



1. Name: Hamza Badshah

2. Intern ID: TN/IN01/PY/001

3. Email ID : hamzabadshah2592@gmail.com

4. Internship Domain : Python Development

5. Task Week : 04

6. Instructor Name : Mr Hassan Ali

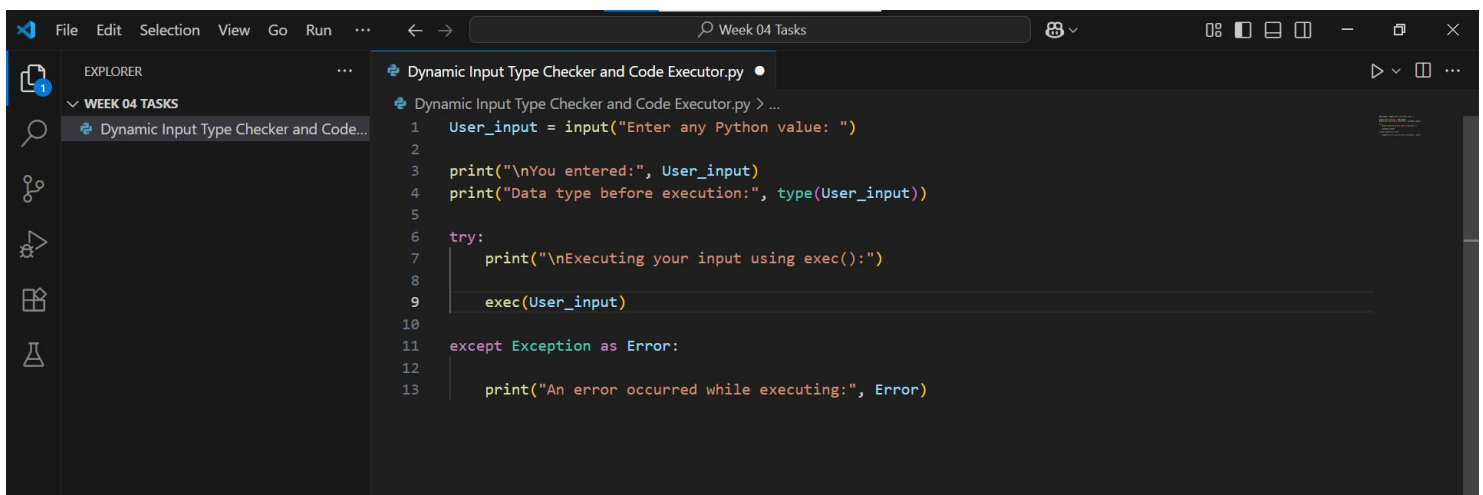
Task 1 :

Ask user to input any value. Use type() to check its data type. Use exec() to execute a string as Python code.

What I Did:

- Took input from the user using input().
- Displayed the input and its type using type().
- Used exec() to execute the input as Python code.
- Added try-except to handle execution errors.
- Printed any output or error from the execution.

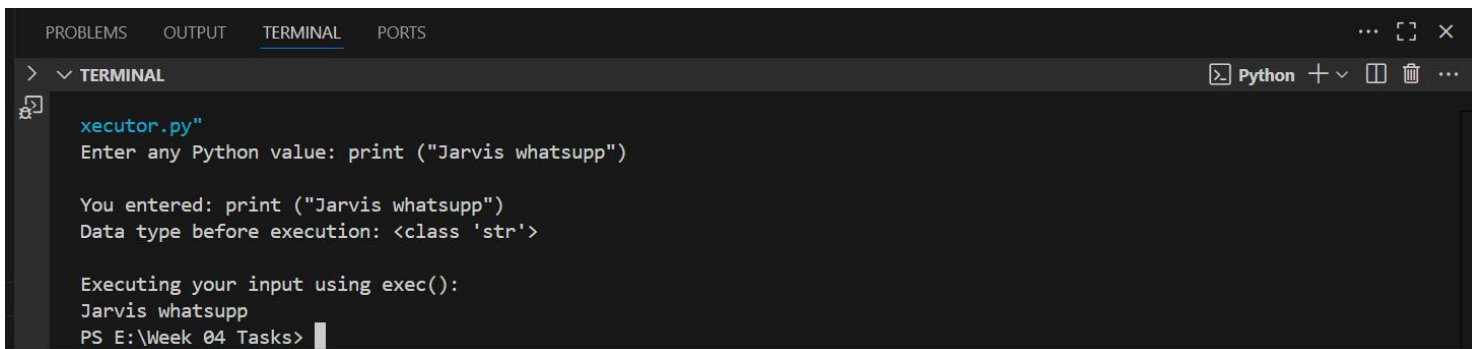
Code Snippets



```
1 User_input = input("Enter any Python value: ")
2
3 print("\nYou entered:", User_input)
4 print("Data type before execution:", type(User_input))
5
6 try:
7     print("\nExecuting your input using exec():")
8
9     exec(User_input)
10
11 except Exception as Error:
12
13     print("An error occurred while executing:", Error)
```



