



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 : 03

Instructor Name : Mr Hassan Ali

Task 1 :

Create a calculator that accepts two numbers and an operator (+, -, *, /, %, //, **).

Perform the operation and display the result.

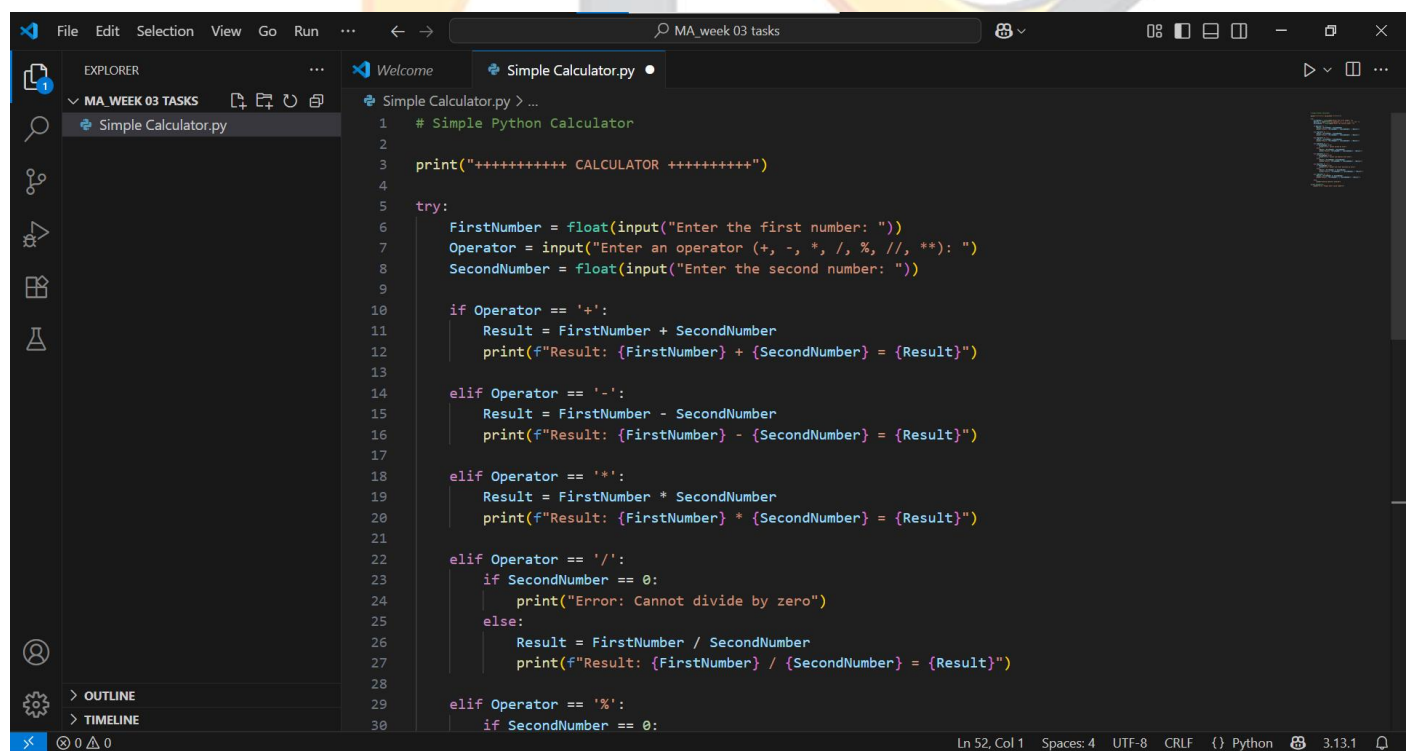
Handle division by zero safely.

Solution :

What I Did (Step by Step):

- Took two numbers and an operator as input from the user.
- Used if-elif and else statements to perform the selected operation.
- Checked for division, modulus, and floor division by zero.
- Displayed the result or appropriate error message.

Code Screenshots



The screenshot shows a Python IDE with a file named 'Simple Calculator.py'. The code is as follows:

```
1 # Simple Python Calculator
2
3 print("+++++++ CALCULATOR ++++++")
4
5 try:
6     FirstNumber = float(input("Enter the first number: "))
7     Operator = input("Enter an operator (+, -, *, /, %, //, **): ")
8     SecondNumber = float(input("Enter the second number: "))
9
10    if Operator == '+':
11        Result = FirstNumber + SecondNumber
12        print(f"Result: {FirstNumber} + {SecondNumber} = {Result}")
13
14    elif Operator == '-':
15        Result = FirstNumber - SecondNumber
16        print(f"Result: {FirstNumber} - {SecondNumber} = {Result}")
17
18    elif Operator == '*':
19        Result = FirstNumber * SecondNumber
20        print(f"Result: {FirstNumber} * {SecondNumber} = {Result}")
21
22    elif Operator == '/':
23        if SecondNumber == 0:
24            print("Error: Cannot divide by zero")
25        else:
26            Result = FirstNumber / SecondNumber
27            print(f"Result: {FirstNumber} / {SecondNumber} = {Result}")
28
29    elif Operator == '%':
30        if SecondNumber == 0:
```

```
28
29
30 elif Operator == '%':
31     if SecondNumber == 0:
32         print("Error: Cannot use modulus with zero")
33     else:
34         Result = FirstNumber % SecondNumber
35         print(f"Result: {FirstNumber} % {SecondNumber} = {Result}")
36
37 elif Operator == '//':
38     if SecondNumber == 0:
39         print("Error: Cannot use floor division by zero")
40     else:
41         Result = FirstNumber // SecondNumber
42         print(f"Result: {FirstNumber} // {SecondNumber} = {Result}")
43
44 elif Operator == '**':
45     Result = FirstNumber ** SecondNumber
46     print(f"Result: {FirstNumber} ** {SecondNumber} = {Result}")
47
48 else:
49     print("Invalid operator entered")
50
51 except ValueError:
52     print("Error: Please enter valid numbers")
```

