



Name: Musfira Ahmed

Intern ID: TN/IN01/PY/002

Email ID : ahmedmusfira3@gmail.com

Internship Domain : Python Development

Task Week : 05

Instructor Name : Mr.Hassan Ali

Task 1 :

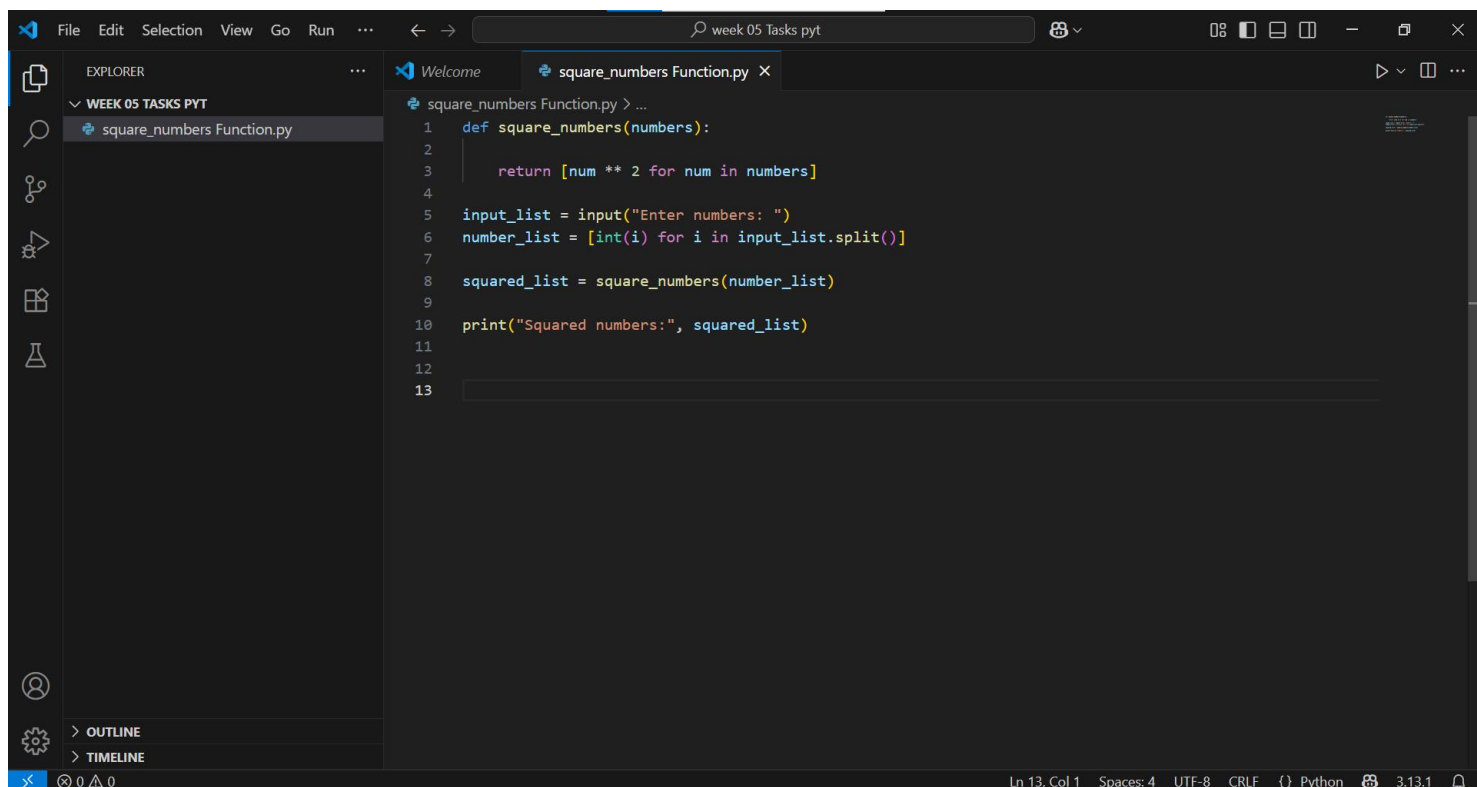
Create a function `square_numbers` that takes a list of numbers and returns a list of their squares.

Solution :

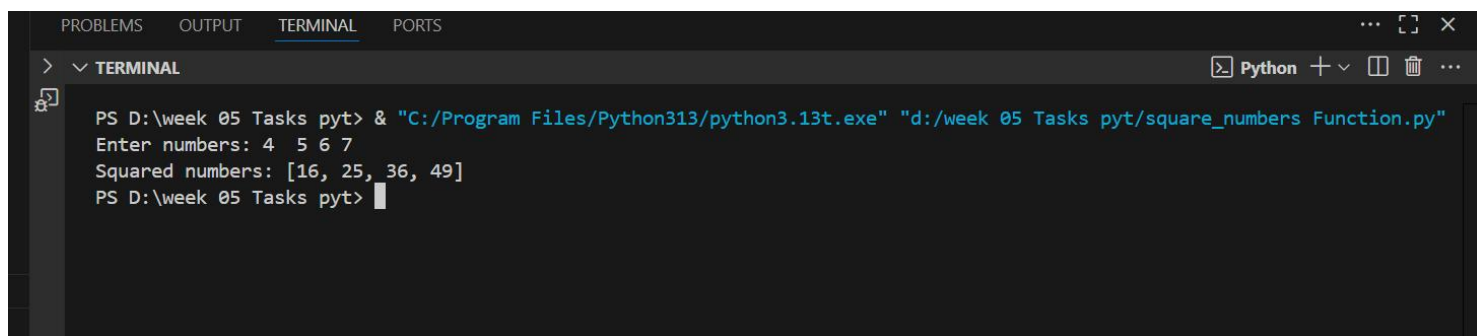
What I Did (Step by Step):

