



TECHNIK NEST

INNOVATIVE MINDS, NESTING SUCCESS

Name: Musfira Ahmed

Intern ID: TN/IN01/PY/002

Email ID : ahmedmusfira3@gmail.com

Internship Domain : Python Development

Task Week : 02

Instructor Name : 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.

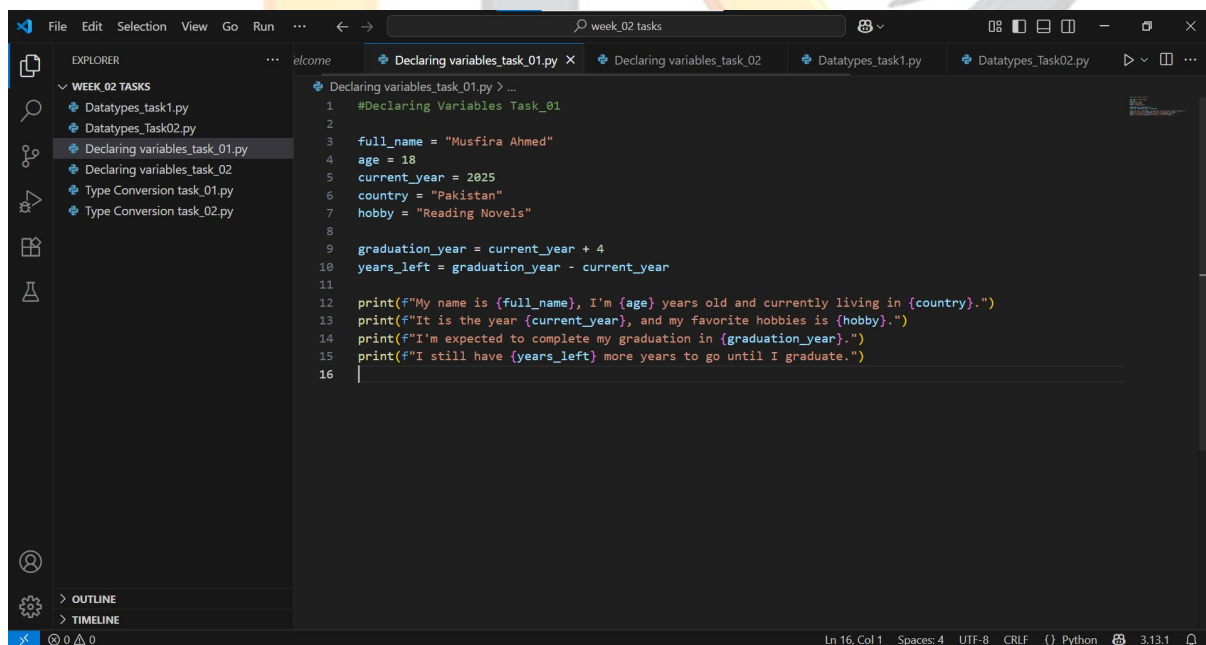
Also print how many years are left till graduation.

Solution :

What I Did (Step by Step):

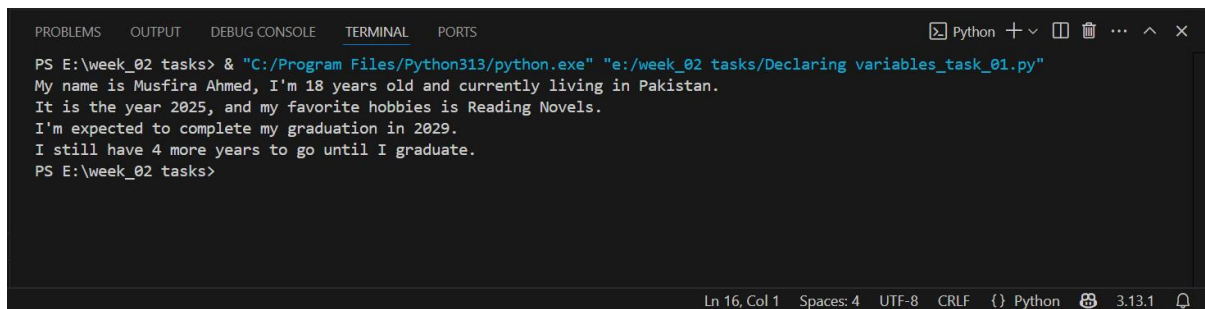
1. Defined variables for user profile (name, age, year, etc.).
2. Calculated the expected graduation year.
3. Printed the details in proper sentence format.

Code Screenshots



```
1 #Declaring Variables Task_01
2
3 full_name = "Musfira Ahmed"
4 age = 18
5 current_year = 2025
6 country = "Pakistan"
7 hobby = "Reading Novels"
8
9 graduation_year = current_year + 4
10 years_left = graduation_year - current_year
11
12 print(f"My name is {full_name}, I'm {age} years old and currently living in {country}.")
13 print(f"It is the year {current_year}, and my favorite hobbies is {hobby}.")
14 print(f"I'm expected to complete my graduation in {graduation_year}.")
15 print(f"I still have {years_left} more years to go until I graduate.")
16
```

Output Screenshot

A screenshot of a Python terminal window. The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active. The command prompt shows the execution of a Python script: `PS E:\week_02 tasks> & "C:/Program Files/Python313/python.exe" "e:/week_02 tasks/Declaring variables_task_01.py"`. The output of the script is displayed below the command: `My name is Musfira Ahmed, I'm 18 years old and currently living in Pakistan.
It is the year 2025, and my favorite hobbies is Reading Novels.
I'm expected to complete my graduation in 2029.
I still have 4 more years to go until I graduate.
PS E:\week_02 tasks>`. The status bar at the bottom indicates the current line and column (Ln 16, Col 1), the number of spaces (Spaces: 4), the encoding (UTF-8), the line ending (CRLF), the file type (Python), and the version (3.13.1).

