



TECHNIK NEST

INNOVATIVE MINDS, NESTING SUCCESS

Name: Maaz Sher Muhammad

Intern ID: TN/1N01/003

Email ID : maazshermuhammadofficial@gmail.com

Internship Domain : Python Internee

Task Week : 5

Instructor Name : Hassan Ali

Task 1 :

Code Snippet & Screenshot

```
def square_numbers(numbers):
```

```
    squared = []
```

```
    for num in numbers:
```

```
        squared.append(num ** 2)
```

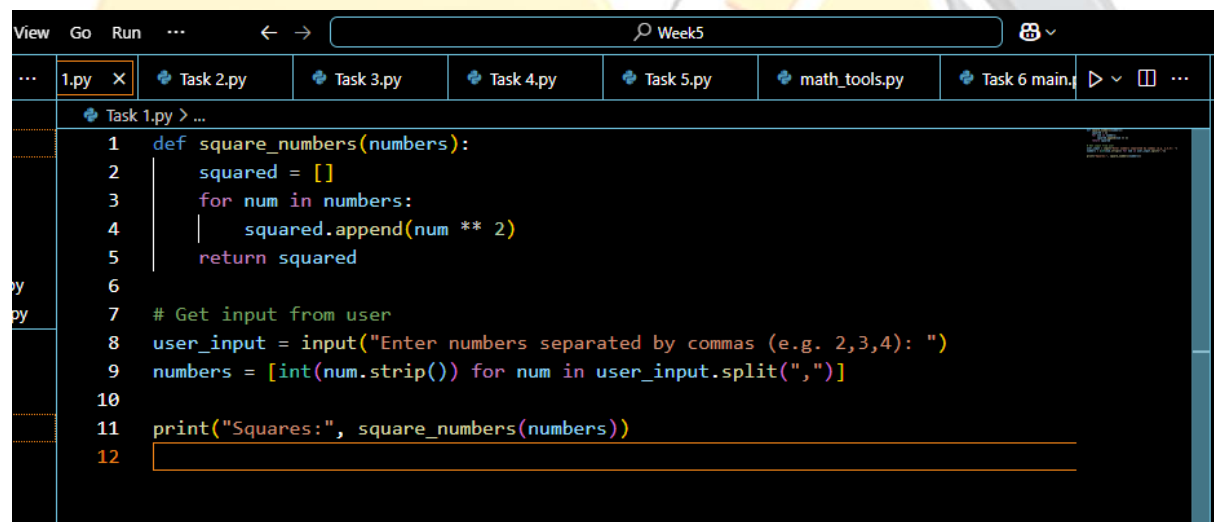
```
    return squared
```

```
# Get input from user
```

```
user_input = input("Enter numbers separated by commas (e.g. 2,3,4): ")
```

```
numbers = [int(num.strip()) for num in user_input.split(",")]
```

```
print("Squares:", square_numbers(numbers))
```



```
View Go Run ... < -> Week5
... 1.py x Task 2.py Task 3.py Task 4.py Task 5.py math_tools.py Task 6 main.
Task 1.py > ...
1 def square_numbers(numbers):
2     squared = []
3     for num in numbers:
4         squared.append(num ** 2)
5     return squared
6
7 # Get input from user
8 user_input = input("Enter numbers separated by commas (e.g. 2,3,4): ")
9 numbers = [int(num.strip()) for num in user_input.split(",")]
10
11 print("Squares:", square_numbers(numbers))
12
```

```
Enter numbers separated by commas (e.g. 2,3,4): 5,7,8
Squares: [25, 49, 64]
```

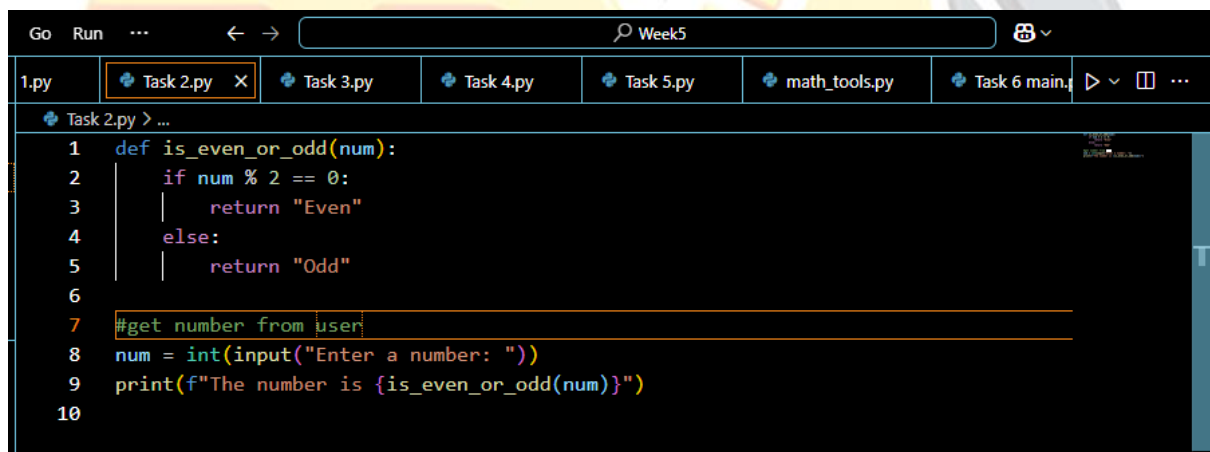
Challenges:

- ☐ Had to learn how to **split and clean user input** from a string to integers.
- ☐ Faced issues when users entered **spaces or non-numeric characters**, causing crashes.
- ☐ Initially forgot to **convert input into a list of integers** using list comprehension.

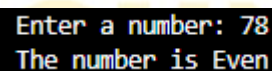
Task 2 :

Code Snippet & Screenshot

```
def is_even_or_odd(num):  
    if num % 2 == 0:  
        return "Even"  
    else:  
        return "Odd"  
  
#get number from user  
num = int(input("Enter a number: "))  
print(f"The number is {is_even_or_odd(num)}")
```



```
1 def is_even_or_odd(num):  
2     if num % 2 == 0:  
3         return "Even"  
4     else:  
5         return "Odd"  
6  
7 #get number from user  
8 num = int(input("Enter a number: "))  
9 print(f"The number is {is_even_or_odd(num)}")  
10
```



```
Enter a number: 78  
The number is Even
```

Challenges:

- ☐ Forgot to wrap input() inside int() — caused type errors.
- ☐ Had to understand how to handle **odd/even logic using modulo %**.
- ☐ Entered float values like 4.5 initially, which caused unexpected results or errors.

Task 3 :

Code Snippet & Screenshot

```

import math

def calculate_area(radius):

    return math.pi * (radius ** 2)

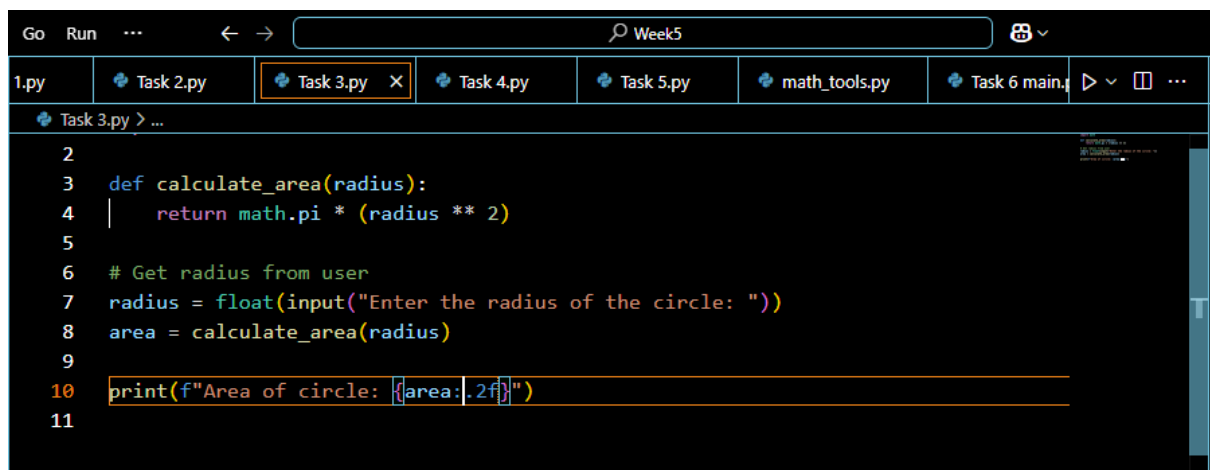
# Get radius from user

radius = float(input("Enter the radius of the circle: "))

area = calculate_area(radius)

print(f"Area of circle: {area:.2f}")

```



```

2
3 def calculate_area(radius):
4     return math.pi * (radius ** 2)
5
6 # Get radius from user
7 radius = float(input("Enter the radius of the circle: "))
8 area = calculate_area(radius)
9
10 print(f"Area of circle: {area:.2f}")
11

```

```

Enter the radius of the circle: 6767676
Area of circle: 143889462530015.34

```

Challenges:

- ☐ Was confused whether to use math.pi or just 3.14.
- ☐ Inputting negative numbers gave wrong results — had to decide whether to **validate radius**.
- ☐ Got very long decimals in output — learned to format with `:.2f`.