- Defined a function called `square_numbers(numbers)`.
- Took space-separated input from the user and converted it to a list.
- Called the function with the user's list as input.
- Stored the result in a variable `squared_list`.
- Printed the squared numbers clearly.

Code Screenshot



Output Screenshot



Learning and Challenges:

1. Learned how to write reusable functions in Python.
2. Practiced converting string input into a list of integers.
3. Faced a small challenge with `split()` and `int()` conversion.
4. Fixed it using list comprehension to handle all numbers easily.
5. Improved understanding of how to work with lists and functions together.

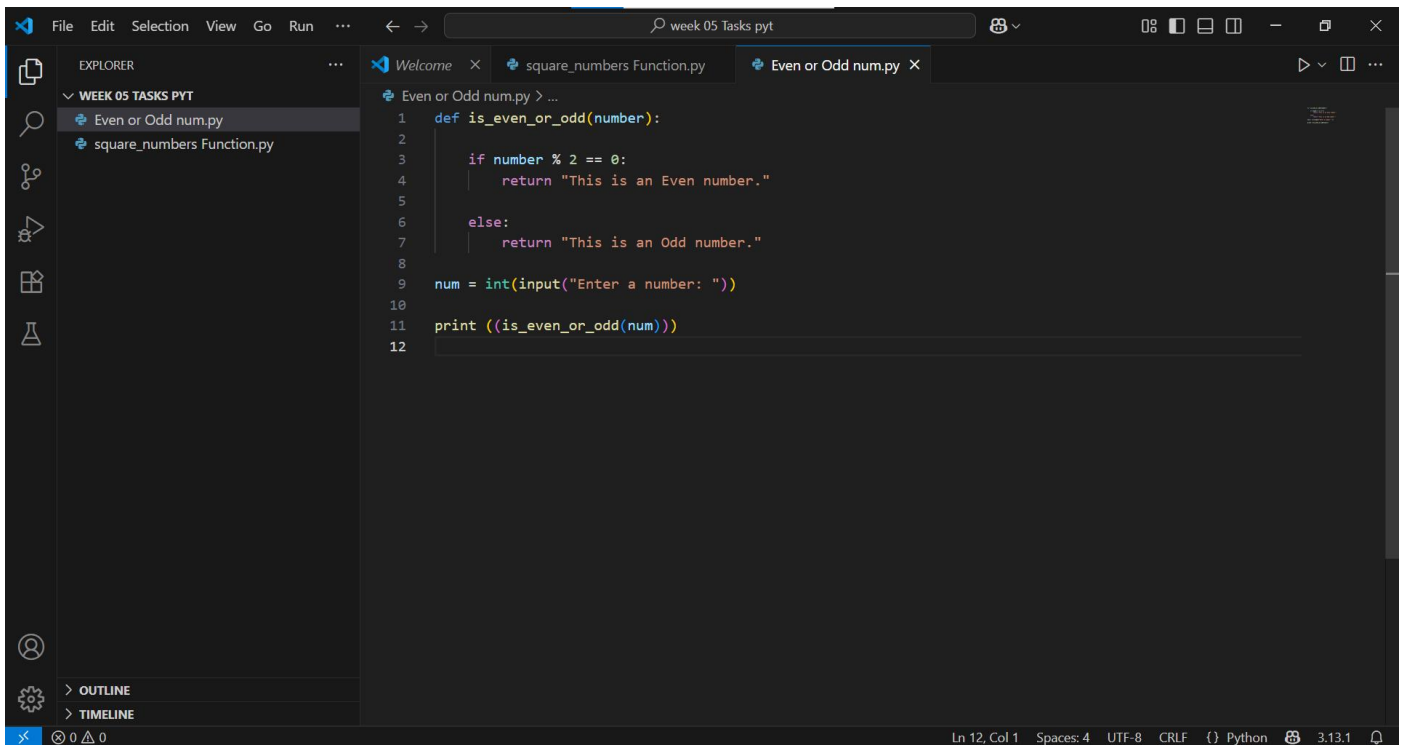
Task 02:

Create a function ``is_even_or_odd`` that takes a number and returns whether it is even or odd.

What I Did (Step by Step):

- Created a function called `is_even_or_odd(number)`.
- Used the modulus `%` operator to check if the number is divisible by 2.
- Returned "Even" or "Odd" based on the result.
- Took input from the user using `input()`.
- Printed whether the number is even or odd.

