AmazingDevOps

Notes for Exam 1

4 Programs to Learn and Run.

[Multiplication, Subtraction, Addition and Division]

Program #1

```
#!/usr/bin/python3

def add_numbers():
    numbers = []
    while True:
        num = input("Enter a number (or 'q' to quit): ")
        if num.lower() == 'q':
            break
        try:
            num = float(num)
            numbers.append(num)
        except ValueError:
            print("Invalid input. Please enter a number or 'q' to quit.")

    total = sum(numbers)
    print("The sum of the numbers is:", total)

# Call the add_numbers function
My_homework_tonight=add_numbers()

#Print the result on the screen

print(My_homework_tonight)
```

Detailed explanation of how the code works

This code defines a function called "add_numbers()" that allows the user to input a series of numbers and calculates their sum. Let's break down the code step by step:

1. The function begins by initializing an empty list called "numbers". This list will store the numbers inputted by the user.

2. The code enters a while loop that will continue running until the user decides to quit. The loop prompts the user to enter a number or 'q' to quit by using the input() function.
3. Inside the loop, the code checks if the user input is equal to 'q' (ignoring the case by converting it to lowercase). If so, the break statement is executed, which terminates the loop and proceeds to the next line of code.
4. If the user input is not 'q', the code attempts to convert it into a floating-point number using the float() function. If the conversion is successful, the number is appended to the "numbers" list.
5. However, if the conversion raises a ValueError, it means that the user input is not a valid number. In this case, an error message is printed, informing the user about the invalid input. The loop continues to the next iteration, prompting the user for another input.
6. Once the user decides to quit by entering 'q', the loop terminates, and the code proceeds to calculate the sum of the numbers using the sum() function. The "numbers" list contains all the valid numbers entered by the user.
7. Finally, the total sum of the numbers is printed using the print() function, along with an accompanying message.

To summarize, this code allows the user to input multiple numbers until they choose to quit. It handles invalid inputs and calculates the sum of the valid numbers entered by the user.

The **try** statement in Python is used for exception handling. It allows you to catch and handle specific types of errors that might occur during the execution of your code. By using the **try** statement, you can write code that can potentially raise an exception and provide a way to handle that exception gracefully.

In the given code snippet, the **try** statement is used to handle the possibility of a **ValueError** being raised when converting the user input to a float. Here's how it works:

1. The user input is initially captured as a string with the **input()** function.
2. Inside the **try** block, the code attempts to convert the user input to a float using the **float()** function. If the conversion is successful, the number is appended to the **numbers** list.
3. If the conversion raises a **ValueError**, it means that the user input is not a valid number. In this case, the code jumps to the **except** block, where a specific error

message is printed to inform the user about the invalid input. The **except ValueError** specifies that this block should only handle **ValueError** exceptions.

By using the **try-except** construct, you can prevent the program from crashing when an invalid input is provided. Instead, it gracefully handles the error by printing a user-friendly message. This allows the program to continue running and prompt the user for the next input.

In summary, the **try-except** block in this code snippet is used to handle potential conversion errors when converting user input to a float, providing a way to handle those errors and continue the program's execution without interruption.

Expected Output

```
PS C:\Users\Forex\OneDrive\Desktop\practice.py> & C:/Users/Forex/AppData/Local/Microsoft/Wi
dowsApps/python3.10.exe c:/Users/Forex/OneDrive/Desktop/practice.py/josh/number-addition.py
Enter a number (or 'q' to quit): 3
Enter a number (or 'q' to quit): 4
Enter a number (or 'q' to quit): 7
Enter a number (or 'q' to quit): 9
Enter a number (or 'q' to quit): q
The sum of the numbers is: 23.0
None
```

Program Number #2

AmazingDevOps

```python
#!/usr/bin/python3
def multiply_numbers():
    numbers = []
    while True:
        num = input("Enter a number (or 'q' to quit): ")
        if num.lower() == 'q':
            break
        try:
            num = float(num)
            numbers.append(num)
        except ValueError:
            print("Invalid input. Please enter a number or 'q' to quit.")

    total = 1
    for num in numbers:
        total *= num

    print("The multiplication of the numbers is:", total)

#Let us call the function

multiply_numbers()
```

Important points to note.