Output Screenshot

```
PS D:\MA_week 03 tasks> & "C:/Program Files/Python313/python.exe" "d:/MA_week 03 tasks/Simple Calculator.py"
+++++++ CALCULATOR ++++++++
Enter the first number: 4
Enter an operator (+, -, *, /, %, //, **): *
Enter the second number: 5
Result: 4.0 * 5.0 = 20.0
PS D:\MA_week 03 tasks>
```

Learning and Challenges:

- Learned basic arithmetic operations and conditional logic.
- Faced issues with division by zero and handled them using conditions.
- Understood how to convert inputs and print results clearly.
- Practiced simple error handling with try-except for invalid input.

Task 02:

Take marks of 3 subjects.

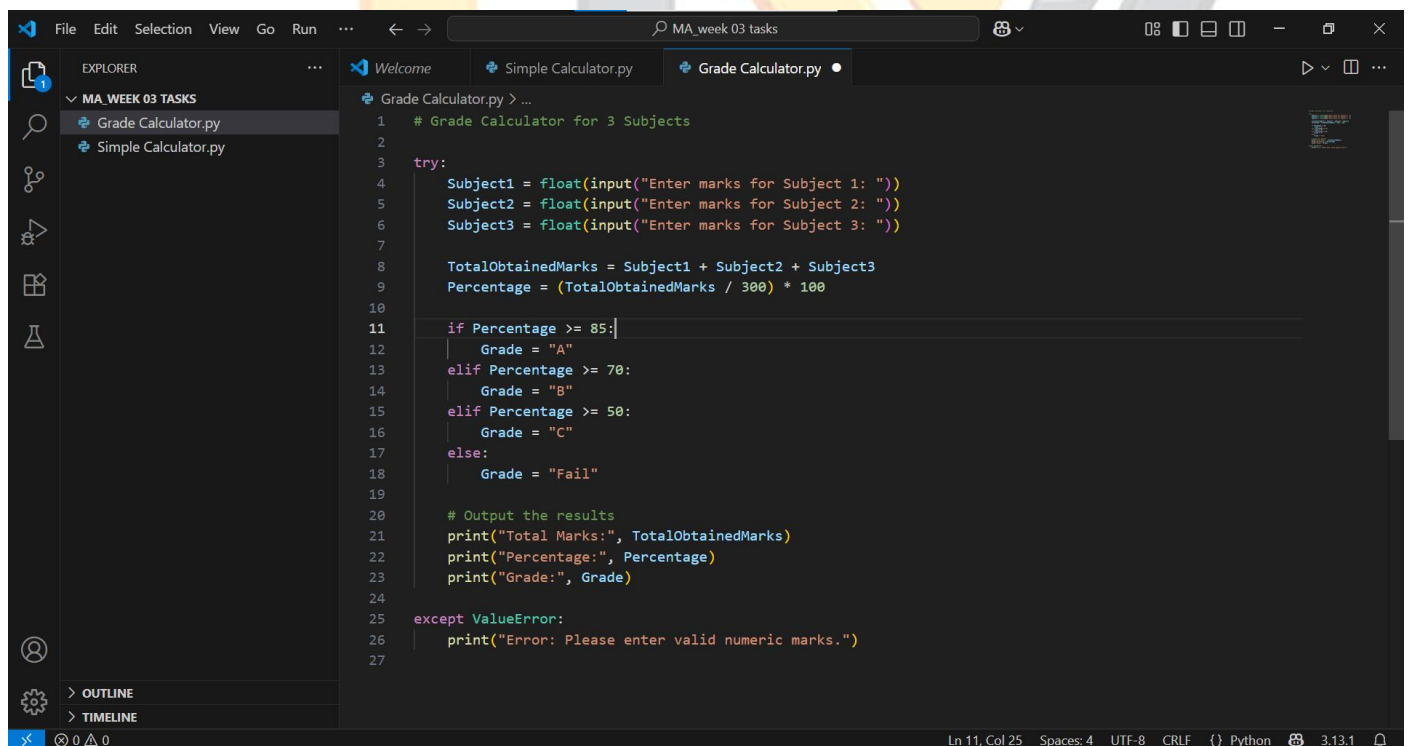
Calculate total, percentage and assign grade:

A (≥ 85), B (≥ 70), C (≥ 50), Fail (< 50).

What I Did (Step by Step):

- Took marks for 3 subjects as input from the user.
- Calculated the total and percentage assuming each subject is out of 100.
- Used if-elif statements to assign grades based on percentage.
- Displayed total, percentage, and grade with proper error handling.