Code Screenshots

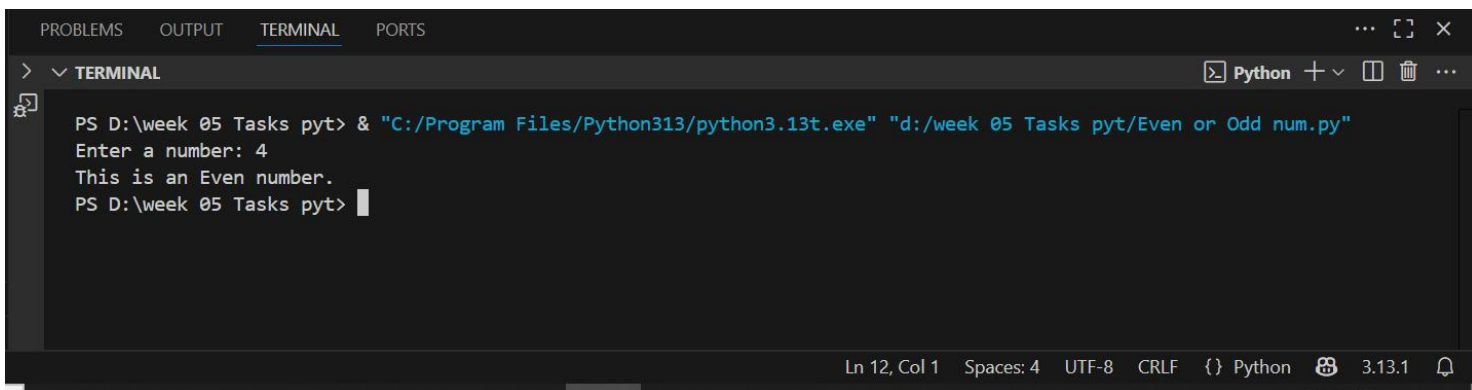


The screenshot shows a code editor with three tabs: 'Welcome', 'square_numbers Function.py', and 'Even or Odd num.py'. The 'Even or Odd num.py' tab is active, displaying the following Python code:

```
1 def is_even_or_odd(number):
2
3     if number % 2 == 0:
4         return "This is an Even number."
5
6     else:
7         return "This is an Odd number."
8
9 num = int(input("Enter a number: "))
10
11 print ((is_even_or_odd(num)))
12
```

The left sidebar shows the 'EXPLORER' view with a folder named 'WEEK 05 TASKS PYT' containing two files: 'Even or Odd num.py' and 'square_numbers Function.py'. The bottom status bar indicates 'Ln 12, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and '3.13.1'.

Output Screenshot



The screenshot shows a terminal window with the following output:

```
PS D:\week 05 Tasks pyt> & "C:/Program Files/Python313/python3.13t.exe" "d:/week 05 Tasks pyt/Even or Odd num.py"
Enter a number: 4
This is an Even number.
PS D:\week 05 Tasks pyt>
```

The terminal window has tabs for 'PROBLEMS', 'OUTPUT', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active. The bottom status bar indicates 'Ln 12, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and '3.13.1'.

Learning and Challenges:

- 1) Learned how to check even/odd using %.
- 2) Practiced writing and calling a function with one input.
- 3) Faced no issues, logic was easy and clear.
- 4) Understood how return values work inside a function.
- 5) Improved my conditional thinking and function writing skills.

