

Name: Musfira Ahmed

Intern ID: TN/IN01/PY/002

Email ID : ahmedmusfira3@gmail.com

Internship Domain : Python Development

Task Week : 04

Instructor Name : Mr Hassan Ali

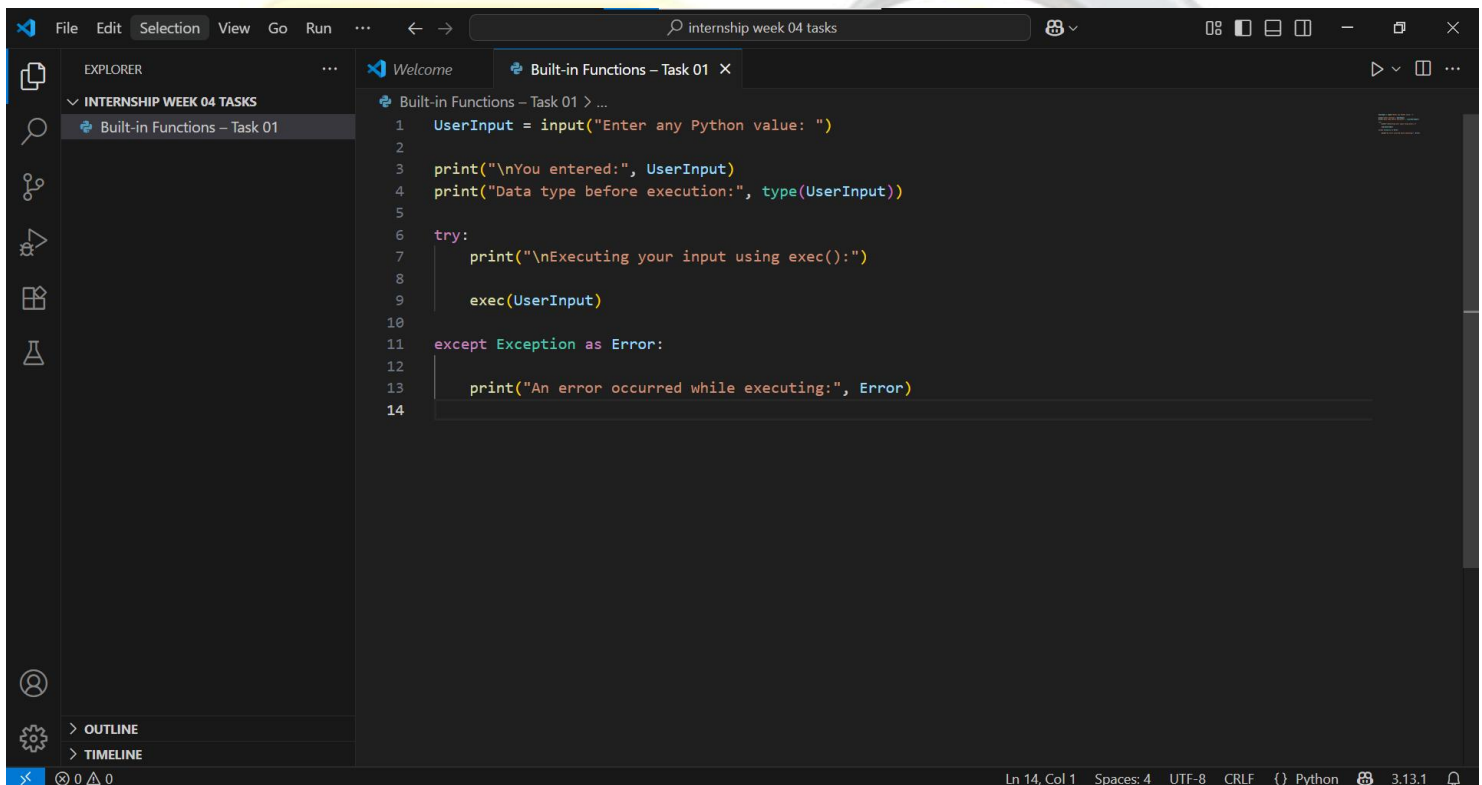
Task 1 :

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 (Step by Step):

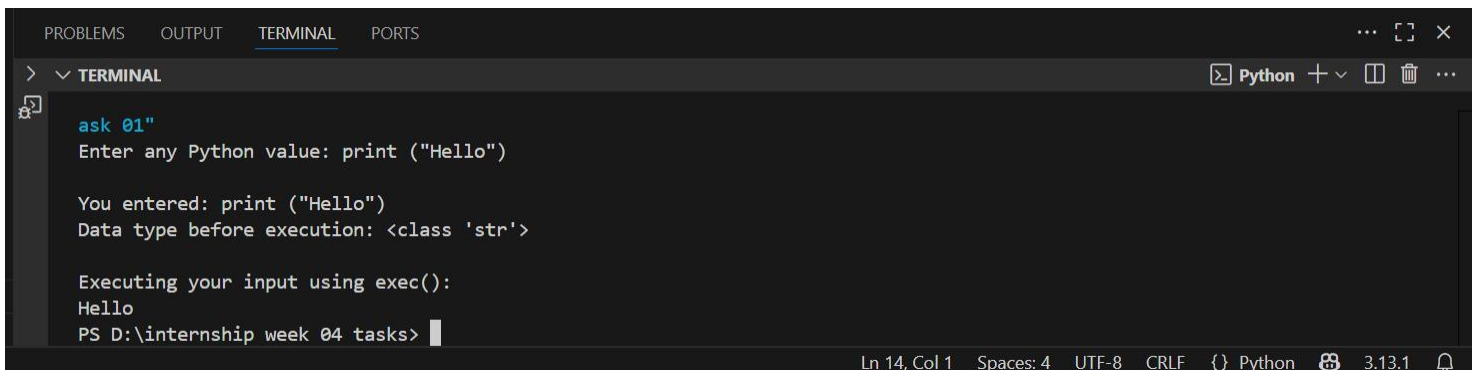
- Took input from the user using `input()`.
- Displayed the input and its type using `type()`.
- Used `exec()` to execute the input as Python code.
- Added try-except to handle execution errors.
- Printed any output or error from the execution.