Output Snippet



```
PROBLEMS OUTPUT TERMINAL PORTS
> v TERMINAL
Python + v [ ] [ ] [ ] [ ]
xecutor.py
Enter any Python value: print ("Jarvis whatsapp")

You entered: print ("Jarvis whatsapp")
Data type before execution: <class 'str'>

Executing your input using exec():
Jarvis whatsapp
PS E:\Week 04 Tasks>
```

Learning and Challenges:

- Learned that all `input()` values are strings by default.
- Understood how `exec()` runs dynamic Python code.
- Faced issues when invalid code caused syntax errors.
- Used error handling to avoid crashes during execution.
- Realized `exec()` can be dangerous if not used safely.

Task 02:

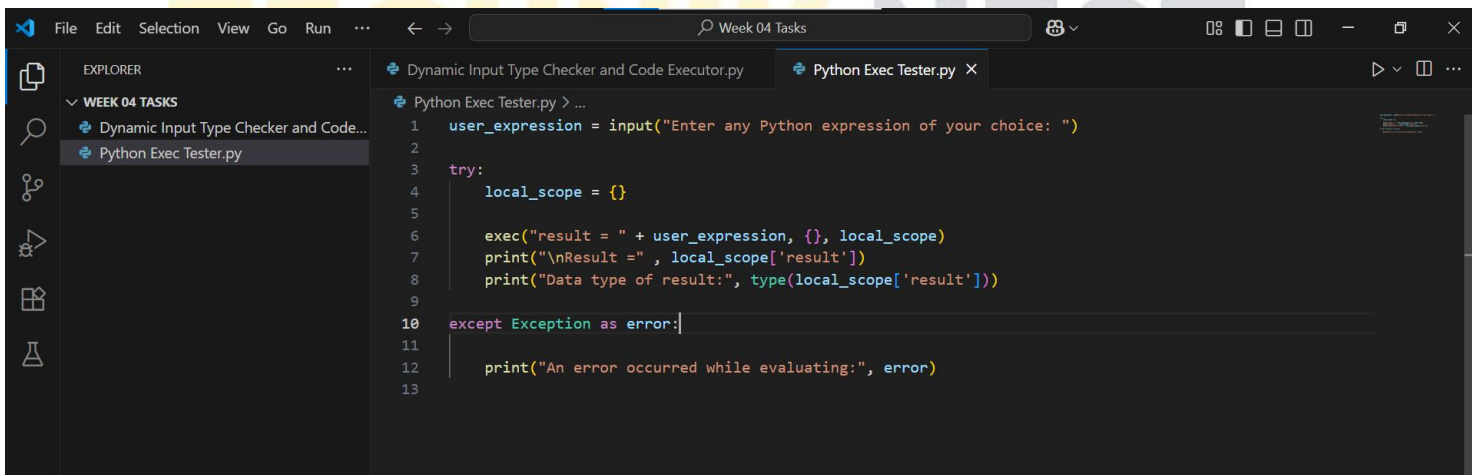
Ask user for a Python expression as a string (like '2 + 3 * 4') and evaluate it using `exec()`.

Show result.

What I Did:

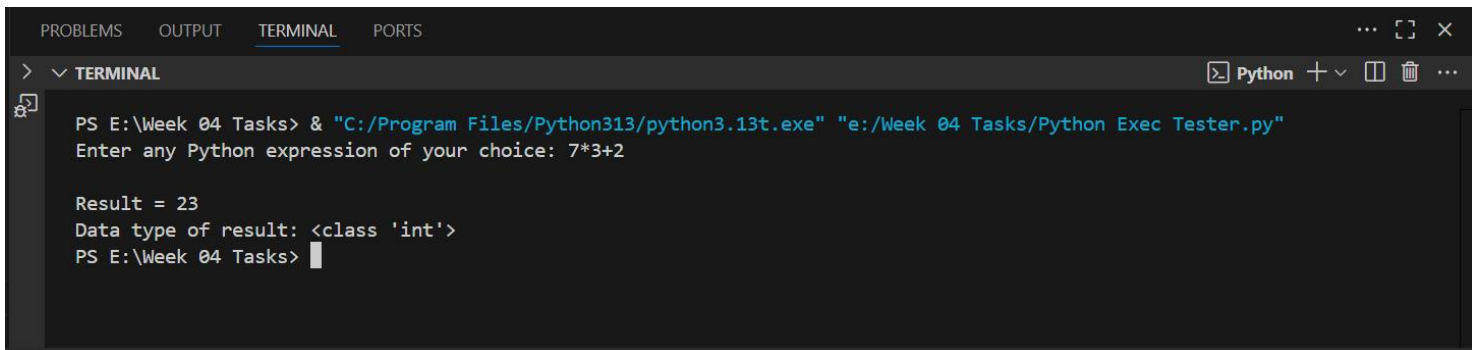
- Took a math expression as a string input from the user.
- Combined it into a statement using `exec()` to assign the result to a variable.
- Executed it safely using a try-except block.
- Retrieved the result from the dynamically created variable.
- Displayed the result or any execution error.