Code Screenshots

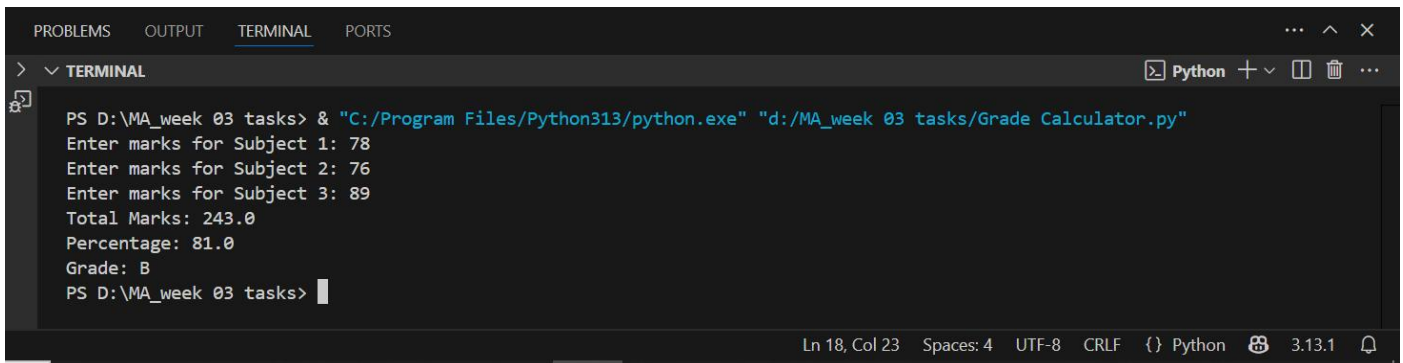


The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer panel on the left shows a project named 'MA_WEEK 03 TASKS' containing two files: 'Grade Calculator.py' and 'Simple Calculator.py'. The main editor area displays the code for 'Grade Calculator.py'. The code is a Python script that takes marks for three subjects as input, calculates the total and percentage, and assigns a grade based on the percentage. It includes error handling for non-numeric input.

```
1 # Grade Calculator for 3 Subjects
2
3 try:
4     Subject1 = float(input("Enter marks for Subject 1: "))
5     Subject2 = float(input("Enter marks for Subject 2: "))
6     Subject3 = float(input("Enter marks for Subject 3: "))
7
8     TotalObtainedMarks = Subject1 + Subject2 + Subject3
9     Percentage = (TotalObtainedMarks / 300) * 100
10
11     if Percentage >= 85:
12         Grade = "A"
13     elif Percentage >= 70:
14         Grade = "B"
15     elif Percentage >= 50:
16         Grade = "C"
17     else:
18         Grade = "Fail"
19
20     # Output the results
21     print("Total Marks:", TotalObtainedMarks)
22     print("Percentage:", Percentage)
23     print("Grade:", Grade)
24
25 except ValueError:
26     print("Error: Please enter valid numeric marks.")
27
```

The status bar at the bottom indicates the current line and column (Ln 11, Col 25), the number of spaces (4), the encoding (UTF-8), the line ending (CRLF), the language (Python), and the version (3.13.1).

Output Screenshot



```
PROBLEMS OUTPUT TERMINAL PORTS
> TERMINAL
Python + - [ ] [ ] [ ]
PS D:\MA_week 03 tasks> & "C:/Program Files/Python313/python.exe" "d:/MA_week 03 tasks/Grade Calculator.py"
Enter marks for Subject 1: 78
Enter marks for Subject 2: 76
Enter marks for Subject 3: 89
Total Marks: 243.0
Percentage: 81.0
Grade: B
PS D:\MA_week 03 tasks> |
Ln 18, Col 23 Spaces: 4 UTF-8 CRLF {} Python 3.13.1
```

Learning and Challenges:

- . Learned how to work with multiple inputs and calculations.
- . Practiced using conditions to categorize values (grade logic).
- . Faced minor issues with invalid input and fixed them using try-except.
- . Understood how to structure output for better readability.

Task 03:

Ask user for monthly income and expenses.