Code Screenshots



```
1  userInput = input("Enter any Python value: ")
2
3  print("\nYou entered:", userInput)
4  print("Data type before execution:", type(userInput))
5
6  try:
7      print("\nExecuting your input using exec():")
8
9      exec(userInput)
10
11 except Exception as Error:
12
13     print("An error occurred while executing:", Error)
14
```

Output Screenshot



The screenshot shows a terminal window with the following text:

```
ask 01"
Enter any Python value: print ("Hello")

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

Executing your input using exec():
Hello
PS D:\internship week 04 tasks>
```

The terminal window has tabs for PROBLEMS, OUTPUT, TERMINAL, and PORTS. The TERMINAL tab is active. The status bar at the bottom shows 'Ln 14, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and '3.13.1'.

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.

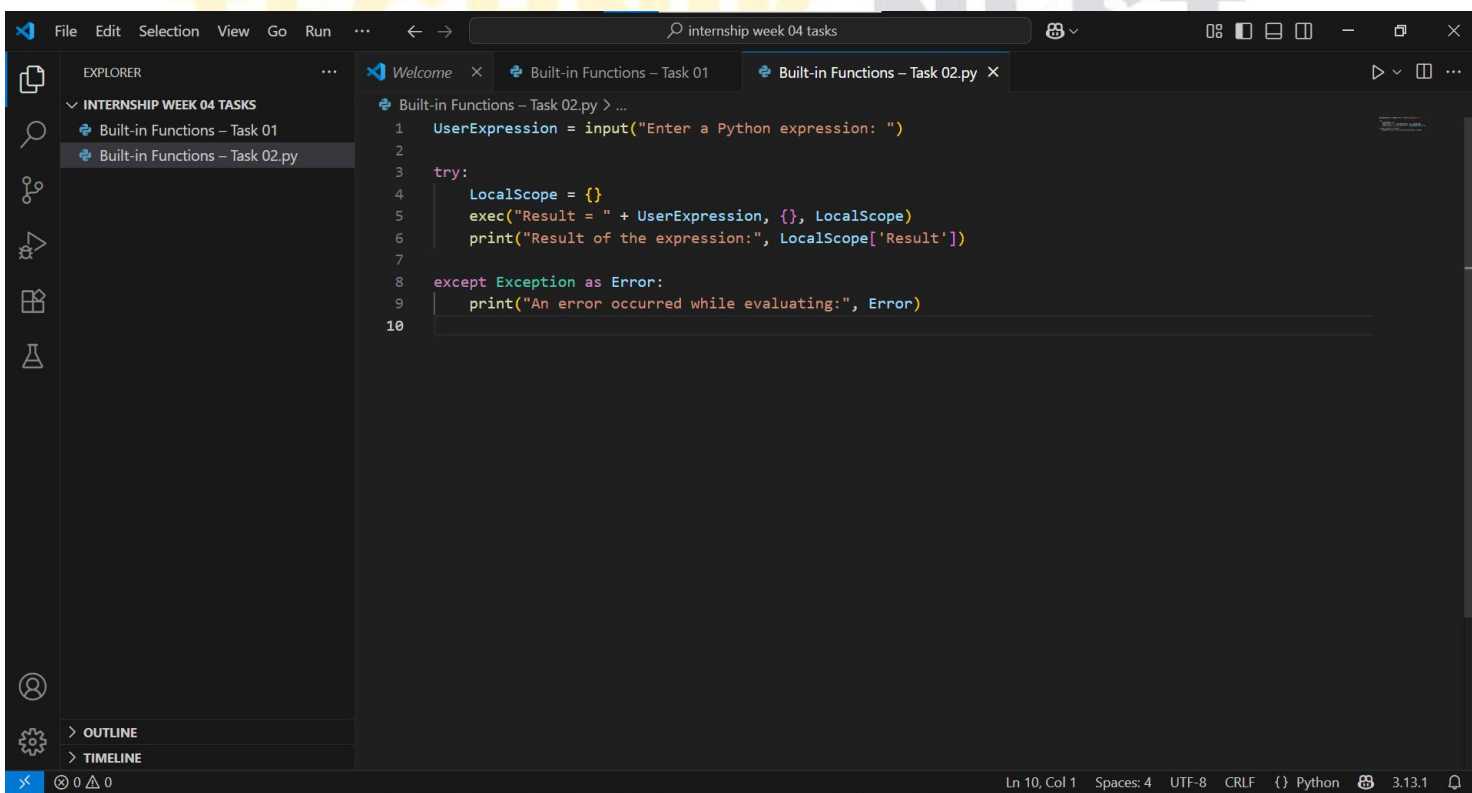
TECHNIK NEST

Task 02:

What I Did (Step by Step):

