TECHNIK NEST
INNOVATIVE MINDS, NESTING SUCCESS

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

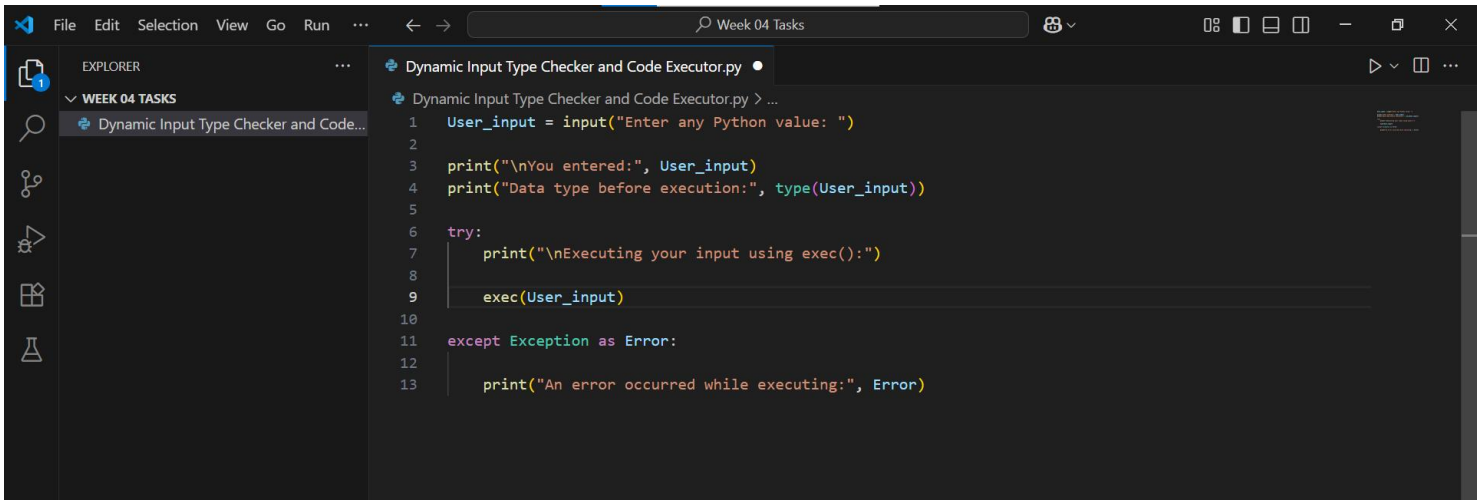
Task 1 :

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



Output Snippet

```
PROBLEMS    OUTPUT    TERMINAL    PORTS
TERMINAL                                                    Python

xecutor.py"
Enter any Python value: print ("Jarvis whatsupp")

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

Executing your input using exec():
Jarvis whatsupp
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:

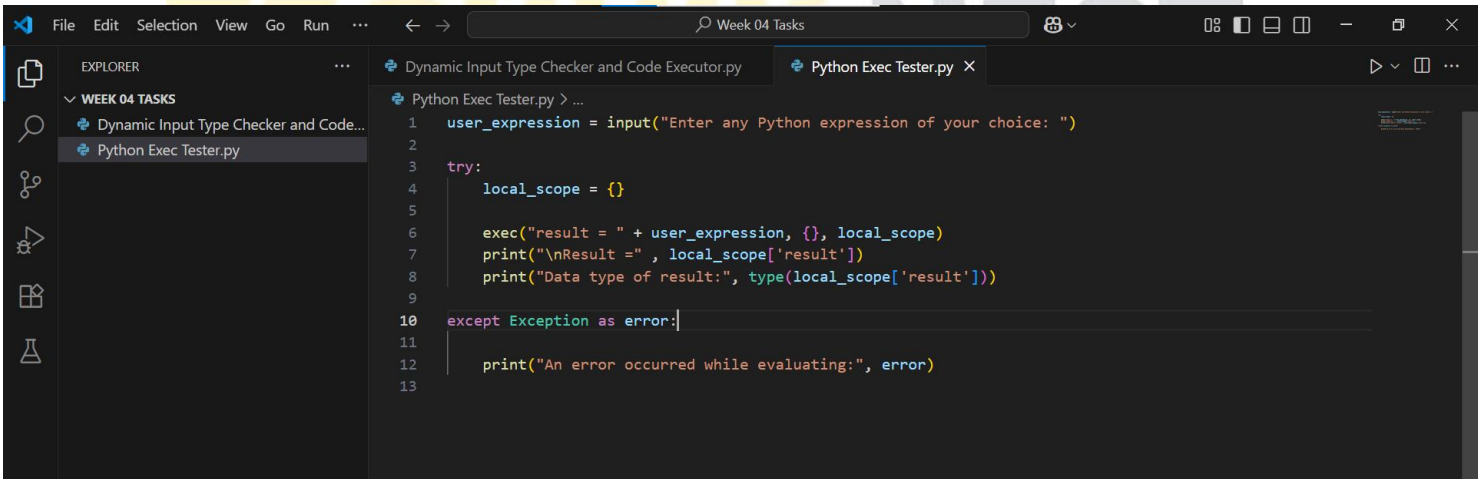
Ask user for a Python expression as a string (like '2 + 3 * 4') and evaluate it using exec().

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



Output Snippet