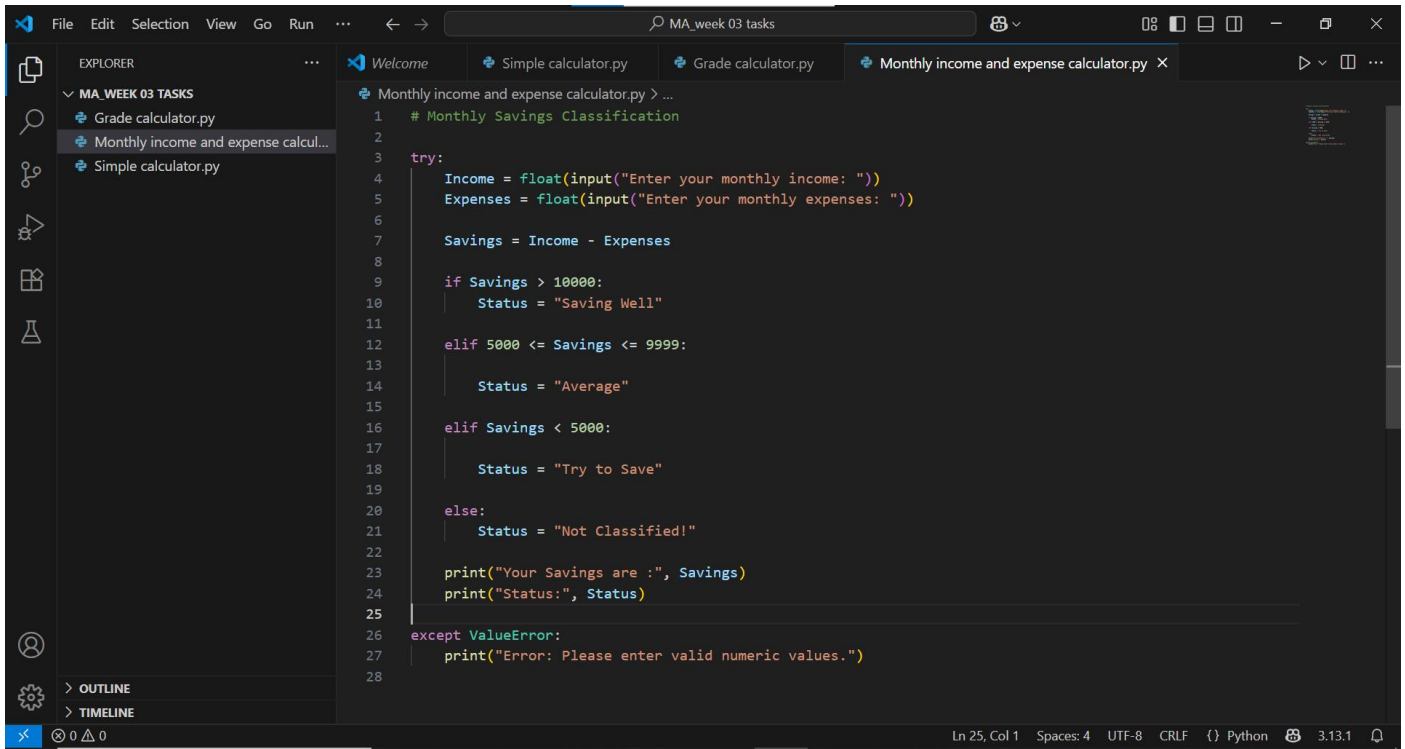
Calculate savings and classify:

>10000 = Saving Well, 5000–9999 = Average, <5000 = Try to Save.

What I Did (Step by Step):

- Took income and expense values as input from the user.
- Subtracted expenses from income to calculate savings.
- Classified savings using if-elif conditions.
- Displayed the savings amount and status.

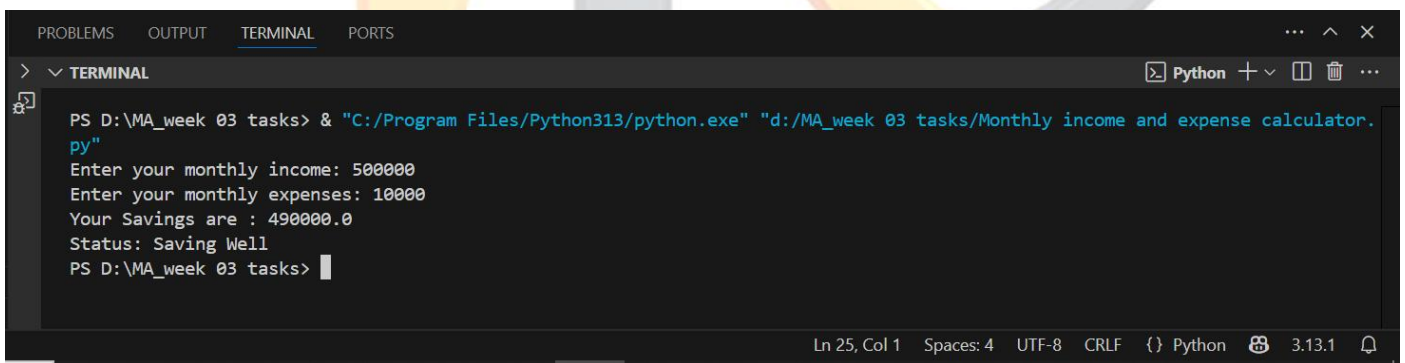
Code Screenshots



The screenshot shows the Visual Studio Code editor with the file 'Monthly income and expense calculator.py' open. The code is a Python script that calculates savings and classifies the user's financial status based on their monthly income and expenses. The Explorer sidebar on the left shows the project structure with 'MA_WEEK 03 TASKS' containing 'Grade calculator.py', 'Monthly income and expense calcul...', and 'Simple calculator.py'. The code in the editor is as follows:

```
1 # Monthly Savings Classification
2
3 try:
4     Income = float(input("Enter your monthly income: "))
5     Expenses = float(input("Enter your monthly expenses: "))
6
7     Savings = Income - Expenses
8
9     if Savings > 10000:
10         Status = "Saving Well"
11
12     elif 5000 <= Savings <= 9999:
13
14         Status = "Average"
15
16     elif Savings < 5000:
17
18         Status = "Try to Save"
19
20     else:
21         Status = "Not Classified!"
22
23     print("Your Savings are :", Savings)
24     print("Status:", Status)
25
26 except ValueError:
27     print("Error: Please enter valid numeric values.")
28
```

Output Screenshot



The screenshot shows the terminal output of the Python script. The user has entered a monthly income of 50000 and monthly expenses of 10000. The script calculates the savings as 49000.0 and classifies the status as 'Saving Well'.

```
PS D:\MA_week 03 tasks> & "C:/Program Files/Python313/python.exe" "d:/MA_week 03 tasks/Monthly income and expense calculator.py"
Enter your monthly income: 50000
Enter your monthly expenses: 1000
Your Savings are : 49000.0
Status: Saving Well
PS D:\MA_week 03 tasks>
```

Learnings and Challenges:

- Learned how to use basic arithmetic and conditional logic in real-life scenarios.
- Faced input validation issues and handled them with try-except.
- Practiced displaying results in a clear and readable format.
- Understood how thresholds can be used to categorize user input

Task 04:

Build a login system. Ask username & password.