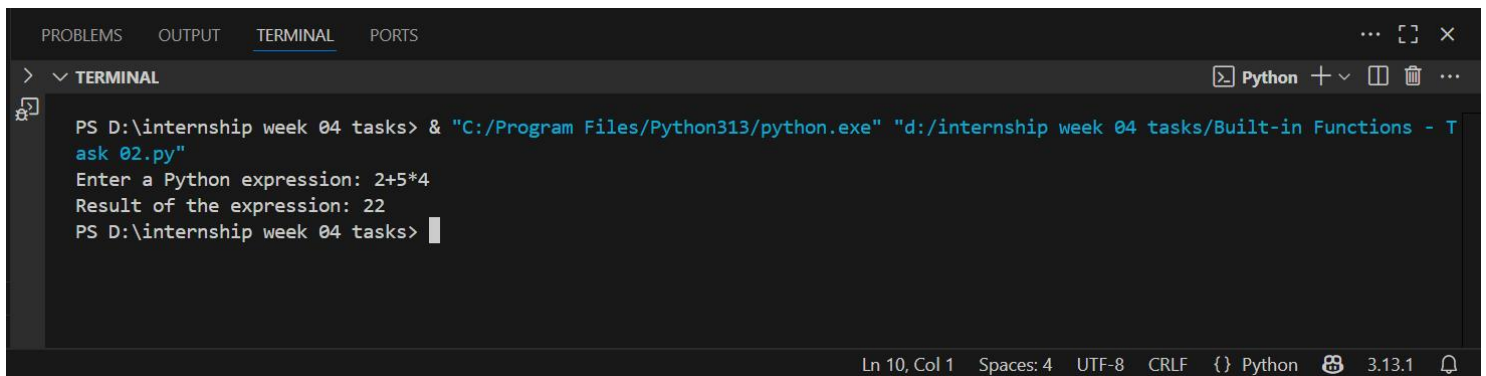
- Took a math expression as a string input from the user.
- Combined it into a statement using `exec()` to assign the result to a variable.
- Executed it safely using a try-except block.
- Retrieved the result from the dynamically created variable.
- Displayed the result or any execution error.

Code Screenshots



```
1 UserExpression = input("Enter a Python expression: ")
2
3 try:
4     LocalScope = {}
5     exec("Result = " + UserExpression, {}, LocalScope)
6     print("Result of the expression:", LocalScope['Result'])
7
8 except Exception as Error:
9     print("An error occurred while evaluating:", Error)
10
```