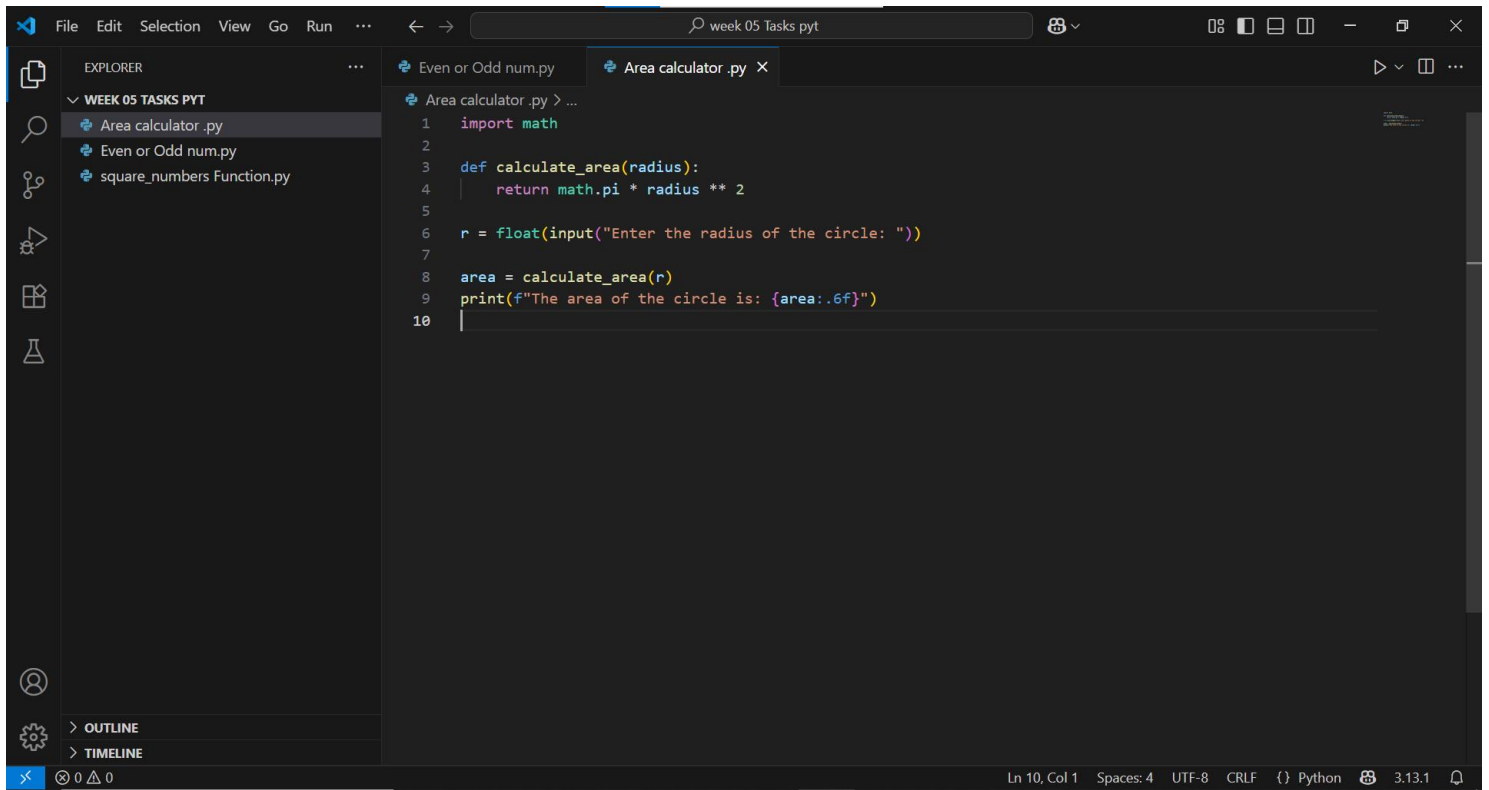
Task 03:

Write a function `calculate_area` that takes radius and returns area of a circle.

What I Did (Step by Step):

- Imported the math module to use math.pi.
- Created a function called calculate_area(radius).
- Used the formula $\pi \times \text{radius}^2$ to calculate area.
- Took radius input from the user.
- Called the function and printed the result.

Code Screenshots

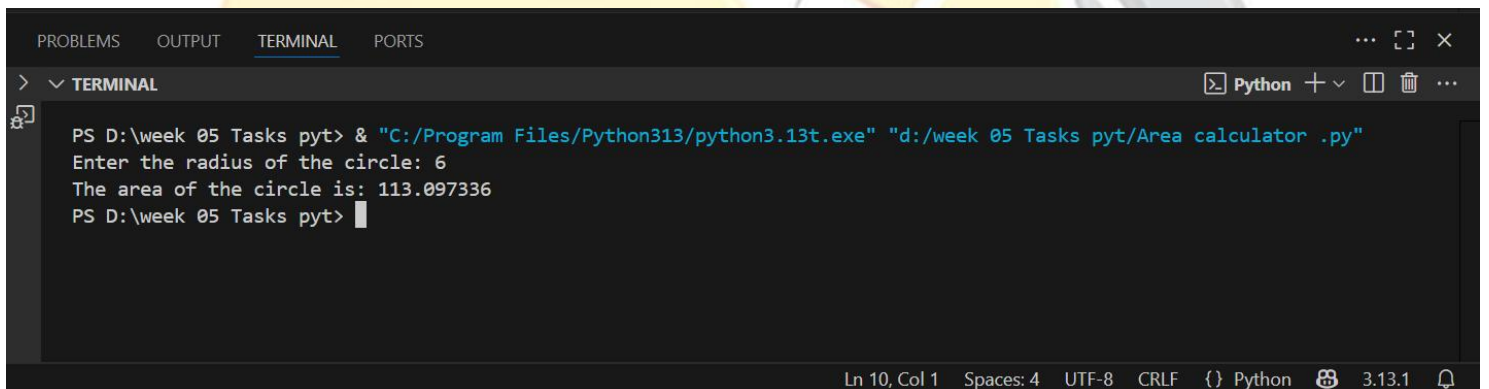


The screenshot shows a code editor with the following content:

```
File Edit Selection View Go Run ... week 05 Tasks.pyt
EXPLORER
WEEK 05 TASKS.PYT
  Area calculator .py
  Even or Odd num.py
  square_numbers Function.py
  OUTLINE
  TIMELINE
Area calculator .py x
1 import math
2
3 def calculate_area(radius):
4     return math.pi * radius ** 2
5
6 r = float(input("Enter the radius of the circle: "))
7
8 area = calculate_area(r)
9 print(f"The area of the circle is: {area:.6f}")
10
```