Code Snippets



```
1 user_expression = input("Enter any Python expression of your choice: ")
2
3 try:
4     local_scope = {}
5
6     exec("result = " + user_expression, {}, local_scope)
7     print("\nResult =", local_scope['result'])
8     print("Data type of result:", type(local_scope['result']))
9
10 except Exception as error:
11
12     print("An error occurred while evaluating:", error)
13
```

Output Snippet



```
PROBLEMS OUTPUT TERMINAL PORTS
> v TERMINAL Python + v [ ] [ ] ...
PS E:\Week 04 Tasks> & "C:/Program Files/Python313/python3.13t.exe" "e:/Week 04 Tasks/Python Exec Tester.py"
Enter any Python expression of your choice: 7*3+2

Result = 23
Data type of result: <class 'int'>
PS E:\Week 04 Tasks> |
```

Learning and Challenges:

- Learned how to evaluate expressions using `exec()` by building an assignment string.
- Understood that `exec()` doesn't return values directly like `eval()`.
- Faced a challenge accessing variables created inside `exec()`.
- Solved it by assigning to a known variable name like `Result`.
- Realized that `eval()` is simpler for expressions, but `exec()` offers more flexibility.

Task 03:

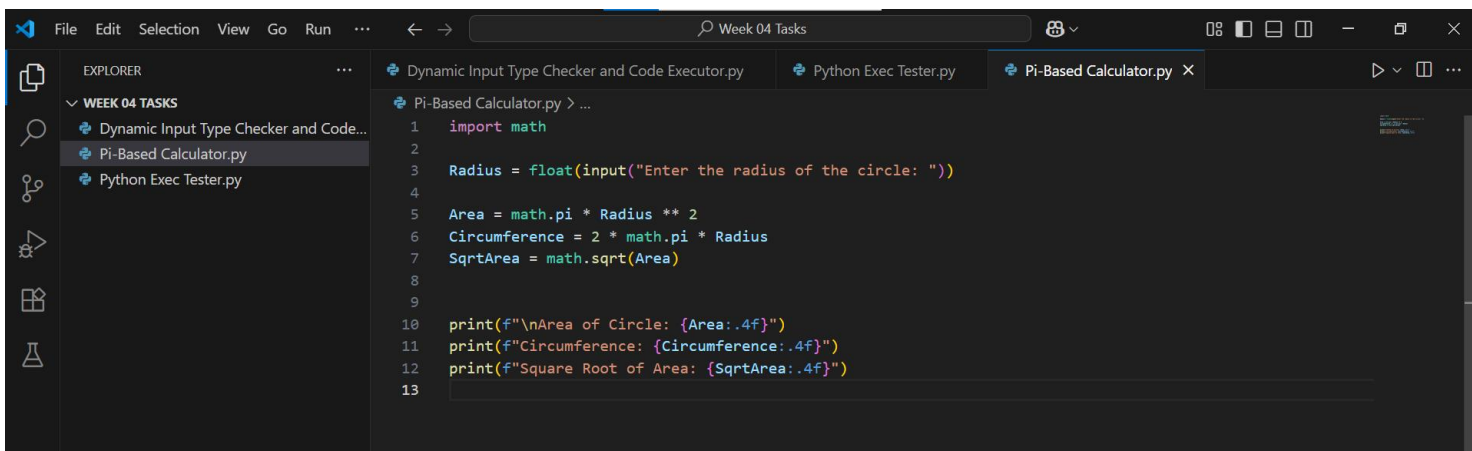
Use math module to take a radius input from user and calculate:
- Area of circle, circumference, and square root of area.

What I Did:

- Imported the `math` module for advanced mathematical functions.
- Took radius input from the user and converted it to `float`.

- Calculated area using πr^2 and circumference using $2\pi r$.
- Found square root of area using `math.sqrt()`.
- Printed all three calculated values clearly.

