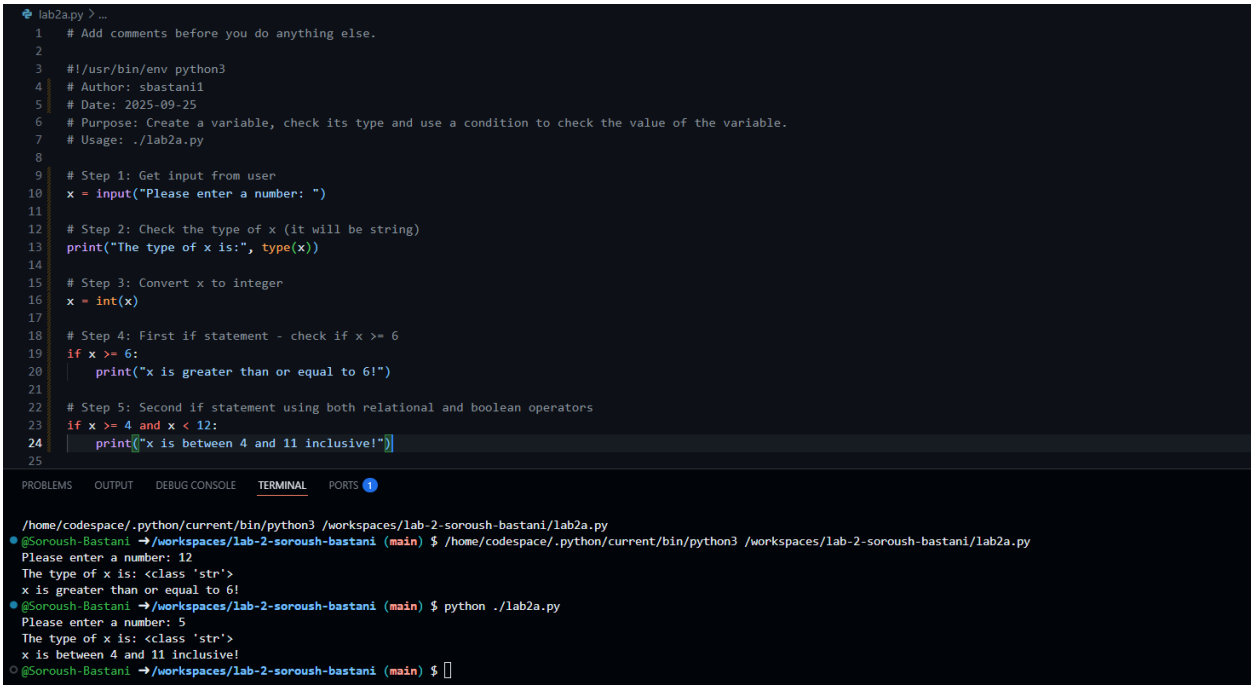
Sbastani1 – Soroush Bastani – 2025/09/25

