



TECHNIK NEST
INNOVATIVE MINDS, NESTING SUCCESS

Name: Hamza Badshah

Intern ID: TN/IN01/PY/001

Email ID : hamzabadshah2592@gmail.com

Internship Domain : Python Development

Task Week : 02

Instructor Name : Mr Hassan Ali

Task 1 :

Create a mini profile for a fictional user using variables. Store the following information:

Full name , Age , Current year, Country, Hobby, Expected graduation year (calculate it from current year + 4)

Print all details in a proper sentence format.

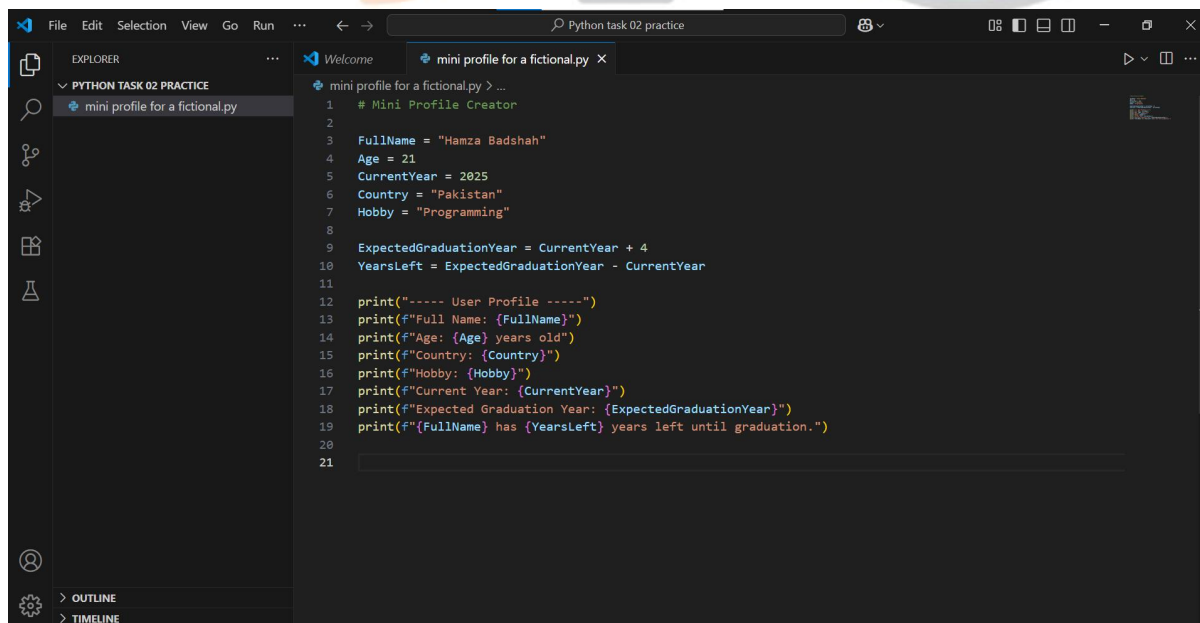
Also print how many years are left till graduation.

Solution :

What I Did (Step by Step):

1. **Created variables** to store the user's full name, age, current year, country, and hobby.
2. **Used a formula** to calculate the expected graduation year by adding 4 to the current year.
3. **Calculated the remaining years** till graduation by subtracting the current year from the expected graduation year.
4. **Printed all details** using formatted strings (f-strings) to form proper sentences.
5. Made sure that the program gives meaningful, human-readable output,

Code Screenshots



```
1 # Mini Profile Creator
2
3 FullName = "Hamza Badshah"
4 Age = 21
5 CurrentYear = 2025
6 Country = "Pakistan"
7 Hobby = "Programming"
8
9 ExpectedGraduationYear = CurrentYear + 4
10 YearsLeft = ExpectedGraduationYear - CurrentYear
11
12 print("---- User Profile ----")
13 print(f"Full Name: {FullName}")
14 print(f"Age: {Age} years old")
15 print(f"Country: {Country}")
16 print(f"Hobby: {Hobby}")
17 print(f"Current Year: {CurrentYear}")
18 print(f"Expected Graduation Year: {ExpectedGraduationYear}")
19 print(f"{FullName} has {YearsLeft} years left until graduation.")
20
21
```