Output Screenshot



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 content shows a command prompt 'PS D:\internship week 04 tasks>' followed by a command to run a Python script: '& "C:/Program Files/Python313/python.exe" "d:/internship week 04 tasks/Built-in Functions - Task 02.py"'. The script prompts 'Enter a Python expression: 2+5*4', and the terminal shows the output 'Result of the expression: 22'. The prompt returns to 'PS D:\internship week 04 tasks>'. At the bottom right of the terminal, status information is displayed: 'Ln 10, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1'.

```
PS D:\internship week 04 tasks> & "C:/Program Files/Python313/python.exe" "d:/internship week 04 tasks/Built-in Functions - Task 02.py"
Enter a Python expression: 2+5*4
Result of the expression: 22
PS D:\internship 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.

TECHNIK NEST

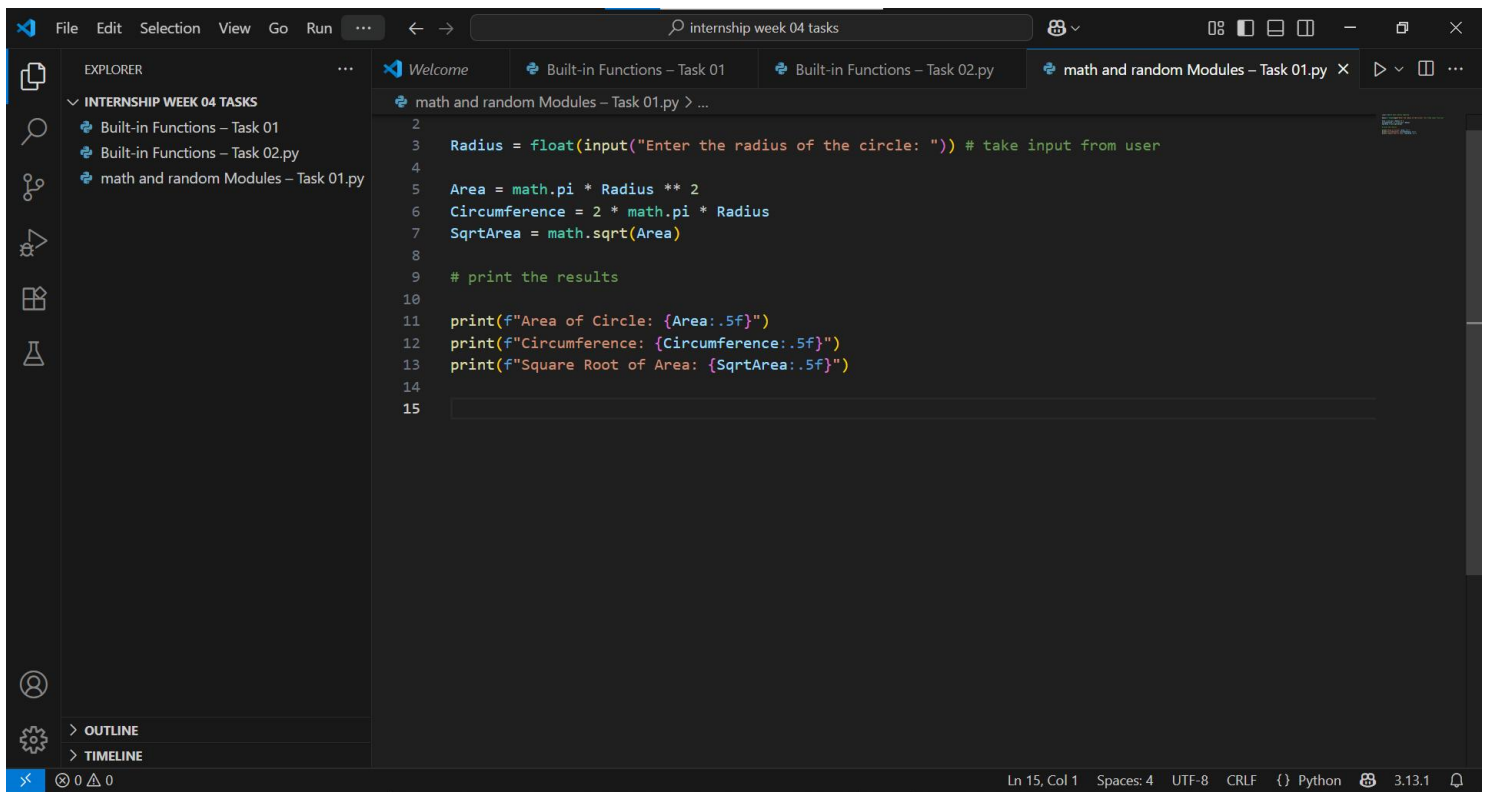
Task 03:

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

What I Did (Step by Step):

- 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 Screenshots

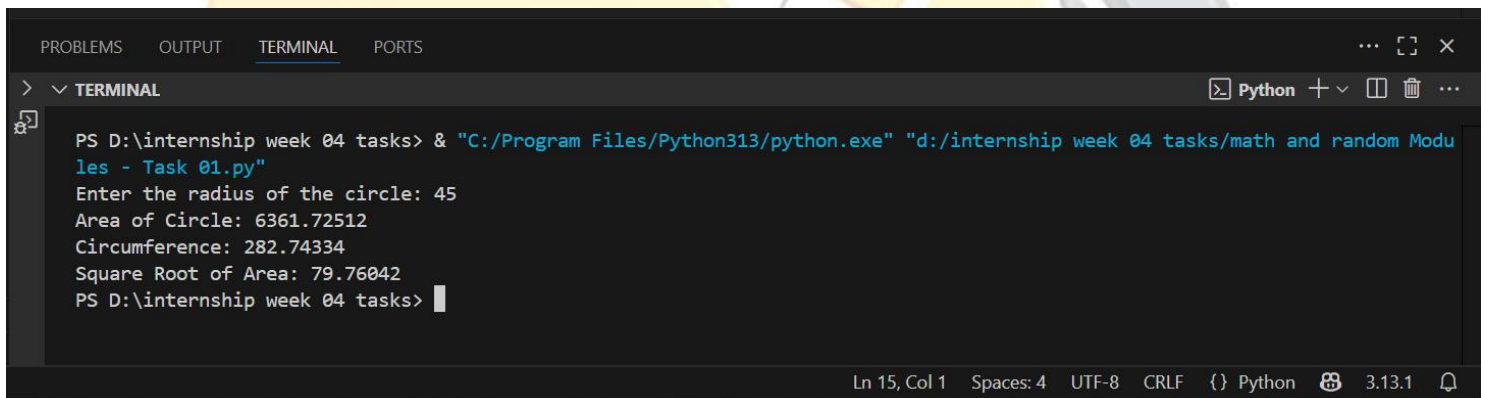


The screenshot shows the Visual Studio Code editor interface. The Explorer pane on the left displays a project named 'INTERNSHIP WEEK 04 TASKS' with three files: 'Built-in Functions - Task 01', 'Built-in Functions - Task 02.py', and 'math and random Modules - Task 01.py'. The main editor window is open to 'math and random Modules - Task 01.py'. The code in the editor is as follows:

```
2
3 Radius = float(input("Enter the radius of the circle: ")) # take input from user
4
5 Area = math.pi * Radius ** 2
6 Circumference = 2 * math.pi * Radius
7 SqrtArea = math.sqrt(Area)
8
9 # print the results
10
11 print(f"Area of Circle: {Area:.5f}")
12 print(f"Circumference: {Circumference:.5f}")
13 print(f"Square Root of Area: {SqrtArea:.5f}")
14
15
```

The status bar at the bottom indicates the current line and column (Ln 15, Col 1), indentation (Spaces: 4), encoding (UTF-8), line endings (CRLF), language (Python), and version (3.13.1).

Output Screenshot



The screenshot shows the VS Code terminal window with the 'TERMINAL' tab selected. The terminal output is as follows:

```
PS D:\internship week 04 tasks> & "C:/Program Files/Python313/python.exe" "d:/internship week 04 tasks/math and random Modules - Task 01.py"
Enter the radius of the circle: 45
Area of Circle: 6361.72512
Circumference: 282.74334
Square Root of Area: 79.76042
PS D:\internship week 04 tasks>
```

The status bar at the bottom of the terminal window shows the same information as the code editor: Ln 15, Col 1, Spaces: 4, UTF-8, CRLF, Python, 3.13.1.

TECHNIK NEST

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.

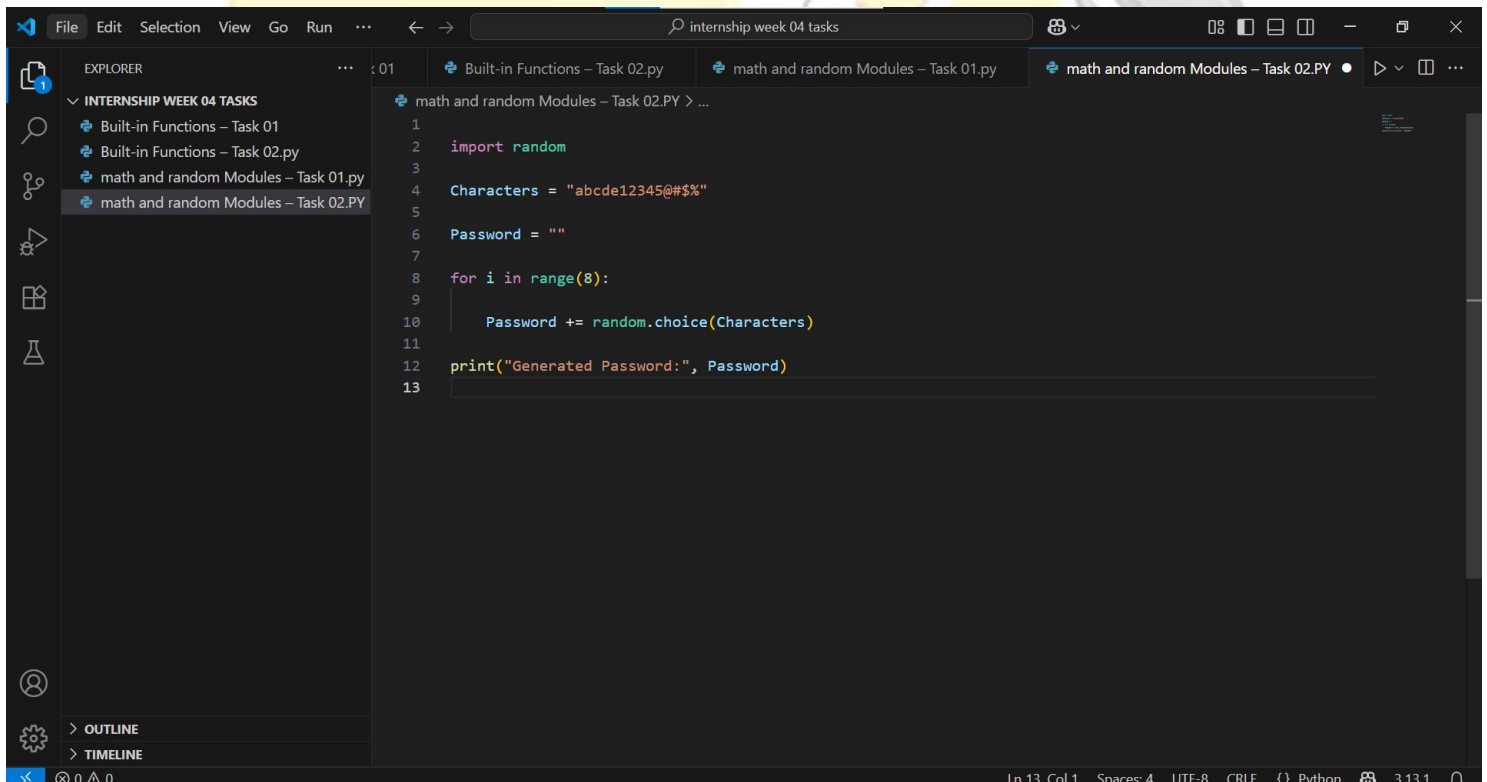
Task 04:

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

What I Did (Step by Step):

- 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 Screenshots

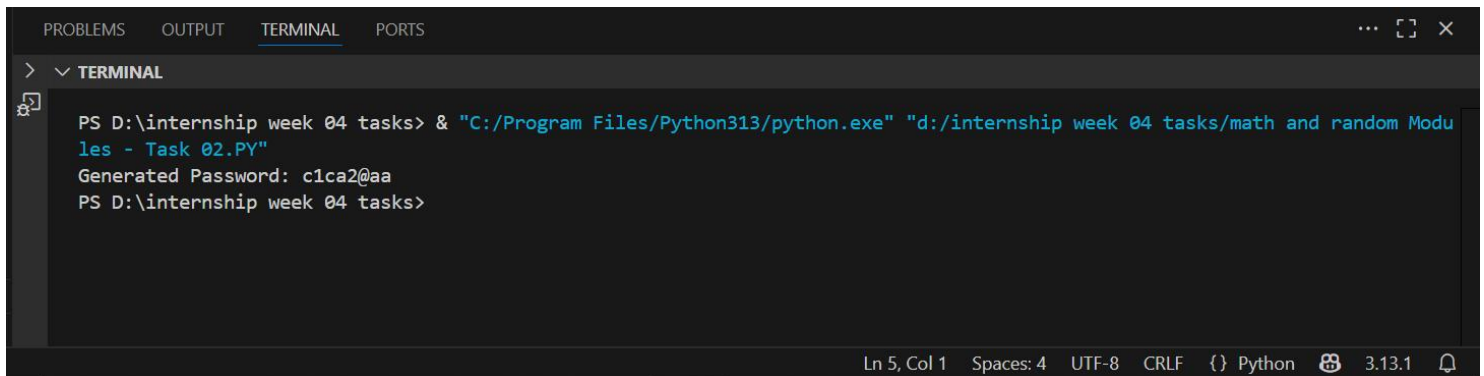


The screenshot shows a Visual Studio Code editor window with the following content:

- Explorer Panel:** Shows a folder named "INTERNSHIP WEEK 04 TASKS" containing four files: "Built-in Functions – Task 01", "Built-in Functions – Task 02.py", "math and random Modules – Task 01.py", and "math and random Modules – Task 02.PY".
- Editor Panel:** Displays the code for "math and random Modules – Task 02.PY". The code is as follows:

