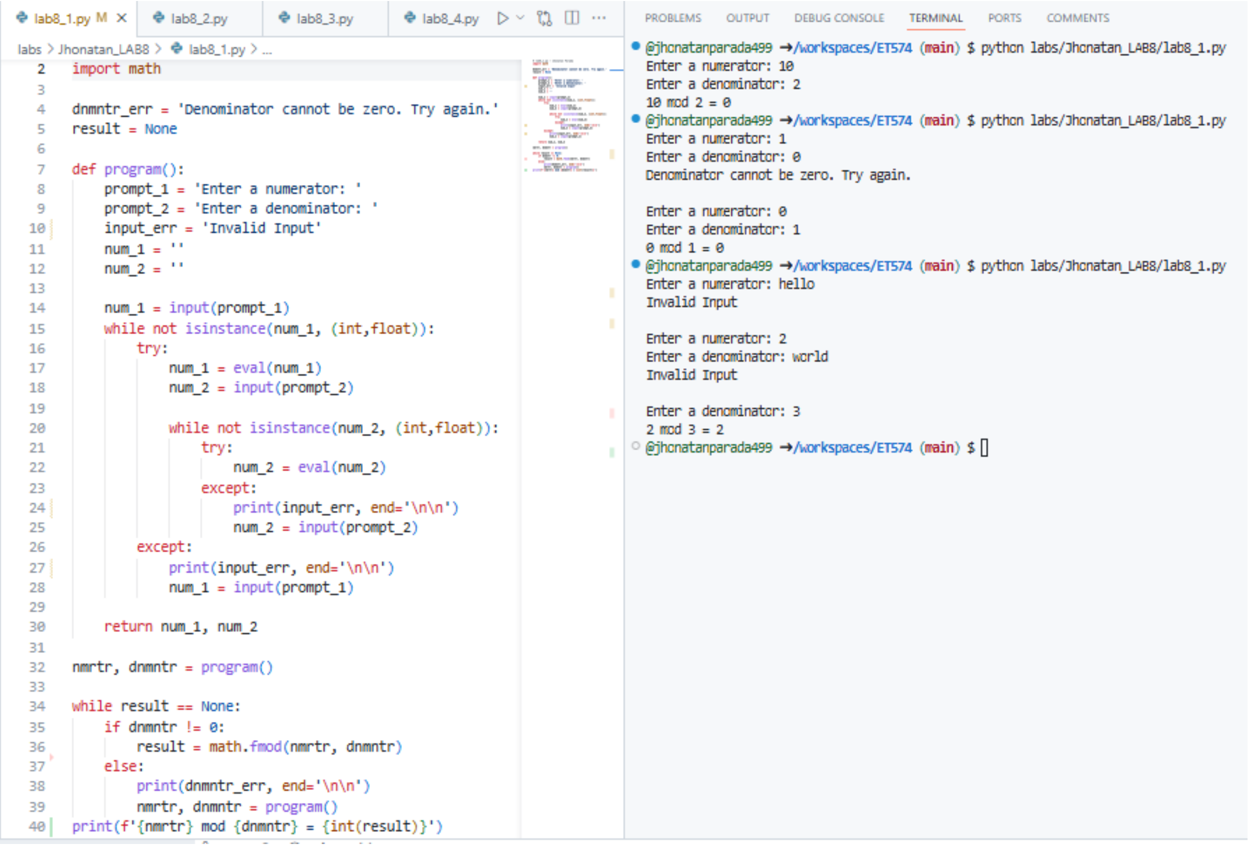
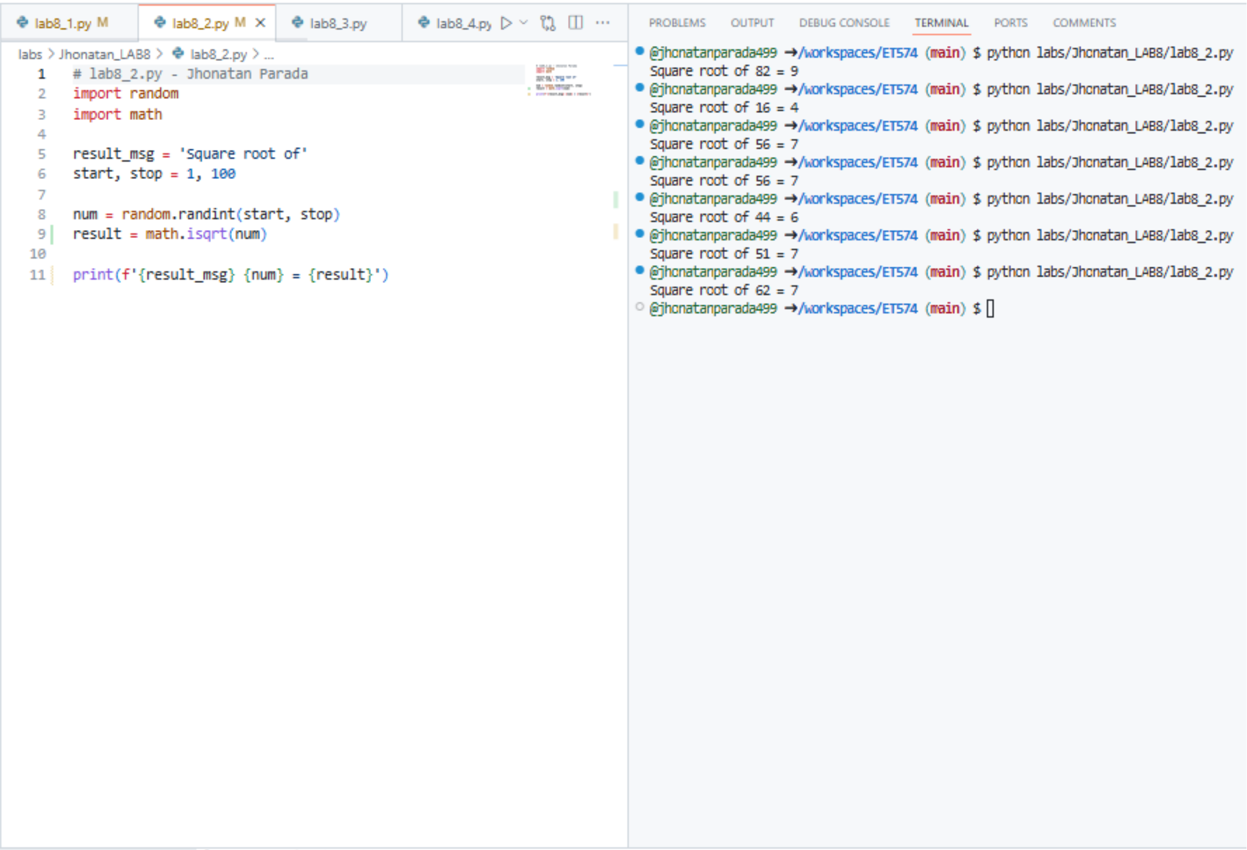
Lab 8 Summary



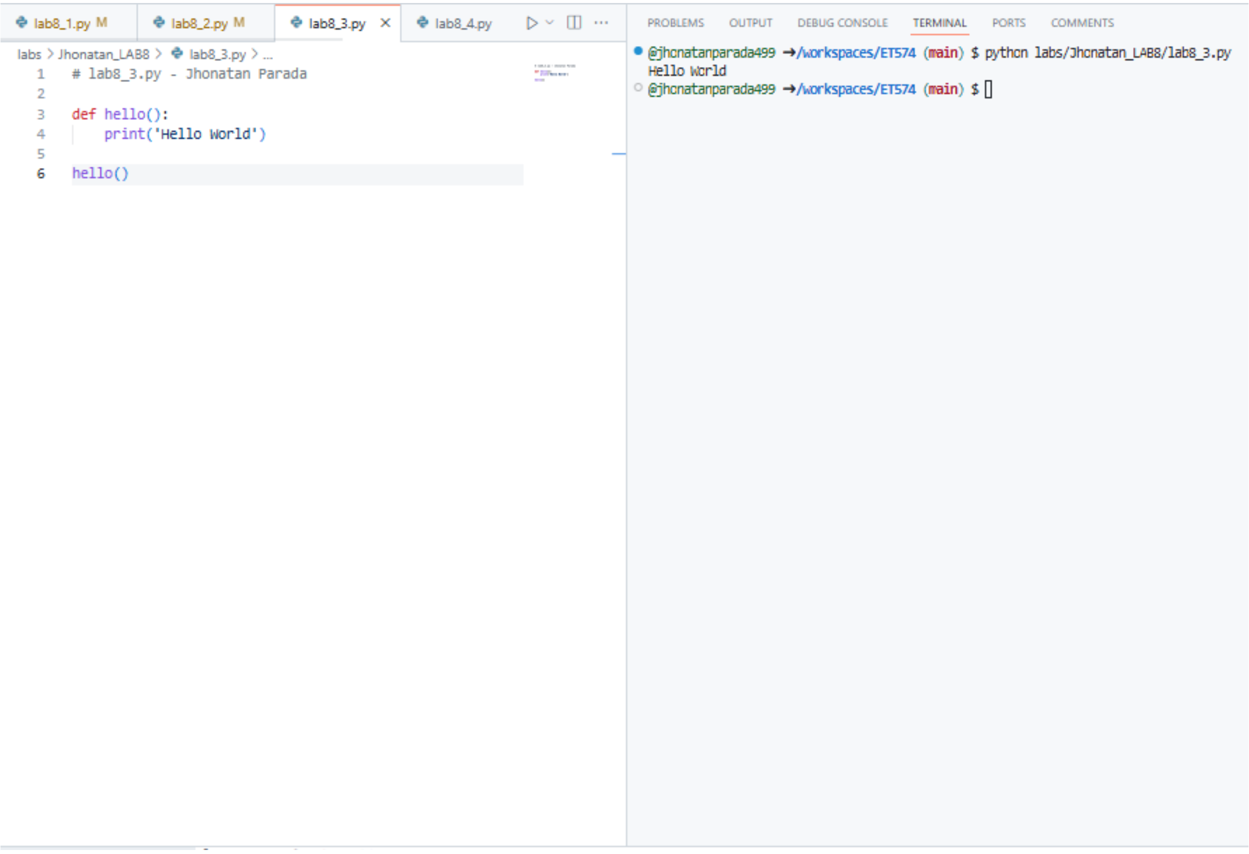
```python
import math

dnmntr_err = 'Denominator cannot be zero. Try again.'
result = None

def program():
    prompt_1 = 'Enter a numerator: '
    prompt_2 = 'Enter a denominator: '
    input_err = 'Invalid Input'
    num_1 = ''
    num_2 = ''

    num_1 = input(prompt_1)
    while not isinstance(num_1, (int,float)):
        try:
            num_1 = eval(num_1)
            num_2 = input(prompt_2)

            while not isinstance(num_2, (int,float)):
                try:
                    num_2 = eval(num_2)
                except:
                    print(input_err, end='\n\n')
                    num_2 = input(prompt_2)
        except:
            print(input_err, end='\n\n')
            num_1 = input(prompt_1)

    return num_1, num_2

nmrtr, dnmntr = program()

while result == None:
    if dnmntr != 0:
        result = math.fmod(nmrtr, dnmntr)
    else:
        print(dnmntr_err, end='\n\n')
        nmrtr, dnmntr = program()
print(f'{nmrtr} mod {dnmntr} = {int(result)}')
```

```
● @jhonatanparada499 ➜/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_1.py
Enter a numerator: 10
Enter a denominator: 2
10 mod 2 = 0
● @jhonatanparada499 ➜/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_1.py
Enter a numerator: 1
Enter a denominator: 0
Denominator cannot be zero. Try again.