Output Screenshot

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [ ] [ ] ... ^ x
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/mini profile for a fictiona
1.py"
----- User Profile -----
Full Name: Hamza Badshah
Age: 21 years old
Country: Pakistan
Hobby: Programming
Current Year: 2025
Expected Graduation Year: 2029
Hamza Badshah has 4 years left until graduation.
PS D:\Python task 02 practice>
```

Learnings:

- 1) Understood how to use variables to store and manage data.
- 2) Practiced basic arithmetic operations to perform calculations with variables.
- 3) Learned how to use f-strings for clean and readable output.
- 4) Got familiar with the concept of derived values, like calculating future years using current values.

Challenges:

1. Initially, I had to remember to correctly use variable names and not mix them up.
2. I had to ensure that all printed sentences were grammatically correct and logically ordered.
3. Faced a minor issue with formatting, but using f-strings made it easier to organize

Task 02:

Design a command-line survey that:

Asks the user 5 different questions (e.g., name, favorite food, birth year, favorite number, favorite language)

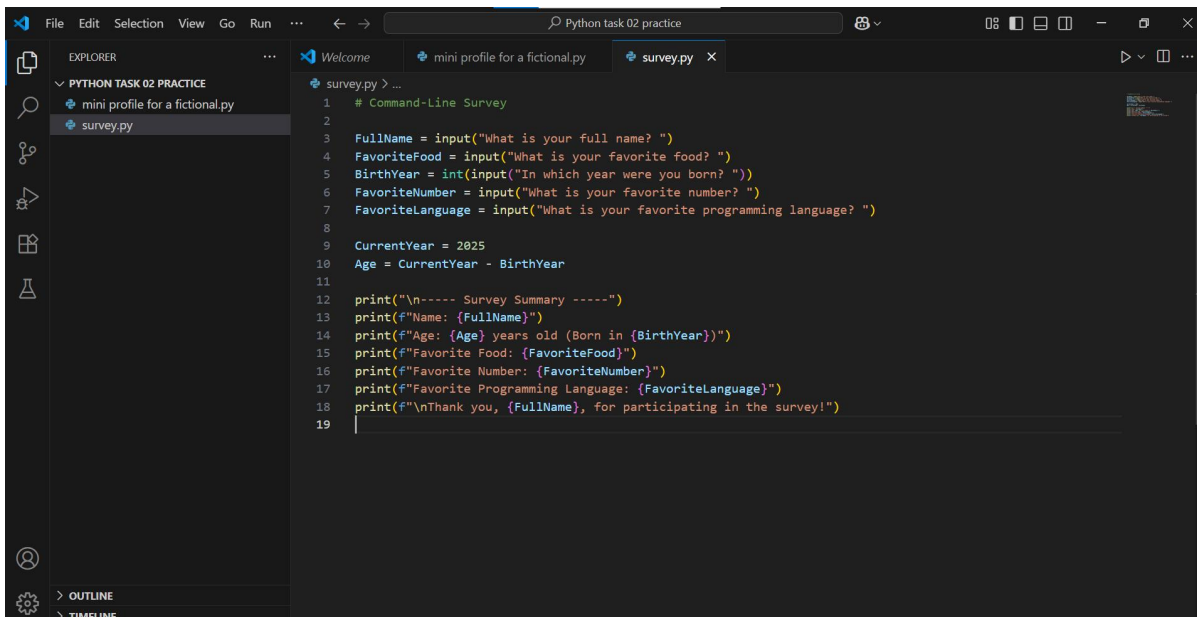
Then prints a summary of all responses in sentence format.

Use formatting to make the output look professional (e.g., f-strings).

What I Did (Step by Step):

- 1) Asked the user 5 questions using input().
- 2) Converted the birth year to an integer and calculated age.
- 3) Stored each response in separate variables.
- 4) Printed a summary using f-strings for clear formatting.
- 5) Used visual separators (like =) to make the output look professional.

Code Screenshots

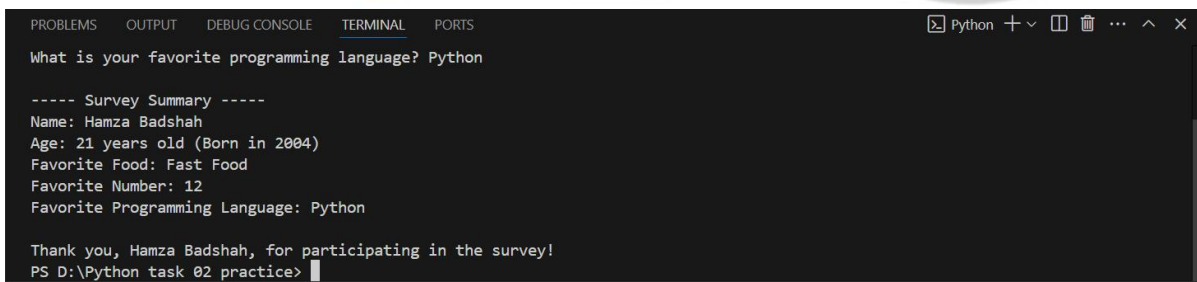


```
1 # Command-Line Survey
2
3 FullName = input("What is your full name? ")
4 FavoriteFood = input("What is your favorite food? ")
5 BirthYear = int(input("In which year were you born? "))
6 FavoriteNumber = input("What is your favorite number? ")
7 FavoriteLanguage = input("What is your favorite programming language? ")
8
9 CurrentYear = 2025
10 Age = CurrentYear - BirthYear
11
12 print("\n----- Survey Summary -----")
13 print(f"Name: {FullName}")
14 print(f"Age: {Age} years old (Born in {BirthYear})")
15 print(f"Favorite Food: {FavoriteFood}")
16 print(f"Favorite Number: {FavoriteNumber}")
17 print(f"Favorite Programming Language: {FavoriteLanguage}")
18 print(f"\nThank you, {FullName}, for participating in the survey!")
19
```

Output Screenshot



```
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/survey.py"
What is your full name? Hamza Badshah
What is your favorite food? Fast Food
In which year were you born? 2004
What is your favorite number? 12
What is your favorite programming language? Python
```



```
----- Survey Summary -----
Name: Hamza Badshah
Age: 21 years old (Born in 2004)
Favorite Food: Fast Food
Favorite Number: 12
Favorite Programming Language: Python