Task 4 :

Code Snippet & Screenshot

```

def greet_user(name, age):

    return f"Hello {name}, you are {age} years old."

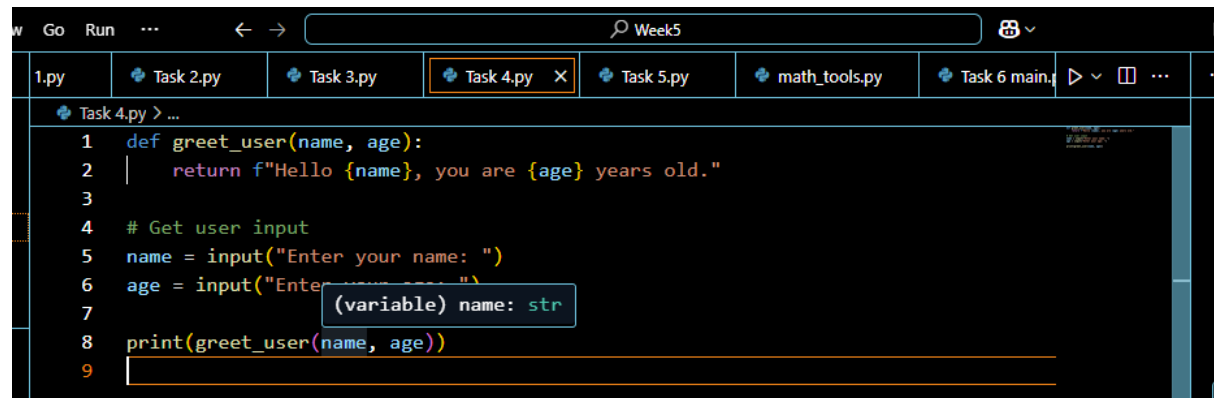
```

```
# Get user input

name = input("Enter your name: ")

age = input("Enter your age: ")

print(greet_user(name, age))
```



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

```
Enter your name: maaz sher muhammad
Enter your age: 21
Hello maaz sher muhammad, you are 21 years old.
```

Challenges:

- ☐ Initially used + for string joining — later learned f-strings are cleaner and better.
- ☐ Forgot to cast age as a string when joining — caused TypeError.
- ☐ Tried to add validation to stop empty names or non-numeric ages but wasn't sure how.

Task 5 :

Code Snippet & Screenshot

```
counter = 0 # Global variable
```

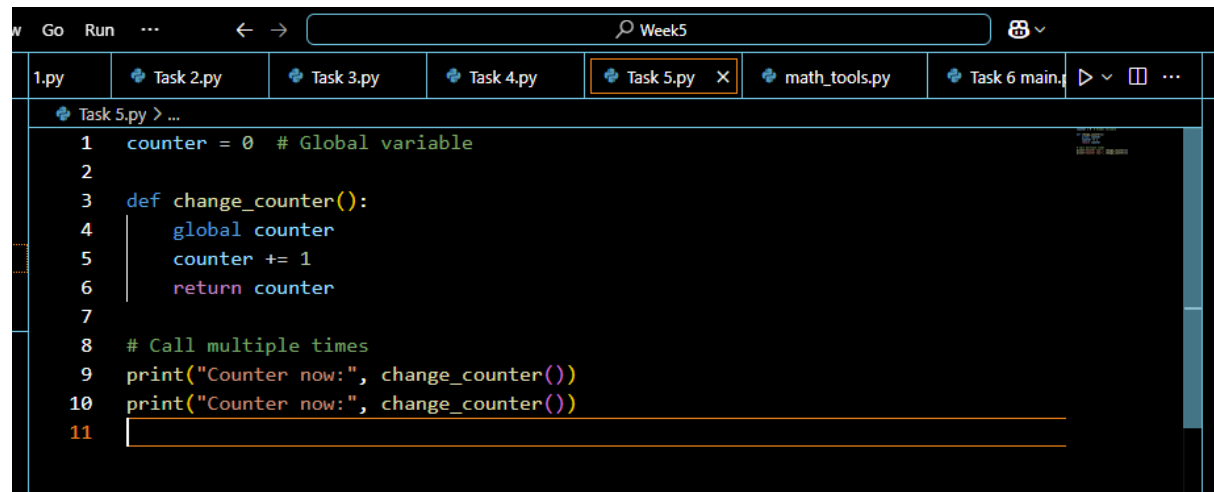
```
def change_counter():
```

```
    global counter
```

```
    counter += 1
```

```
    return counter

# Call multiple times
print("Counter now:", change_counter())
print("Counter now:", change_counter())
```

A screenshot of a code editor window titled 'Week5'. The editor shows a Python script with the following code:

```
1 counter = 0 # Global variable
2
3 def change_counter():
4     global counter
5     counter += 1
6     return counter
7
8 # Call multiple times
9 print("Counter now:", change_counter())
10 print("Counter now:", change_counter())
11
```

The script defines a global variable 'counter' and a function 'change_counter()' that increments it and returns the value. It then calls the function twice. The editor tabs show 'Task 5.py' is the active file.

```
Counter now: 1
Counter now: 2
```

Challenges:

- ☐ Didn't understand why the function needed global — thought the variable would update automatically.
- ☐ Tried using counter += 1 without declaring it global — resulted in UnboundLocalError.
- ☐ Wasn't sure where the global variable should be declared in the script.