```
PROBLEMS    OUTPUT    TERMINAL    PORTS
TERMINAL                                                    Python

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

```python
import math

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

Area = math.pi * Radius ** 2
Circumference = 2 * math.pi * Radius
SqrtArea = math.sqrt(Area)


print(f"\nArea of Circle: {Area:.4f}")
print(f"Circumference: {Circumference:.4f}")
print(f"Square Root of Area: {SqrtArea:.4f}")
```

Output Snippets

```
PS E:\Week 04 Tasks> & "C:/Program Files/Python313/python3.13t.exe" "e:/Week 04 Tasks/Pi-Based Calculator.py"
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

```python
# Random Password Generator using random library

import random

Characters = "abcde12345@#$%"

Password = ""

for i in range(10):

    Password += random.choice(Characters)

print(F"Generated Random Password is:{Password}")
```

Output Snippets

```
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.

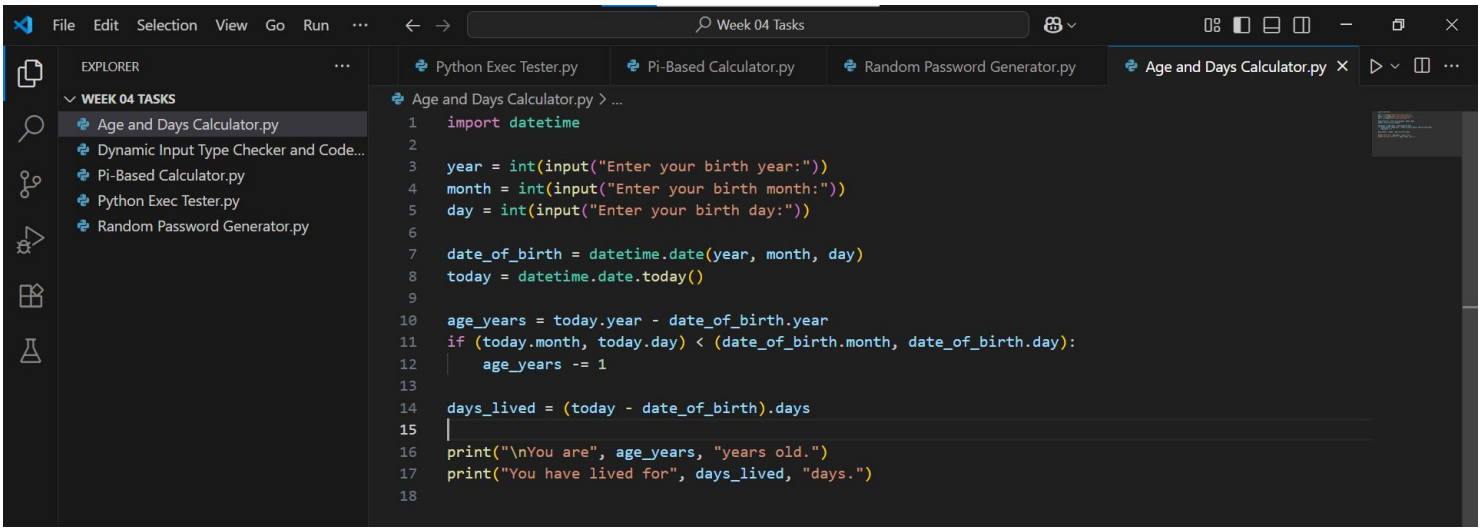
Task 05:

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



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.

<u>Task 06:</u>

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

```python
import os
import re

folder_path = input("Enter folder path: ")

if os.path.isdir(folder_path):

    print("\nMatching .txt files that start with 'report':")

    for file_name in os.listdir(folder_path):

        if file_name.endswith(".txt") and re.match(r'^report.*\.txt$', file_name):
            print(file_name)
else:

    print("Folder not found. Invalid path entered!")
```

Output Snippets

```
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.