Thank you, Hamza Badshah, for participating in the survey!
PS D:\Python task 02 practice>
```

Learnings:

1. Learned to collect user input using input().
2. Understood type conversion (str to int) for calculations.
3. Practiced using f-strings for neat and readable output.
4. Improved formatting techniques for command-line applications.

Challenges:

- 1) Faced minor confusion while converting birth year input to integer.
- 2) Needed to adjust spacing to make the summary visually clean.
- 3) Ensured proper variable naming to avoid mistakes during printing.

Task 03:

Ask the user to:

Enter their year of birth , Calculate their age (based on current year) ,Check if the user is eligible to vote (18+ years)

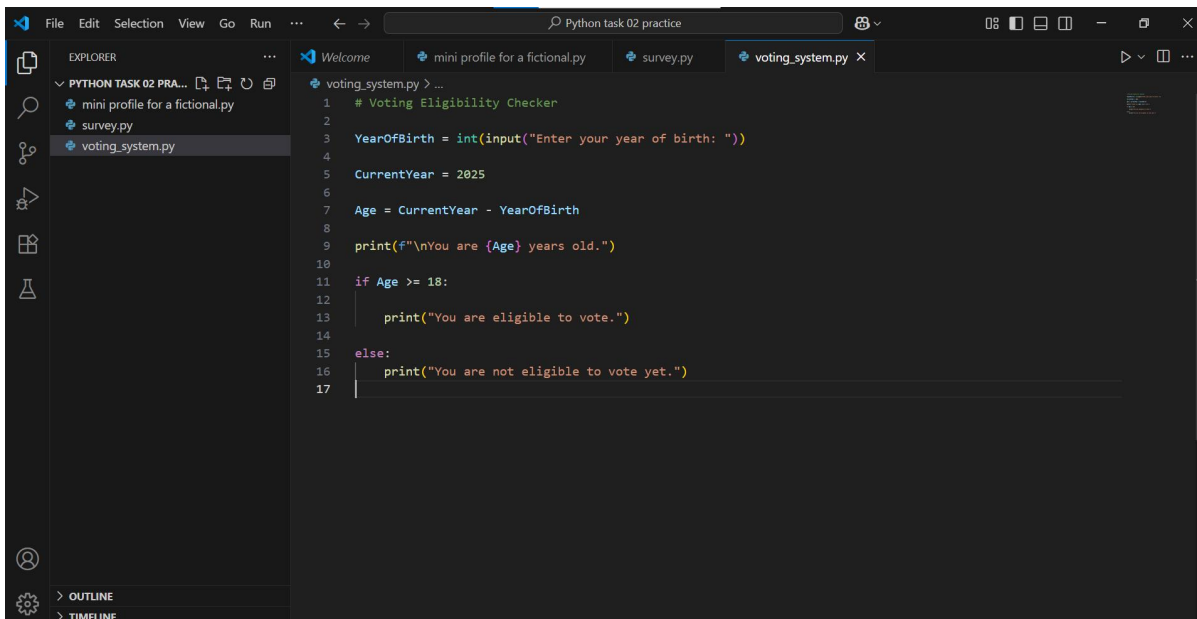
Display a message:

"You are eligible to vote." or "You are not eligible to vote yet."

What I Did (Step by Step):

1. Asked the user to enter their year of birth using input().
2. Converted the input to an integer using int().
3. Defined the current year manually as 2025.
4. Calculated the user's age by subtracting the birth year from the current year.
5. Used an if-else statement to check if the age is 18 or more.
6. Printed a message stating whether the user is eligible to vote or not

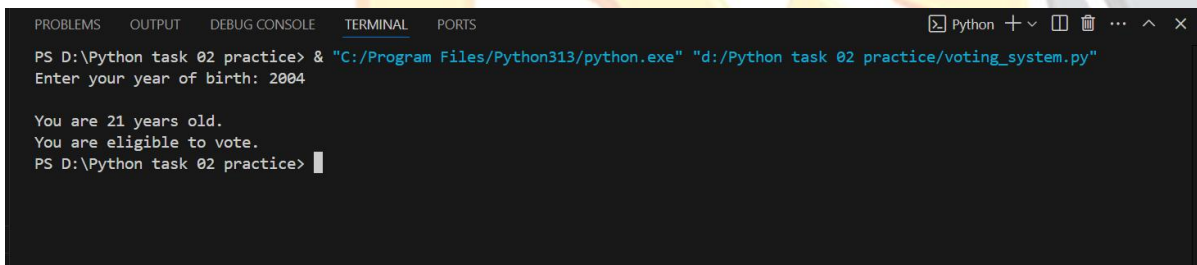
Code Screenshots



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'PYTHON TASK 02 PRA...' with files 'mini profile for a fictional.py', 'survey.py', and 'voting_system.py'. The 'voting_system.py' file is open in the editor. The code is as follows:

```
1 # Voting Eligibility Checker
2
3 YearOfBirth = int(input("Enter your year of birth: "))
4
5 CurrentYear = 2025
6
7 Age = CurrentYear - YearOfBirth
8
9 print(f"\nYou are {Age} years old.")
10
11 if Age >= 18:
12     print("You are eligible to vote.")
13
14 else:
15     print("You are not eligible to vote yet.")
16
17
```

Output Screenshot



The screenshot shows a terminal window with the following output:

```
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/voting_system.py"
Enter your year of birth: 2004

