**DAY 3 TASKS**

**Task 21: Method Overriding**

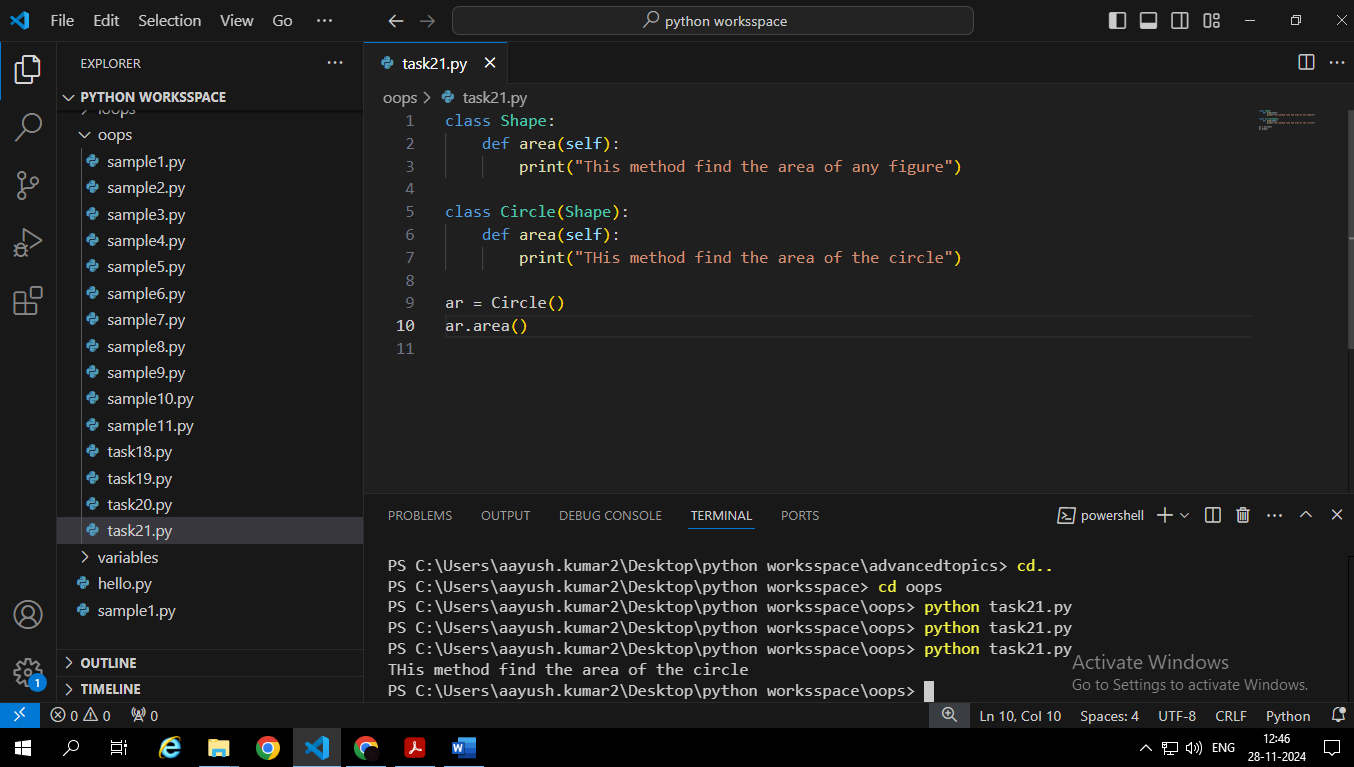
**Objective:** Learn how to override methods in a derived class.

**Steps:**

1. Create a base class Shape with a method area().

2. Create a derived class Circle and override the area() method.

3. Create an object of Circle and call the area() method.



**Question:** How would you call the method of the base class from the derived class?

To call the method of the base class from the derived class we use the **super()** function.

**A screenshot of a computer

Description automatically generated**

**Task 22 : Static Methods in Classes**

**Objective:** Learn how to use static methods in Python classes.

**Steps:**

1. Create a class Calculator with a static method add().

2. Call the static method without creating an instance of the class.

A screenshot of a computer

Description automatically generated

**Question:** What is the difference between static methods and instance methods?

**Static methods** are often used for utility functions that don't require access to specific object data, such as mathematical calculations or formatting operations.

**Instance methods** are used to perform actions on specific objects, such as modifying their properties or performing calculations based on their state.

