used to access its attributes and return a string representation of the object.
	
	Here is an example program that demonstrates the use of __init__() and __str__() methods in Python:
	
	Code Snippet
	
	class Person:
	def __init__(self, name, age):
	self.name = name
	self.age = age
	
	def __str__(self):
	return f"{self.name} ({self.age} years old)"
	person1 = Person("John", 30)
	person2 = Person("Jane", 25)
	print(person1) # Output: John (30 years old)
	print(person2) # Output: Jane (25 years old)
	
	In this program, we define a Person class that has two attributes: name and age. We use the
		5	CO4	L2

	__init__() method to initialize these attributes when an instance of the class is created. We also define the __str__() method to return a string representation of the object. When we print an ce of the Person class using the print() function, the __str__() method is called to return a string entation of the object.			

3 (a)	**What is operator overloading? Explain with an example Program.**
	
	Explanation – [2 Marks]
	Example code– [3 Marks]
	
	Solution:
	
	Operator overloading is the ability of a programming language to define operators for user-defined types or objects. In Python, the "+" operator can be overloaded to add two objects of a user-defined class.
	
	Here's an example program that demonstrates operator overloading for a Point class: Python
	
	Code Snippet
	
	class Point:
	def __init__(self, x=0, y=0):
	self.x = x
	self.y = y
	def __add__(self, other):
	return Point(self.x + other.x, self.y + other.y)
	
	def __str__(self):
	return f'({self.x}, {self.y})'
	p1 = Point(2, 3)
	p2 = Point(-1, 5)
	p3 = p1 + p2
	print(p1) # Output: (2, 3)
	print(p2) # Output: (-1, 5)
	print(p3) # Output: (1, 8)
	
	In this example, we define a Point class with __add__() and __str__() methods. __add__() method is used to overload the "+" operator to add two Point objects, while __str__() method is used to overload the str() function to display the Point object as an ordered pair.
	
	We create two Point objects p1 and p2, and then add them together using the "+" operator, which calls the __add__() method of the Point class. The resulting Point object is stored in p3 and then displayed using the __str__() method.
		5	CO4	L3

(b)	**Write a Python program to convert seconds into hours, minutes and seconds using class [Define Time Class with attribute hour, minute, second]**
	Sample input / output
	Enter time in seconds 5690
	Time is: 01:34:50
	
	Full Program- [3 Marks]
	Logic- [2 Marks]
	
	Solution:
	
	Code Snippet:
	
	class Time:
	def __init__ (self, hour, minute, second):
	self.hour = hour
	self.minute = minute
	self.second = second
	def to_seconds(self):
		5	CO4	L3

```
 return self.hour * 3600 + self.minute * 60 + self.second
 # Get input from user
 hours, minutes, seconds = input ("Enter time hrs, min and sec: ").split()
 hours, minutes, seconds = int(hours), int(minutes), int(seconds)
   # Create a Time object and convert to seconds
   time_obj = Time(hours, minutes, seconds)
   total_seconds = time_obj.to_seconds()

 # Display output
   print("Time in seconds:", total_seconds)
```

Output:

```
Enter time hrs, min and sec: 01 34 50
Time in seconds: 5690
```

4(a) **Write a function named DivExp which takes TWO parameters a, b and returns a value c (c=a/b). Write suitable assertion for a>0 in function DivExp and raise an exception for when b=0. Develop a suitable program which reads two values from the console and calls a function DivExp.**

Full Program- [3 Marks]
Logic -[2 Marks]

Solution:

Code Snippet:

```
import sys

def DivExp(a,b):
assert a>0, "a should be greater than 0"
try:
c = a/b
except ZeroDivisionError:
print("Value of b cannot be zero")
sys.exit(0)
else:
return c
val1 = int(input("Enter a value for a : "))
val2 = int(input("Enter a value for b : "))
val3 = DivExp(val1, val2)
print(val1, "/", val2, "=", val3)
```

Output:

```
Enter a value for a: 7
Enter a value for b: 6
7/6=1.166666666666666667
```

5 CO4 L3

(b) **Discuss the methods isinstance () and hasattr () with suitable example?**

Explanation – [2 Marks]
Example code– [3 Marks]

Solution:

isinstance() function:

The **isinstance () function** is used to determine whether an object is an instance of a specific class or not. It returns True if the object is an instance of the specified class, otherwise returns False.

Code Snippet

```
class Person:
def __init__(self, name, age):
```

5 CO3 L2

	self.name = name
	self.age = age
	p = Person("John", 36)
	print(isinstance(p, Person)) **# Output: True**
	print(isinstance("hello", Person)) **# Output: False**
	
	hasattr() function:
	The hasattr() function is used to determine whether an object has a given named attribute or not. It returns True if the object has the named attribute, otherwise returns False.
	