Learnings and Challenges:

- 1) Learned about variables, string formatting using f-strings.
- 2) Practiced combining text and calculations in output.

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. Collected user input using `input()` for name, food, birth year, number, and language.
2. Converted birth year to `int` and calculated age using `2025 - birth_year`.
3. Used f-strings for a clean, formatted summary with emojis for better readability.

Code Screenshots

```
1 print("=== Welcome to the Quick Survey ===\n")
2
3
4 name = input("1. What is your name? ")
5 fav_food = input("2. What is your favorite food? ")
6 birth_year = input("3. What year were you born? ")
7 fav_number = input("4. What is your favorite number? ")
8 fav_language = input("5. What is your favorite programming language? ")
9
10 print("\n=== Survey Summary ===\n")
11 print(f"Hello {name}, it's great to know about you!")
12 print(f"You love eating {fav_food}.")
13 print(f"You were born in {birth_year}, which makes you {2025 - int(birth_year)} years old (if it's 2025).")
14 print(f"Your favorite number is {fav_number}.")
15 print(f"And your favorite programming language is {fav_language}.")
16 print("\nThank you for participating in the survey!")
17
```

Output Screenshot

```
PS E:\week_02 tasks> & "C:/Program Files/Python313/python.exe" "e:/week_02 tasks/input_output_task-01.py"
=== Welcome to the Quick Survey ===

1. What is your name? Musfira Ahmed
2. What is your favorite food? Biryani
3. What year were you born? 2006
4. What is your favorite number? 5
5. What is your favorite programming language? Python

=== Survey Summary ===
```

```
=== Survey Summary ===

Hello Musfira Ahmed, it's great to know about you!
You love eating Biryani.
You were born in 2006, which makes you 19 years old (if it's 2025).
Your favorite number is 5.
And your favorite programming language is Python.

Thank you for participating in the survey!
PS E:\week_02 tasks>
```

Learnings and Challenges:

1. Learned to collect user input and store it in variables.
2. Practiced converting strings to integers for age calculation.
3. Improved use of f-strings for clean, personalized output.
4. Focused on user-friendly formatting with emojis and spacing.

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)

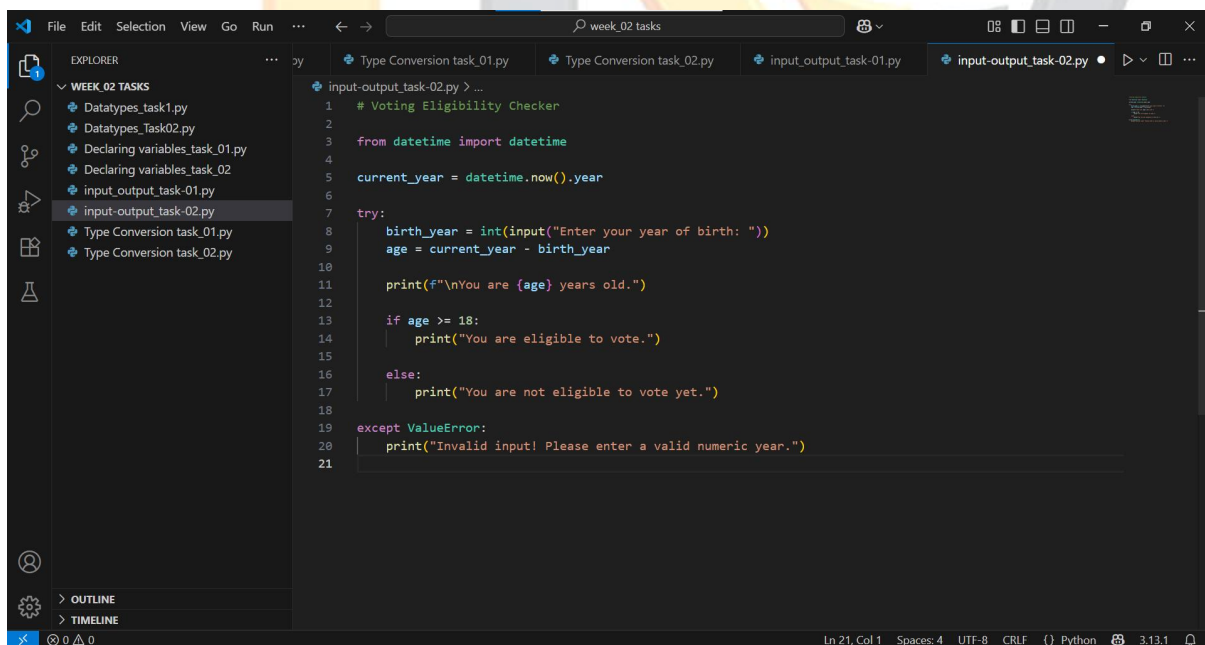
Display a message:

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

What I Did (Step by Step):