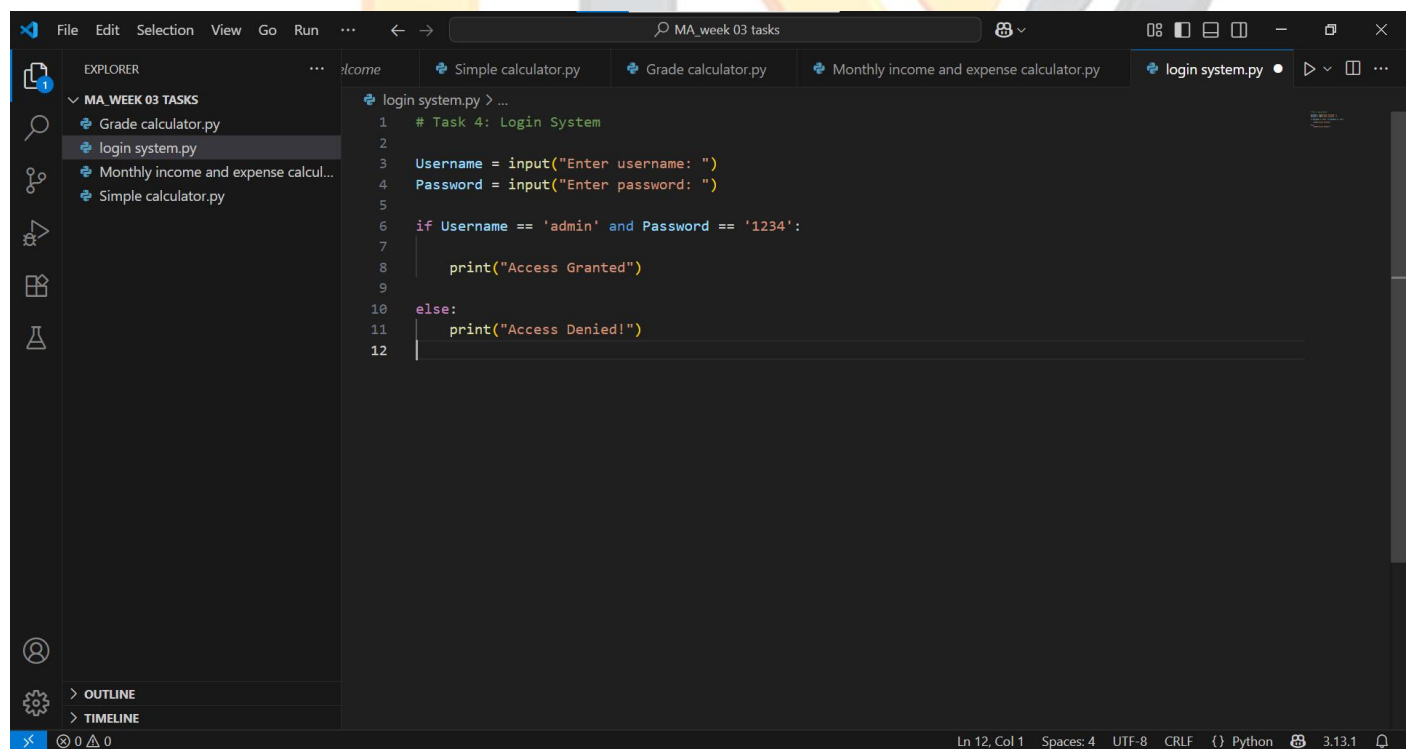
If **username** = 'admin' and **password** = '1234', print Access Granted.

Else, Access Denied.

What I Did (Step by Step):

- Took username and password input from the user.
- Checked if the entered values matched the required credentials.
- Used an if statement to grant or deny access.
- Displayed the result based on the input match.