Ln 10, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1

Output Screenshot



The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT TERMINAL PORTS
> v TERMINAL Python + v
PS D:\week 05 Tasks.pyt> & "C:/Program Files/Python313/python3.13t.exe" "d:/week 05 Tasks.pyt/Area calculator .py"
Enter the radius of the circle: 6
The area of the circle is: 113.097336
PS D:\week 05 Tasks.pyt>
```

Ln 10, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1

TECHNIK NEST

Learnings and Challenges:

- 1) Learned how to use the math module in Python.
- 2) Practiced writing functions with mathematical formulas.
- 3) Faced no issues, just used correct input and float type.
- 4) Understood how return values work in real-world formulas.
- 5) Improved confidence in writing math-based functions.

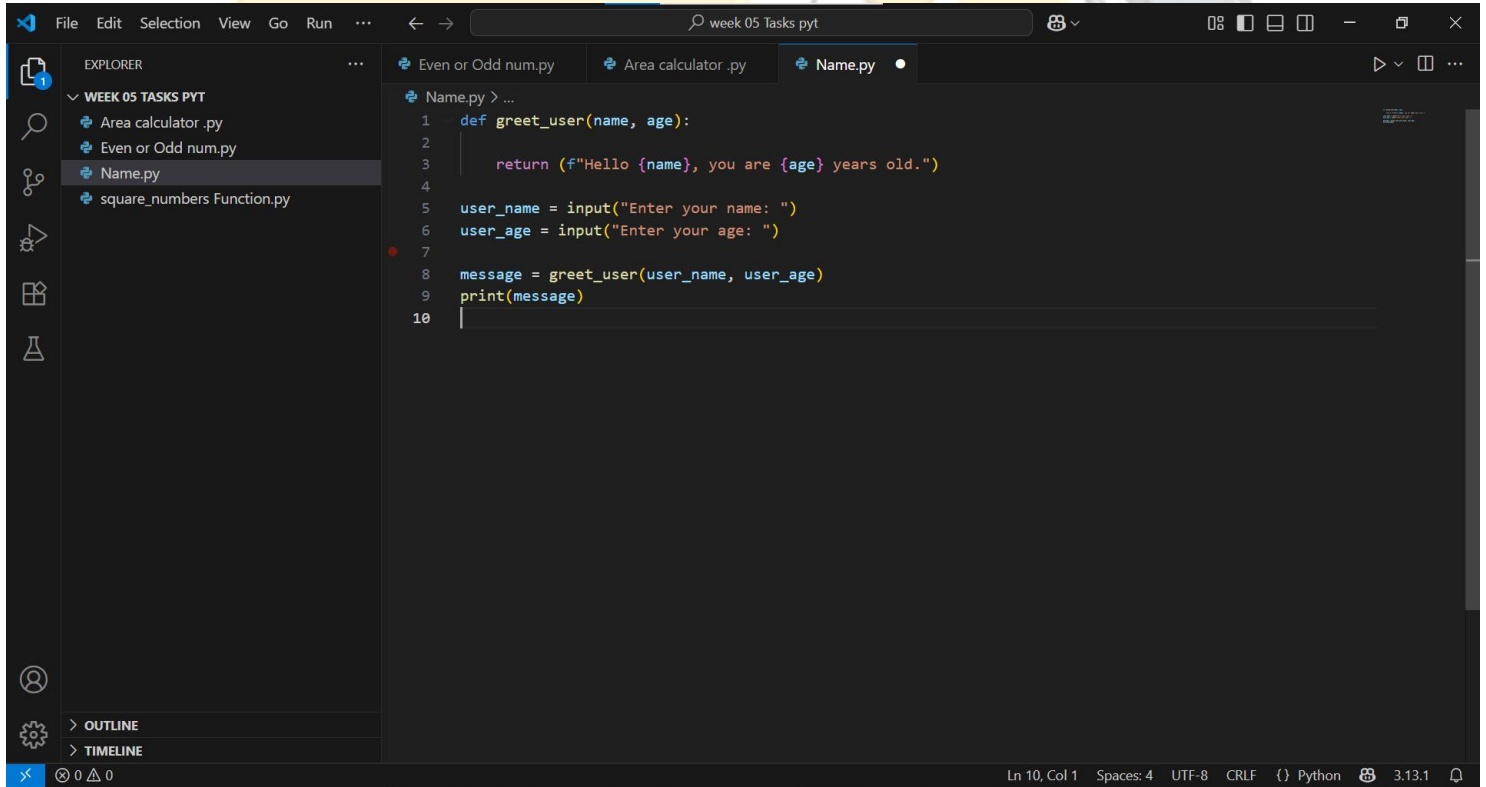
Task 04:

Write a function ``greet_user(name, age)`` that returns a greeting like: 'Hello Ali, you are 20 years old.'

What I Did (Step by Step):

- Defined a function `greet_user(name, age)` with two parameters.
- Used an f-string to format the greeting message.
- Took name and age as input from the user.
- Called the function with those inputs.
- Printed the personalized message.

Code Screenshots



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left displays a project named 'WEEK 05 TASKS PYT' containing four files: 'Area calculator .py', 'Even or Odd num.py', 'Name.py', and 'square_numbers Function.py'. The 'Name.py' file is selected and its code is visible in the main editor. The code defines a function 'greet_user' that takes 'name' and 'age' as arguments and returns a formatted string. It then prompts the user for their name and age, calls the function, and prints the result.

