## **DAY 10 - 20.6.25 Python Coding Task**

## **Q1. Understanding Access Specifiers**

Create a class `Student` with the following properties:  
  
 Class Requirements:  
 1. `name` → Public attribute   
 2. `\_roll\_number` → Protected attribute   
 3. `\_\_marks` → Private attribute   
  
 Implement the following methods:  
 - Constructor to initialize all attributes.  
 - `display\_details()` → Public method to display all attribute values.  
 - `\_update\_roll\_number(new\_roll)` → Protected method to update roll number.  
 - `\_\_update\_marks(new\_marks)` → Private method to update marks.  
 - `access\_private\_method(new\_marks)` → Public method that uses the private method `\_\_update\_marks`.

**ANSWER**

class Student:

def \_\_init\_\_(self, name, roll\_number, marks):

self.name = name

self.\_roll\_number = roll\_number

self.\_\_marks = marks

def display\_details(self):

print(f"Name: {self.name}")

print(f"Roll Number: {self.\_roll\_number}")

print(f"Marks: {self.\_\_marks}")

def \_update\_roll\_number(self, new\_roll):

self.\_roll\_number = new\_roll

print(f"Roll number updated to: {self.\_roll\_number}")

def \_\_update\_marks(self, new\_marks):

self.\_\_marks = new\_marks

print(f"Marks updated to: {self.\_\_marks}")

def access\_private\_method(self, new\_marks):

self.\_\_update\_marks(new\_marks)

## **Q2. Demonstrate Access**

In the main section:  
 - Create an object of the `Student` class.  
 - Modify and print the `name` directly.  
 - Modify and print the `\_roll\_number` directly.  
 - Try accessing `\_\_marks` directly and observe the result.

**ANSWER**

student1 = Student("Amar", 101, 88)

student1.name = "Amari"

print(f"Modified Name: {student1.name}")

student1.\_roll\_number = 202

print(f"Modified Roll Number: {student1.\_roll\_number}")

try:

print(student1.\_\_marks)

except AttributeError as e:

print(f"Error accessing \_\_marks directly: {e}")

## **Q3. Inheritance and Access Control**

Create a subclass `Topper` that inherits from `Student` and includes:  
 - A method `try\_access()` that attempts to access `\_roll\_number` and `\_\_marks` from the subclass.  
 - Show what works and what doesn't.

**ANSWERS**

class Topper(Student):

def try\_access(self):

print(f"Accessing protected \_roll\_number from subclass: {self.\_roll\_number}")

try:

print(f"Trying to access private \_\_marks: {self.\_\_marks}")

except AttributeError as e:

print(f"Cannot access \_\_marks from subclass: {e}")

## **Q4. Use of Name Mangling**

Demonstrate how to access the private attribute `\_\_marks` using name mangling technique from outside the class.

**ANSWERS**

print(f"Accessing \_\_marks via name mangling: {student1.\_Student\_\_marks}")

## **Q5. Reflection**

Answer the following short questions:  
 1. Why can’t private members be accessed directly?  
Ans: Because they’re meant to be hidden. Python changes their name in the background (called name mangling) so that they don’t get messed with by accident. It’s a way to keep the data safe inside the class.

2. What is the purpose of using protected members in class design?  
Ans : Protected members are kind of a “warning” to other programmers. It says: “You can use this, but you probably shouldn’t unless you’re inside this class or a class that inherits from it.” It helps keep the code cleaner and more organized.

3. How does name mangling help with private members in Python?

Ans:It changes the name of private variables behind the scenes. For example, \_\_marks becomes \_Student\_\_marks. This doesn’t make it impossible to access, but it makes it harder, so people don’t accidentally use or change them.

## **Python Question Paper : Task 2**

## **Section A: Basic File Operations (Q1 - Q3)**

Q1. Write a Python program to create a text file named `sample.txt`, write your name and a message into it, and then close the file.

with open("sample.txt", "w") as f:

f.write("YourName\n")

f.write("This is a message written to the file.\n")

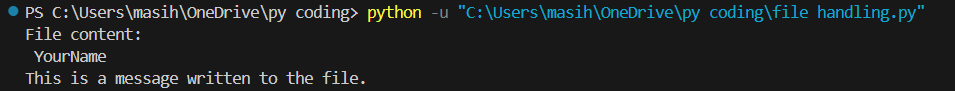
**Output : **

Q2. Write a program to read and display the contents of `sample.txt`.

with open("sample.txt", "r") as f:

content = f.read()

print("File content:\n", content)

**Output : **

Q3. Write a Python script to append a new line `"This is an appended line"` to `sample.txt` and display the updated content.