	Code Snippet
	
	class Person:
	def __init__(self, name, age):
	self.name = name
	self.age = age
	p = Person("John", 36)
	print(hasattr(p, "name")) **# Output: True**
	print(hasattr(p, "gender")) **# Output: False**			

5(a)	**What is operator overloading? Write a program to add two points objects by operator + overloading?**
	Explanation – [2 Marks]
	Example code– [3 Marks]
	
	Solution:
	
	Operator overloading is the ability of a programming language to define operators for user-defined types or objects. In Python, the "+" operator can be overloaded to add two objects of a user-defined class.
	
	Here's an example program that demonstrates operator overloading for a Point class:
	
	Code Snippet
	
	class Point:
	def __init__(self, x=0, y=0):
	self.x = x
	self.y = y
	def __add__(self, other):
	return Point(self.x + other.x, self.y + other.y)
	
	def __str__(self):
	return f'({self.x}, {self.y})'
	p1 = Point(2, 3)
	p2 = Point(-1, 5)
	p3 = p1 + p2
	
	print(p1**) # Output: (2, 3)**
	print(p2) **# Output: (-1, 5)**
	print(p3) **# Output: (1, 8)**
	
	In this example, we define a Point class with __add__() and __str__() methods. __add__() method is used to overload the "+" operator to add two Point objects, while __str__() method is used to overload the str() function to display the Point object as an ordered pair.
	
	We create two Point objects p1 and p2, and then add them together using the "+" operator, which calls the __add__() method of the Point class. The resulting Point object is stored in p3 and then displayed using the __str__() method.
		5	CO3	L3

(b) **What is compressing files? Explain Reading, extracting and creating ZIP files with code snippets**

Explanation – [2 Marks]
Example code– [3 Marks]

Solution:

Compressing files means reducing the size of one or more files to save disk space and make them easier to transfer over the internet. One way to compress files is to use a file archive format like ZIP.

A ZIP file is a compressed archive that can contain one or more files and directories. It is a popular file format used for compressing and archiving files on Windows, macOS, and Linux.

We can use the built-in zipfile module. Here's how you can read, extract, and create ZIP files with code snippets

1. Reading a ZIP file: To read the contents of a ZIP file in Python, you can use the zipfile.ZipFile class

Code Snippet

```
import zipfile
# Open the ZIP file
with zipfile.ZipFile('example.zip', 'r') as zip_file:
# Print the list of files in the ZIP file
print(zip_file.namelist())
# Print the contents of a specific file in the ZIP file
with zip_file.open('example.txt') as file:
print(file.read())
```

we use the with statement to open the ZIP file in read mode. We then use the namelist() method to print the list of files in the ZIP file. Finally, we use the open() method to open a specific file in the ZIP file and read its contents

2.Extracting a ZIP file: To extract the contents of a ZIP file in Python, you can use the zipfile.ZipFile.extractall() method

Code Snippet

```
import zipfile
# Open the ZIP file
with zipfile.ZipFile('example.zip', 'r') as zip_file:
# Extract all the files in the ZIP file to a directory
zip_file.extractall('extracted_files')
```

3.Creating a ZIP file: To create a new ZIP file in Python, you can use the zipfile. ZipFile class in write mode.

Code Snippet

```
import zipfile
# Create a new ZIP file
with zipfile.ZipFile('new.zip', 'w') as zip_file:
# Add a file to the ZIP file
zip_file.write('example.txt')
```

We use the with statement to create a new ZIP file in write mode. We then use the write() method to add a file called 'example.txt' to the ZIP file.

5 CO3 L2

6(a) **What is the use of send2trash module's send2trash method? How it is different from os. unlink ()? Explain with code snippet.**

Explanation – [2 Marks]
Example code– [3 Marks]

Solution:

Using send2trash, we can send files to the Trash or Recycle Bin instead of permanently deleting them. The OS module's unlink(), remove() and rmdir() functions can be used to delete files or folders. But, these functions delete the files permanently.

The send2trash functions will move a file or folder to the recycle bin, while shutil functions will permanently delete files and folders

Using send2trash, we can send files to the Trash or Recycle Bin instead of permanently deleting them. The OS module’s unlink(), remove() and rmdir() functions can be used to delete files or folders. But, these functions delete the files permanently.

Modules required
OS : The OS module in Python provides functions for interacting with the operating system. OS module comes with Python’s Standard Library.
send2trash : Send2Trash is a small package that sends files to the Trash (or Recycle Bin) natively and on all platforms. To install it type the below command in the terminal.

