ASSIGNMENT-13.2

NAME:DEVI PRIYA.G

2403A52067

BATCH-04

TASK-1:

```python
import math

def calculate_rectangle_area(length, width):
    return length * width

def calculate_square_area(side):
    return side * side

def calculate_circle_area(radius):
    return math.pi * radius * radius

def calculate_area(shape, *args):
    area_calculators = {
        "rectangle": calculate_rectangle_area,
        "square": calculate_square_area,
        "circle": calculate_circle_area
    }
    if shape in area_calculators:
        return area_calculators[shape](*args)
    else:
        return "Invalid shape"

# Example Usage
print(f"Area of rectangle (4, 5): {calculate_area('rectangle', 4, 5)}")
print(f"Area of square (4): {calculate_area('square', 4)}")
print(f"Area of circle (3): {calculate_area('circle', 3)}")
print(f"Area of triangle (3, 4, 5): {calculate_area('triangle', 3, 4, 5)}")
```

OUTPUT:

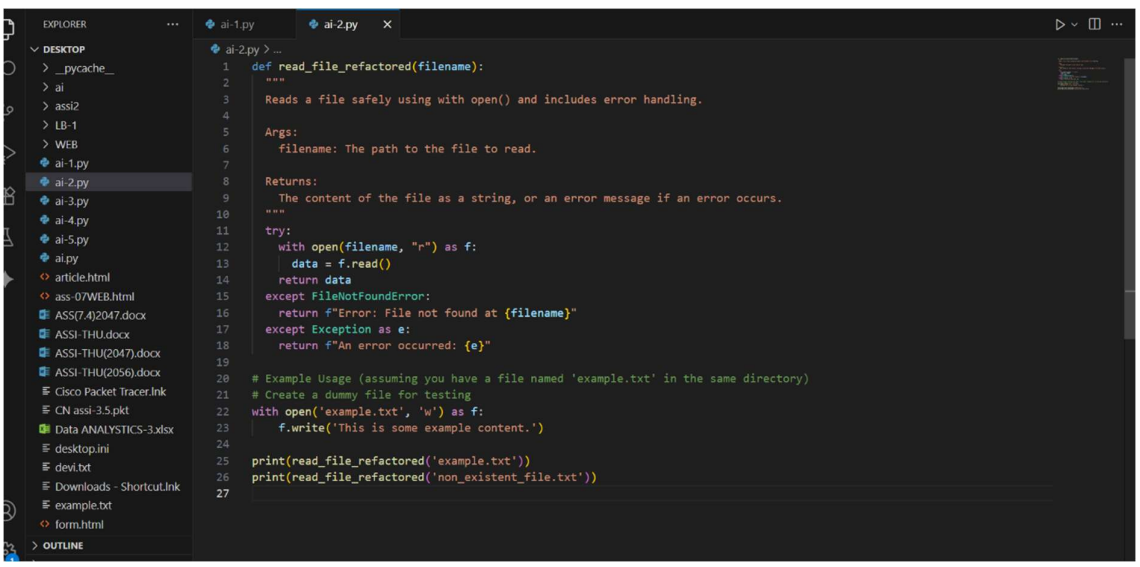
```
Area of rectangle (4, 5): 20
Area of rectangle (4, 5): 20
Area of square (4): 16
Area of circle (3): 28.274333882308138
Area of circle (3): 28.274333882308138
Area of triangle (3, 4, 5): Invalid shape
PS C:\Users\Devi\OneDrive\Desktop>
```

EXPLANATION:

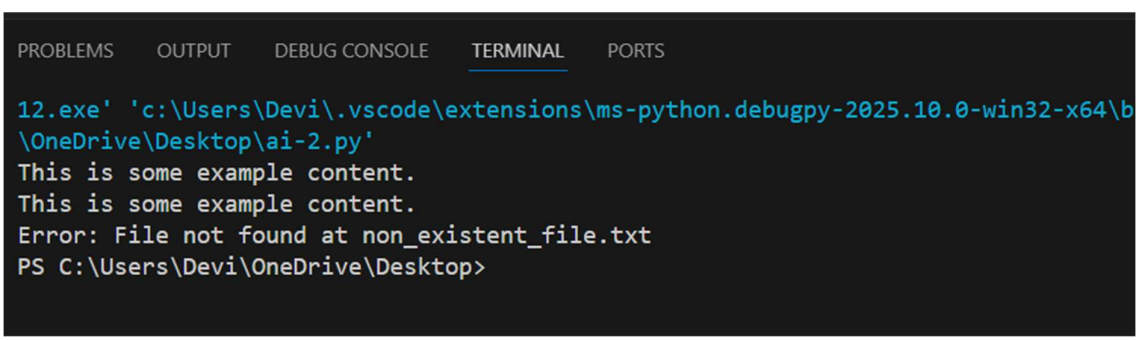
1. **Import math**: This line imports the math module, which is needed to use math.pi for the circle area calculation.

