

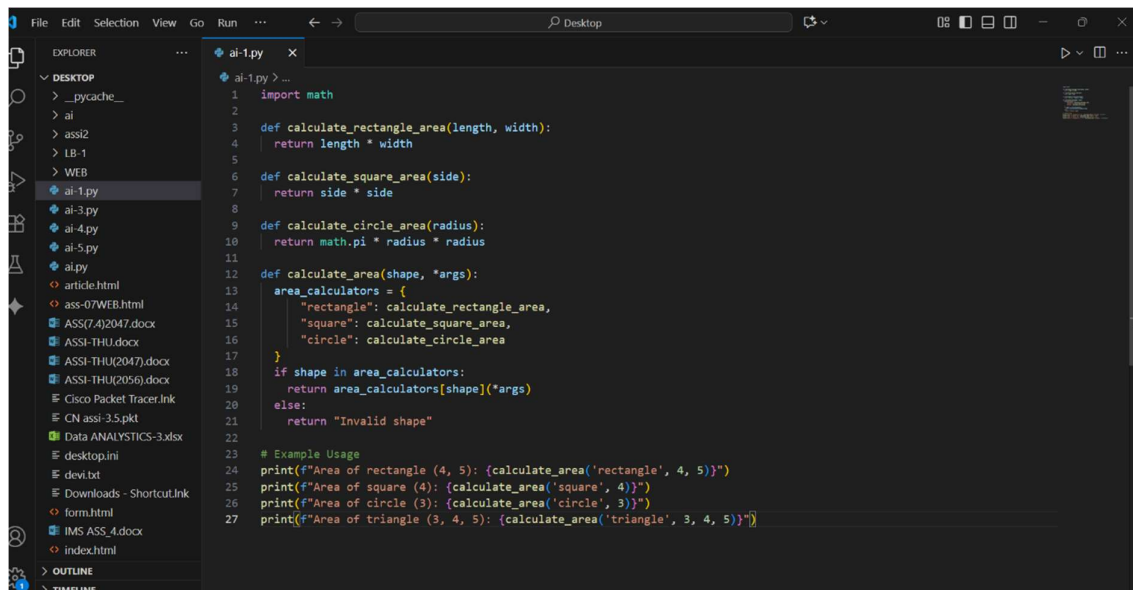
# ASSIGNMENT-13.2

NAME:DEVI PRIYA.G

2403A52067

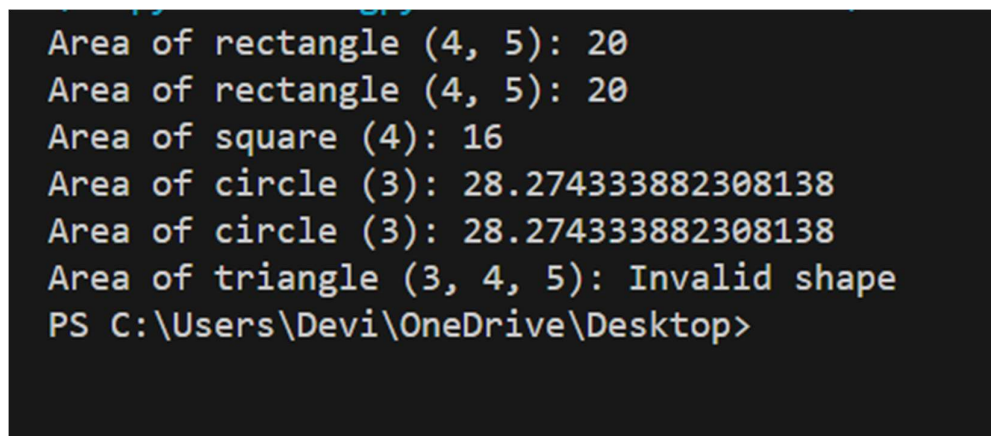
BATCH-04

TASK-1:



```
1 import math
2
3 def calculate_rectangle_area(length, width):
4     return length * width
5
6 def calculate_square_area(side):
7     return side * side
8
9 def calculate_circle_area(radius):
10    return math.pi * radius * radius
11
12 def calculate_area(shape, *args):
13    area_calculators = {
14        "rectangle": calculate_rectangle_area,
15        "square": calculate_square_area,
16        "circle": calculate_circle_area
17    }
18    if shape in area_calculators:
19        return area_calculators[shape](*args)
20    else:
21        return "Invalid shape"
22
23 # Example Usage
24 print(f"Area of rectangle (4, 5): {calculate_area('rectangle', 4, 5)}")
25 print(f"Area of square (4): {calculate_area('square', 4)}")
26 print(f"Area of circle (3): {calculate_area('circle', 3)}")
27 print(f"Area of triangle (3, 4, 5): {calculate_area('triangle', 3, 4, 5)}")
```