You are 21 years old.
You are eligible to vote.
PS D:\Python task 02 practice>
```

Learnings:

- 1) Practiced converting user input from string to integer.
- 2) Understood how to calculate age using basic arithmetic.
- 3) Learned how to use if-else statements for decision-making.
- 4) Improved user interaction through formatted output.

Challenges:

1. Had to make sure the birth year input is correctly converted to an integer.
2. Needed to manually update the current year (can be dynamic in advanced versions).
3. Ensured the message displayed clearly matches the condition outcome.

Task 04:

Create 3 different user profiles (using variables). For each profile, include:

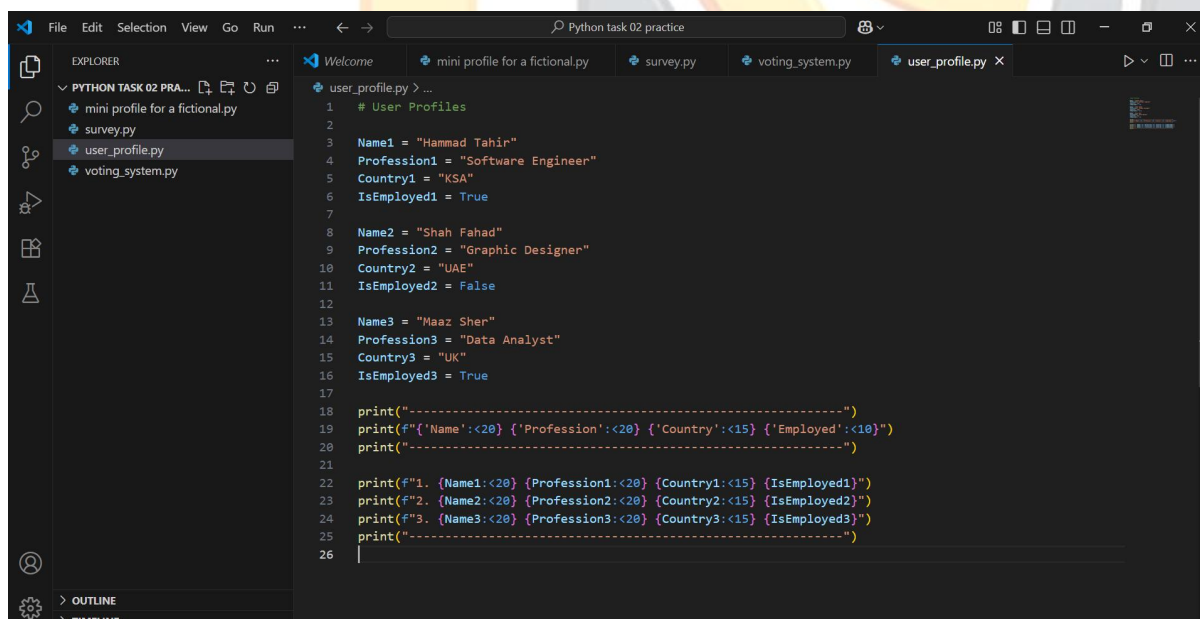
Name, profession, country, is_employed (Boolean)

Print their data in a tabular format using print() (not with external libraries).

What I Did (Step by Step):

- 1) Created 3 separate user profiles using variables for each: Name, Profession, Country, and IsEmployed.
- 2) Used print() statements to output a table header with column names.
- 3) Formatted each line using f-strings with alignment (<15, <20, etc.) to keep columns neat.
- 4) Printed a separator line (=) above and below the table for better readability.
- 5) Converted the Boolean values to strings to display True/False clearly in the table.

Code Screenshots



```
1 # User Profiles
2
3 Name1 = "Hammad Tahir"
4 Profession1 = "Software Engineer"
5 Country1 = "KSA"
6 IsEmployed1 = True
7
8 Name2 = "Shah Fahad"
9 Profession2 = "Graphic Designer"
10 Country2 = "UAE"
11 IsEmployed2 = False
12
13 Name3 = "Maaz Sher"
14 Profession3 = "Data Analyst"
15 Country3 = "UK"
16 IsEmployed3 = True
17
18 print("-----")
19 print(f"{'Name':<20} {'Profession':<20} {'Country':<15} {'Employed':<10}")
20 print("-----")
21
22 print(f"1. {Name1:<20} {Profession1:<20} {Country1:<15} {IsEmployed1}")
23 print(f"2. {Name2:<20} {Profession2:<20} {Country2:<15} {IsEmployed2}")
24 print(f"3. {Name3:<20} {Profession3:<20} {Country3:<15} {IsEmployed3}")
25 print("-----")
26
```

Output Screenshots

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/user_profile.py"
-----
Name           Profession      Country      Employed
-----
1. Hammad Tahir   Software Engineer   KSA          True
2. Shah Fahad     Graphic Designer    UAE          False
3. Maaz Sher      Data Analyst        UK           True
-----
PS D:\Python task 02 practice>
```

Learnings

1. Practiced creating multiple user profiles using individual variables.
2. Learned how to format data into a table using f-strings and alignment operators.
3. Improved understanding of how to control column width and spacing in console output.

Challenges

- 1) Had to carefully align column widths to avoid messy output.
- 2) Needed to convert Boolean values to string for proper display.
- 3) Managed multiple similar variables without using more advanced structures (like lists or dictionaries).

Task 05:

Write a program that:

Declares five different variables , Stores a different data type in each (e.g., string, integer, float, boolean, complex)

Prints their values and data types

Then, converts each variable to a different type (where possible) and prints the new types

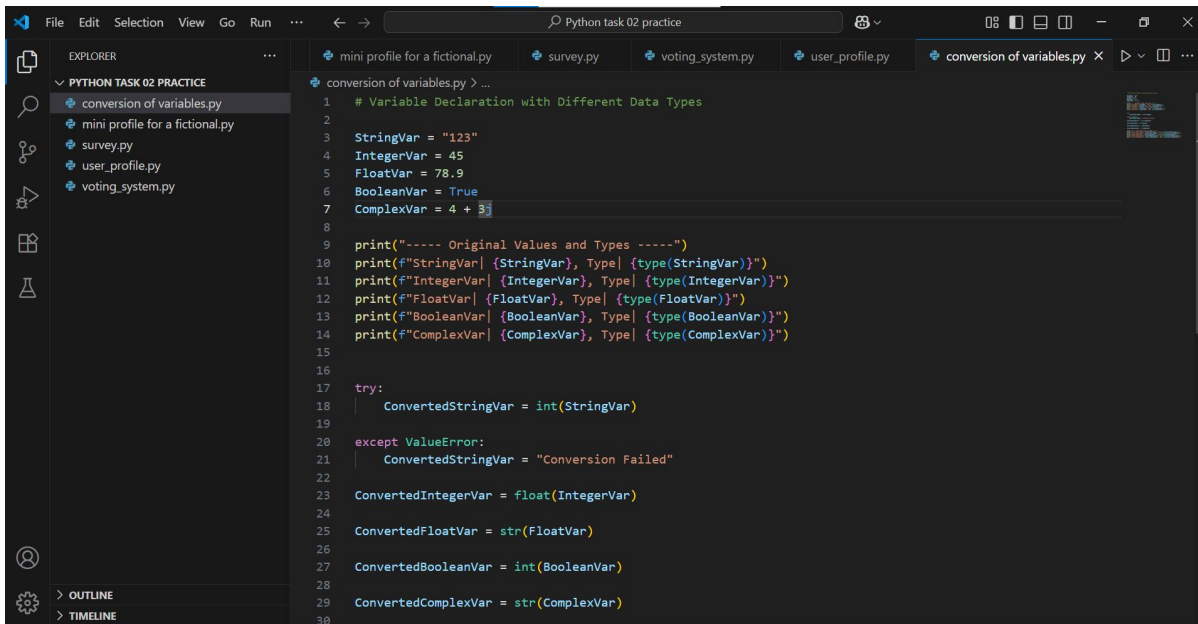
Note: You may not be able to convert all types — handle errors or comment why.