Code Snippet

```
import send2trash

send2trash.send2trash("/location/to/file")


import os
import send2trash

# walking through the directory
for folder, subfolders, files in os.walk('/Users/tithighosh/Documents'):

        for file in files:

                # checking if file is
                # of .txt type
                if file.endswith('.txt'):
                        path = os.path.join(folder, file)

                        # printing the path of the file
                        # to be deleted
                        print('deleted : ', path )

                        # deleting the file
                        send2trash.send2trash(path)
```

Output:

```
deleted :  /Users/tithighosh/Documents/cfile.txt
deleted :  /Users/tithighosh/Documents/e_also_big_output.txt
deleted :  /Users/tithighosh/Documents/res.txt
deleted :  /Users/tithighosh/Documents/tk.txt
```

5 CO3 L2

(b) **Define assertions. What does an assert statement in python consists of? Explain how assertions can be used with Python code snippets**

Explanation – [2 Marks]
Example code– [3 Marks]

5 CO3 L3

Solution:

Assertions are statements in Python that can be used to check if a condition is true. They are often used as a debugging aid to ensure that the assumptions made by the programmer about the state of the program are correct. If the assertion is false, an AssertError is raised, indicating that there is a bug in the program.

An assert statement consists of the assert keyword, followed by a condition that is expected to be true.

Example.

```
x = 5
assert x == 5, "x is not 5"
```

Here, we use the assert statement to check that the variable x has the value 5. If the condition x == 5 is false, an AssertionError is raised with the message "x is not 5"

Assertions can be used to add automated tests in codes to catch bugs early in the development process. For example, we can use assertions to check that the output of a function is correct for a given input.

Code Snippet:

```
def square(x):
return x ** 2
assert square(2) == 4, "square(2) should be 4"
assert square(3) == 9, "square(3) should be 9"
assert square(-2) == 4, "square(-2) should be 4"
```

Here, we define a function square that returns the square of a number. We then use assert statements to check that the output of the function is correct for different input values.

7(a) **Differentiate aliasing & copying objects and deep copy & shallow copy with example program in Python.**

Explanation – [3 Marks]
Example code– [2 Marks]

Solution:

An object will be aliased whenever there an object is assigned to another object of same class. This may happen in following situations –

- ➢ Direct object assignment (like p2=p1)
- ➢ When an object is passed as an argument to a function
- ➢ When an object is returned from a function

```
>>> class Point:
pass
>>> p1=Point()
>>> p1.x=10
>>> p1.y=20
>>> p2=p1
>>> print(p1)
< main .Point object at 0x01581BF0>
>>> print(p2)
< main .Point object at 0x01581BF0>
```

If we need a copy of an object, but not an alias, do this, Python provides a module called copy and a method called copy(). Consider the below given program to understand the concept.

5 CO4 L2

```
>>> class Point:
pass
>>> p1=Point()
>>> p1.x=10
>>> p1.y=20
>>> import copy #import module copy
>>> p3=copy.copy(p1) #use the method copy()
>>> print(p1)

< main .Point object at 0x01581BF0>

>>> print(p3)

< main .Point object at 0x02344A50>

>>> print(p3.x,p3.y)
```

This is called shallow copying.

The copy() method of copy module duplicates the object.

```
import copy
class Point:
""" This is a class Point representing coordinate point
"""
class Rectangle:
""" This is a class Rectangle. Attributes: width, height and Corner Point """
box1=Rectangle()
box1.corner=Point()
box1.width=100
box1.height=200
box1.corner.x=0
box1.corner.y=0
box2=copy.copy(box1)
print(box1 is box2) #prints False
print(box1.corner is box2.corner) #prints True
```

If we use copy.copy()inner object point will not be copied. It will be shared by box1 and box2 objects. That's is the reason second print statement output True

If we use copy.deepcopy() inner object point(corner) is also copied and occupy separate memory. This is deep copying.

```
box3=copy.deepcopy(box1)
print(box1 is box3)#prints False
print(box1.corner is box3.corner)
#prints False
```

(b) **How class can be instantiated in python? Write a python program to express instances as return values to define a class RECTANGLE with member's width, height, corner_x, corner_y, and member function to find area and perimeter of a rectangle.**

Explanation – [3 Marks]
Example code– [2 Marks]

Solution:

To instantiate a class in Python, you need to create an object of the class by calling the class name followed by parentheses.

Here's a quick example: class MyClass: def __init__(self): self. message = "Hello, World."
def greet(self): p

5 CO4 L3

	Code Snippet
	class Rectangle:
	"""Represents a rectangle.
	attributes: width, height, corner.
	"""
	The docstring lists the attributes: width and height are numbers; corner is a Point object that specifies the lower-left corner.
	
	To represent a rectangle, you have to instantiate a Rectangle object and assign values to the attributes:
	box = Rectangle()
	box.width = 100.0
	box.height = 200.0
	box.corner = Point()
	box.corner.x = 0.0
	box.corner.y = 0.0
	
	Functions can return instances. For example, find_center takes a Rectangle as an argument and returns a Point that contains the coordinates of the center of the Rectangle:
	
	def find_center(rect):
	p = Point()
	p.x = rect.corner.x + rect.width/2
	p.y = rect.corner.y + rect.height/2
	return p			