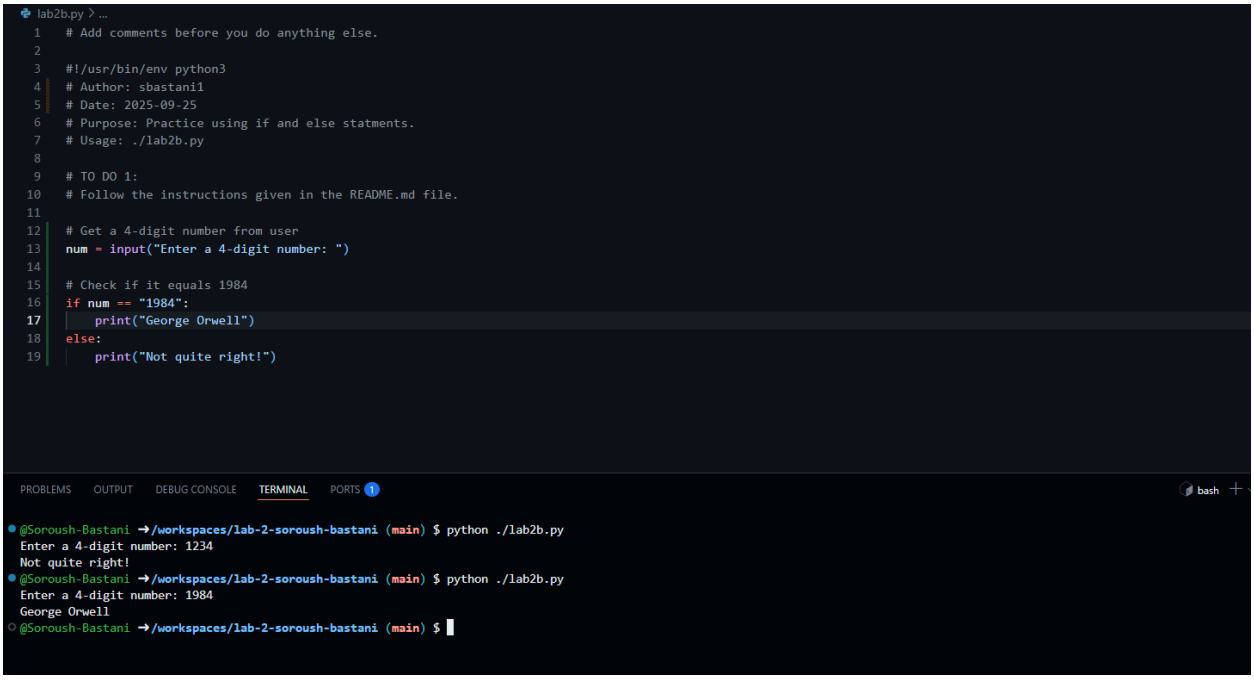
lab2a



```python
# Add comments before you do anything else.

#!/usr/bin/env python3
# Author: sbastani1
# Date: 2025-09-25
# Purpose: Create a variable, check its type and use a condition to check the value of the variable.
# Usage: ./lab2a.py

# Step 1: Get input from user
x = input("Please enter a number: ")

# Step 2: Check the type of x (it will be string)
print("The type of x is:", type(x))

# Step 3: Convert x to integer
x = int(x)

# Step 4: First if statement - check if x >= 6
if x >= 6:
    print("x is greater than or equal to 6!")

# Step 5: Second if statement using both relational and boolean operators
if x >= 4 and x < 12:
    print("x is between 4 and 11 inclusive!")
```

```
/home/codespace/.python/current/bin/python3 /workspaces/lab-2-soroush-bastani/lab2a.py
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ /home/codespace/.python/current/bin/python3 /workspaces/lab-2-soroush-bastani/lab2a.py
Please enter a number: 12
The type of x is: <class 'str'>
x is greater than or equal to 6!
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2a.py
Please enter a number: 5
The type of x is: <class 'str'>
x is between 4 and 11 inclusive!
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $
```

lab2b



```python
# Add comments before you do anything else.

#!/usr/bin/env python3
# Author: sbastani1
# Date: 2025-09-25
# Purpose: Practice using if and else statments.
# Usage: ./lab2b.py

# TO DO 1:
# Follow the instructions given in the README.md file.

# Get a 4-digit number from user
num = input("Enter a 4-digit number: ")

# Check if it equals 1984
if num == "1984":
    print("George Orwell")
else:
    print("Not quite right!")
```

```
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2b.py
Enter a 4-digit number: 1234
Not quite right!
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2b.py
Enter a 4-digit number: 1984
George Orwell
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $
```

lab2c

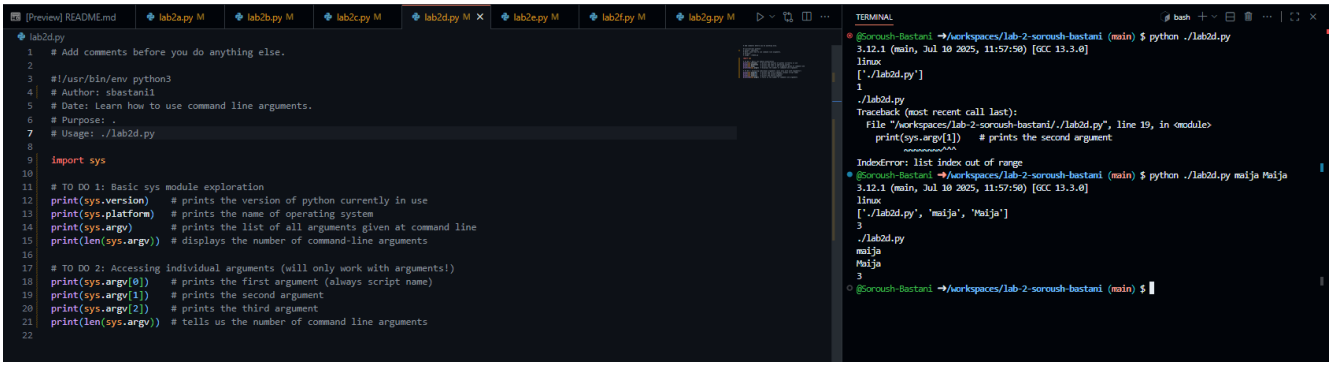
lab2c.py > ...

```python
# Add comments before you do anything else.

#!/usr/bin/env python3
# Author: sbastani1
# Date: 2025-09-25
# Purpose: Practice using if, elif, and else statments.
# Usage: ./lab2c.py

# Prompt the user to enter a sentence, save it in the variable str1
str1 = input("Enter a sentence: ")

# Prompt the user to enter another sentence, save it in the variable str2
str2 = input("Enter another sentence: ")

# Use if, elif, and else statements with the len() function to check which of the 2 is longer
if len(str1) > len(str2):
    print("str1 is longer than str2!")
elif len(str2) > len(str1):
    print("str2 is longer than str1!")
else:
    print("str1 and str2 are of equal length!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1