Code Screenshots

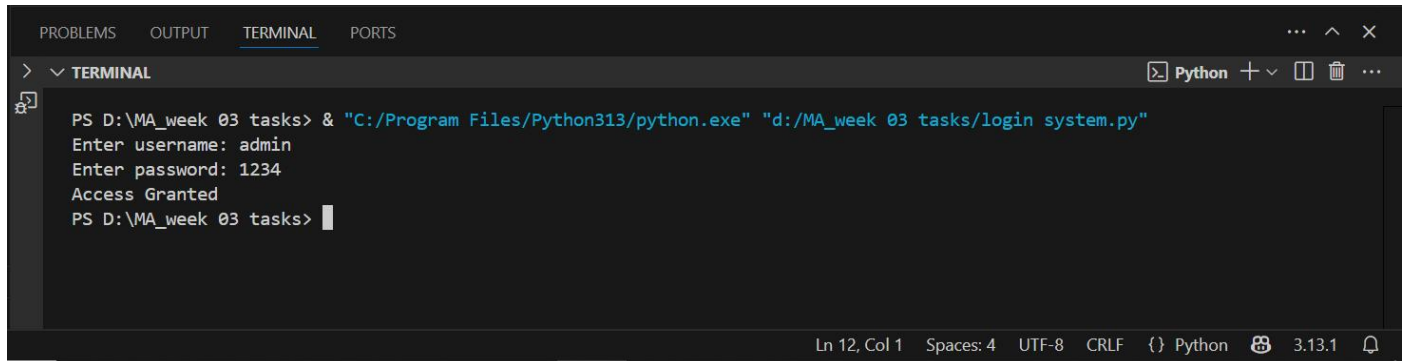


The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a project named 'MA_WEEK 03 TASKS' with files: 'Grade calculator.py', 'login system.py', 'Monthly income and expense calcul...', and 'Simple calculator.py'. The 'login system.py' file is selected and open in the editor. The code in the editor is as follows:

```
1 # Task 4: Login System
2
3 Username = input("Enter username: ")
4 Password = input("Enter password: ")
5
6 if Username == 'admin' and Password == '1234':
7     print("Access Granted")
8
9
10 else:
11     print("Access Denied!")
12
```

The status bar at the bottom indicates 'Ln 12, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', '{} Python', and '3.13.1'.

Output Screenshots



The screenshot shows a terminal window with the following content:

```
PS D:\MA_week 03 tasks> & "C:/Program Files/Python313/python.exe" "d:/MA_week 03 tasks/login system.py"
Enter username: admin
Enter password: 1234
Access Granted
PS D:\MA_week 03 tasks> |
```

The terminal window has tabs for PROBLEMS, OUTPUT, TERMINAL, and PORTS. The TERMINAL tab is active. The status bar at the bottom indicates 'Ln 12, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and version '3.13.1'.

Learning and Challenges

- Learned how to compare multiple conditions using and.
- Practiced basic input validation and string comparison.
- Faced no major issues due to the simple structure.
- Understood basic logic used in authentication systems.

Task 05:

Ask user for attendance (%) and final marks.

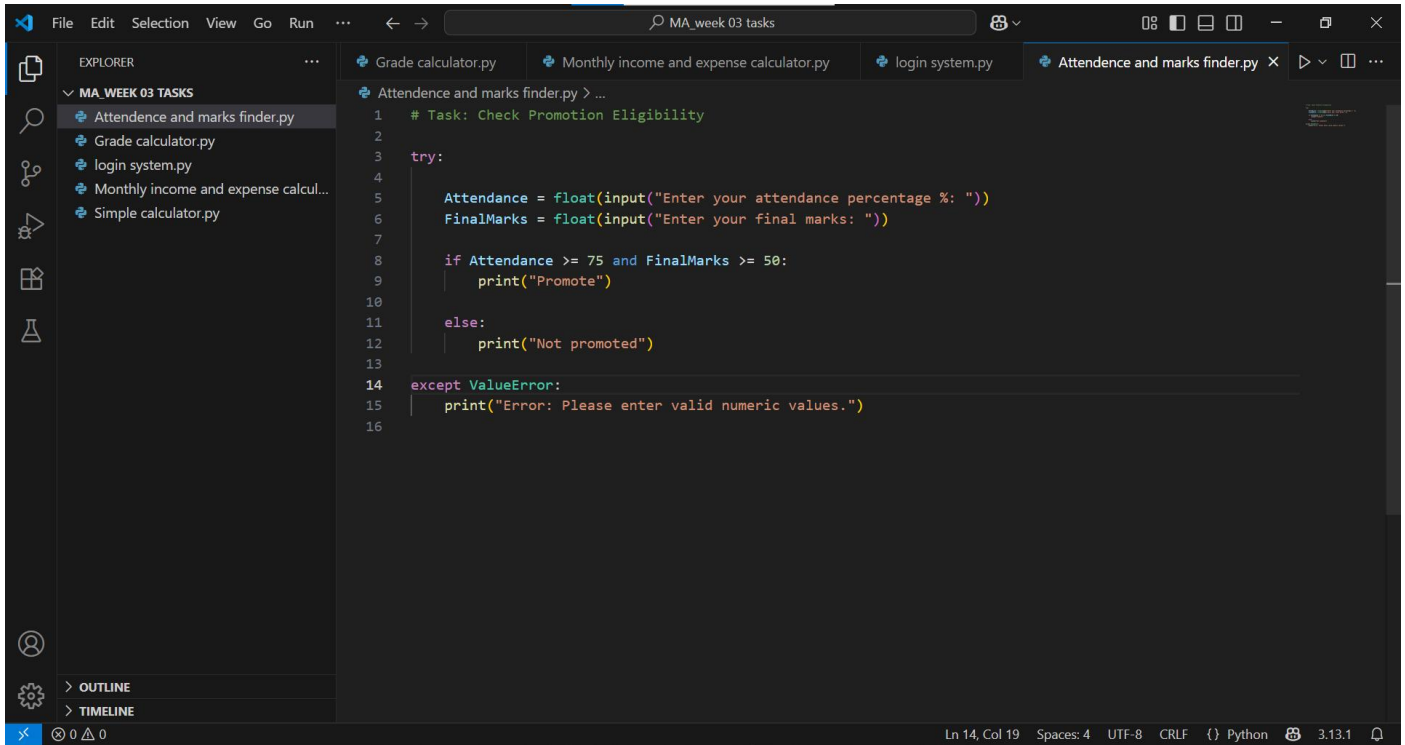
If attendance ≥ 75 and marks $\geq 50 \rightarrow$ Promote

Else \rightarrow Not promoted.