What I Did (Step by Step):

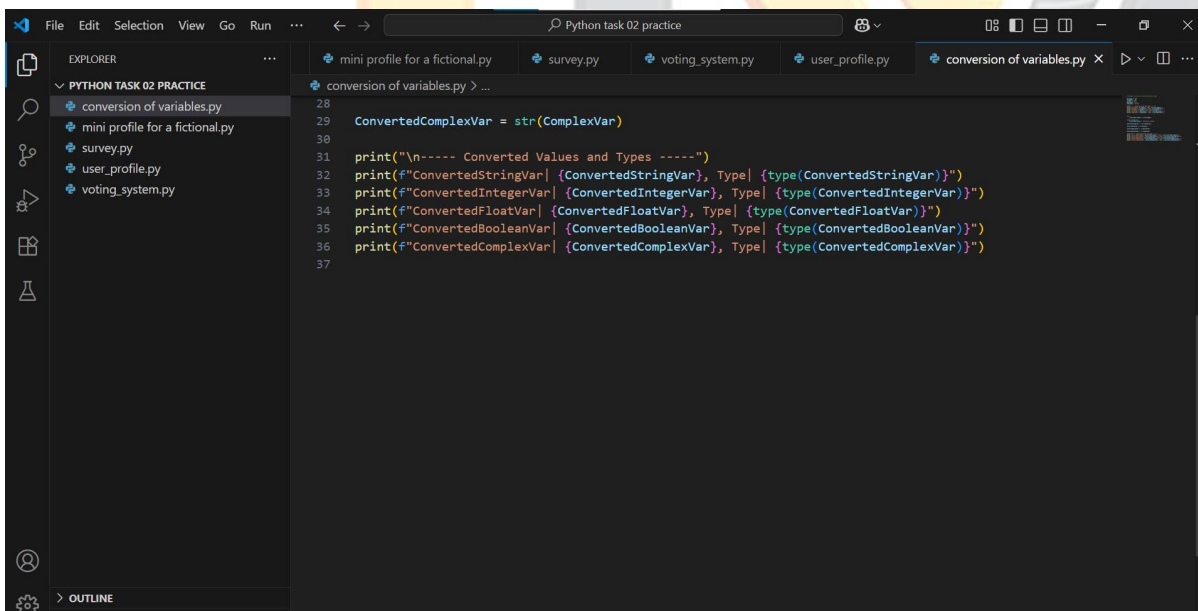
1. Declared 5 variables of different types: string, integer, float, boolean, and complex.
2. Printed each variable's value along with its data type using type().

3. Converted each variable (where possible) into another type using built-in functions like `float()`, `int()`, or `str()`.
4. Printed the new values and their types.
5. Commented out the conversion of string to integer for non-numeric data to avoid errors.

Code Screenshots



```
1 # Variable Declaration with Different Data Types
2
3 StringVar = "123"
4 IntegerVar = 45
5 FloatVar = 78.9
6 BooleanVar = True
7 ComplexVar = 4 + 3j
8
9 print("----- Original Values and Types -----")
10 print(f"StringVar| {StringVar}, Type| {type(StringVar)}")
11 print(f"IntegerVar| {IntegerVar}, Type| {type(IntegerVar)}")
12 print(f"FloatVar| {FloatVar}, Type| {type(FloatVar)}")
13 print(f"BooleanVar| {BooleanVar}, Type| {type(BooleanVar)}")
14 print(f"ComplexVar| {ComplexVar}, Type| {type(ComplexVar)}")
15
16
17 try:
18     ConvertedStringVar = int(StringVar)
19
20 except ValueError:
21     ConvertedStringVar = "Conversion Failed"
22
23 ConvertedIntegerVar = float(IntegerVar)
24
25 ConvertedFloatVar = str(FloatVar)
26
27 ConvertedBooleanVar = int(BooleanVar)
28
29 ConvertedComplexVar = str(ComplexVar)
30
```



```
28
29 ConvertedComplexVar = str(ComplexVar)
30
31 print("\n----- Converted Values and Types -----")
32 print(f"ConvertedStringVar| {ConvertedStringVar}, Type| {type(ConvertedStringVar)}")
33 print(f"ConvertedIntegerVar| {ConvertedIntegerVar}, Type| {type(ConvertedIntegerVar)}")
34 print(f"ConvertedFloatVar| {ConvertedFloatVar}, Type| {type(ConvertedFloatVar)}")
35 print(f"ConvertedBooleanVar| {ConvertedBooleanVar}, Type| {type(ConvertedBooleanVar)}")
36 print(f"ConvertedComplexVar| {ConvertedComplexVar}, Type| {type(ConvertedComplexVar)}")
37
```

Output Screenshots

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [ ] [ ] ... ^ x
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/conversion of variables.py"

----- Original Values and Types -----
StringVar| 123, Type| <class 'str'>
IntegerVar| 45, Type| <class 'int'>
FloatVar| 78.9, Type| <class 'float'>
BooleanVar| True, Type| <class 'bool'>
ComplexVar| (4+3j), Type| <class 'complex'>
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [ ] [ ] ... ^ x
FloatVar| 78.9, Type| <class 'float'>
BooleanVar| True, Type| <class 'bool'>
ComplexVar| (4+3j), Type| <class 'complex'>