```
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2c.py
Enter a sentence: hey love
Enter another sentence: hey love
str1 and str2 are of equal length!
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $
```

lab2d



[Preview] README.md | lab2a.py M | lab2b.py M | lab2c.py M | lab2d.py M | lab2e.py M | lab2f.py M | lab2g.py M

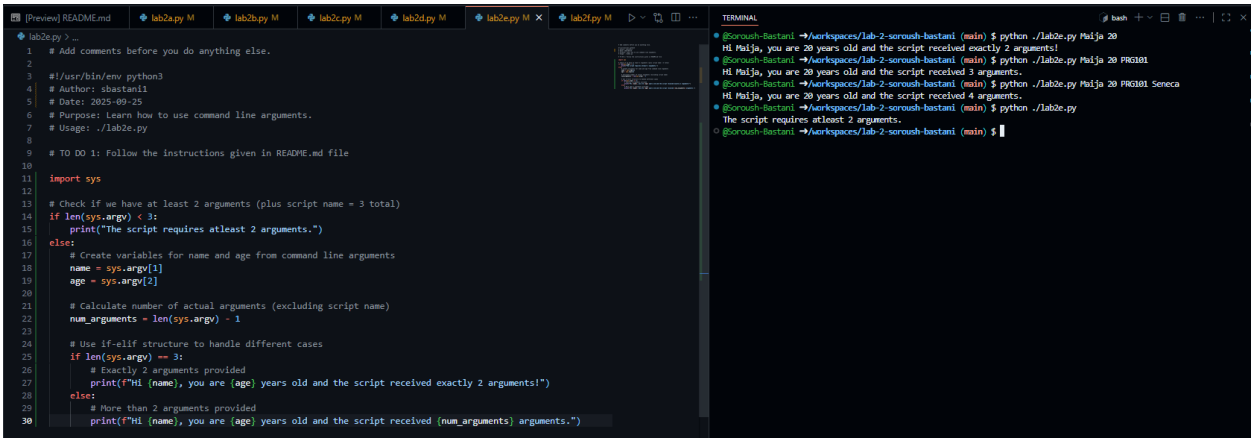
lab2d.py

```python
# Add comments before you do anything else.

#!/usr/bin/env python3
# Author: sbastani1
# Date: Learn how to use command line arguments.
# Purpose: .
# Usage: ./lab2d.py

import sys

# TO DO 1: Basic sys module exploration
print(sys.version)    # prints the version of python currently in use
print(sys.platform)   # prints the name of operating system
print(sys.argv)       # prints the list of all arguments given at command line
print(len(sys.argv))  # displays the number of command-line arguments