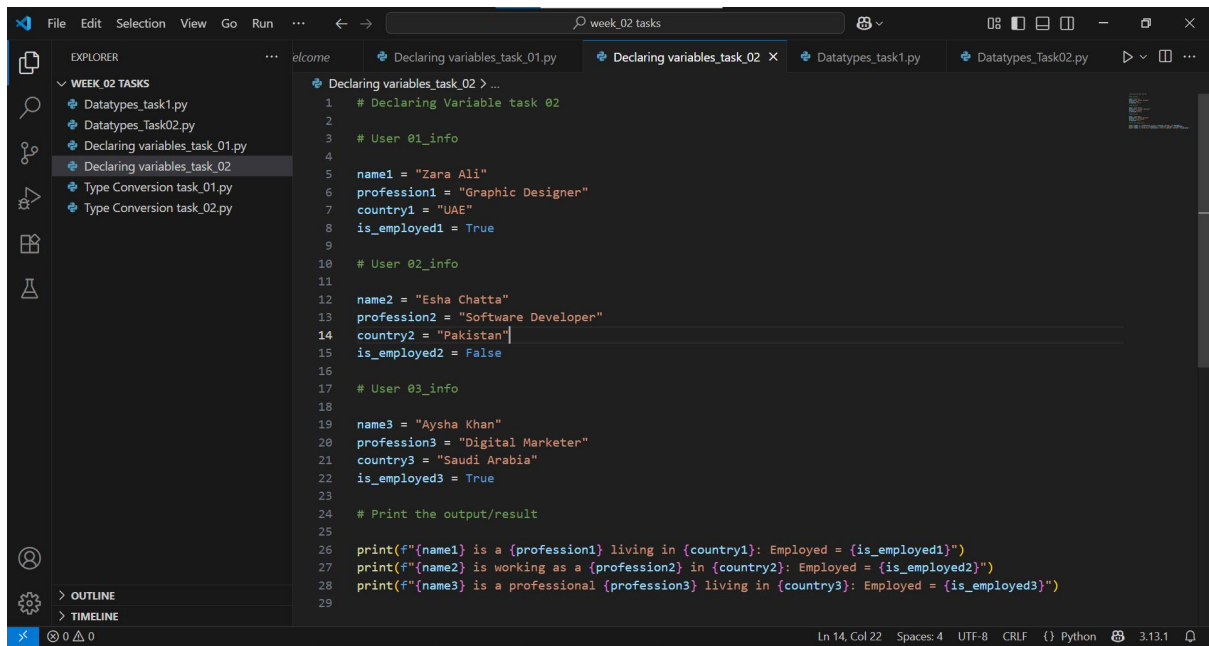
- 1) Asks for the user's year of birth,
- 2) Calculates their age,
- 3) Checks if they're eligible to vote (18+),
- 4) And prints a clear message based on the result.

Code Screenshots

A screenshot of a Python IDE (VS Code) showing a file named 'input-output_task-02.py'. The code is a 'Voting Eligibility Checker' that prompts the user for their birth year, calculates their age based on the current year, and prints a message indicating if they are eligible to vote (18+ years old). The code includes error handling for invalid input.

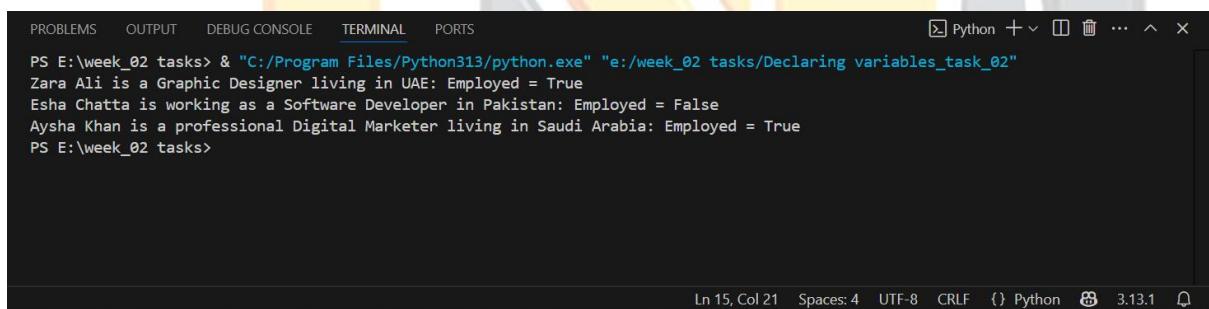
```
1 # Voting Eligibility Checker
2
3 from datetime import datetime
4
5 current_year = datetime.now().year
6
7 try:
8     birth_year = int(input("Enter your year of birth: "))
9     age = current_year - birth_year
10
11     print(f"\nYou are {age} years old.")
12
13     if age >= 18:
14         print("You are eligible to vote.")
15     else:
16         print("You are not eligible to vote yet.")
17
18 except ValueError:
19     print("Invalid input! Please enter a valid numeric year.")
20
21
```

Output Screenshot



```
1 # Declaring Variable task 02
2
3 # User 01_info
4
5 name1 = "Zara Ali"
6 profession1 = "Graphic Designer"
7 country1 = "UAE"
8 is_employed1 = True
9
10 # User 02_info
11
12 name2 = "Esha Chatta"
13 profession2 = "Software Developer"
14 country2 = "Pakistan"
15 is_employed2 = False
16
17 # User 03_info
18
19 name3 = "Aysha Khan"
20 profession3 = "Digital Marketer"
21 country3 = "Saudi Arabia"
22 is_employed3 = True
23
24 # Print the output/result
25
26 print(f"name1 is a {profession1} living in {country1}: Employed = {is_employed1}")
27 print(f"name2 is working as a {profession2} in {country2}: Employed = {is_employed2}")
28 print(f"name3 is a professional {profession3} living in {country3}: Employed = {is_employed3}")
29
```

Output Screenshots



```
PS E:\week_02 tasks> & "C:/Program Files/Python313/python.exe" "e:/week_02 tasks/Declaring variables_task_02"
Zara Ali is a Graphic Designer living in UAE: Employed = True
Esha Chatta is working as a Software Developer in Pakistan: Employed = False
Aysha Khan is a professional Digital Marketer living in Saudi Arabia: Employed = True
PS E:\week_02 tasks>
```

Learnings and Challenges:

1. Understood how to manage and format multiple sets of data.
2. Learned about boolean values and consistent formatting.

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

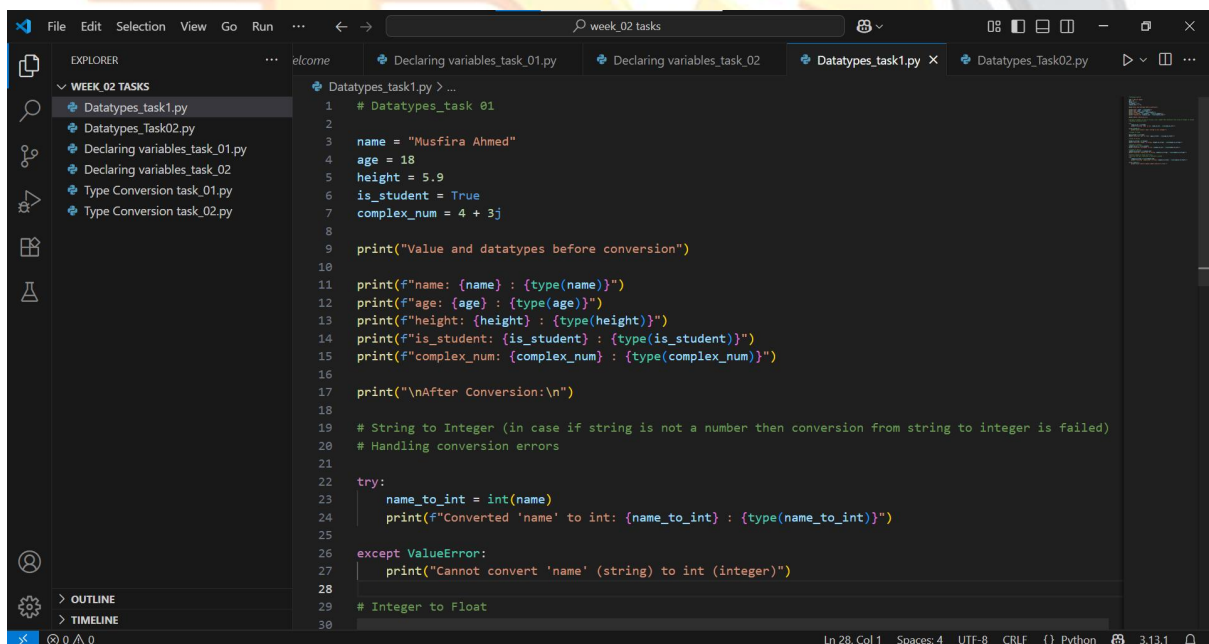
Then, converts each variable to a different type (where possible) and prints the new types

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

What I Did (Step by Step:

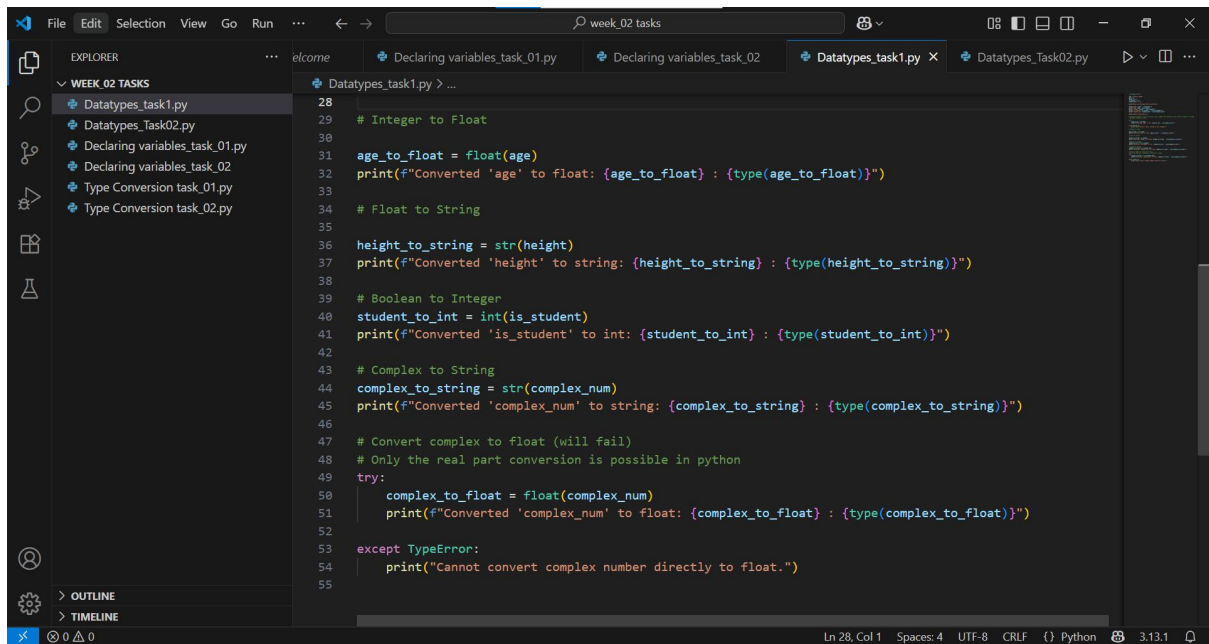
- 1) Declared variables of 5 different types.
- 2) Printed their original types.
- 3) Attempted conversions with try-except.

Code Screenshots



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'WEEK_02 TASKS' with several files. The code editor displays a Python script named 'Datatypes_task1.py'. The script defines five variables: 'name' (string), 'age' (integer), 'height' (float), 'is_student' (boolean), and 'complex_num' (complex number). It then prints the values and types of these variables. After a separator, it attempts to convert the 'name' variable to an integer using a try-except block. The try block fails, and the except block catches a 'ValueError' and prints a message: 'Cannot convert 'name' (string) to int (integer)'. The script also includes a comment about handling conversion errors and a section for converting an integer to a float.