Enter a numerator: 0
Enter a denominator: 1
0 mod 1 = 0
● @jhonatanparada499 ➜/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_1.py
Enter a numerator: hello
Invalid Input

Enter a numerator: 2
Enter a denominator: world
Invalid Input

Enter a denominator: 3
2 mod 3 = 2
○ @jhonatanparada499 ➜/workspaces/ET574 (main) $ 
```



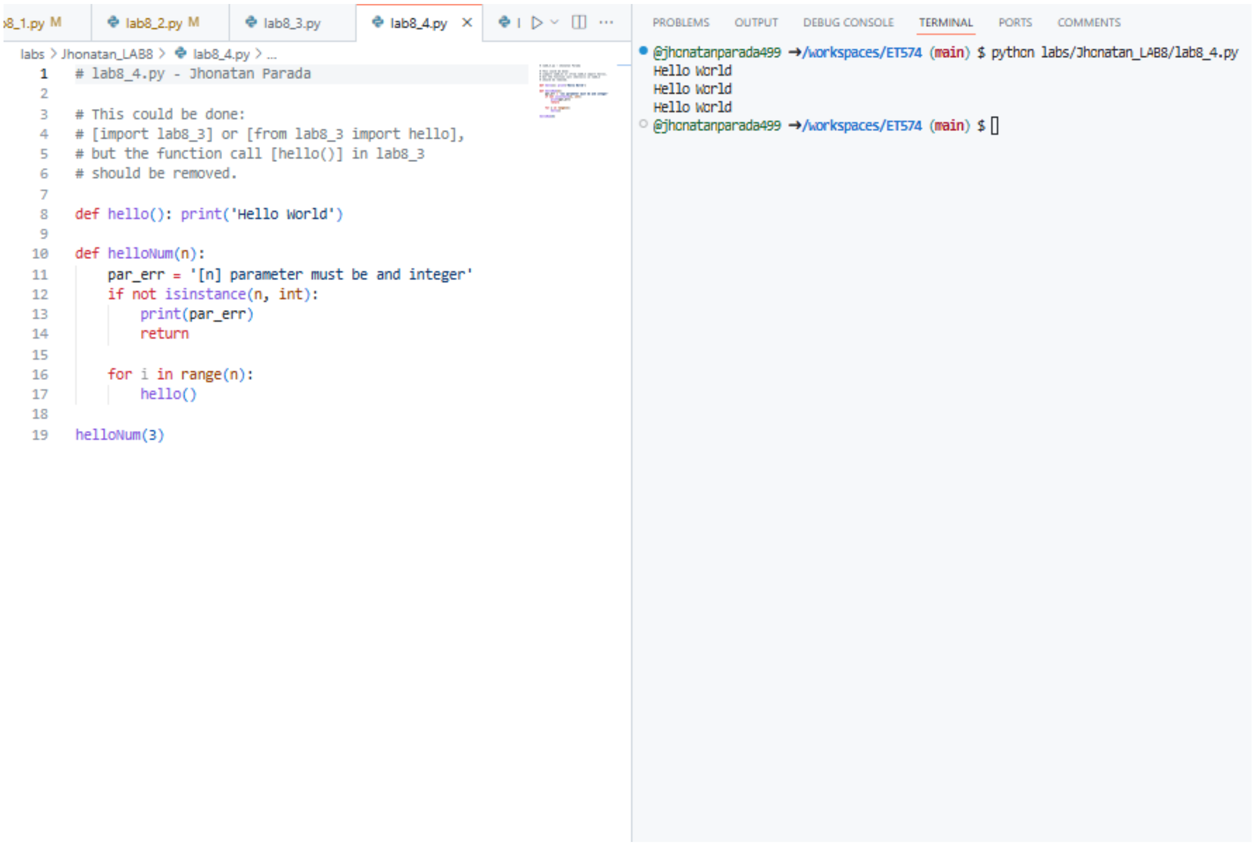
```python
# lab8_2.py - Jhonatan Parada
import random
import math

result_msg = 'Square root of'
start, stop = 1, 100

num = random.randint(start, stop)