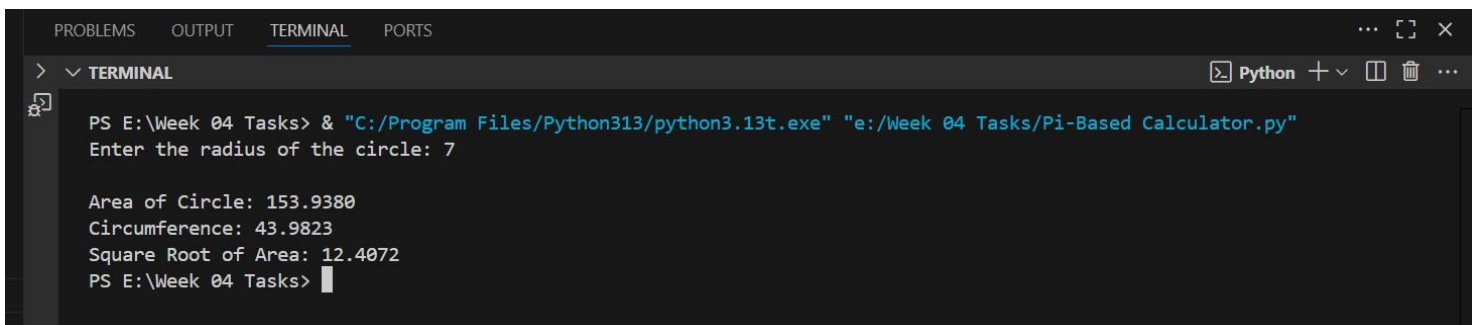
Code Snippets



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a project named 'WEEK 04 TASKS' with three files: 'Dynamic Input Type Checker and Code...', 'Pi-Based Calculator.py', and 'Python Exec Tester.py'. The 'Pi-Based Calculator.py' file is selected and its code is displayed in the main editor. The code is as follows:

```
1 import math
2
3 Radius = float(input("Enter the radius of the circle: "))
4
5 Area = math.pi * Radius ** 2
6 Circumference = 2 * math.pi * Radius
7 SqrtArea = math.sqrt(Area)
8
9
10 print(f"\nArea of Circle: {Area:.4f}")
11 print(f"Circumference: {Circumference:.4f}")
12 print(f"Square Root of Area: {SqrtArea:.4f}")
13
```

Output Snippets



The screenshot shows a terminal window with a dark theme. The terminal title is 'Python'. The command prompt is 'PS E:\Week 04 Tasks> & "C:/Program Files/Python313/python3.13t.exe" "e:/Week 04 Tasks/Pi-Based Calculator.py"'. The output of the program is as follows:

```
Enter the radius of the circle: 7

Area of Circle: 153.9380
Circumference: 43.9823
Square Root of Area: 12.4072
PS E:\Week 04 Tasks>
```

Learnings and Challenges:

- Learned how to use `math.pi` for accurate π value.
- Understood the importance of data type conversion (float).
- Faced issue with square root before calculating area first.
- Practiced chaining formulas and printing multiple outputs.
- Strengthened understanding of geometry in real Python usage.

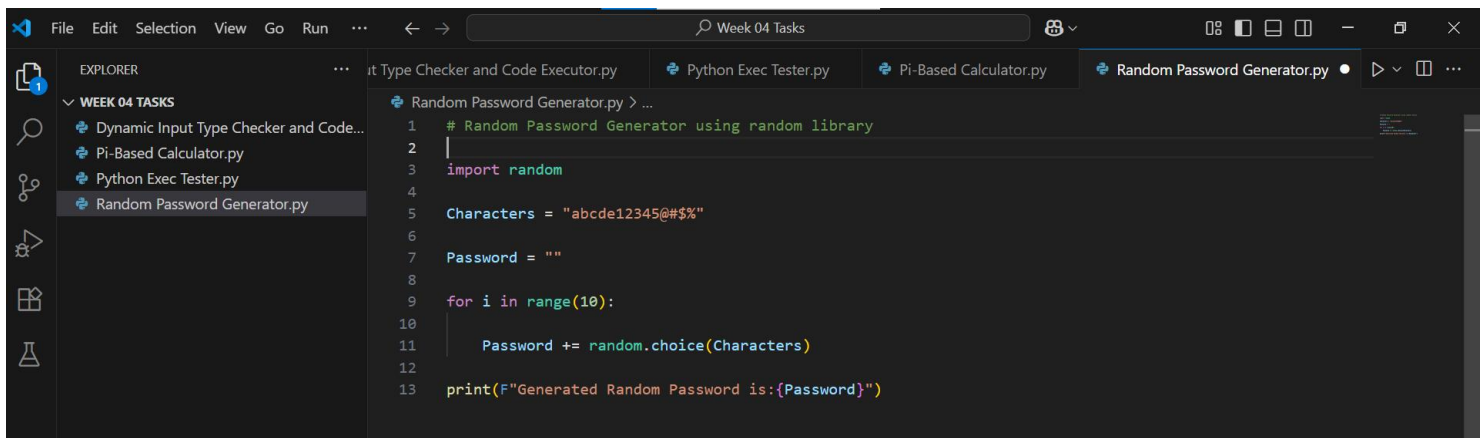
Task 04:

Use random module to generate a random 8-character password using letters, numbers, and symbols.

What I Did:

- Used random module to pick random characters.
- Created a simple string with letters, numbers, and symbols.
- Initialized an empty password variable.
- Used a loop to add 8 random characters.
- Printed the final password to the screen.

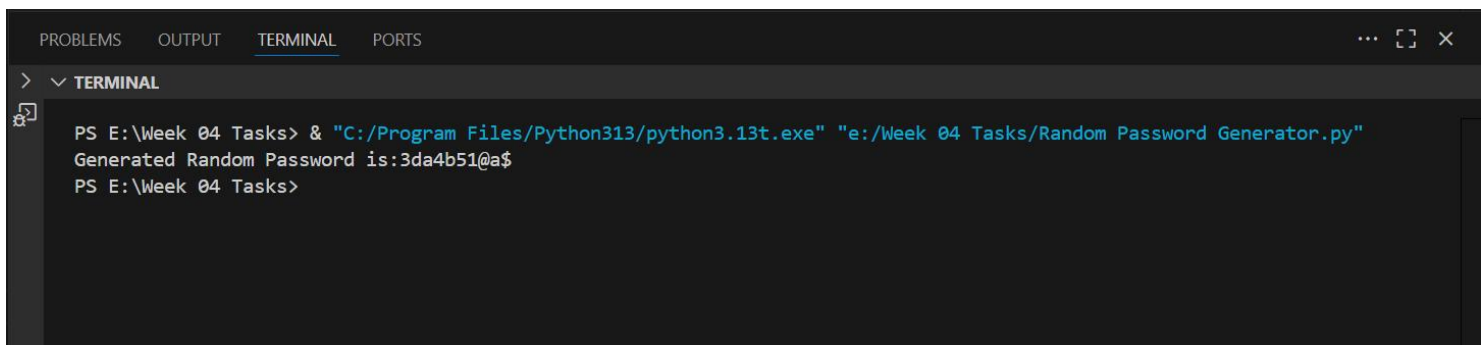
Code Snippets



The screenshot shows a code editor with a dark theme. The Explorer pane on the left shows a project named 'WEEK 04 TASKS' with four files: 'Dynamic Input Type Checker and Code...', 'Pi-Based Calculator.py', 'Python Exec Tester.py', and 'Random Password Generator.py'. The main editor area shows the code for 'Random Password Generator.py'.

```
1 # Random Password Generator using random library
2 |
3 import random
4
5 Characters = "abcde12345@#$$%"
6
7 Password = ""
8
9 for i in range(10):
10     Password += random.choice(Characters)
11
12
13 print(F"Generated Random Password is:{Password}")
```