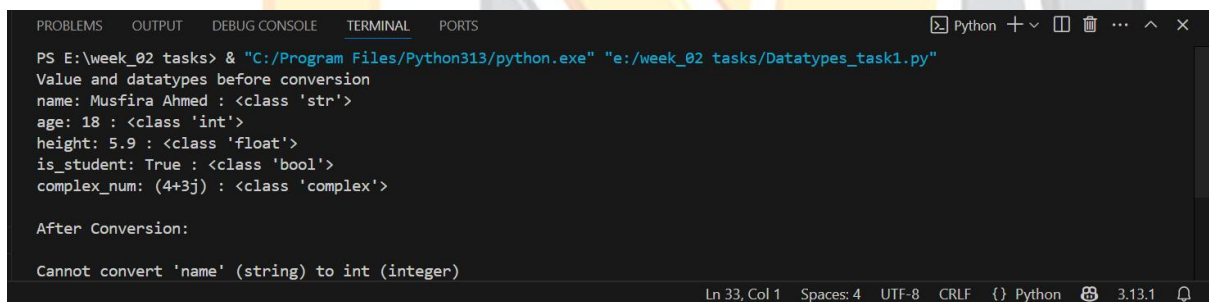
```
1 # Datatypes_task 01
2
3 name = "Musfira Ahmed"
4 age = 18
5 height = 5.9
6 is_student = True
7 complex_num = 4 + 3j
8
9 print("Value and datatypes before conversion")
10
11 print(f"name: {name} : {type(name)}")
12 print(f"age: {age} : {type(age)}")
13 print(f"height: {height} : {type(height)}")
14 print(f"is_student: {is_student} : {type(is_student)}")
15 print(f"complex_num: {complex_num} : {type(complex_num)}")
16
17 print("\nAfter Conversion:\n")
18
19 # String to Integer (in case if string is not a number then conversion from string to integer is failed)
20 # Handling conversion errors
21
22 try:
23     name_to_int = int(name)
24     print(f"Converted 'name' to int: {name_to_int} : {type(name_to_int)}")
25
26 except ValueError:
27     print("Cannot convert 'name' (string) to int (integer)")
28
29 # Integer to Float
30
```

The screenshot shows a VS Code editor with a file explorer on the left containing files for 'WEEK 02 TASKS'. The main editor displays the code for 'Datatypes_task1.py'. The code defines variables for name, age, height, is_student, and complex_num, then attempts to convert them to different data types. It includes comments for each conversion and a try-except block for the complex number conversion.

```
28
29 # Integer to Float
30
31 age_to_float = float(age)
32 print(f"Converted 'age' to float: {age_to_float} : {type(age_to_float)}")
33
34 # Float to String
35
36 height_to_string = str(height)
37 print(f"Converted 'height' to string: {height_to_string} : {type(height_to_string)}")
38
39 # Boolean to Integer
40 student_to_int = int(is_student)
41 print(f"Converted 'is_student' to int: {student_to_int} : {type(student_to_int)}")
42
43 # Complex to String
44 complex_to_string = str(complex_num)
45 print(f"Converted 'complex_num' to string: {complex_to_string} : {type(complex_to_string)}")
46
47 # Convert complex to float (will fail)
48 # Only the real part conversion is possible in python
49 try:
50     complex_to_float = float(complex_num)
51     print(f"Converted 'complex_num' to float: {complex_to_float} : {type(complex_to_float)}")
52 except TypeError:
53     print("Cannot convert complex number directly to float.")
54
55
```

Output Screenshots

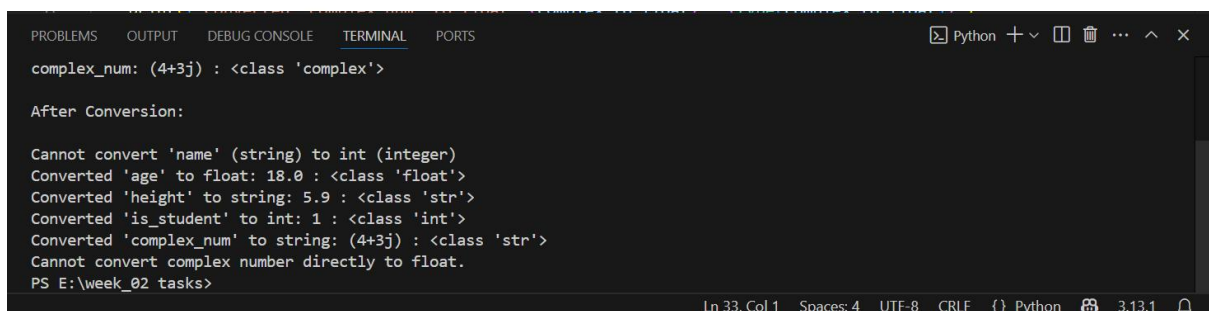


The terminal window shows the command to run the script and its output. The output displays the initial values and types of the variables, followed by the results of the conversions. An error message is shown for the 'name' variable conversion.

```
PS E:\week_02 tasks> & "C:/Program Files/Python313/python.exe" "e:/week_02 tasks/Datatypes_task1.py"
Value and datatypes before conversion
name: Musfira Ahmed : <class 'str'>
age: 18 : <class 'int'>
height: 5.9 : <class 'float'>
is_student: True : <class 'bool'>
complex_num: (4+3j) : <class 'complex'>

After Conversion:

Cannot convert 'name' (string) to int (integer)
```



This screenshot continues the terminal output from the previous one, showing the successful conversions of 'age', 'height', and 'is_student', and the final error message for the 'name' variable.

```
complex_num: (4+3j) : <class 'complex'>

After Conversion:

Cannot convert 'name' (string) to int (integer)
Converted 'age' to float: 18.0 : <class 'float'>
Converted 'height' to string: 5.9 : <class 'str'>
Converted 'is_student' to int: 1 : <class 'int'>
Converted 'complex_num' to string: (4+3j) : <class 'str'>
Cannot convert complex number directly to float.
PS E:\week_02 tasks>
```

Learnings and Challenges:

- 1) Learned about data types and what conversions are allowed.
- 2) Learned that complex numbers can't be directly converted to float.
- 3) Improved error handling skills.

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()`).