```
1 def greet_user(name, age):
2     return (f"Hello {name}, you are {age} years old.")
3
4
5 user_name = input("Enter your name: ")
6 user_age = input("Enter your age: ")
7
8 message = greet_user(user_name, user_age)
9 print(message)
10
```

The status bar at the bottom indicates the current position is Line 10, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.13.1 interpreter, and a warning icon.

Output Screenshots



The screenshot shows the VS Code terminal window with the 'TERMINAL' tab selected. The terminal displays the command to run the script and its output. The user enters 'Musfira' for the name and '19' for the age, resulting in the message 'Hello Musfira, you are 19 years old.'.

```
PS D:\week 05 Tasks pyt> & "C:/Program Files/Python313/python3.13t.exe" "d:/week 05 Tasks pyt/Name.py"
Enter your name: Musfira
Enter your age: 19
Hello Musfira, you are 19 years old.
PS D:\week 05 Tasks pyt>
```

The status bar at the bottom indicates the current position is Line 6, Column 37, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.13.1 interpreter, and a warning icon.

Learning and Challenges

1. Practiced using f-strings for clean formatting.
2. Faced no issues logic was clear and direct.
3. Understood how to return and use strings from a function.
4. Improved basic input/output handling and formatting.
5. Learned how to use multiple arguments in a function.

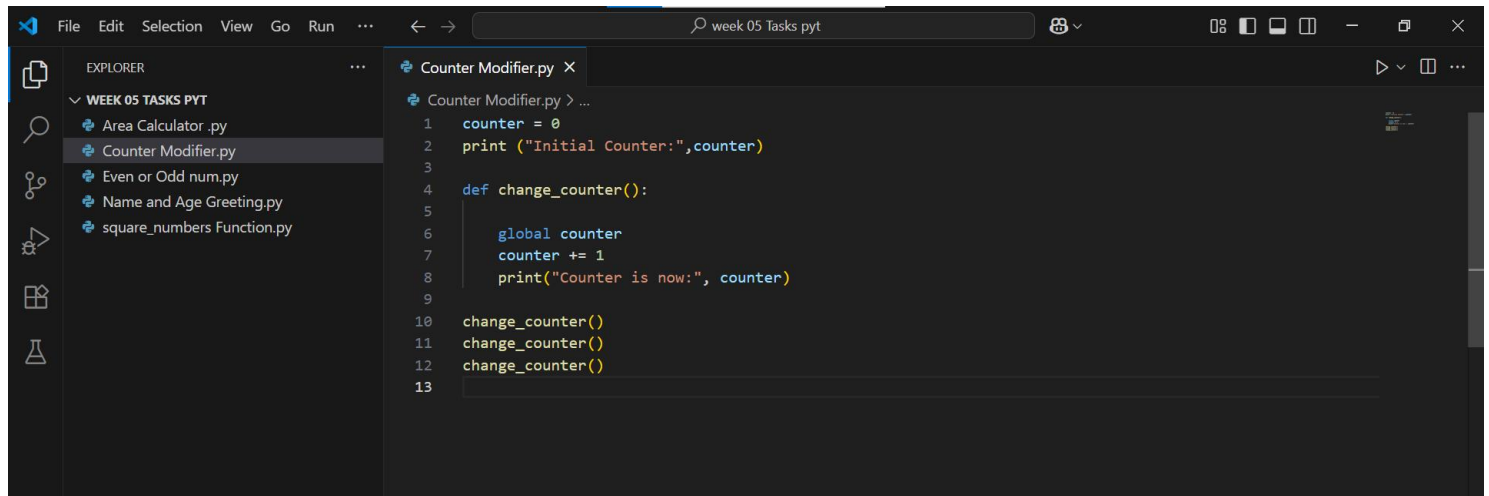
Task 05:

Create a function `change_counter()` that modifies a global counter variable.

What I Did (Step by Step):

- Declared a global variable counter with initial value 0.
- Created a function `change_counter()` that uses global keyword.
- Increased the value of counter by 1 inside the function.
- Called the function multiple times to show the change.
- Printed the updated counter value after each call.

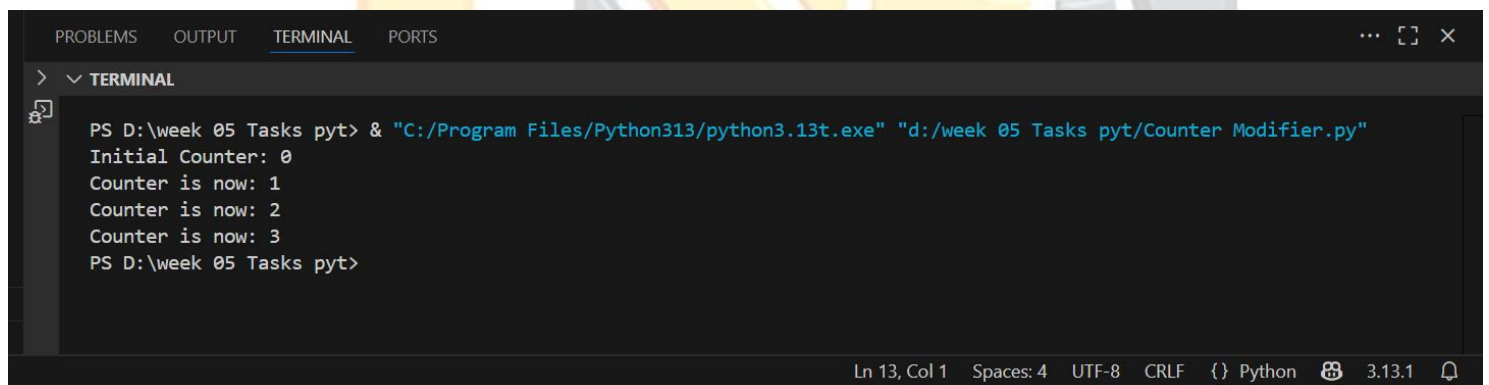
Code Screenshots



The screenshot shows a code editor with a dark theme. The Explorer panel on the left lists files under 'WEEK 05 TASKS PYT', including 'Counter Modifier.py'. The main editor window displays the code for 'Counter Modifier.py'. The code initializes a global 'counter' variable to 0, prints its initial value, defines a 'change_counter()' function that increments the global counter and prints its new value, and then calls the function three times.