Output Snippets

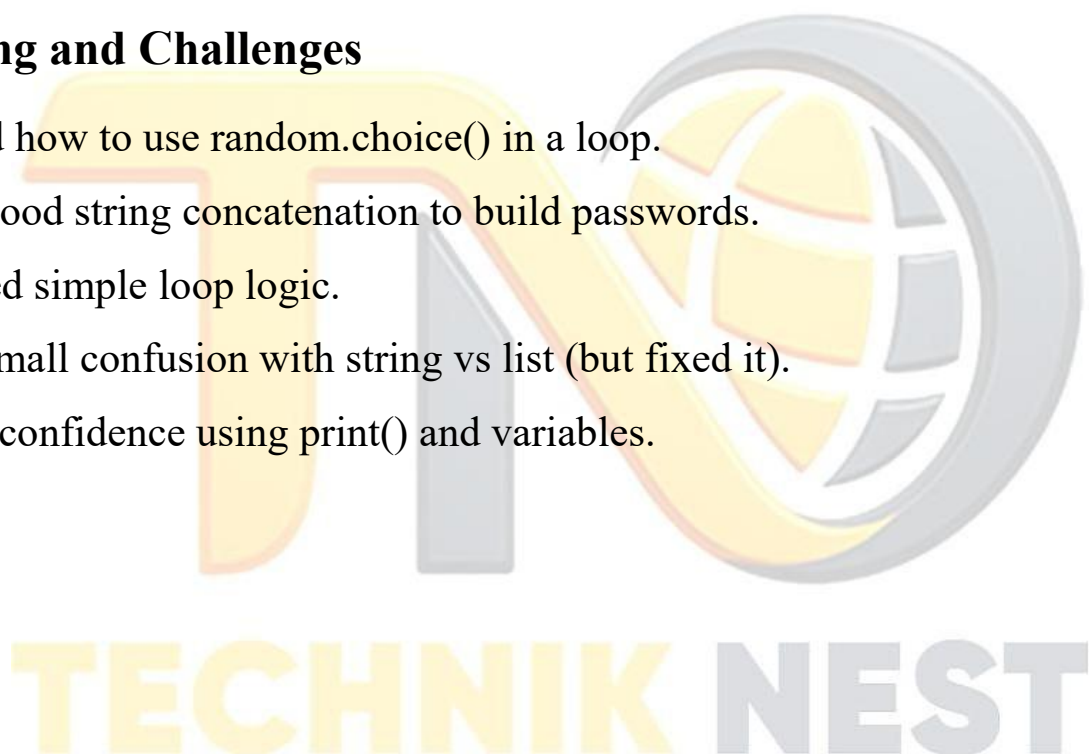


The screenshot shows a terminal window with the 'TERMINAL' tab selected. The command prompt shows the execution of the Python script, and the output displays a generated random password.

```
PS E:\Week 04 Tasks> & "C:/Program Files/Python313/python3.13t.exe" "e:/Week 04 Tasks/Random Password Generator.py"
Generated Random Password is:3da4b51@a$
PS E:\Week 04 Tasks>
```

Learning and Challenges

- Learned how to use `random.choice()` in a loop.
- Understood string concatenation to build passwords.
- Practiced simple loop logic.
- Faced small confusion with string vs list (but fixed it).
- Gained confidence using `print()` and variables.



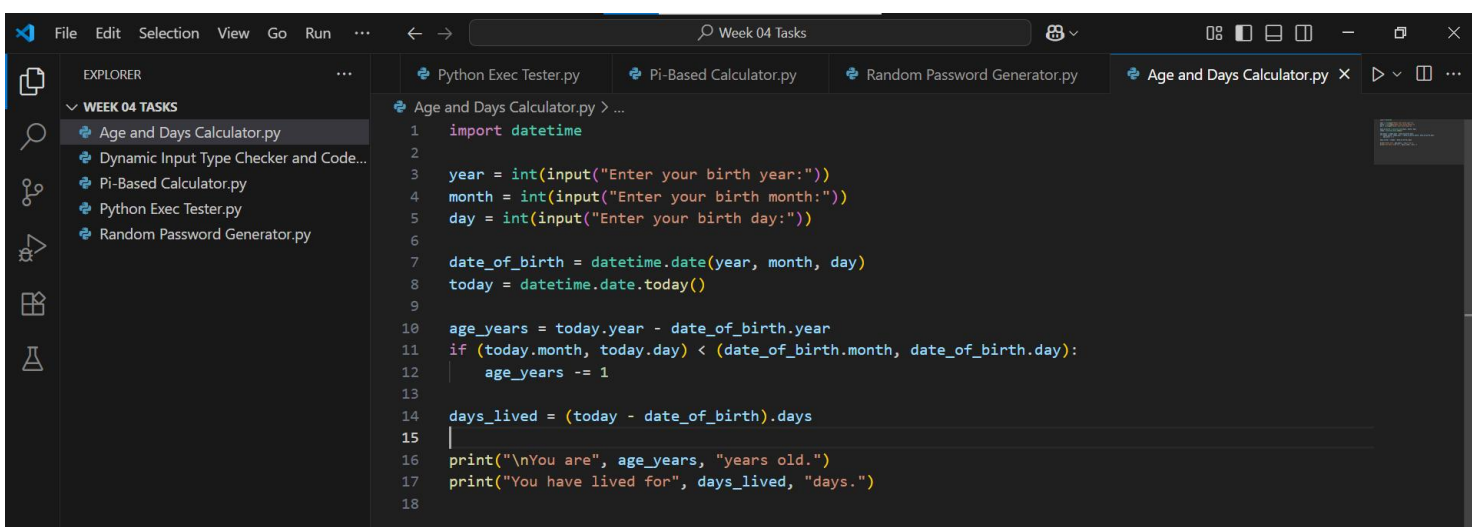
Task 05:

Using datetime module, ask user for their birth date and show:
- Their age in years and number of days lived.