----- Converted Values and Types -----
ConvertedStringVar| 123, Type| <class 'int'>
ConvertedIntegerVar| 45.0, Type| <class 'float'>
ConvertedFloatVar| 78.9, Type| <class 'str'>
ConvertedBooleanVar| 1, Type| <class 'int'>
ConvertedComplexVar| (4+3j), Type| <class 'str'>
PS D:\Python task 02 practice>
```

Learnings

- 1) Declared 5 variables with different data types: str, int, float, bool, complex.
- 2) Printed their original values and data types using type().
- 3) Converted each to another type where possible (e.g., int to float, float to int, etc.).
- 4) Printed the new values and their types after conversion.
- 5) Skipped or commented conversions that are not valid (like string "Hamza" to int).

Challenges

1. Faced an error when trying to convert a non-numeric string to int or float.
2. Needed to understand which type conversions are logically and syntactically allowed.
3. Ensured error-prone conversions are avoided or clearly commented.

Task 06:

Create a data type tester:

Ask the user to input any value. , Detect and print what Python guesses its type as (use type()).

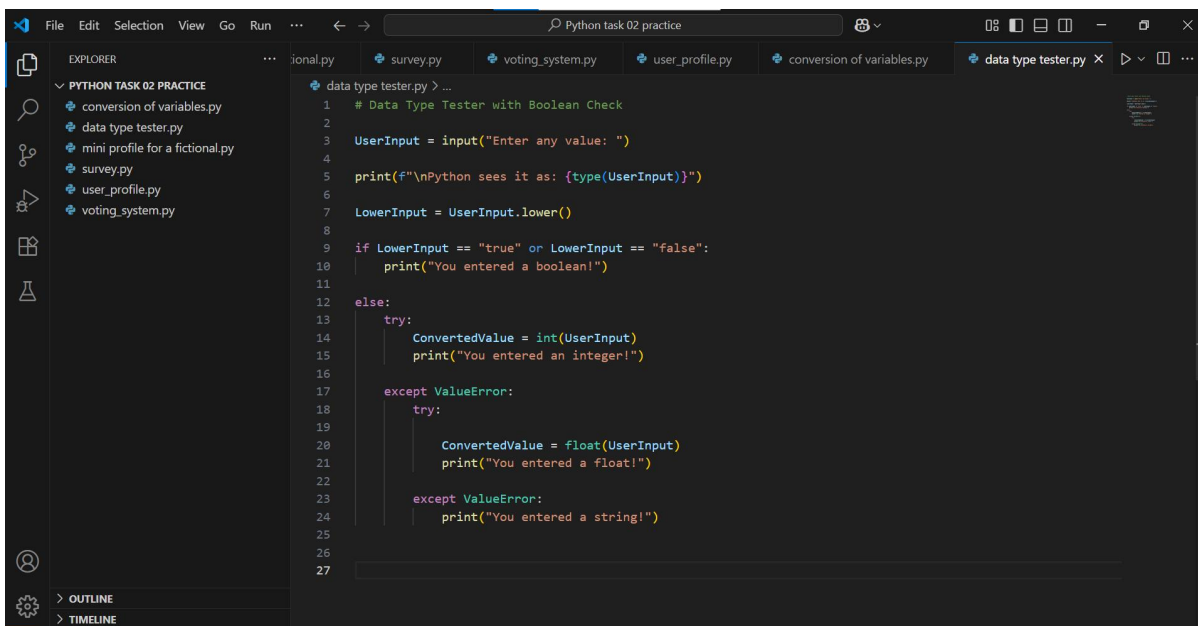
Add conditions to identify if it's likely an integer, float, or string, and print a message like:

"You entered a float!"

What I Did (Step by Step):

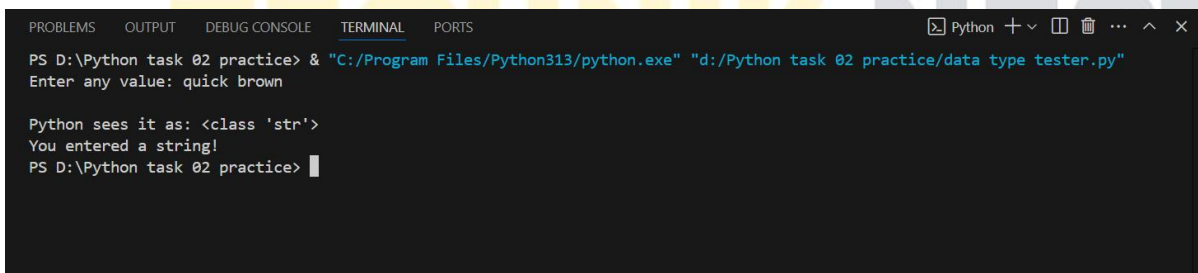
- 1) Used `input()` to take a value from the user as a string.
- 2) Tried to convert the input to an int using `int()`.
- 3) If it failed, tried to convert it to a float using `float()`.
- 4) If both conversions failed, treated the value as a string.
- 5) Printed the detected Python type using `type()` and showed a friendly message like "You entered a float!".

Code Screenshots



```
1 # Data Type Tester with Boolean Check
2
3 UserInput = input("Enter any value: ")
4
5 print(f"\nPython sees it as: {type(UserInput)}")
6
7 LowerInput = UserInput.lower()
8
9 if LowerInput == "true" or LowerInput == "false":
10     print("You entered a boolean!")
11
12 else:
13     try:
14         ConvertedValue = int(UserInput)
15         print("You entered an integer!")
16
17     except ValueError:
18         try:
19             ConvertedValue = float(UserInput)
20             print("You entered a float!")
21
22         except ValueError:
23             print("You entered a string!")
24
25
26
27
```

Output Screenshots