```
1
2 import random
3
4 Characters = "abcde12345@#$$%"
5
6 Password = ""
7
8 for i in range(8):
9
10     Password += random.choice(Characters)
11
12 print("Generated Password:", Password)
13
```
- Bottom Panel:** Shows the "OUTLINE" and "TIMELINE" tabs.
- Status Bar:** At the bottom, it indicates "Ln 13, Col 1", "Spaces: 4", "UTF-8", "CRLF", "Python", and "3.13.1".

Output Screenshots



```
PROBLEMS OUTPUT TERMINAL PORTS ... [ ] X  
> ▾ TERMINAL  
PS D:\internship week 04 tasks> & "C:/Program Files/Python313/python.exe" "d:/internship week 04 tasks/math and random Modules - Task 02.PY"  
Generated Password: c1ca2@aa  
PS D:\internship week 04 tasks>  
Ln 5, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1
```

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.

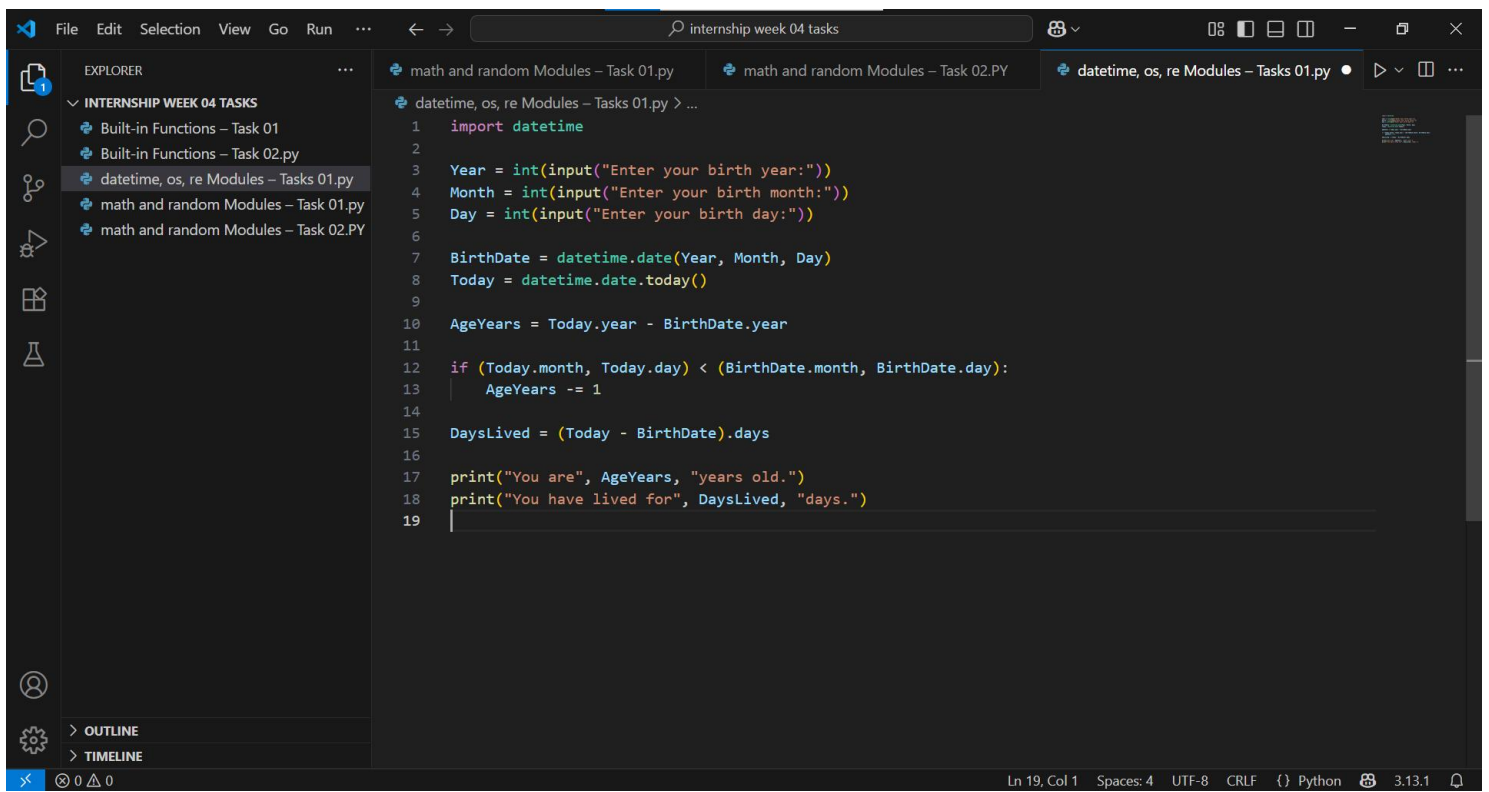
Task 05:

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

What I Did (Step by Step):

- 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.

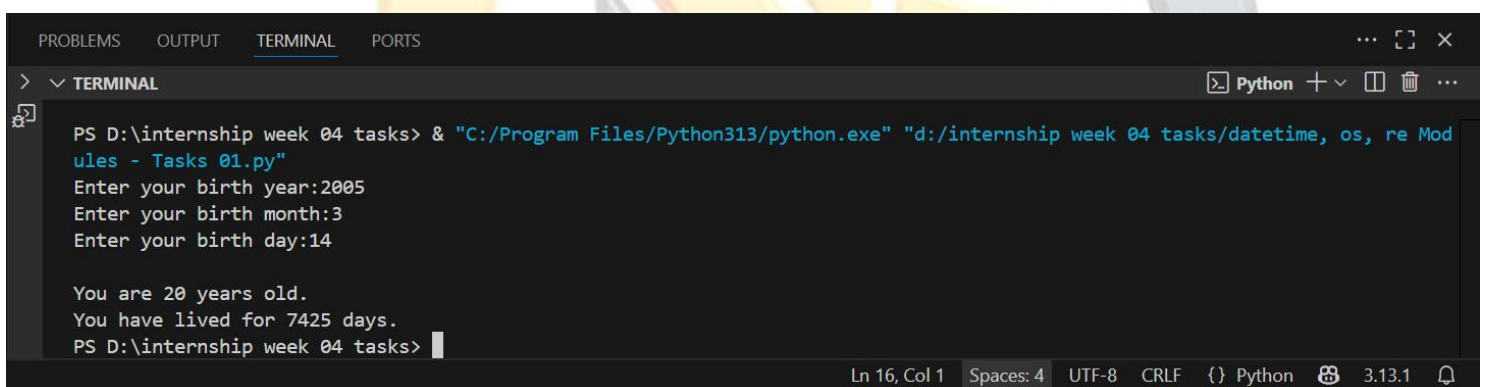
Code Screenshots



The screenshot shows a code editor with a dark theme. The Explorer panel on the left lists files under 'INTERNSHIP WEEK 04 TASKS'. The main editor displays a Python script named 'datetime, os, re Modules - Tasks 01.py'. The script prompts the user for birth year, month, and day, calculates the current age in years, and the number of days lived. The status bar at the bottom indicates the cursor is at line 19, column 1.