# TO DO 2: Accessing individual arguments (will only work with arguments!)
print(sys.argv[0])    # prints the first argument (always script name)
print(sys.argv[1])    # prints the second argument
print(sys.argv[2])    # prints the third argument
print(len(sys.argv))  # tells us the number of command line arguments

```

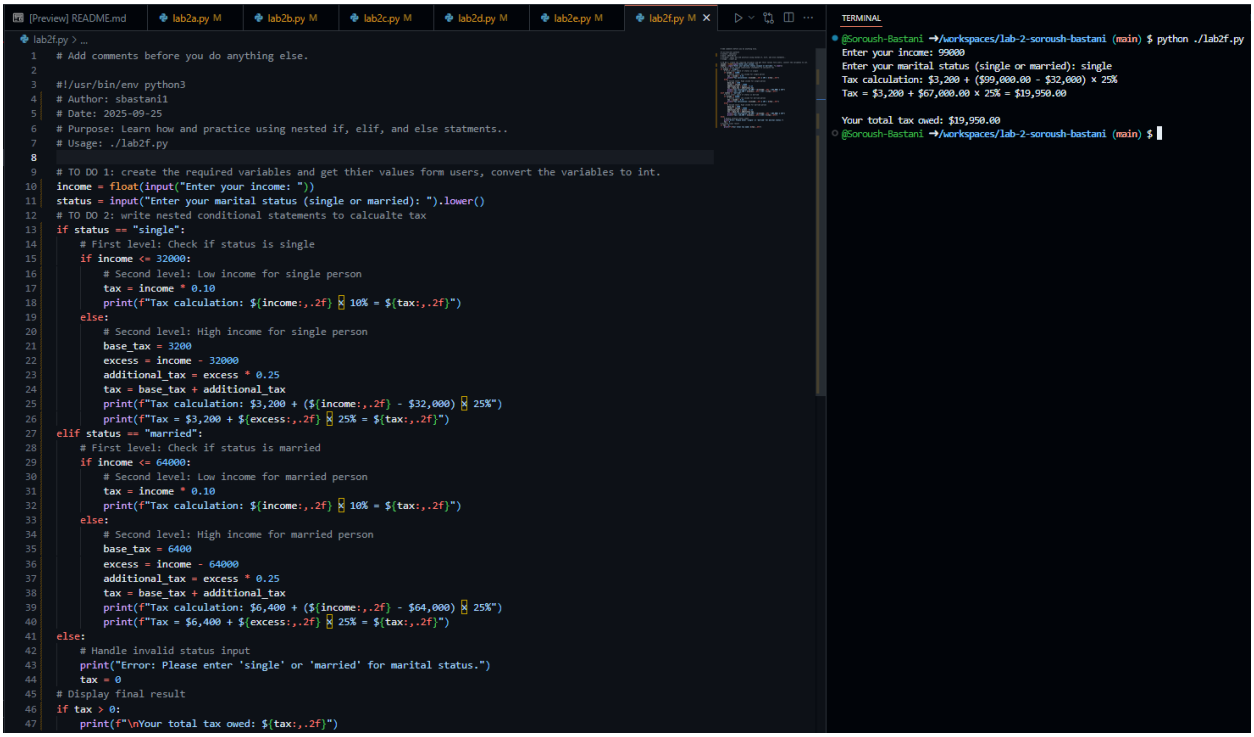
TERMINAL — bash