What I Did (Step by Step):

- Took attendance percentage and final marks as input.
- Used if condition with and to check promotion eligibility.
- Displayed result based on the input values.
- Handled invalid input using try-except.

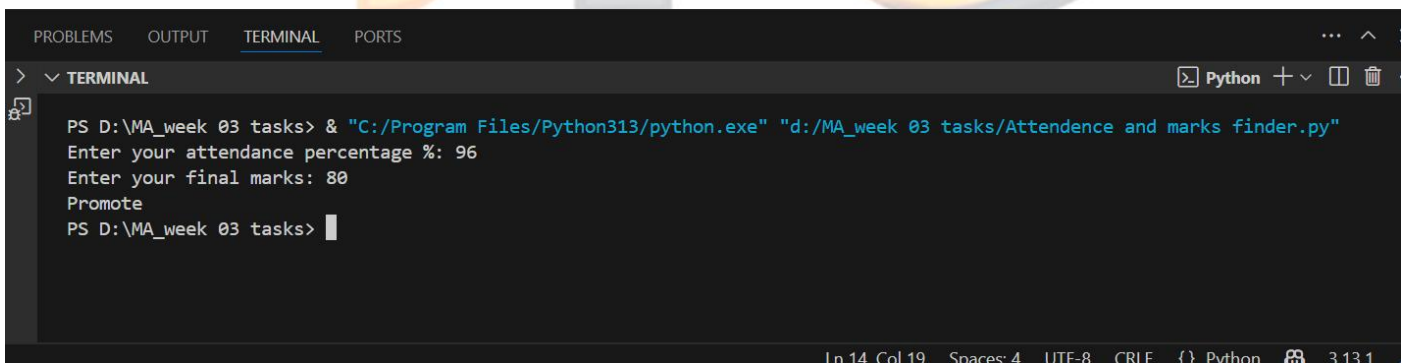
Code Screenshots



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a folder named 'MA_WEEK 03 TASKS' containing several Python files. The main editor window displays the code for 'Attendance and marks finder.py'. The code is a Python script that prompts the user for attendance percentage and final marks, then checks if the user is eligible for promotion based on these values. It includes a try-except block to handle potential ValueError exceptions.

```
1 # Task: Check Promotion Eligibility
2
3 try:
4
5     Attendance = float(input("Enter your attendance percentage %: "))
6     FinalMarks = float(input("Enter your final marks: "))
7
8     if Attendance >= 75 and FinalMarks >= 50:
9         print("Promote")
10
11     else:
12         print("Not promoted")
13
14 except ValueError:
15     print("Error: Please enter valid numeric values.")
16
```

Output Screenshots



The screenshot shows a terminal window with a dark theme. The terminal displays the command to run the Python script and the subsequent input and output. The user enters '96' for attendance percentage and '80' for final marks. The script outputs 'Promote'.

```
PS D:\MA_week 03 tasks> & "C:/Program Files/Python313/python.exe" "d:/MA_week 03 tasks/Attendance and marks finder.py"
Enter your attendance percentage %: 96
Enter your final marks: 80
Promote
PS D:\MA_week 03 tasks>
```

Learning and Challenges

- Learned how to apply multiple conditions using logical and.
- Practiced input handling and numeric comparison.
- Ensured correct logic flow for real-life decision-making.
- Resolved input type issues using float() and error handling.

Task 06:

Billing system:

Take number of products and total price.

If price > 1000 and products > 3 → 15% discount

If price > 500 → 10% discount

Else → No discount.