```
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 BirthDate = datetime.date(Year, Month, Day)
8 Today = datetime.date.today()
9
10 AgeYears = Today.year - BirthDate.year
11
12 if (Today.month, Today.day) < (BirthDate.month, BirthDate.day):
13     AgeYears -= 1
14
15 DaysLived = (Today - BirthDate).days
16
17 print("You are", AgeYears, "years old.")
18 print("You have lived for", DaysLived, "days.")
19
```

Output Screenshots



The screenshot shows a terminal window with the command prompt. The user has executed the Python script, and the output shows the calculated age and days lived. The status bar at the bottom indicates the cursor is at line 16, column 1.

```
PS D:\internship week 04 tasks> & "C:/Program Files/Python313/python.exe" "d:/internship week 04 tasks/datetime, os, re Modules - Tasks 01.py"
Enter your birth year:2005
Enter your birth month:3
Enter your birth day:14

You are 20 years old.
You have lived for 7425 days.
PS D:\internship week 04 tasks>
```

TECHNIK NEST

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.

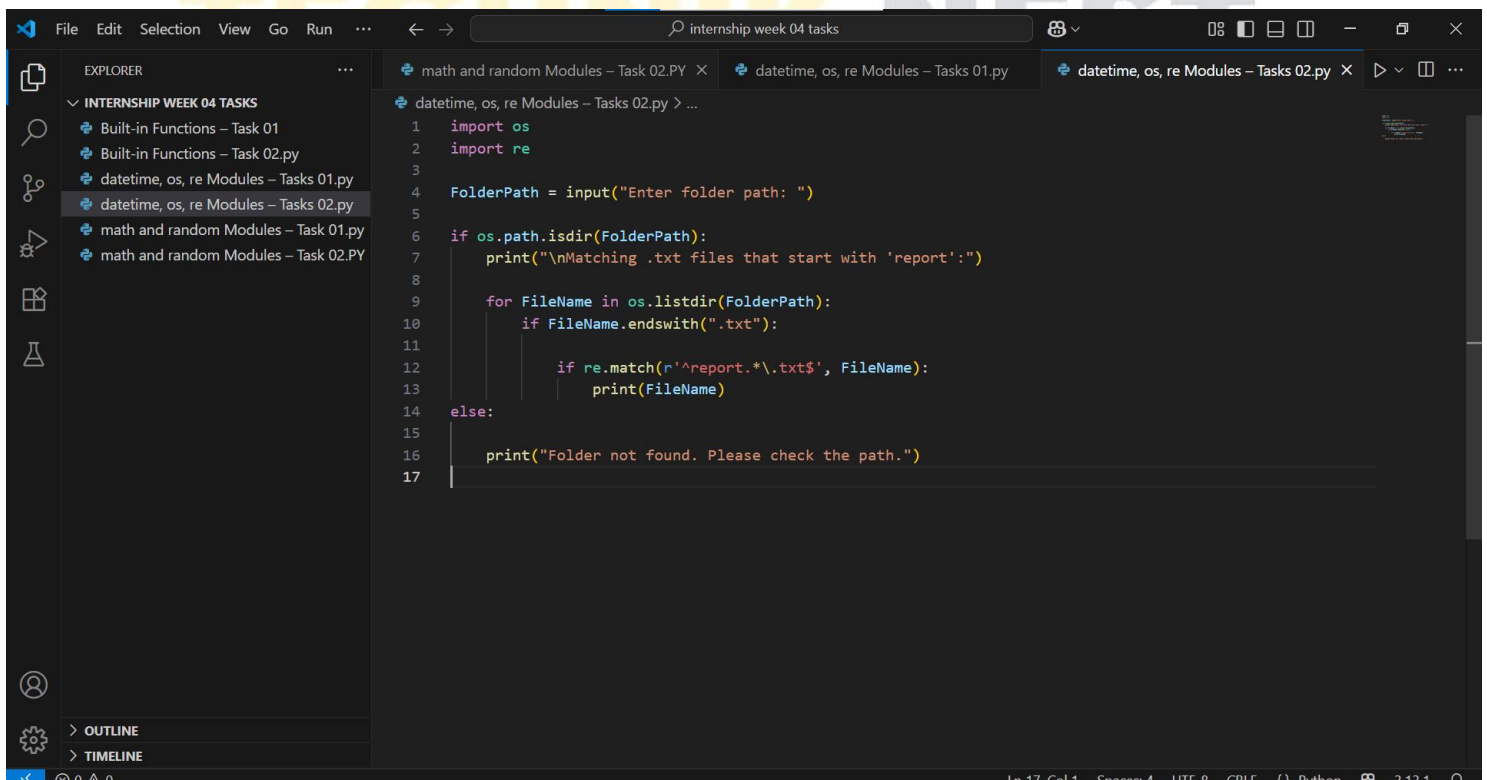
Task 06:

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 (Step by Step):

- 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.

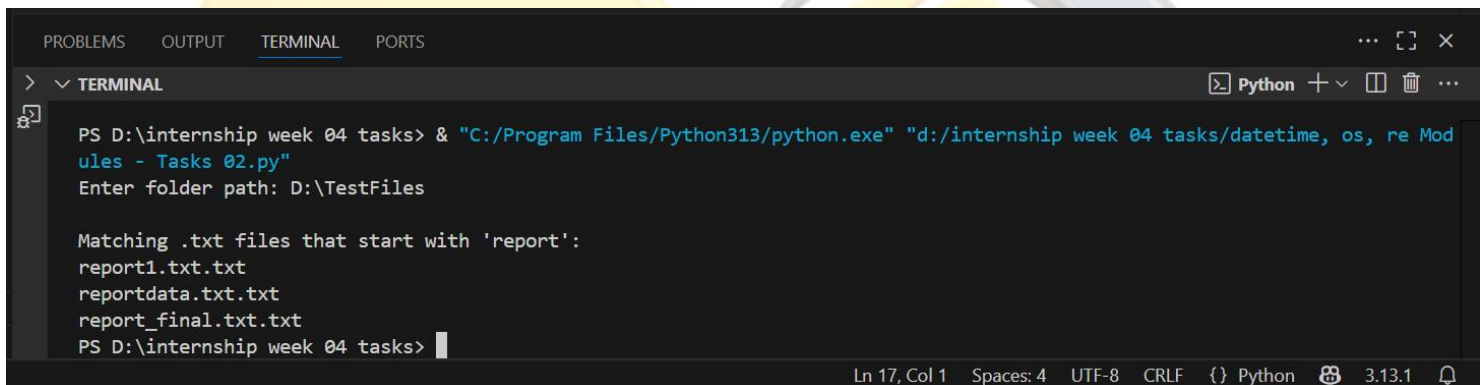
Code Screenshots



```
File Edit Selection View Go Run ... internship week 04 tasks