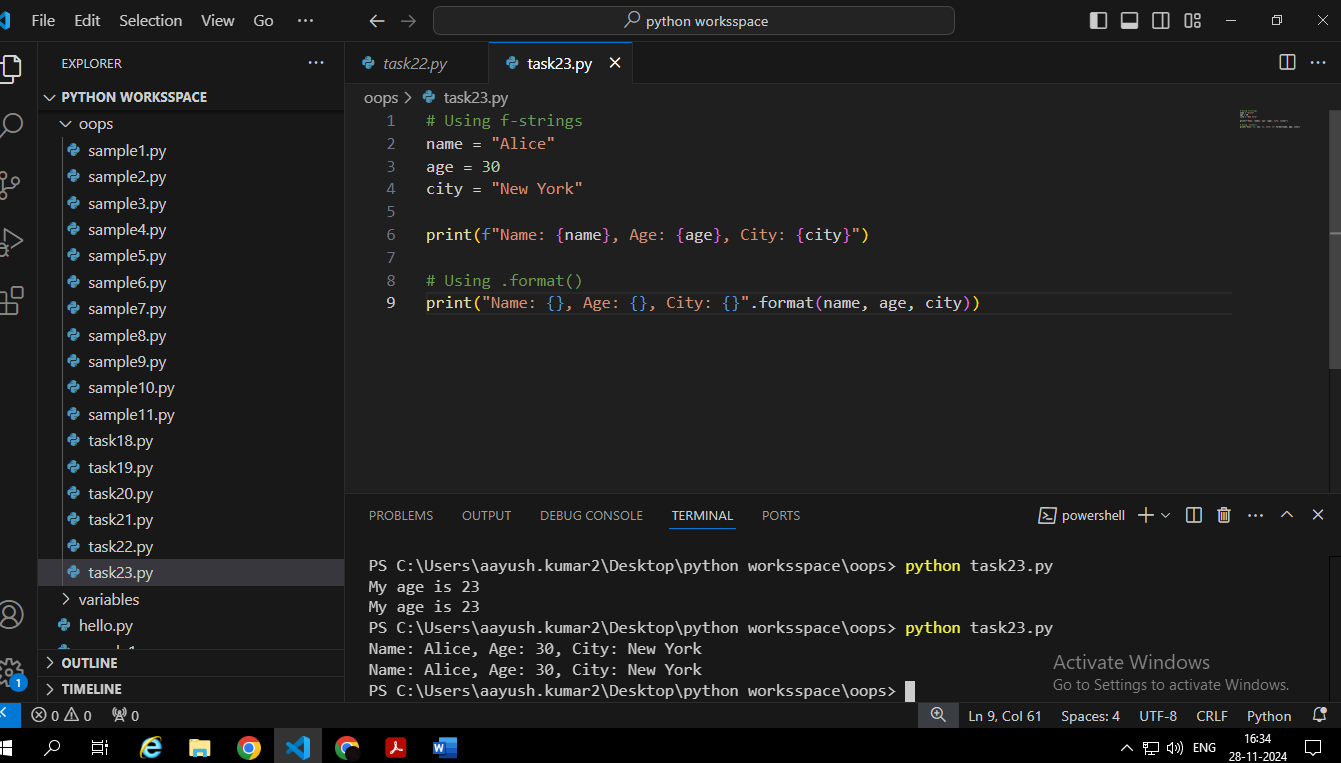
**Task 23: String Formatting**

**Objective:** Practice string formatting in Python.

 **Steps:**

1. Use f-strings to print variables in a formatted string.

2. Use .format() to achieve the same result.



**Question:** How does f-string formatting improve readability in comparison to other methods?

F-strings offer several advantages over .format():

* **Readability:** F-strings are more concise and easier to read, especially for complex formatting.
* **Conciseness:** They eliminate the need for extra formatting characters and braces.
* **Direct Variable Substitution:** You can directly embed variables within the string, making the code more intuitive.
* **Expression Evaluation:** You can perform calculations and expressions within f-strings, providing flexibility.

**Task 24: Reading and Writing Files**

**Objective:** Learn to read and write data to files.

**Steps:**

1. Write a program that reads data from a file.

2. Write a program that writes data to a file.

A screenshot of a computer

Description automatically generated

**Question:** What happens if you try to open a file that doesn’t exist?

If we try to open a file that doesn't exist in read mode ("r"), a **FileNotFoundError** will be raised.

**Task 25: List Comprehension**

**Objective:** Understand how to use list comprehensions.

**Steps:**

1. Write a list comprehension to create a list of squares from 1 to 10.

A screenshot of a computer

Description automatically generated

1. Use a conditional expression in the list comprehension to filter out odd numbers.

A screenshot of a computer

Description automatically generated

**Question:** How would you modify the list comprehension to create a list of even numbers?

A screenshot of a computer

Description automatically generated

**26. Task: Lambda Functions**

**Objective:** Learn how to use lambda functions for simple operations.

**Steps:**

1. Write a lambda function that returns the square of a number.

A screenshot of a computer

Description automatically generated

1. Use the lambda function in a map() operation to square all numbers in a list.

A screenshot of a computer

Description automatically generated

**Question:** How does a lambda function differ from a regular function in terms of usage?

Lambda functions are concise, anonymous functions defined in a single line, often used for short operations. Regular functions are more verbose, named functions suitable for complex logic and reusability.

While lambda functions offer conciseness, regular functions provide flexibility and structure for larger-scale programming.