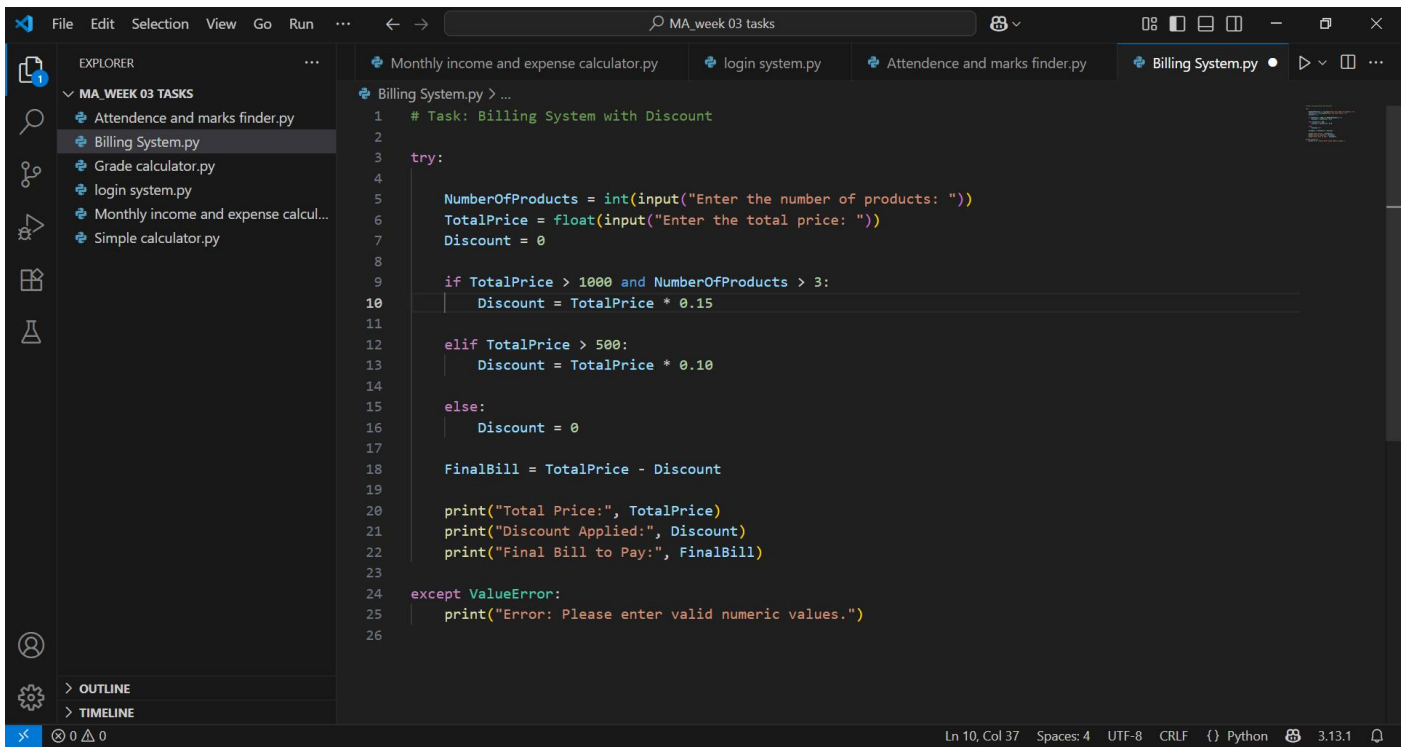
Show final bill.

What I Did (Step by Step):

- Took user input for product count and total price.
- Used conditions to apply the correct discount rule.
- Calculated and displayed the discount and final bill.
- Handled invalid input types safely using try-except.

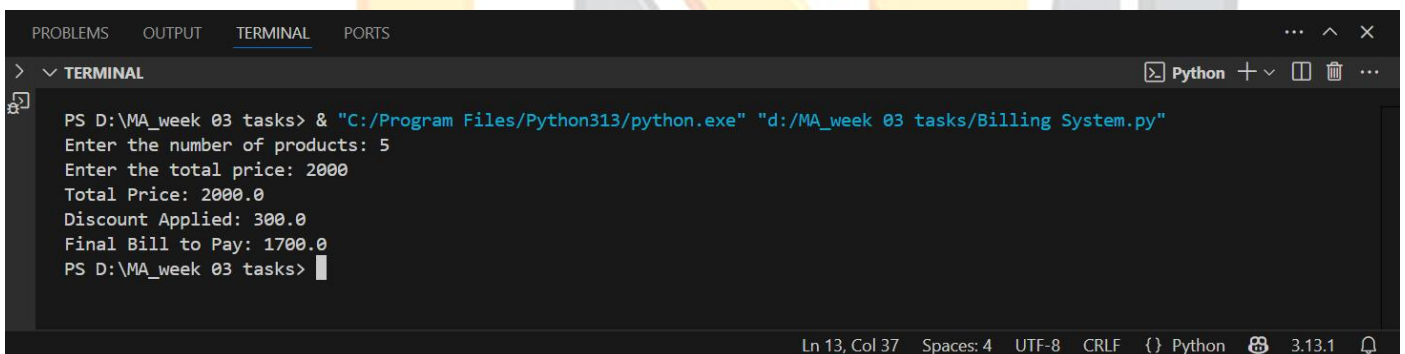
Code Screenshots

TECHNIK NEST



```
1 # Task: Billing System with Discount
2
3 try:
4
5     NumberOfProducts = int(input("Enter the number of products: "))
6     TotalPrice = float(input("Enter the total price: "))
7     Discount = 0
8
9     if TotalPrice > 1000 and NumberOfProducts > 3:
10         Discount = TotalPrice * 0.15
11
12     elif TotalPrice > 500:
13         Discount = TotalPrice * 0.10
14
15     else:
16         Discount = 0
17
18     FinalBill = TotalPrice - Discount
19
20     print("Total Price:", TotalPrice)
21     print("Discount Applied:", Discount)
22     print("Final Bill to Pay:", FinalBill)
23
24 except ValueError:
25     print("Error: Please enter valid numeric values.")
26
```

Output Screenshots



```
PS D:\MA_week 03 tasks> & "C:/Program Files/Python313/python.exe" "d:/MA_week 03 tasks/Billing System.py"
Enter the number of products: 5
Enter the total price: 2000
Total Price: 2000.0
Discount Applied: 300.0
Final Bill to Pay: 1700.0
PS D:\MA_week 03 tasks>
```

Learning and Challenges

- . Learned how to calculate and apply percentage-based discounts.
- . Practiced combining multiple conditions using and and elif.
- . Faced input conversion issues and fixed using proper data types.
- . Applied real-world billing logic to make the program functional.