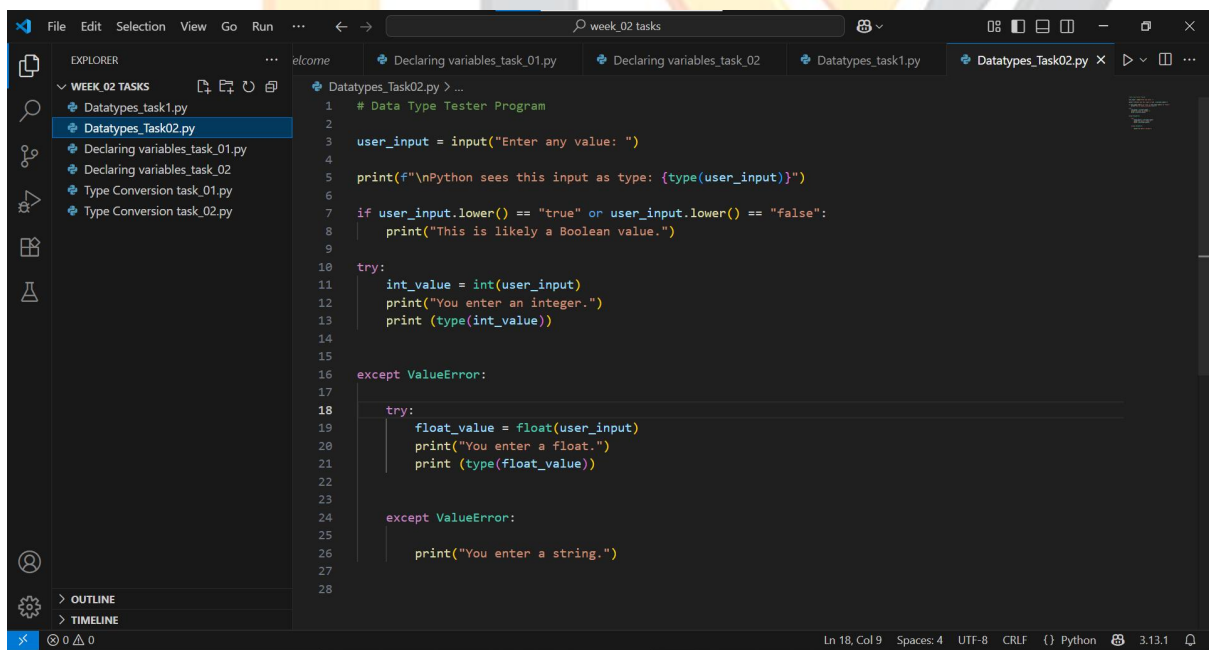
Add conditions to identify if it's likely an integer, float, or string, and print a message like:

"You entered a float!"

What I Did (Step by Step):

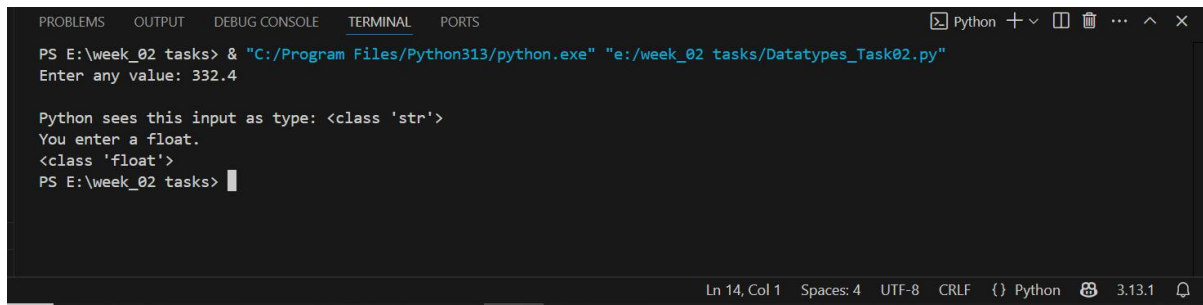
1. Took user input and checked the type.
2. Detected booleans, integers, floats, or strings.

Code Screenshots

A screenshot of a Python IDE (VS Code) showing a file named 'Datatypes_Task02.py'. The code is a 'Data Type Tester Program' that prompts the user for input and checks its type. It uses a combination of direct type checking for booleans and try-except blocks for integers, floats, and strings. The IDE interface includes an Explorer sidebar on the left with a project structure, a main editor window with the code, and a status bar at the bottom showing 'Ln 18, Col 9' and other details.

```
1 # Data Type Tester Program
2
3 user_input = input("Enter any value: ")
4
5 print(f"\nPython sees this input as type: {type(user_input)}")
6
7 if user_input.lower() == "true" or user_input.lower() == "false":
8     print("This is likely a Boolean value.")
9
10
11 try:
12     int_value = int(user_input)
13     print("You enter an integer.")
14     print (type(int_value))
15
16 except ValueError:
17
18     try:
19         float_value = float(user_input)
20         print("You enter a float.")
21         print (type(float_value))
22
23     except ValueError:
24
25         print("You enter a string.")
26
27
28
```

Output Screenshots



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [ ] [ ] ... ^ x

PS E:\week_02 tasks> & "C:/Program Files/Python313/python.exe" "e:/week_02 tasks/Datatypes_Task02.py"
Enter any value: 332.4