```
1 counter = 0
2 print ("Initial Counter:",counter)
3
4 def change_counter():
5
6     global counter
7     counter += 1
8     print("Counter is now:", counter)
9
10 change_counter()
11 change_counter()
12 change_counter()
13
```

Output Screenshots



The screenshot shows a terminal window with the command prompt. The command executed is 'PS D:\week 05 Tasks pyt> & "C:/Program Files/Python313/python3.13t.exe" "d:/week 05 Tasks pyt/Counter Modifier.py"'. The output shows the initial counter value of 0, followed by three increments to 1, 2, and 3, each printed as 'Counter is now: [value]'. The terminal window also shows the file path and the command used to run the script.

```
PS D:\week 05 Tasks pyt> & "C:/Program Files/Python313/python3.13t.exe" "d:/week 05 Tasks pyt/Counter Modifier.py"
Initial Counter: 0
Counter is now: 1
Counter is now: 2
Counter is now: 3
PS D:\week 05 Tasks pyt>
```

Learning and Challenges

1. Learned how to use and modify global variables in a function.
2. Practiced using the global keyword correctly.
3. Faced no issues syntax is simple and clear.
4. Understood the difference between local and global scope.
5. Gained better control over how variables behave across code.

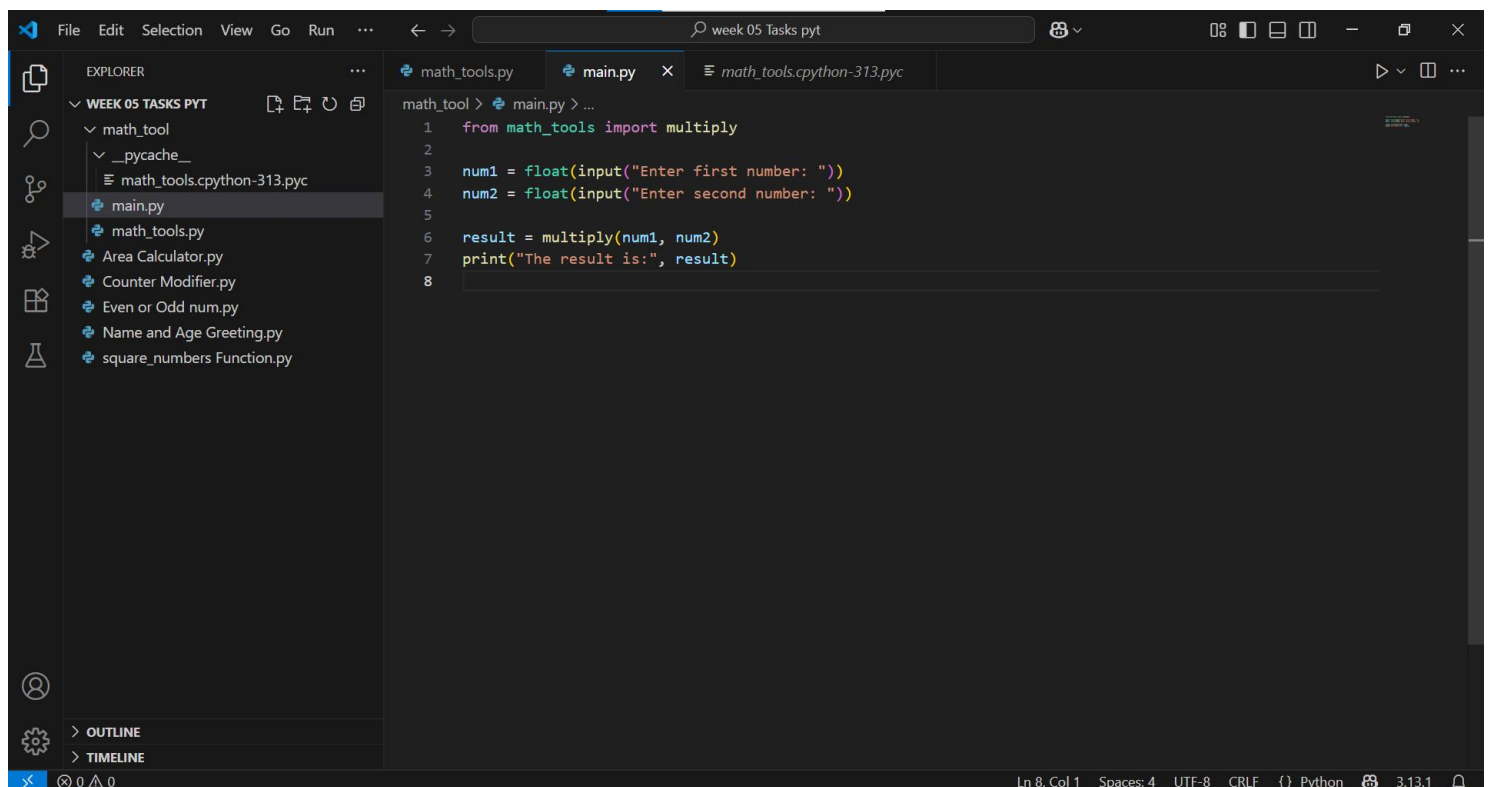
Task 06:

Create a module named `math_tools.py` with a function `multiply(x, y)` and use it in another script.

What I Did (Step by Step):

- Created a module file `math_tools.py` with a function `multiply(x, y)`.
- Wrote a second file `main.py` to import and use that function.
- Took two numbers from user input using `input()`.
- Called the function and stored the result.