1. The first thing is to define the empty function using def.
2. We are going to use list []. This will allow us to enter numbers as a list.
3. While True: The loop will allow you to enter numbers until you enter ‘q’, which will terminate the loop instantly and jump to the print.
4. Try is used to handle the error that can occur during the process of converting the input, which is a string into float. A float is a number with a decimal.
5. In case an error occurs during the process of converting the input to a float, the loop will just go to the except line which is very important in preventing the program from crashing. Then the except line will print the message under the ValueError.
6. Total = 1 is just there to initialize the variable total with a value of 1.
7. The last line is used to call the function to perform the action that we defined in the function’s block.

Output

```
PS C:\Users\Forex\OneDrive\Desktop\practice.py> & C:/Users/Forex/AppData/Local/Microsoft/Win
dowsApps/python3.10.exe c:/Users/Forex/OneDrive/Desktop/practice.py/josh/multiplication.py
Enter a number (or 'q' to quit): 2
Enter a number (or 'q' to quit): 3
Enter a number (or 'q' to quit): 5
Enter a number (or 'q' to quit): 8
Enter a number (or 'q' to quit): q
The multiplication of the numbers is: 240.0
PS C:\Users\Forex\OneDrive\Desktop\practice.py>
```

Program #3 – Subtraction

Input

```python
#!/usr/bin/python3

def subtract_numbers():
    numbers = []
    while len(numbers) < 2:  # Continue until exactly two numbers are entered
        num = input("Enter a number: ")
        try:
            num = float(num)
            numbers.append(num)
        except ValueError:
            print("Invalid input. Please enter a number.")

    result = numbers[0] - numbers[1]  # Subtract the second number from the first numb

    print("The subtraction of the two numbers is:", result)

# Let us call the function

subtract_numbers()
```

Important points to note.

1. The first line involves defining a function called subtract_numbers ()
2. The variable numbers will store a list of numbers but in this case just two numbers will be stored. We are using the square bracket to donate the fact that this is a list. When working with a list, it is vital to know that positioning is one key feature that will be vital in this program. The first number will get position [0] and the second number will get position [1].
3. We are using len() function to make sure the input has a character of less than 2.

4. Try is used to handle the error that can occur during the process of converting the input, which is a string into float. A float is a number with a decimal.
5. In case an error occurs during the process of converting the input to a float, the loop will just go to the except line which is very important in preventing the program from crashing. Then the except line will print the message under the ValueError.
6. Total = 1 is just there to initialize the variable total with a value of 1.
7. The last line is used to call the function to perform the action that we defined in the function's block.

Output

```
PS C:\Users\Forex\OneDrive\Desktop\practice.py> & C:/Users/Forex/AppData/Local/Microsoft/Win
dowsApps/python3.10.exe c:/Users/Forex/OneDrive/Desktop/practice.py/josh/subtraction.py
Enter a number: 40
Enter a number: 50
The subtraction of the two numbers is: -10.0
PS C:\Users\Forex\OneDrive\Desktop\practice.py>
```

Program #4– Division

Input

```
#!/usr/bin/python3
def divide_numbers():
    numbers = []
    while True: # Continue until exactly two numbers are entered
        num = input("Enter a number: ")
        try:
            num = float(num)
            numbers.append(num)
        except ValueError:
            print("Invalid input. Please enter a number.")
        if len(numbers) >= 2:
            break

    total = numbers[0] / numbers[1]  # Divide the first number by the second number

    print("The division of the two numbers is:", total)

# Let us call the function
divide_numbers()
```

Important points to note.

1. The first line involves defining a function called divide_numbers ()
2. The variable numbers will store a list of numbers but in this case just two numbers will be stored. We are using the square bracket to donate the fact that this is a list. When working with a list, it is vital to know that positioning is one key feature that will be vital in this program. The first number will get position [0] and the second number will get position [1].
3. We are using len() function to make sure the input has at least 2 character.
4. Try is used to handle the error that can occur during the process of converting the input, which is a string into float. A float is a number with a decimal.
5. In case an error occurs during the process of converting the input to a float, the loop will just go to the except line which is very important in preventing the program from crashing. Then the except line will print the message under the ValueError.
6. Total = 1 is just there to initialize the variable total with a value of 1.
7. The last line is used to call the function to perform the action that we defined in the function's block.

Output