```
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2d.py
3.12.1 (main, Jul 10 2025, 11:57:50) [GCC 13.3.0]
linux
['./lab2d.py']
1
./lab2d.py
Traceback (most recent call last):
  File "/workspaces/lab-2-soroush-bastani/./lab2d.py", line 19, in <module>
    print(sys.argv[1])    # prints the second argument
          ~~~~~~~~^^^
IndexError: list index out of range
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2d.py maija Maija
3.12.1 (main, Jul 10 2025, 11:57:50) [GCC 13.3.0]
linux
['./lab2d.py', 'maija', 'Maija']
3
./lab2d.py
maija
Maija
3
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $
```

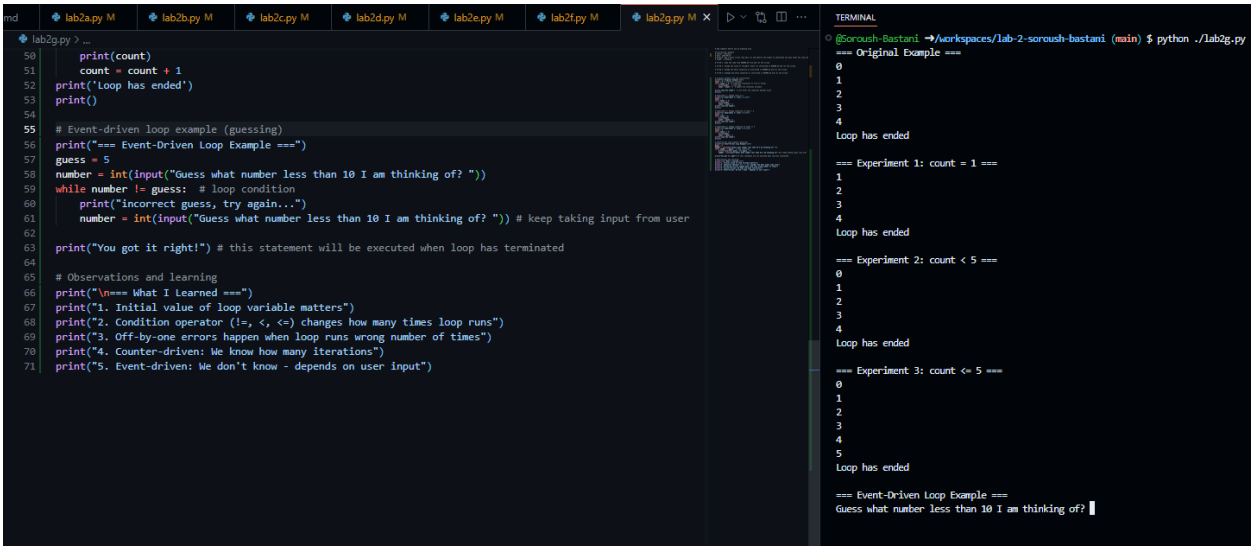
lab2e


[Preview] README.md
lab2a.py M
lab2b.py M
lab2c.py M
lab2d.py M
lab2e.py M
lab2f.py M
lab2e.py > ...
Add comments before you do anything else.
#!/usr/bin/env python3
# Author: sbastani1
# Date: 2025-09-25
# Purpose: Learn how to use command line arguments.
# Usage: ./lab2e.py
# TO DO 1: Follow the instructions given in README.md file
import sys
# Check if we have at least 2 arguments (plus script name = 3 total)
if len(sys.argv) < 3:
print("The script requires atleast 2 arguments.")
else:
# Create variables for name and age from command line arguments
name = sys.argv[1]
age = sys.argv[2]
# Calculate number of actual arguments (excluding script name)
num_arguments = len(sys.argv) - 1
# Use if-elif structure to handle different cases
if len(sys.argv) == 3:
# Exactly 2 arguments provided
print(f"Hi {name}, you are {age} years old and the script received exactly 2 arguments!")
else:
# More than 2 arguments provided
print(f"Hi {name}, you are {age} years old and the script received {num_arguments} arguments.")
TERMINAL
bash
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2e.py Maija 20
Hi Maija, you are 20 years old and the script received exactly 2 arguments!
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2e.py Maija 20 PRG101
Hi Maija, you are 20 years old and the script received 3 arguments.
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2e.py Maija 20 PRG101 Seneca
Hi Maija, you are 20 years old and the script received 4 arguments.
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2e.py
The script requires atleast 2 arguments.
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $

lab2f


[Preview] README.md
lab2a.py M
lab2b.py M
lab2c.py M
lab2d.py M
lab2e.py M
lab2f.py M
lab2f.py > ...
Add comments before you do anything else.
#!/usr/bin/env python3
# Author: sbastani1
# Date: 2025-09-25
# Purpose: Learn how and practice using nested if, elif, and else statments..
# Usage: ./lab2f.py
# TO DO 1: create the required variables and get thier values form users, convert the variables to int.
income = float(input("Enter your income: "))
status = input("Enter your marital status (single or married): ").lower()
# TO DO 2: write nested conditional statements to calcualte tax
if status == "single":
# First level: Check if status is single
if income <= 32000:
# Second level: Low income for single person
tax = income * 0.10
print(f"Tax calculation: ${income:,.2f} × 10% = ${tax:,.2f}")
else:
# Second level: High income for single person
base_tax = 3200
excess = income - 32000
additional_tax = excess * 0.25
tax = base_tax + additional_tax
print(f"Tax calculation: $3,200 + (${income:,.2f} - $32,000) × 25%")
print(f"Tax = $3,200 + ${excess:,.2f} × 25% = ${tax:,.2f}")
elif status == "married":
# First level: Check if status is married
if income <= 64000:
# Second level: Low income for married person
tax = income * 0.10
print(f"Tax calculation: ${income:,.2f} × 10% = ${tax:,.2f}")
else:
# Second level: High income for married person
base_tax = 6400
excess = income - 64000
additional_tax = excess * 0.25
tax = base_tax + additional_tax
print(f"Tax calculation: $6,400 + (${income:,.2f} - $64,000) × 25%")
print(f"Tax = $6,400 + ${excess:,.2f} × 25% = ${tax:,.2f}")
else:
# Handle invalid status input
print("Error: Please enter 'single' or 'married' for marital status.")
tax = 0
# Display final result
if tax > 0:
print(f"\nYour total tax owed: ${tax:,.2f}")
TERMINAL
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2f.py
Enter your income: 99000
Enter your marital status (single or married): single
Tax calculation: $3,200 + ($99,000.00 - $32,000) × 25%
Tax = $3,200 + $67,000.00 × 25% = $19,950.00
Your total tax owed: $19,950.00
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $

lab2g



```
    print(count)
    count = count + 1
print('Loop has ended')
print()

# Event-driven loop example (guessing)
print("=== Event-Driven Loop Example ===")
guess = 5
number = int(input("Guess what number less than 10 I am thinking of? "))
while number != guess:  # loop condition
    print("incorrect guess, try again...")
    number = int(input("Guess what number less than 10 I am thinking of? ")) # keep taking input from user

print("You got it right!") # this statement will be executed when loop has terminated