**Task 27: Using zip() Function**

**Objective:** Understand how to use the zip() function.

**Steps:**

1. Create two lists and zip them together.

2. Convert the result into a list of tuples and print it.

A screenshot of a computer

Description automatically generated

**Question:** How can you use zip() to combine three lists?

A screenshot of a computer

Description automatically generated

**28. Task: Dictionary Comprehension**

**Objective:** Learn how to use dictionary comprehensions.

**Steps:**

1. Create a dictionary comprehension that creates key-value pairs of numbers and their squares.

2. Print the resulting dictionary.

A screenshot of a computer

Description automatically generated

**Question:** How would you modify the comprehension to create a dictionary where the key is the number and the value is the cube?

A screenshot of a computer

Description automatically generated

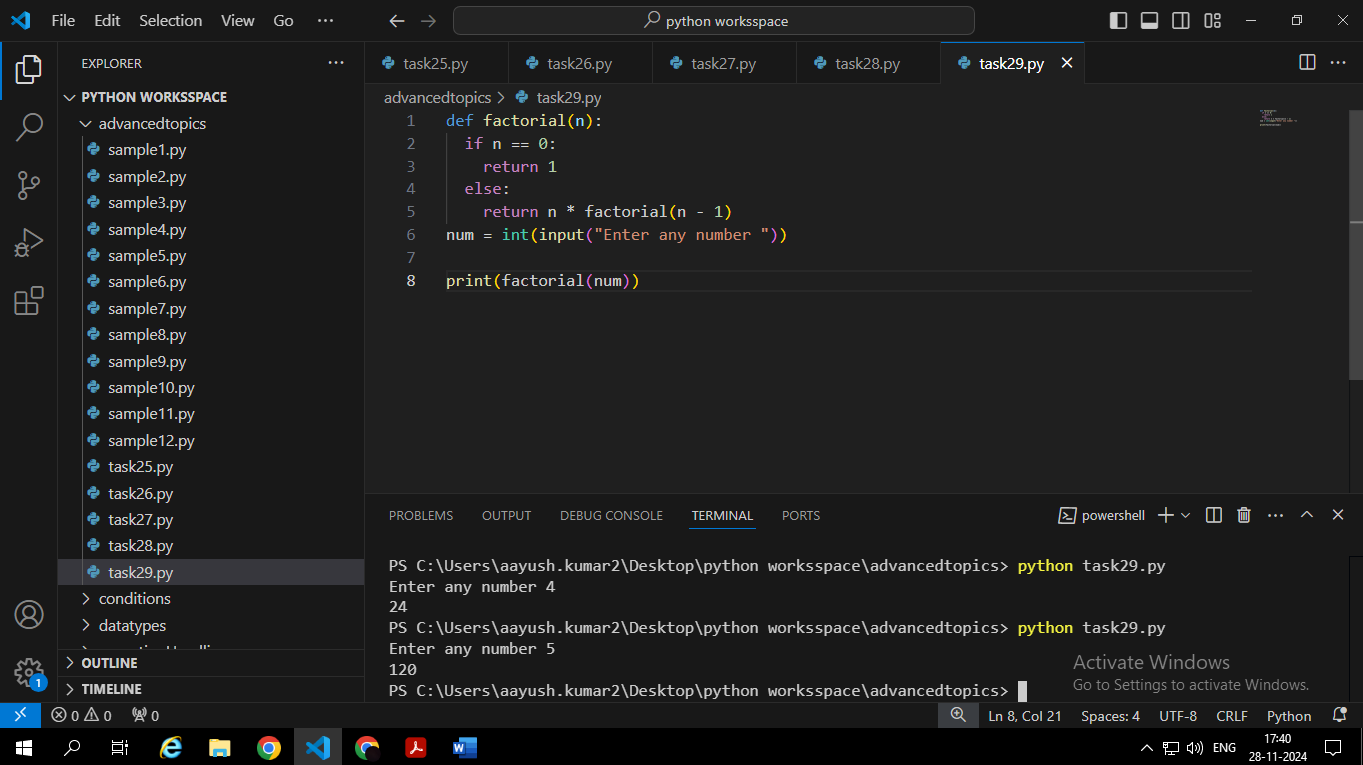
**Task 29: Recursion**

**Objective:** Understand the concept of recursion in Python.

**Steps:**

1. Write a recursive function to calculate the factorial of a number.

2. Call the function with different inputs.



**Question:** What happens if the base case of a recursive function is missing or incorrect?

If the base case is missing or incorrect, the recursion will continue indefinitely, leading to a **Stack Overflow** error.

**30. Task: Handling Multiple Exceptions**

**Objective:** Handle multiple exceptions in a program.

**Steps:**

1. Write a program that handles both ZeroDivisionError and ValueError.

2. Test the program with different inputs.

A screenshot of a computer

Description automatically generated

**Question:** How does the order of exception blocks affect the program's behavior?

The order of exception blocks affects behaviour as Python checks them sequentially. Specific exceptions should come first to handle targeted errors.

General exceptions (like Exception ) must follow to act as a fallback, avoiding them overriding more specific handlers.