What I Did:

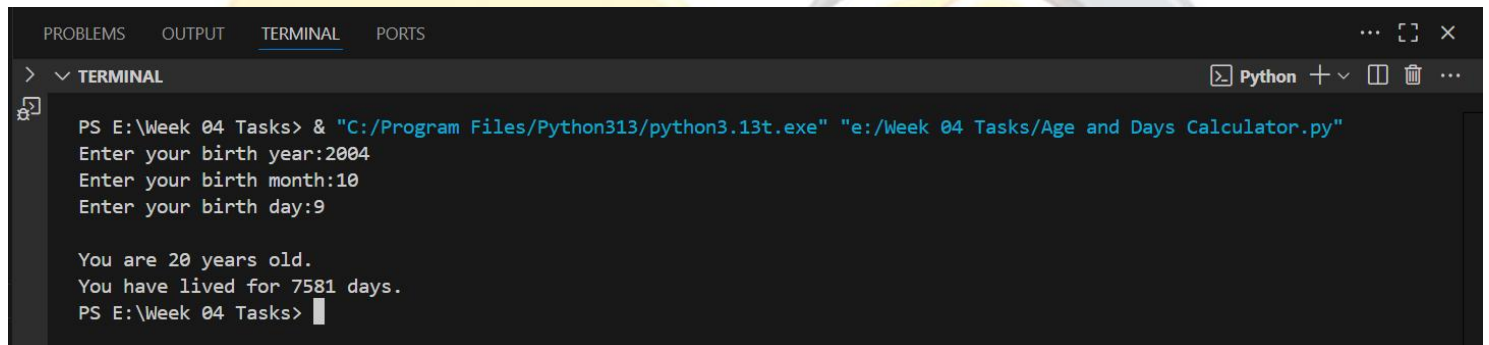
- Imported the datetime module to use dates.
- Asked the user for their birth year, month, and day.
- Created their birthdate and got today's date.
- Calculated their age in years and total days lived.
- Printed the age and number of days to the screen.

Code Snippets



```
1 import datetime
2
3 year = int(input("Enter your birth year:"))
4 month = int(input("Enter your birth month:"))
5 day = int(input("Enter your birth day:"))
6
7 date_of_birth = datetime.date(year, month, day)
8 today = datetime.date.today()
9
10 age_years = today.year - date_of_birth.year
11 if (today.month, today.day) < (date_of_birth.month, date_of_birth.day):
12     age_years -= 1
13
14 days_lived = (today - date_of_birth).days
15
16 print("\nYou are", age_years, "years old.")
17 print("You have lived for", days_lived, "days.")
18
```

Output Snippets



```
PROBLEMS OUTPUT TERMINAL PORTS
> ▼ TERMINAL Python + ▢ 🗑️ ...
PS E:\Week 04 Tasks> & "C:/Program Files/Python313/python3.13t.exe" "e:/Week 04 Tasks/Age and Days Calculator.py"
Enter your birth year:2004
Enter your birth month:10
Enter your birth day:9

You are 20 years old.
You have lived for 7581 days.
PS E:\Week 04 Tasks> |
```

Learning and Challenges

- Learned how to work with real dates using datetime.
- Practiced asking for multiple inputs from the user.
- Faced a small challenge checking if birthday passed this year.
- Learned how to subtract dates to get days difference.
- Understood how age calculation can change based on the date.

Task 06:

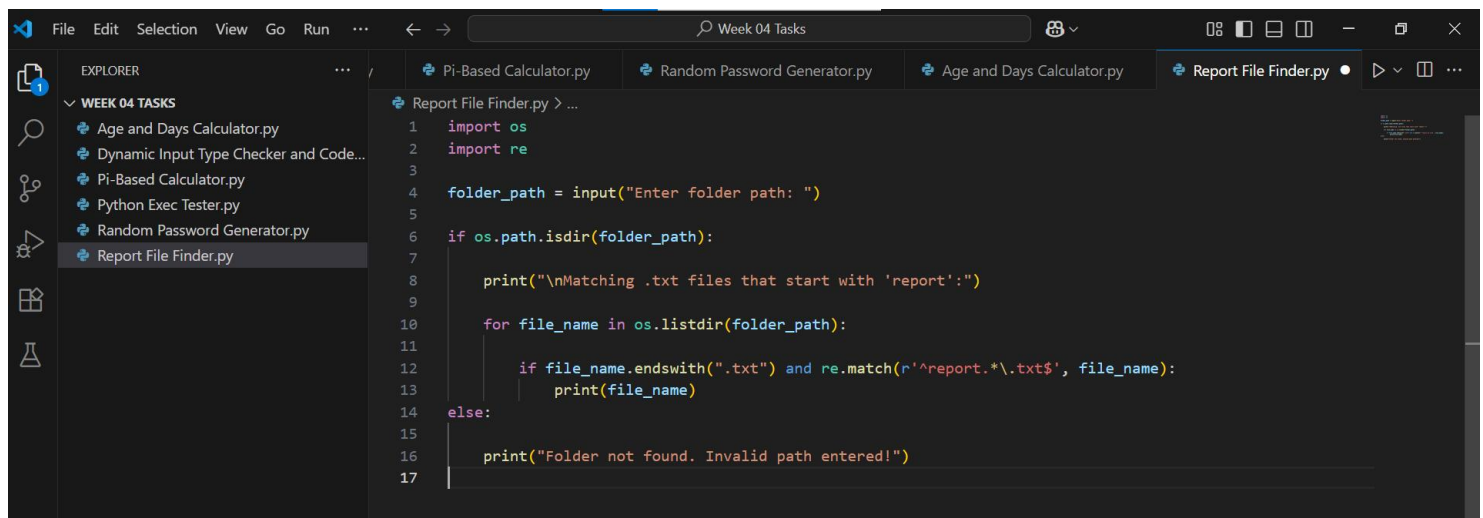
Create a script using os and re that lists all `.txt` files from a folder and filters only those that match a pattern (e.g., start with 'report').