OUTPUT:



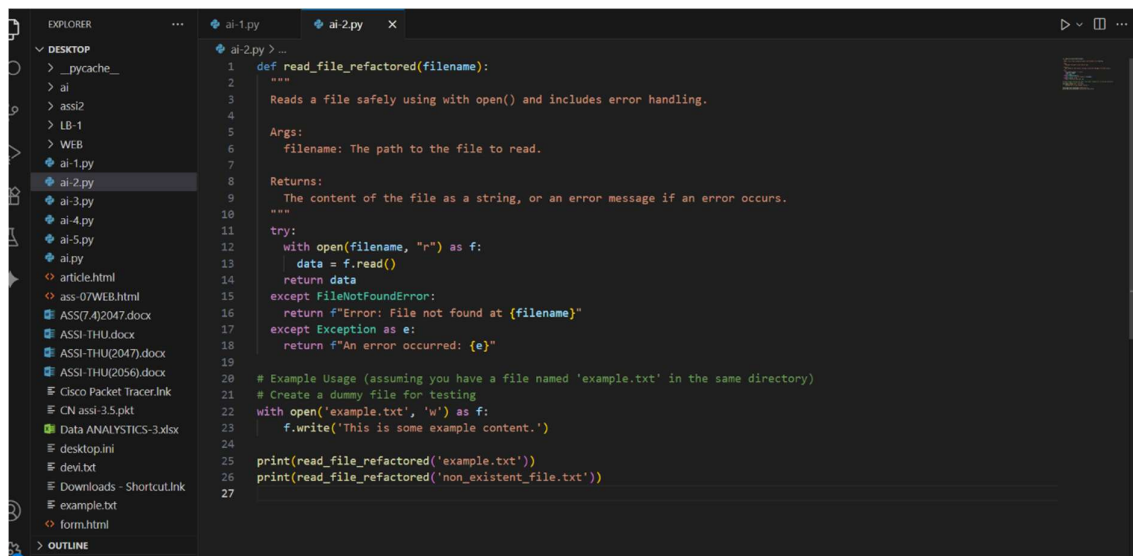
```
Area of rectangle (4, 5): 20
Area of rectangle (4, 5): 20
Area of square (4): 16
Area of circle (3): 28.274333882308138
Area of circle (3): 28.274333882308138
Area of triangle (3, 4, 5): Invalid shape
PS C:\Users\Devi\OneDrive\Desktop>
```

EXPLANATION:

1. **Import math:** This line imports the math module, which is needed to use math.pi for the circle area calculation.

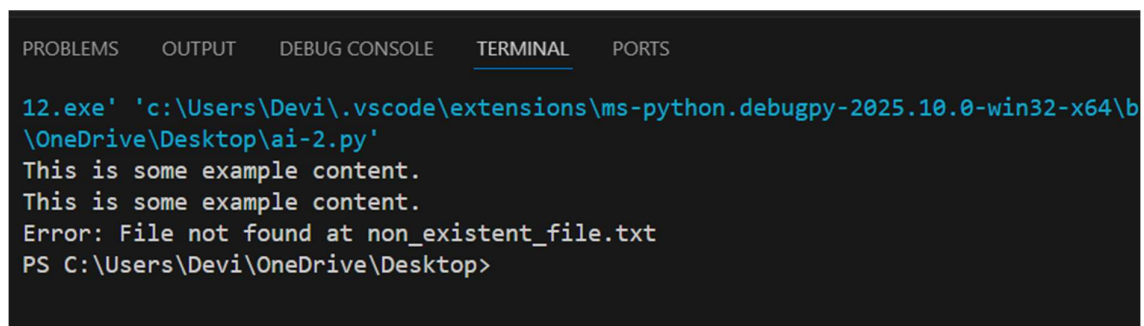
2. **def calculate\_rectangle\_area(length, width)::** This function takes length and width as arguments and returns their product, which is the area of a rectangle.
3. **def calculate\_square\_area(side)::** This function takes side as an argument and returns the square of the side, which is the area of a square.
4. **def calculate\_circle\_area(radius)::** This function takes radius as an argument and returns the area of a circle using the formula  $\pi * \text{radius}^2$ . `math.pi` provides the value of pi.