```
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/data type tester.py"
Enter any value: quick brown

Python sees it as: <class 'str'>
You entered a string!
PS D:\Python task 02 practice>
```

Learnings

1. Learned how to use try-except to handle input conversion errors.
2. Understood the concept of dynamic typing in Python using `type()`.
3. Practiced conditional logic to identify and label data types.

Challenges

- 1) Needed to carefully order type checking (int before float) to avoid incorrect detection.
- 2) Faced potential errors when converting input that's not purely numeric.
- 3) Had to make sure the output was user-friendly and accurate.

Task 07:

Create a marks percentage calculator:

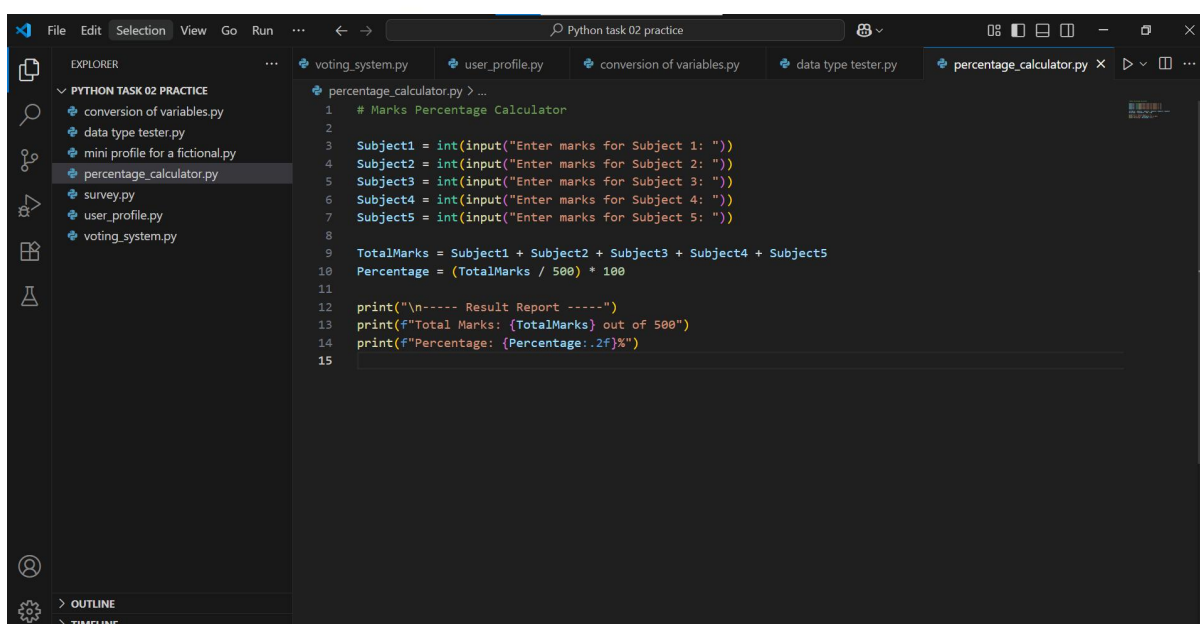
Ask user to input marks for 5 subjects (input as strings) , Convert them to integers

Calculate the total and percentage

What I Did (Step by Step):

1. Used input() to take marks for 5 subjects from the user.
2. Converted the input strings to integers using int().
3. Calculated the total marks by adding all 5 subject marks.
4. Calculated the percentage using the formula: $(\text{TotalMarks} / 500) * 100$.
5. Displayed the total and percentage using formatted print statements.

Code Screenshots



```
1 # Marks Percentage Calculator
2
3 Subject1 = int(input("Enter marks for Subject 1: "))
4 Subject2 = int(input("Enter marks for Subject 2: "))
5 Subject3 = int(input("Enter marks for Subject 3: "))
6 Subject4 = int(input("Enter marks for Subject 4: "))
7 Subject5 = int(input("Enter marks for Subject 5: "))
8
9 TotalMarks = Subject1 + Subject2 + Subject3 + Subject4 + Subject5
10 Percentage = (TotalMarks / 500) * 100
11
12 print("\n----- Result Report -----")
13 print(f"Total Marks: {TotalMarks} out of 500")
14 print(f"Percentage: {Percentage:.2f}%")
15
```

Output Screenshots

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/percentage_calculator.py"
Enter marks for Subject 1: 56
Enter marks for Subject 2: 76
Enter marks for Subject 3: 54
Enter marks for Subject 4: 67
Enter marks for Subject 5: 87

----- Result Report -----
Total Marks: 340 out of 500
Percentage: 68.00%
PS D:\Python task 02 practice> |
```

Learnings

- 1) Practiced taking and converting multiple user inputs.
- 2) Learned how to calculate total and percentage using basic arithmetic.
- 3) Improved formatting skills using f-strings with decimal precision.

Challenges

1. Needed to ensure all inputs were converted to integers to avoid type errors.
2. Remembered to use the correct total (500) when calculating the percentage.
3. Could improve by adding input validation to handle non-numeric entries.

Task 08:

Create a temperature converter:

Ask the user to input temperature in Celsius.