- Printed the final result clearly.

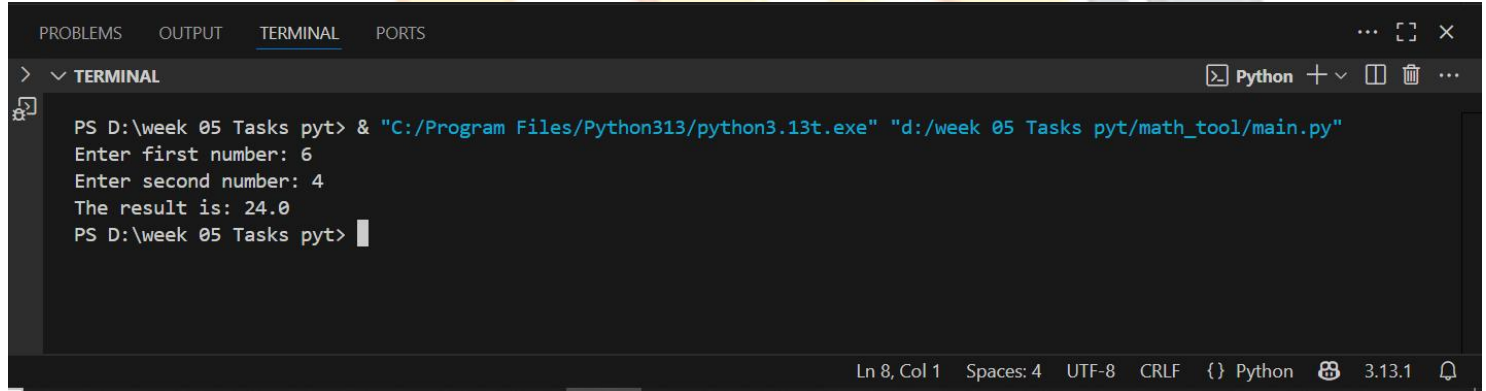
Code Screenshots



The screenshot shows a Python IDE with two files open: `math_tools.py` and `main.py`. The `EXPLORER` pane on the left shows a project structure with a folder `WEEK 05 TASKS PYT` containing `math_tool` (with `__pycache__` and `math_tools.cpython-313.pyc`) and `main.py`. The `main.py` file is selected, and its content is displayed in the editor. The code in `main.py` imports the `multiply` function from `math_tools`, takes two numbers as input, calls the `multiply` function, and prints the result.

```
1 from math_tools import multiply
2
3 num1 = float(input("Enter first number: "))
4 num2 = float(input("Enter second number: "))
5
6 result = multiply(num1, num2)
7 print("The result is:", result)
8
```

Output Screenshots



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is active), and 'PORTS'. Below the tabs, the terminal displays the following text:

```
PS D:\week 05 Tasks pyt> & "C:/Program Files/Python313/python3.13t.exe" "d:/week 05 Tasks pyt/math_tool/main.py"
Enter first number: 6
Enter second number: 4
The result is: 24.0
PS D:\week 05 Tasks pyt> |
```

At the bottom of the terminal window, there is a status bar that reads: 'Ln 8, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1'.

TECHNIK NEST

Learning and Challenges

1. Learned how to build and import custom modules in Python.
2. Practiced separating code into reusable files.
3. Faced an import error when files weren't in the same folder.
4. Solved it by keeping both files together.
5. Understood the use of modular code for cleaner projects.