# Observations and learning
print("\n=== What I Learned ===")
print("1. Initial value of loop variable matters")
print("2. Condition operator (!=, <, <=) changes how many times loop runs")
print("3. Off-by-one errors happen when loop runs wrong number of times")
print("4. Counter-driven: We know how many iterations")
print("5. Event-driven: We don't know - depends on user input")
```

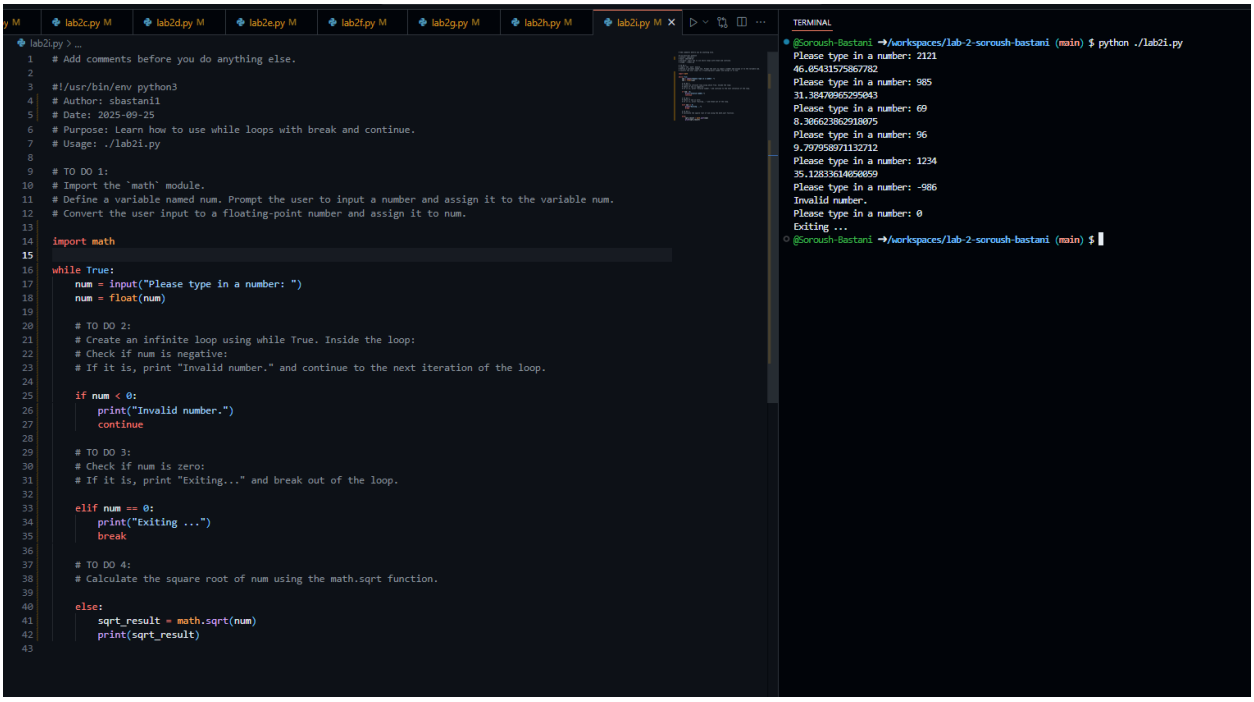
```
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2g.py
=== Original Example ===
0
1
2
3
4
Loop has ended

=== Experiment 1: count = 1 ===
1
2
3
4
Loop has ended

=== Experiment 2: count < 5 ===
0
1
2
3
4
Loop has ended

=== Experiment 3: count <= 5 ===
0
1
2
3
4
5
Loop has ended

=== Event-Driven Loop Example ===
Guess what number less than 10 I am thinking of?
```

lab2h

```
# Add a while loop to create program that wont end until the user enters 1234.
# Follow the specific instructions given in the README.md file.

# Keep asking until user enters correct PIN
while pin != "1234":
    # Check if PIN is exactly 4 digits and all characters are numeric
    if len(pin) != 4 or not pin.isdigit():
        print("Incorrect...enter a 4 digit number")
    else:
        # PIN is 4 digits, but wrong PIN
        print("Incorrect...try again")

    # Add blank line for formatting
    print()

    # Ask for PIN again
    pin = input("Please type in your PIN: ")

# When loop ends, user entered correct PIN
print("Correct PIN, You can enter!")
```

```
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2h.py
Please type in your PIN: 1234
Correct PIN, You can enter!
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2h.py
Please type in your PIN: 99
Incorrect...enter a 4 digit number