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 π * radius². math.pi provides the value of pi.

TASK-2:



OUTPUT:



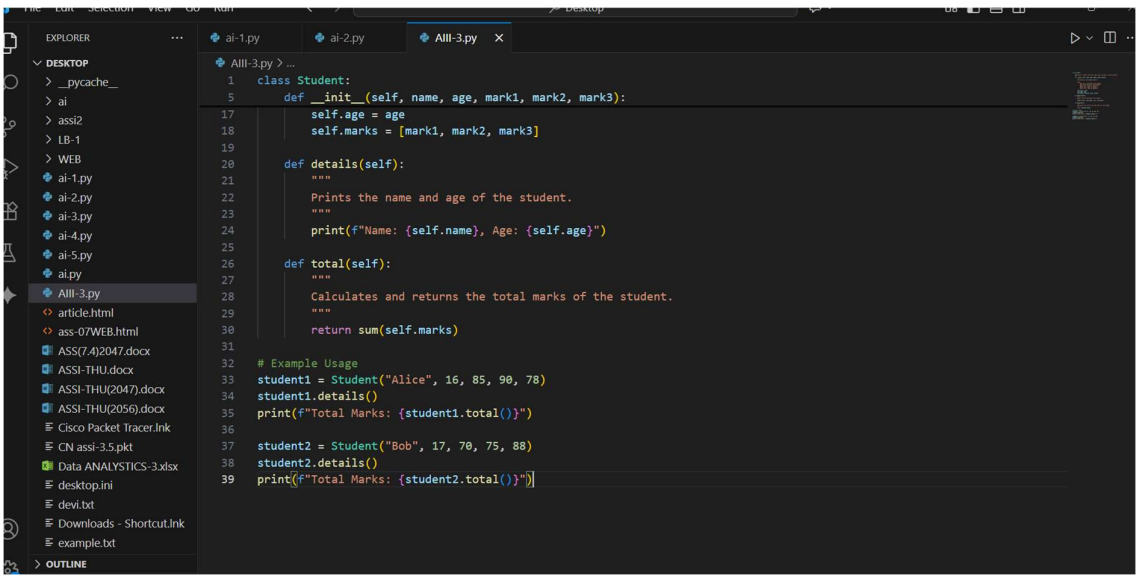
EXPLANATION:

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:



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>
```

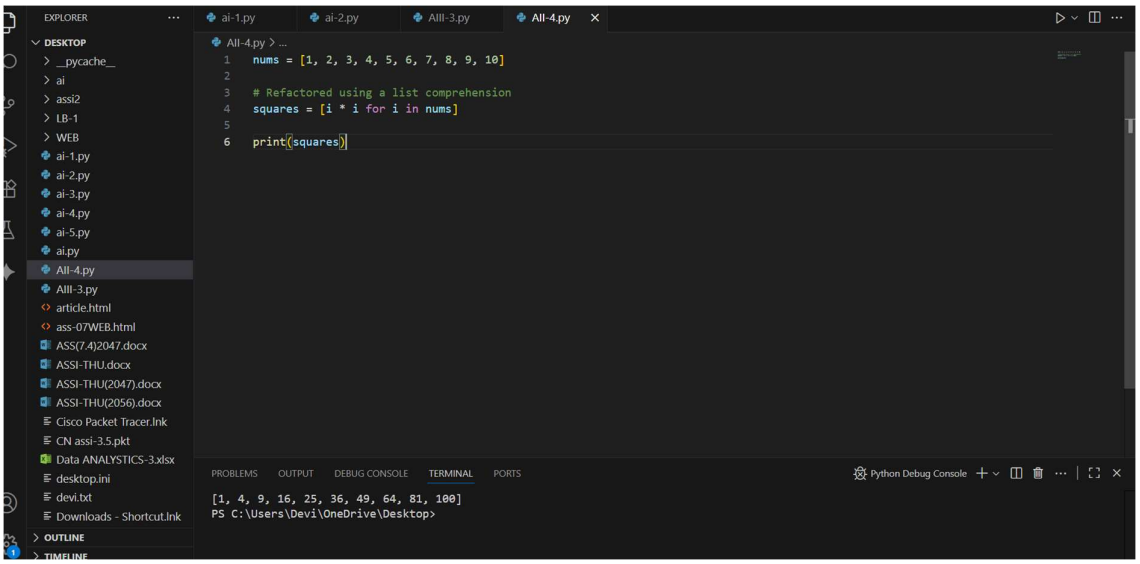
EXPLANATION:

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:



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.