## TASK-2:



```
1 def read_file_refactored(filename):
2     """
3     Reads a file safely using with open() and includes error handling.
4
5     Args:
6         filename: The path to the file to read.
7
8     Returns:
9         The content of the file as a string, or an error message if an error occurs.
10    """
11    try:
12        with open(filename, "r") as f:
13            data = f.read()
14            return data
15    except FileNotFoundError:
16        return f"Error: File not found at {filename}"
17    except Exception as e:
18        return f"An error occurred: {e}"
19
20    # Example Usage (assuming you have a file named 'example.txt' in the same directory)
21    # Create a dummy file for testing
22    with open('example.txt', 'w') as f:
23        f.write('This is some example content.')
24
25    print(read_file_refactored('example.txt'))
26    print(read_file_refactored('non_existent_file.txt'))
27
```

## OUTPUT:

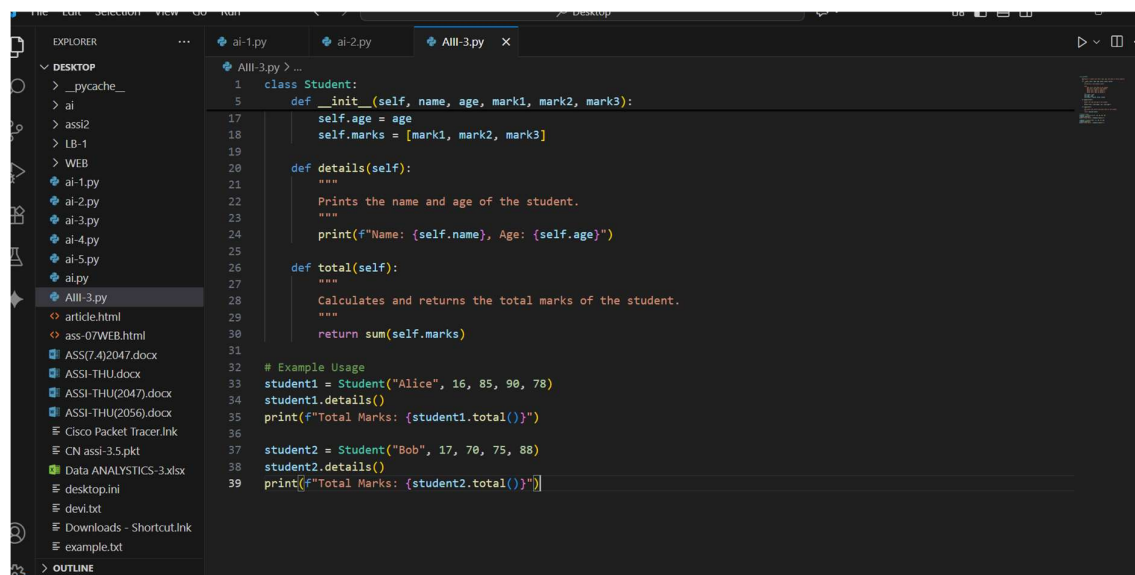


```
12.exe' 'c:\Users\Devi\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\b
\OneDrive\Desktop\ai-2.py'
This is some example content.
This is some example content.
Error: File not found at non_existent_file.txt
PS C:\Users\Devi\OneDrive\Desktop>
```

## EXPLANATION:

1. **def read\_file\_refactored(filename)::** This defines the function named `read_file_refactored` that takes one argument, `filename`, which is the path to the file you want to read.
2. **""" Docstring """:** The triple-quoted string is a docstring that explains what the function does, its arguments (Args), and what it returns (Returns). This is good practice for documenting your code.
3. **try::** This block starts a try...except block, which is used for error handling. Code within the try block is executed, and if an error occurs, the code within the corresponding except block is executed.