Please type in your PIN: 9999
Incorrect...try again

Please type in your PIN: 1234
Correct PIN, You can enter!
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $
```

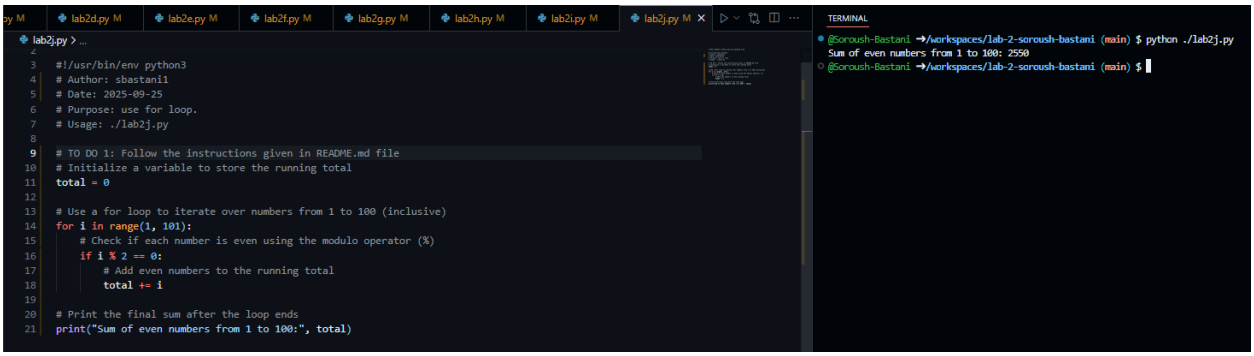
lab2i



```python
# Add comments before you do anything else.

#!/usr/bin/env python3
# Author: sbastani1
# Date: 2025-09-25
# Purpose: Learn how to use while loops with break and continue.
# Usage: ./lab2i.py

# TO DO 1:
# Import the `math` module.
# Define a variable named num. Prompt the user to input a number and assign it to the variable num.
# Convert the user input to a floating-point number and assign it to num.

import math

while True:
    num = input("Please type in a number: ")
    num = float(num)

    # TO DO 2:
    # Create an infinite loop using while True. Inside the loop:
    # Check if num is negative:
    # If it is, print "Invalid number." and continue to the next iteration of the loop.

    if num < 0:
        print("Invalid number.")
        continue

    # TO DO 3:
    # Check if num is zero:
    # If it is, print "Exiting..." and break out of the loop.

    elif num == 0:
        print("Exiting ...")
        break

    # TO DO 4:
    # Calculate the square root of num using the math.sqrt function.

    else:
        sqrt_result = math.sqrt(num)
        print(sqrt_result)
```

```
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2i.py
Please type in a number: 2121
46.05431575867782
Please type in a number: 985
31.38470965295043
Please type in a number: 69
8.306623862918075
Please type in a number: 96
9.797958971132712
Please type in a number: 1234
35.128336140560659
Please type in a number: -986
Invalid number.
Please type in a number: 0
Exiting ...
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $
```

lab2j



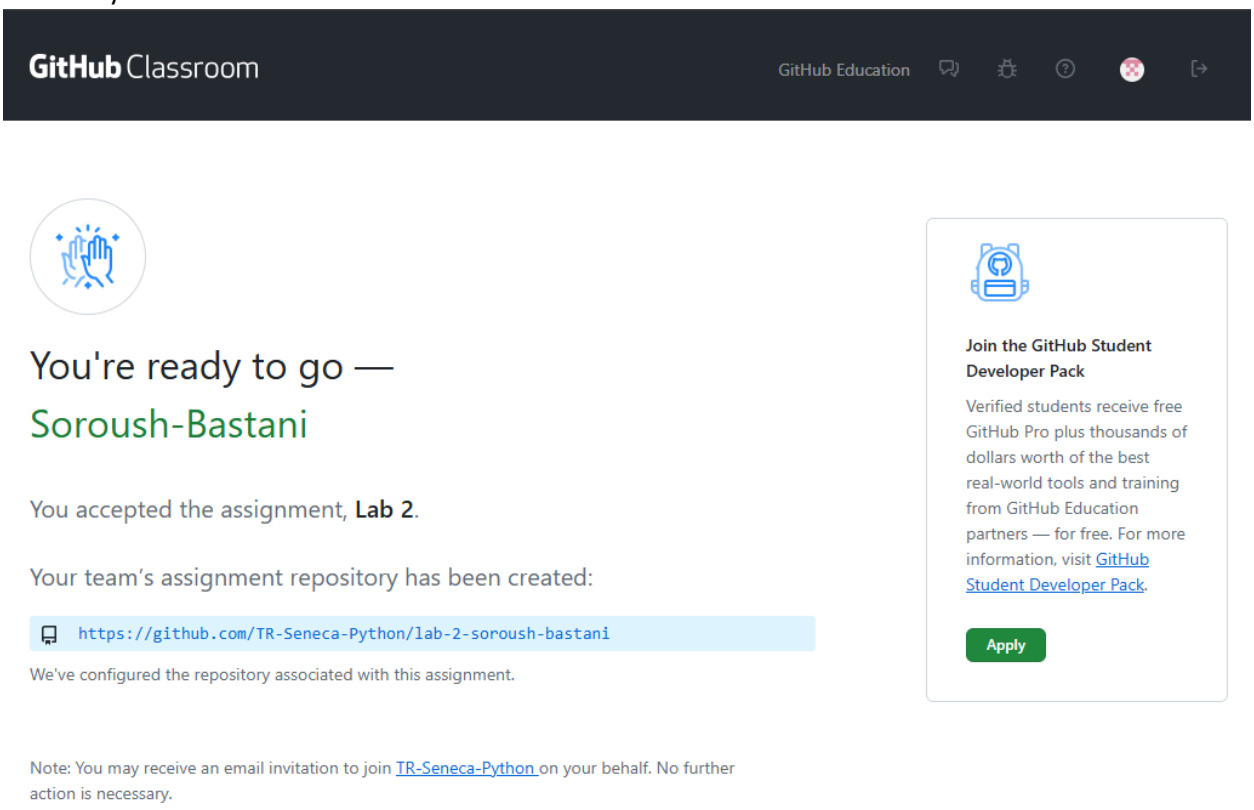
```python
#!/usr/bin/env python3
# Author: sbastani1
# Date: 2025-09-25
# Purpose: use for loop.
# Usage: ./lab2j.py

