SET – 1

1. **Recall the characteristics of Object-Oriented Programming.**

A. **Characteristics of Object Oriented:**

- **Class**
- **Object**
- **Method**
- **Inheritance**
- **Polymorphism**
- **Data Abstraction**
- **Encapsulation**

2. **Define an exception with a suitable example.**

A. **Issues occurring during execution that disrupt program flow.**
Examples: ZeroDivisionError, Type Error, Key Error.

3. **Identify and list the types of inheritance used in Python.**

A. **Types of Inheritance in Python:**

- **Single inheritance**
- **Multiple inheritances**
- **Multilevel inheritance**
- **Hierarchical inheritance**
- **Hybrid inheritance**

4. **Define a class and give a basic syntax example in Python.**

A.
```
class MyClass:
        def greet(self):
                print("Hello, " + self.name)
obj = MyClass("Alice")
obj.greet()
```

5. **Recall the syntax used to open a file in read mode in Python.**

A. **Syntax:** `file = open('example.txt','r')`

6. **Define Tkinter and mention its purpose in Python GUI development.**

A. **Tkinter is the standard graphical user interface (GUI) library for Python. It provides a set of tools that allows you to create window-based applications with various widgets like buttons, labels, text boxes, and more.**

7. **Identify any four commonly used file operations in Python.**

A. **File operations in Python:**

- **Opening a file : ex.file = open('example.txt','r')**
- **Reading from a file: file.read()**
- **Writng to a file: file.write('St Peter's, Welcome)**
- **Appending to a file: file = open('example.txt', a) file.write('\nAppended text')**

8. **Describe a text file and explain its use in file handling**

A. **A text file is a file that stores data in readable text format and is used in Python for reading from or writing to external storage using file handling functions.**

Common Modes:

i)"r" – read
ii)"w" – write (overwrite)
iii) "a" – append
iv) "r+" – read and write

9. **Identify the various meta characters used in the re module in Python.**

A. **Meta characters in re module:**

- **. – Matches any character except a newline.**
- **^ – Matches the start of the string.**
- **$ – Matches the end of the string.**
- *** – Matches 0 or more repetitions of the preceding character.**
- **+ – Matches 1 or more repetitions of the preceding character.**
- **? – Matches 0 or 1 occurrence of the preceding character.**

10. **Define a regular expression and explain its role in pattern matching.**

A. **A regular expression (regex) is a sequence of characters that forms a search pattern. It is used for pattern matching within strings, allowing tasks like searching, extracting, or replacing text based on defined rules. In Python, the re module is used to work with regular expressions.**

SET – 2

1. **Define an object in Python and give a suitable example.**

A. **An 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.**

Example:

```
class Car:
    def __init__(self, brand):
        self.brand = brand
my_car = Car("Toyota")  # my_car is an object of class Car
print(my_car.brand)
```

2. **Describe the purpose of the __init__() method in Python with an example.**

A. **The __init__() method is a special constructor method in Python that is automatically called when a new object of a class is created. Its main purpose is to initialize the attributes of the object with user-defined or default values. It helps set up the object with necessary data at the time of creation.**

Example:

```
class Student:
    def __init__(self, name):
        self.name = name
s1 = Student("Alice")  # my_car is an object of class Car
print(s1.name)
```

3. **Define polymorphism and give a simple Python illustration.**

A. **Polymorphism is an object-oriented programming feature that allows the same function or method to behave differently based on the object calling it.**

Example:

```
class Bird:
    def sound(self):
        print("Chirp")
```

```
class Cow:
    def sound(self):
        print("Moo")
b = Bird()
c = Cow()
b.sound() #Output: Chirp
c.sound() #Output: Moo
```

4. Define a user-defined exception in Python with syntax.

A. A user-defined exception in Python is a custom error type created by inheriting from the built-in Exception class. It is used to handle specific errors in a more meaningful way.

Example:

```
class MyError(Exception):
    pass
try:
    raise MyError("Custom error occurred")
except MyError as e:
    print(e) #Output: Custom error occurred
```

5. Differentiate between readline() and readlines() with syntax and usage.

A. Difference between `readline()` and `readlines()`

No.	Feature	`readline()`	`readlines()`
1	Reads	One line	All lines
2	Returns	String	List of strings
3	Use Case	Read line-by-line	Read all lines at once
4	Syntax	`file.readline()`	`file.readlines()`

Example:

```
with open("sample.txt") as f:
    print(f.readline()) # Output: Apple

with open("sample.txt") as f:
    print(f.readlines()) # Output: ['Apple\n', 'Banana\n',
'Cherry\n']
```