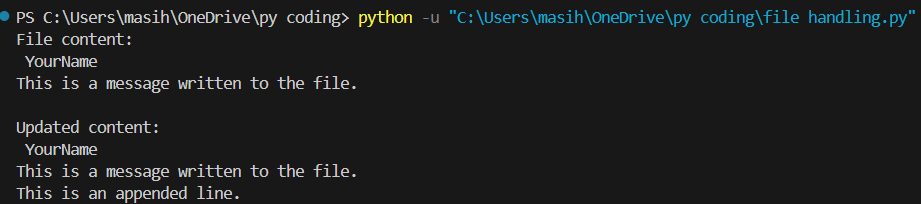
with open("sample.txt", "a") as f:

f.write("This is an appended line.\n")

with open("sample.txt", "r") as f:

content = f.read()

print("Updated content:\n", content)

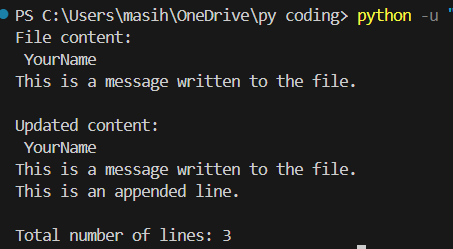
**Output : **

## **Section B: File Processing and Analysis (Q4 - Q7)**

Q4. Write a Python program to count the total number of lines in a given file `sample.txt`.

with open("sample.txt", "r") as f:

lines = f.readlines()

print("Total number of lines:", len(lines))

**Output :**

Q5. Write a Python program that reads a file and prints only those lines that contain the word “Python” (case-sensitive).

with open("sample.txt", "a") as f:

f.write("I am learning Python.\n")

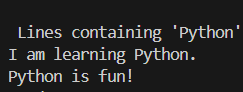
f.write("Python is fun!\n")

print("\n Lines containing 'Python':")

with open("sample.txt", "r") as f:

for line in f:

if "Python" in line:

print(line.strip())

**Output :**

Q6. Write a Python program to count the number of words and characters in the file `sample.txt`.

with open("sample.txt", "r") as f:

text = f.read()

words = text.split()

print("Word count:", len(words))

print("Character count:", len(text))

**Output :**

Q7. Write a program to copy the contents of `sample.txt` to another file `copy\_sample.txt`.

with open("sample.txt", "r") as source, open("copy\_sample.txt", "w") as destination:

destination.write(source.read())

print("Contents copied to 'copy\_sample.txt'")

**Output : **

## **Section C: Advanced File Handling (Q8 - Q10)**

Q8. Write a Python program to display the last 3 lines of a text file.

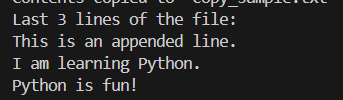
with open("sample.txt", "r") as f:

lines = f.readlines()

print("Last 3 lines of the file:")

for line in lines[-3:]:

print(line.strip())



**Output**

Q9. Write a Python program that reads numbers from a file `numbers.txt`, one per line, and with open("numbers.txt", "w") as f:

f.write("5\n12\n9\n22\n33\n40\n")

with open("numbers.txt", "r") as infile:

with open("even\_numbers.txt", "w") as outfile:

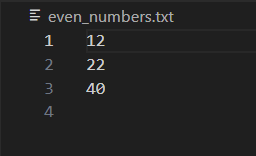
for line in infile:

number = line.strip()

if number.isdigit():

if int(number) % 2 == 0:

outfile.write(number + "\n")

**Output : **

Q10. Create a program that accepts user input (name, age, city) and stores it in a CSV file `users.csv`. Ensure that every new entry is stored on a new line.

import csv

name = input("Enter your name: ")

age = input("Enter your age: ")

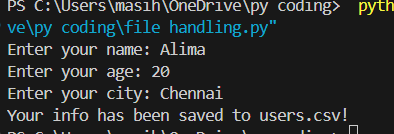
city = input("Enter your city: ")

with open("users.csv", "a", newline='') as f:

writer = csv.writer(f)

writer.writerow([name, age, city])

print("User data saved to 'users.csv'")

**Output : **