What I Did:

- Imported os to list files in a folder.
- Used re (regular expressions) to match filenames.
- Asked the user to enter a folder path.

- Checked if file ends with .txt and starts with 'report'.
- Displayed the matched .txt files

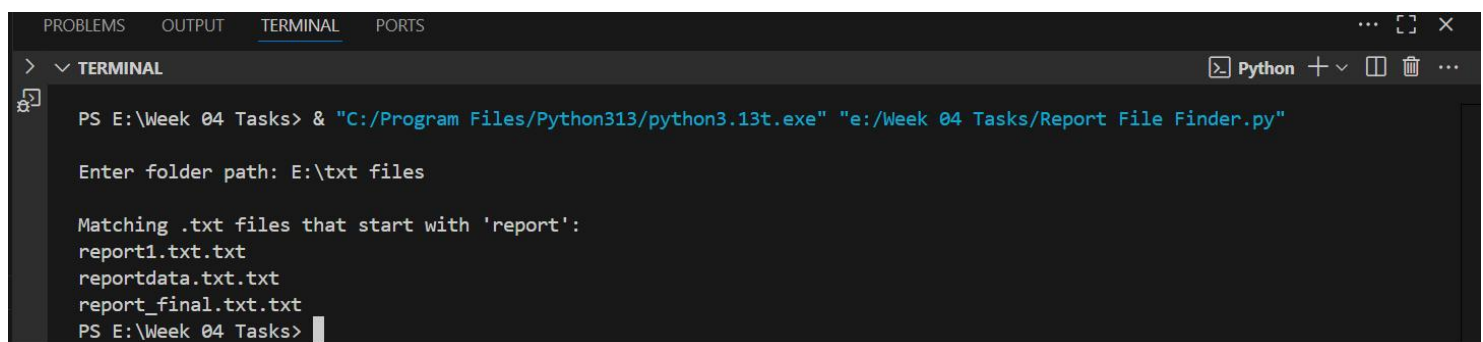
Code Snippets



The screenshot shows the Visual Studio Code editor with the 'Report File Finder.py' file open. The Explorer sidebar on the left shows a project named 'WEEK 04 TASKS' containing several Python files, with 'Report File Finder.py' selected. The main editor area displays the following Python code:

```
1 import os
2 import re
3
4 folder_path = input("Enter folder path: ")
5
6 if os.path.isdir(folder_path):
7
8     print("\nMatching .txt files that start with 'report':")
9
10    for file_name in os.listdir(folder_path):
11
12        if file_name.endswith(".txt") and re.match(r'^report.*\.txt$', file_name):
13            print(file_name)
14    else:
15
16        print("Folder not found. Invalid path entered!")
17
```

Output Snippets



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:

```
PS E:\Week 04 Tasks> & "C:/Program Files/Python313/python3.13t.exe" "e:/Week 04 Tasks/Report File Finder.py"

Enter folder path: E:\txt files

Matching .txt files that start with 'report':
report1.txt.txt
reportdata.txt.txt
report_final.txt.txt
PS E:\Week 04 Tasks>
```

Learning and Challenges

- Learned how to list files using `os.listdir()`.
- Practiced using regex to filter by pattern.
- Faced issues with case sensitivity in `re.match()` (can fix with flags).
- Understood how to combine conditions (`endswith` + regex).
- Gained confidence working with file and folder path.

TECHNIK NEST