6. Explain how to create a Label widget using Tkinter with a simple example.

A.

```
import tkinter as tk
# Step 1: Create the main application window
root = tk.Tk()
root.title("Tkinter Label Example")
# Step 2: Create a label widget
label = tk.Label(root, text="Hello, Tkinter!",
font=("Helvetica", 16))
# Step 3: Add the label to the window using pack geometry
manager
label.pack(pady=20)
# Step 4: Start the main event loop
root.mainloop()
```

7. Identify the different geometry managers used in Tkinter.

A. The three main geometry managers used in Tkinter are:

- **pack() – Organizes widgets in blocks before placing them in the parent.**

- grid() – Places widgets in a table-like structure (rows and columns).
- place() – Positions widgets at specific x and y coordinates.

8. Define a file in Python and list various file modes with their purpose.

A. A file in Python is an external storage resource used to store data permanently. Python provides built-in functions to read from or write to files.

Mode	Purpose
`'r'`	Read-only mode (default)
`'w'`	Write-only (overwrites if exists)
`'a'`	Append (adds to end of file)
`'r+'`	Read and write
`'w+'`	Write and read (overwrites)
`'a+'`	Append and read
`'b'`	Binary mode (used with other modes)

9. Describe the purpose of findall() and sub() functions with syntax and examples.

A. i) findall() : Finds all occurrences of a pattern in a string.

Syntax: `re.findall(pattern, string)`

Example:

```
re.findall("a", "cat and bat")  # ['a', 'a', 'a']
```

ii) sub() : Replaces pattern matches with a replacement string.

Syntax: `re.sub(pattern, replacement, string)`

Example:

```
re.sub("cat", "dog", "cat sat")  # 'dog sat'
```

10. Explain the use of the re module in Python for regular expression operations.

A. The re module in Python is used to perform regular expression operations like searching, matching, and replacing text patterns in strings.

Common Uses:

- re.search() – Searches for a pattern in a string.
- re.match() – Checks if the pattern matches the start of the string.
- re.findall() – Returns all matching patterns.
- re.sub() – Replaces matched patterns with a string.

It helps in text processing, validation, and data extraction.

SET – 3

1. List any four built-in exceptions in Python.

A. ZeroDivisionError, ValueError, TypeError, IndexError.

2. Explain the purpose and advantages of using inheritance in Python.

A. Inheritance in Python allows one class (child) to acquire properties and methods of another class (parent). It promotes code reuse, simplifies code maintenance, and supports hierarchical classification in object-oriented programming.

3. Describe the use of the self keyword in Python classes with an example.

A. The self keyword in Python represents the instance of the class and is used to access variables and methods associated with the object.

```
class Demo:
    def set_value(self, val):
        self.data = val
d = Demo()
d.set_value(5)
print(d.data)  # Output: 5
```

4. Identify the keywords used for exception handling in Python.

A. The keywords are try, except, finally, and raise.

5. Explain the purpose of tell() and seek() functions in file handling with syntax.

A. i) tell(): Returns the current file pointer position.

Syntax: position = file.tell()

ii) seek(): Moves the file pointer to a specified position.

Syntax: file.seek(offset, from_what)

6. List the different ways to read a file in Python.

A. The different ways to read a file in python are read(), readline(), readlines().

7. Identify the different geometry managers used in Tkinter

A. The different geometry managers used in Tkinter are pack, grid, place.

8. Describe the steps to create a Label widget in Tkinter with a code snippet.

A.
```
import tkinter as tk
# Step 1: Create the main application window
root = tk.Tk()
root.title("Tkinter Label Example")
# Step 2: Create a label widget
label = tk.Label(root, text="Hello, Tkinter!", font=("Helvetica", 16))
# Step 3: Add the label to the window using pack geometry manager
label.pack(pady=20)
# Step 4: Start the main event loop
root.mainloop()
```

9. Describe the match() method in Python's re module with its syntax and use case.

A. re.match() : Checks for a match only at the beginning of the string and returns a match object if successful, otherwise None.

Syntax: re.match(pattern, string)

Example:

```
import re
result = re.match("Hi", "Hi there")
print(result.group())  # Output: Hi
```

10. List the commonly used meta characters in Python's re module.

A. Meta characters in re module:

- **. – Matches any character except a newline.**
- **^ – Matches the start of the string.**
- **$ – Matches the end of the string.**
- *** – Matches 0 or more repetitions of the preceding character.**
- **+ – Matches 1 or more repetitions of the preceding character.**
- **? – Matches 0 or 1 occurrence of the preceding character.**