EXPLORER
INTERNSHIP WEEK 04 TASKS
  Built-in Functions – Task 01
  Built-in Functions – Task 02.py
  datetime, os, re Modules – Tasks 01.py
  datetime, os, re Modules – Tasks 02.py
  math and random Modules – Task 01.py
  math and random Modules – Task 02.PY
OUTLINE
TIMELINE

math and random Modules – Task 02.PY x datetime, os, re Modules – Tasks 01.py
datetime, os, re Modules – Tasks 02.py x
datetime, os, re Modules – Tasks 02.py > ...
1 import os
2 import re
3
4 FolderPath = input("Enter folder path: ")
5
6 if os.path.isdir(FolderPath):
7     print("\nMatching .txt files that start with 'report':")
8
9     for FileName in os.listdir(FolderPath):
10         if FileName.endswith(".txt"):
11
12             if re.match(r'^report.*\.txt$', FileName):
13                 print(FileName)
14 else:
15
16     print("Folder not found. Please check the path.")
17
```

Output Screenshots



```
PROBLEMS OUTPUT TERMINAL PORTS
> TERMINAL Python + - [ ] [ ] [ ]
PS D:\internship week 04 tasks> & "C:/Program Files/Python313/python.exe" "d:/internship week 04 tasks/datetime, os, re Modules - Tasks 02.py"
Enter folder path: D:\TestFiles

Matching .txt files that start with 'report':
report1.txt.txt
reportdata.txt.txt
report_final.txt.txt
PS D:\internship 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 paths.