# TO DO 1: Follow the instructions given in README.md file
# Initialize a variable to store the running total
total = 0

# Use a for loop to iterate over numbers from 1 to 100 (inclusive)
for i in range(1, 101):
    # Check if each number is even using the modulo operator (%)
    if i % 2 == 0:
        # Add even numbers to the running total
        total += i

# Print the final sum after the loop ends
print("Sum of even numbers from 1 to 100:", total)
```

```
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $ python ./lab2j.py
Sum of even numbers from 1 to 100: 2550
@Soroush-Bastani ➜/workspaces/lab-2-soroush-bastani (main) $
```

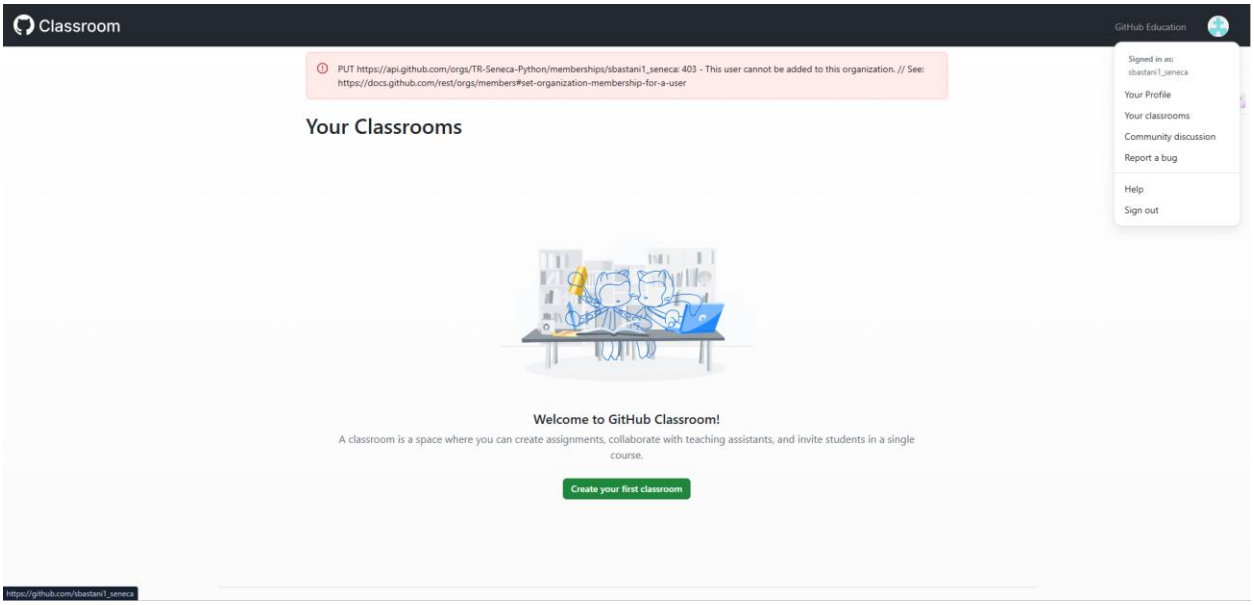
Note:

When I log in with my personal Github Soroush-Bastani, I'm able to accept this lab.

Hence my username is Soroush-Bastani



But when I use the Seneca SSO (by going to https://github.com/senecapolytechnic) and try to accept the lab, this happens:



My Seneca ID is sbastani1

Thanks for reading