Python sees this input as type: <class 'str'>
You enter a float.
<class 'float'>
PS E:\week_02 tasks> |
```

Learnings and Challenges:

- 1) Learnings and Challenges:
- 2) Learned how to determine the actual data type.
- 3) Handled errors smoothly using try-except.

Task 07:

Create a marks percentage calculator:

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

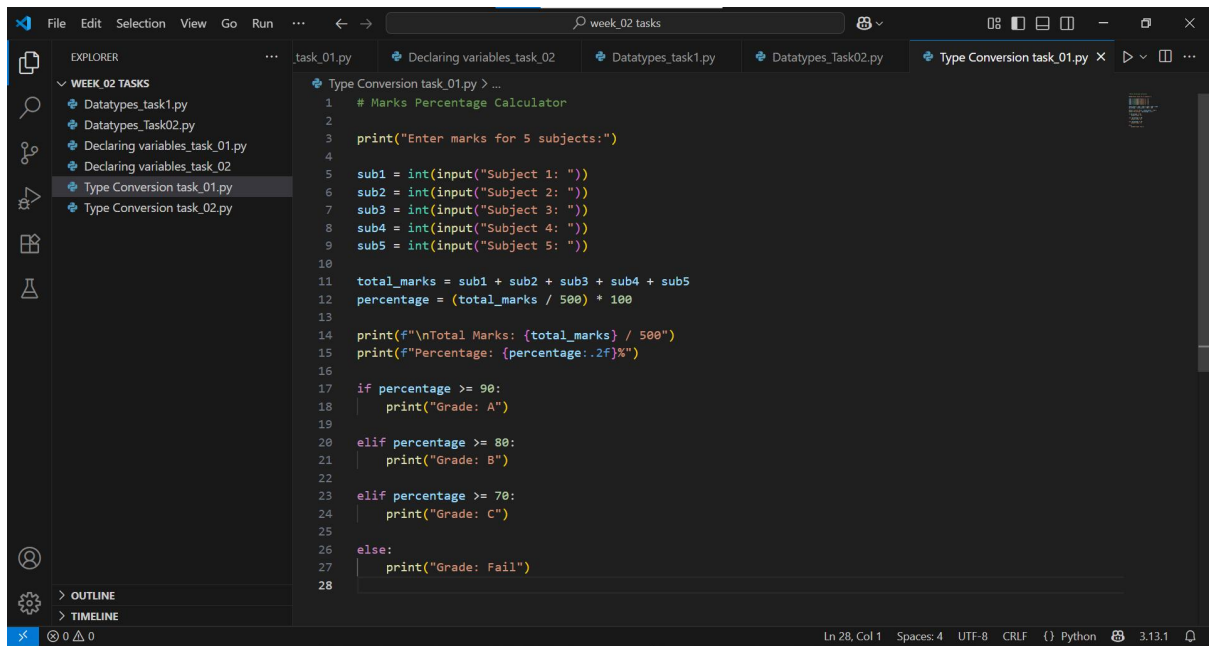
Calculate the total and percentage

Print percentage along with a grade: A (90+), B (80-89), C (70-79), Fail (<70).

What I Did (Step by Step):

1. Collected marks, calculated total and percentage.
2. Used conditions to assign grades.

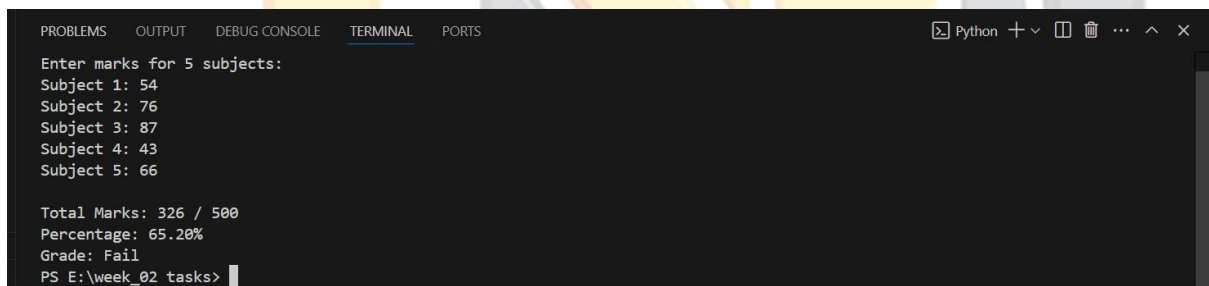
Code Screenshots



The screenshot shows the Visual Studio Code editor with a Python file named 'Type Conversion task_01.py'. The code is a 'Marks Percentage Calculator' that prompts the user to enter marks for 5 subjects. It calculates the total marks, the percentage, and then determines the grade based on the percentage using if-elif-else logic. The status bar at the bottom indicates the cursor is at line 28, column 1, with 4 spaces, using UTF-8 encoding and CRLF line endings.

```
1 # Marks Percentage Calculator
2
3 print("Enter marks for 5 subjects:")
4
5 sub1 = int(input("Subject 1: "))
6 sub2 = int(input("Subject 2: "))
7 sub3 = int(input("Subject 3: "))
8 sub4 = int(input("Subject 4: "))
9 sub5 = int(input("Subject 5: "))
10
11 total_marks = sub1 + sub2 + sub3 + sub4 + sub5
12 percentage = (total_marks / 500) * 100
13
14 print(f"\nTotal Marks: {total_marks} / 500")
15 print(f"Percentage: {percentage:.2f}%")
16
17 if percentage >= 90:
18     print("Grade: A")
19
20 elif percentage >= 80:
21     print("Grade: B")
22
23 elif percentage >= 70:
24     print("Grade: C")
25
26 else:
27     print("Grade: Fail")
28
```

Output Screenshots



The screenshot shows the terminal output of the program. It displays the prompts for 5 subjects, the calculated total marks (326 / 500), the percentage (65.20%), and the resulting grade (Fail). The prompt 'PS E:\week_02 tasks>' is visible at the bottom.