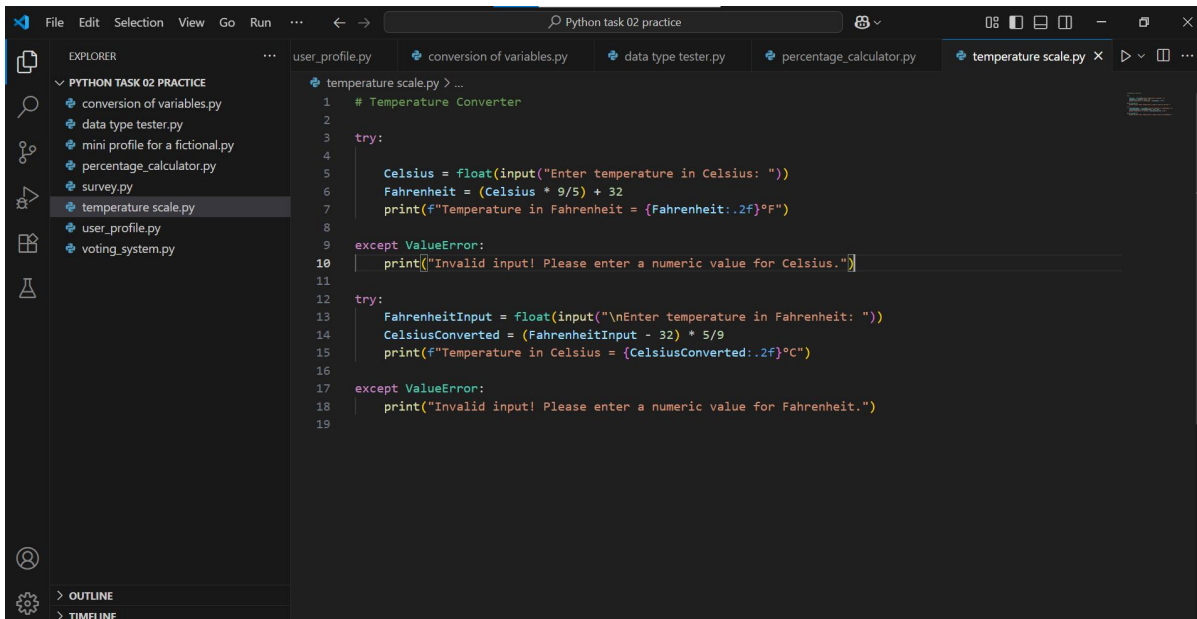
Convert it to Fahrenheit using: $F = (C * 9/5) + 32$, Then reverse: Ask for Fahrenheit, convert it to Celsius.

Handle wrong input types using try-except.

What I Did (Step by Step):

- 1) Used input() to ask the user for a Celsius temperature.
- 2) Converted the Celsius value to float and calculated Fahrenheit using the formula.
- 3) Printed the result using an f-string with 2 decimal places.
- 4) Used a try-except block to handle invalid (non-numeric) input.
- 5) Repeated the same process for Fahrenheit to Celsius conversion.

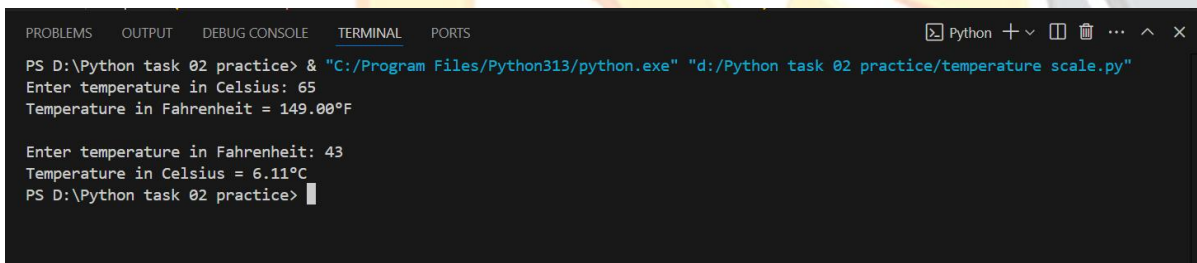
Code Screenshots



The screenshot shows a Python IDE with a file explorer on the left and a code editor on the right. The file explorer lists several Python files under the 'PYTHON TASK 02 PRACTICE' folder, including 'temperature scale.py'. The code editor displays the code for 'temperature scale.py', which is a temperature converter. The code uses try-except blocks to handle user input and convert between Celsius and Fahrenheit.

```
1 # Temperature Converter
2
3 try:
4
5     Celsius = float(input("Enter temperature in Celsius: "))
6     Fahrenheit = (Celsius * 9/5) + 32
7     print(f"Temperature in Fahrenheit = {Fahrenheit:.2f}°F")
8
9 except ValueError:
10     print("Invalid input! Please enter a numeric value for Celsius.")
11
12 try:
13     FahrenheitInput = float(input("\nEnter temperature in Fahrenheit: "))
14     CelsiusConverted = (FahrenheitInput - 32) * 5/9
15     print(f"Temperature in Celsius = {CelsiusConverted:.2f}°C")
16
17 except ValueError:
18     print("Invalid input! Please enter a numeric value for Fahrenheit.")
19
```

Output Screenshots



The screenshot shows a terminal window with the output of the temperature scale converter. The prompt is 'PS D:\Python task 02 practice>'. The user enters '65' for Celsius, and the output is 'Temperature in Fahrenheit = 149.00°F'. The user then enters '43' for Fahrenheit, and the output is 'Temperature in Celsius = 6.11°C'. The prompt is 'PS D:\Python task 02 practice>'.

```
PS D:\Python task 02 practice> & "C:/Program Files/Python313/python.exe" "d:/Python task 02 practice/temperature scale.py"
Enter temperature in Celsius: 65
Temperature in Fahrenheit = 149.00°F

Enter temperature in Fahrenheit: 43
Temperature in Celsius = 6.11°C
PS D:\Python task 02 practice>
```

Learnings

1. Learned how to use temperature conversion formulas in Python.
2. Practiced converting user input to float for accurate calculations.
3. Used f-strings to format the output with decimal precision.
4. Understood how to handle errors gracefully using try-except.

Challenges

- 1) Ensured input was numeric to avoid crashes on invalid entries.
- 2) Remembered correct formulas and applied them in both directions.
- 3) Managed output formatting to keep it clean and user-friendly.