## TASK-3:



```
1 class Student:
2     def __init__(self, name, age, mark1, mark2, mark3):
3         self.age = age
4         self.marks = [mark1, mark2, mark3]
5
6     def details(self):
7         """
8         Prints the name and age of the student.
9         """
10        print(f"Name: {self.name}, Age: {self.age}")
11
12    def total(self):
13        """
14        Calculates and returns the total marks of the student.
15        """
16        return sum(self.marks)
17
18 # Example Usage
19 student1 = Student("Alice", 16, 85, 90, 78)
20 student1.details()
21 print(f"Total Marks: {student1.total()}")
22
23 student2 = Student("Bob", 17, 70, 75, 88)
24 student2.details()
25 print(f"Total Marks: {student2.total()}")
```

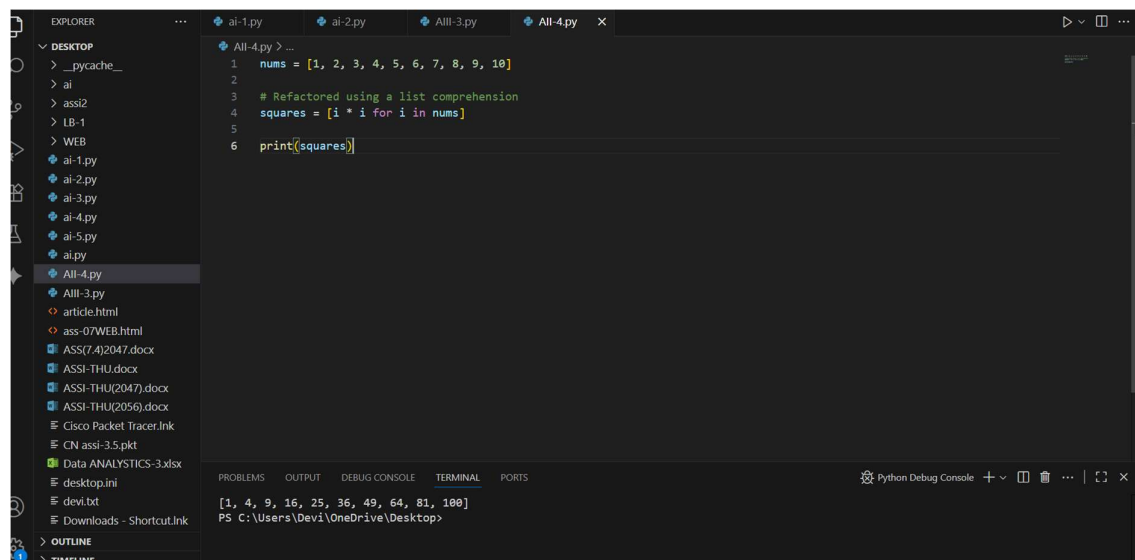
## OUTPUT:

```
PS C:\Users\Devi\OneDrive\Desktop> c::; cd 'c:\Users\Devi\OneDrive\Desktop'; & 'c:\User
12.exe' 'c:\Users\Devi\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled
\OneDrive\Desktop\AIII-3.py'
Name: Alice, Age: 16
Total Marks: 253
Name: Bob, Age: 17
Total Marks: 233
PS C:\Users\Devi\OneDrive\Desktop>
```

## EXPLANATION:

1. **class Student::** This line begins the definition of a class named Student. A class serves as a blueprint for creating objects that represent students.
2. **Class Docstring:** The triple-quoted string right after the class definition is a docstring that explains the overall purpose of the Student class – to represent a student's name, age, and marks.
3. **def \_\_init\_\_(self, name, age, mark1, mark2, mark3)::** This is the constructor method, named \_\_init\_\_. It's a special method that gets called automatically when you create a new Student object.

## TASK-4:



The screenshot shows a Python IDE with a file explorer on the left and a code editor in the center. The file explorer lists various files, including Python scripts (ai-1.py to ai-5.py, All-4.py, All-3.py), HTML files (article.html, ass-07WEB.html), Word documents (ASS(7.4)2047.docx, ASSI-THU.docx, ASSI-THU(2047).docx, ASSI-THU(2056).docx), a Cisco Packet Tracer file (Cisco Packet Tracer.Ink), a PDF file (CN assi-3.5.pkt), an Excel file (Data ANALYTICS-3.xlsx), and a desktop.ini file. The code editor displays the following Python code:

```
1  nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3  # Refactored using a list comprehension
4  squares = [i * i for i in nums]
5
6  print(squares)]
```

The bottom of the IDE shows a terminal window with the output of the code:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
PS C:\Users\Dev1\OneDrive\Desktop>
```

## EXPLANATION:

The selected code snippet initializes a Python list named `nums` with integers from 1 to 10. The comment `# Refactored using a list comprehension` indicates that the following line of code will use a list comprehension to perform an operation on this list. In this specific case, the list comprehension is used to calculate the square of each number in the `nums` list.