Task 6 :

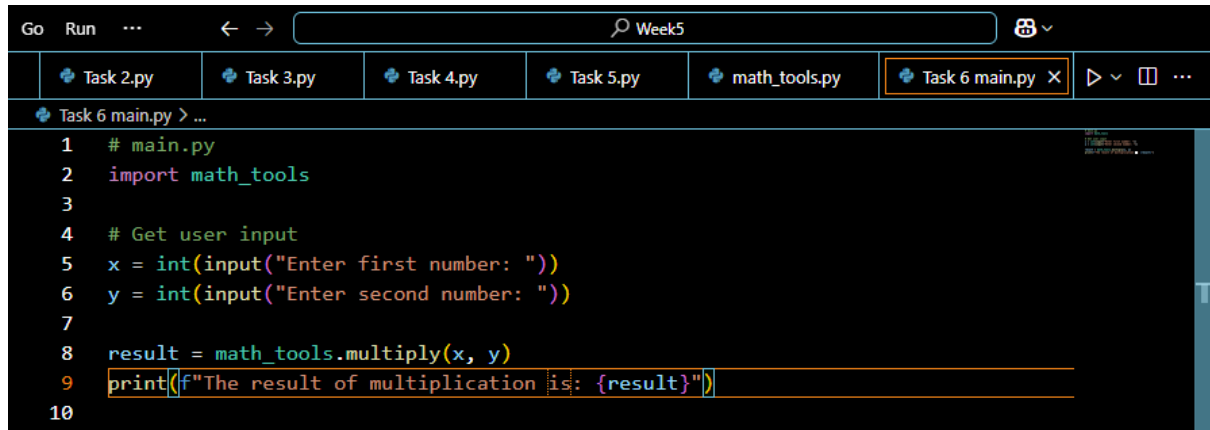
Code Snippet & Screenshot

```
# main.py

import math_tools

# Get user input
```

```
x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
result = math_tools.multiply(x, y)
print(f"The result of multiplication is: {result}")
```

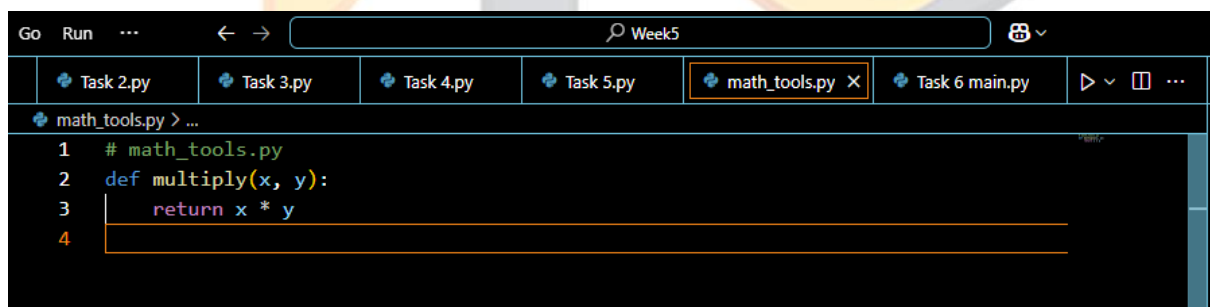


```
Go Run ... Week5
Task 2.py Task 3.py Task 4.py Task 5.py math_tools.py Task 6 main.py x
Task 6 main.py > ...
1 # main.py
2 import math_tools
3
4 # Get user input
5 x = int(input("Enter first number: "))
6 y = int(input("Enter second number: "))
7
8 result = math_tools.multiply(x, y)
9 print(f"The result of multiplication is: {result}")
10
```

```
Enter first number: 01
Enter second number: 21
The result of multiplication is: 21
```

math_tools.py:

```
# math_tools.py
def multiply(x, y):
    return x * y
```



```
Go Run ... Week5
Task 2.py Task 3.py Task 4.py Task 5.py math_tools.py x Task 6 main.py
math_tools.py > ...
1 # math_tools.py
2 def multiply(x, y):
3     return x * y
4
```

Challenges:

- ☐ Faced error `ModuleNotFoundError` because filename had a **space instead of underscore**.
- ☐ Didn't know both `.py` files must be in the **same folder** for import to work.
- ☐ Wasn't sure how to name files properly or avoid using special characters in filenames.