```
Enter marks for 5 subjects:
Subject 1: 54
Subject 2: 76
Subject 3: 87
Subject 4: 43
Subject 5: 66

Total Marks: 326 / 500
Percentage: 65.20%
Grade: Fail
PS E:\week_02 tasks>
```

Learnings and Challenges:

- 1) Learned percentage calculations.
- 2) Practiced using if-elif-else logic.
- 3) Ensured that user input was correctly converted from string to integer.

Task 08:

Create a temperature converter:

Ask the user to input temperature in Celsius.

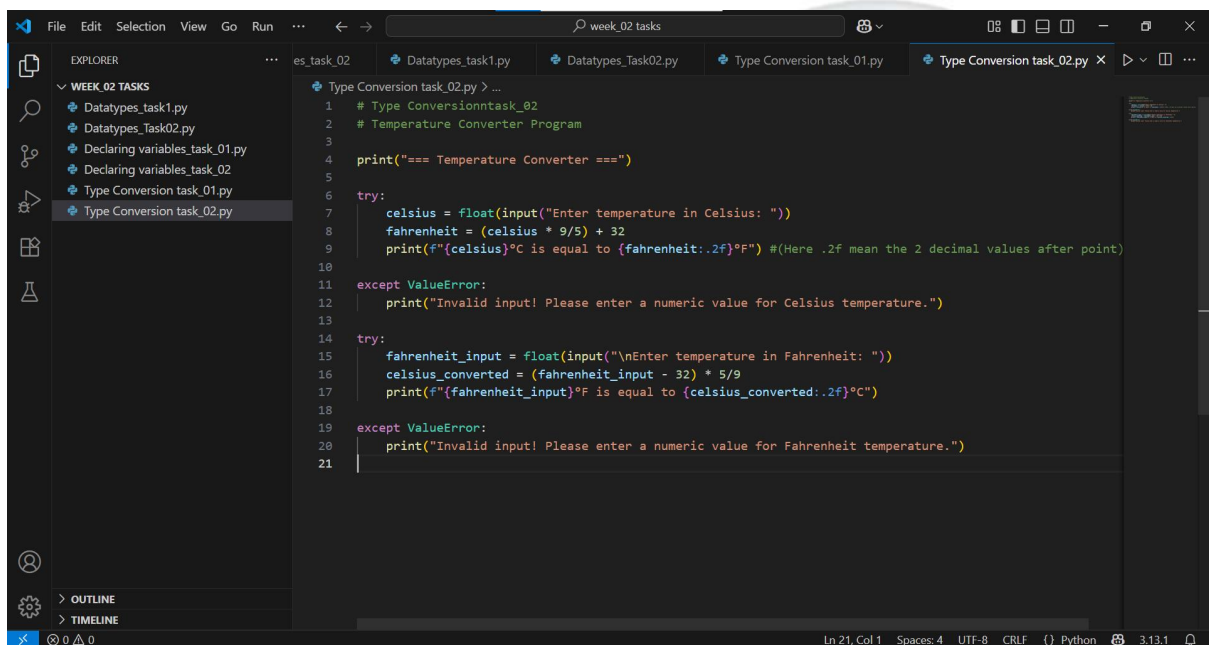
Convert it to Fahrenheit using: $F = (C * 9/5) + 32$, Then reverse: Ask for Fahrenheit, convert it to Celsius.

Handle wrong input types using try-except.

What I Did (Step by Step):

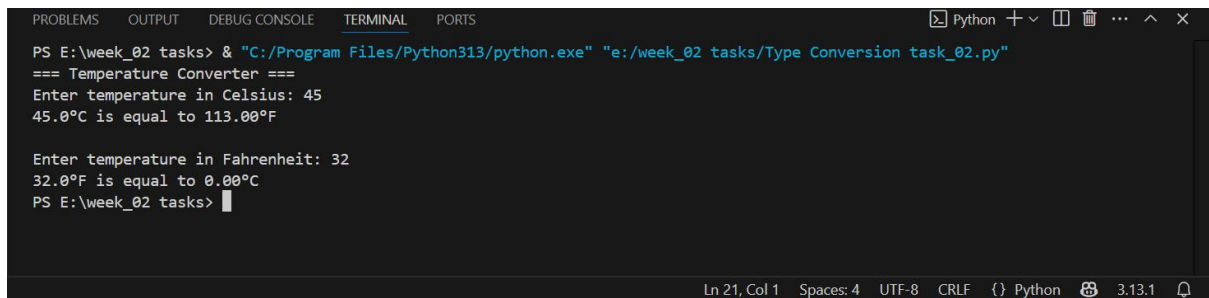
1. Took input in Celsius and Fahrenheit.
2. Applied temperature conversion formulas.
3. Handled invalid inputs with try-except.

Code Screenshots



```
1 # Type Conversion task_02
2 # Temperature Converter Program
3
4 print("=== Temperature Converter ===")
5
6 try:
7     celsius = float(input("Enter temperature in Celsius: "))
8     fahrenheit = (celsius * 9/5) + 32
9     print(f"{celsius}°C is equal to {fahrenheit:.2f}°F") #(Here .2f mean the 2 decimal values after point)
10
11 except ValueError:
12     print("Invalid input! Please enter a numeric value for Celsius temperature.")
13
14 try:
15     fahrenheit_input = float(input("\nEnter temperature in Fahrenheit: "))
16     celsius_converted = (fahrenheit_input - 32) * 5/9
17     print(f"{fahrenheit_input}°F is equal to {celsius_converted:.2f}°C")
18
19 except ValueError:
20     print("Invalid input! Please enter a numeric value for Fahrenheit temperature.")
21
```

Output Screenshots



```
PS E:\week_02 tasks> & "C:/Program Files/Python313/python.exe" "e:/week_02 tasks/Type Conversion task_02.py"
=== Temperature Converter ===
Enter temperature in Celsius: 45
45.0°C is equal to 113.00°F

Enter temperature in Fahrenheit: 32
32.0°F is equal to 0.00°C
PS E:\week_02 tasks>
```

Learnings and Challenges:

- 1) Practiced using mathematical formulas in real applications.
- 2) Learned how to use `.2f` for decimal formatting.
- 3) Improved program reliability using exception handling.