result = math.isqrt(num)

print(f'{result_msg} {num} = {result}')
```

```
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_2.py
Square root of 82 = 9
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_2.py
Square root of 16 = 4
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_2.py
Square root of 56 = 7
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_2.py
Square root of 56 = 7
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_2.py
Square root of 44 = 6
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_2.py
Square root of 51 = 7
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_2.py
Square root of 62 = 7
@jhonatanparada499 ➜ /workspaces/ET574 (main) $
```



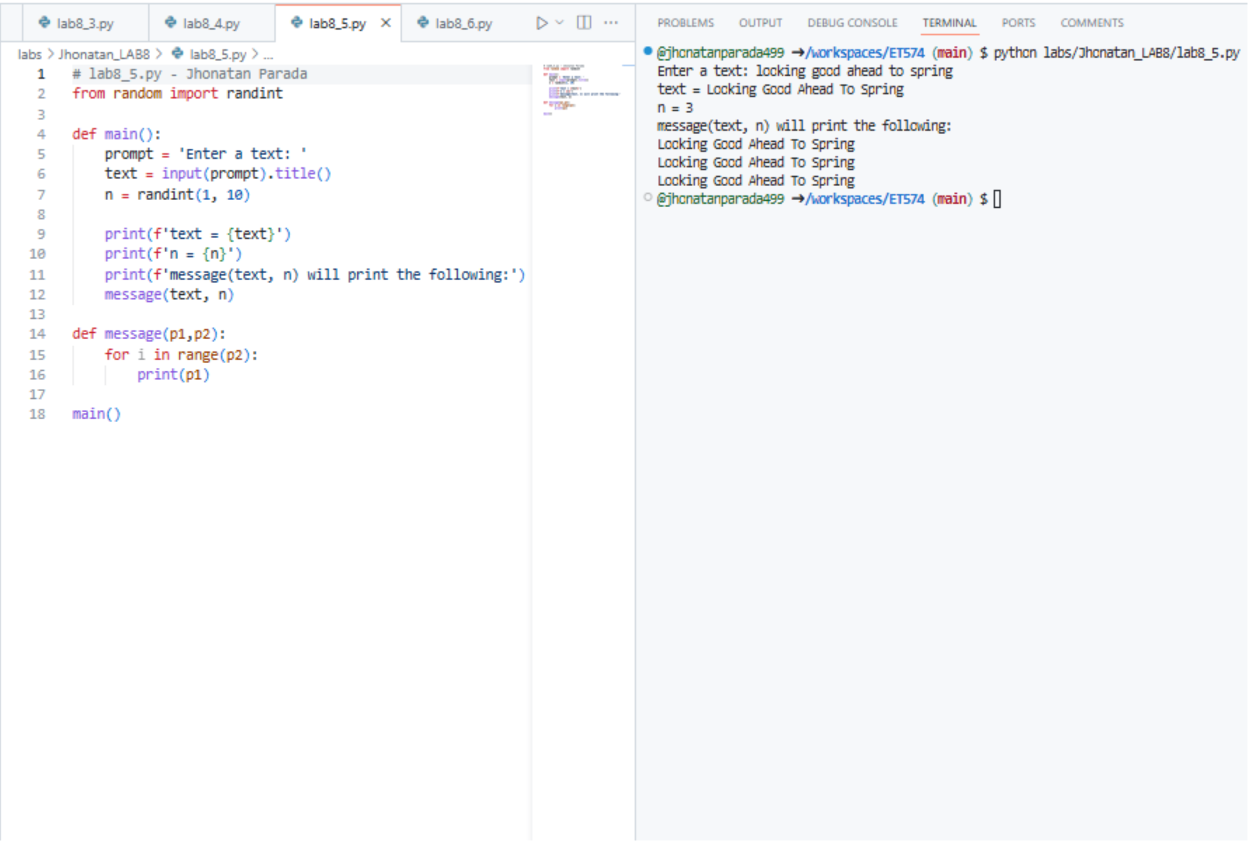
```python
# lab8_3.py - Jhonatan Parada

def hello():
    print('Hello World')

hello()
```

```
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_3.py
Hello World
@jhonatanparada499 ➜ /workspaces/ET574 (main) $
```



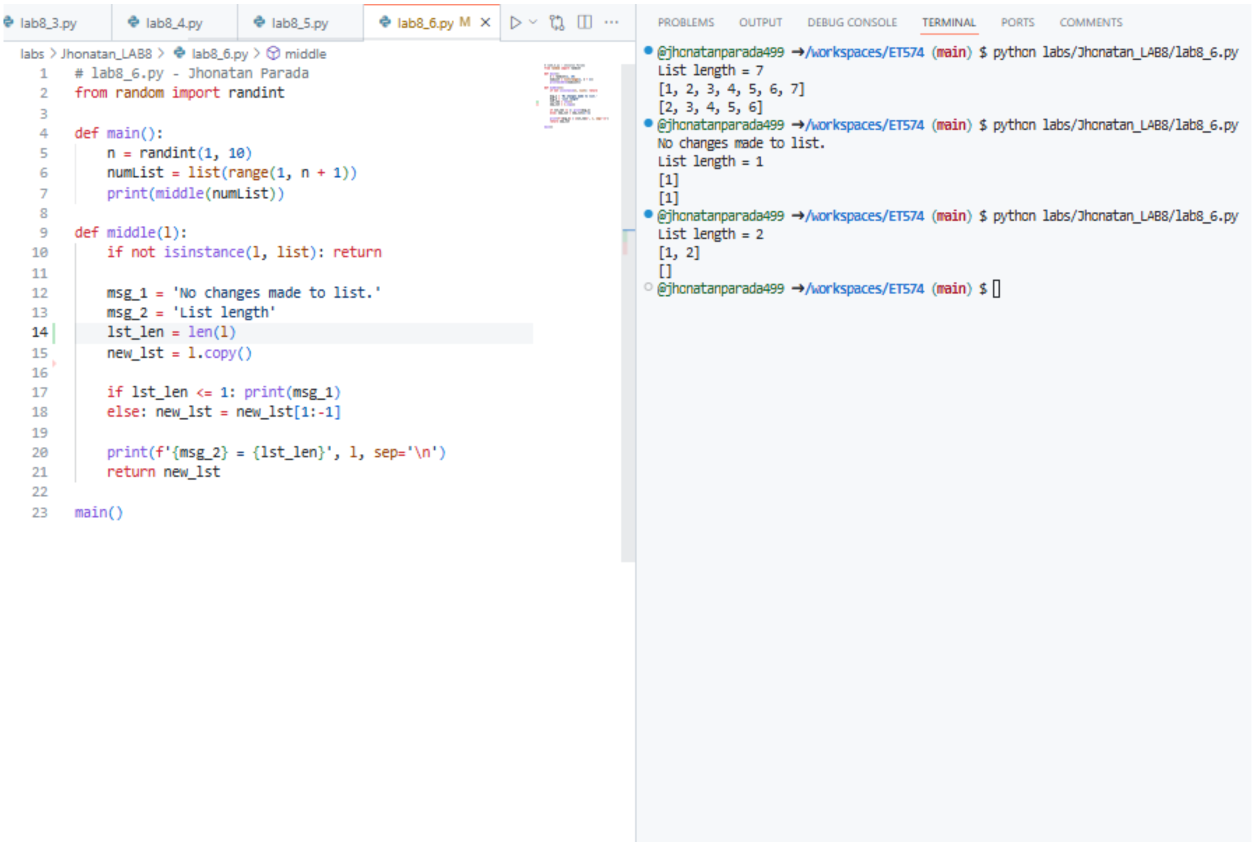
```python
# lab8_4.py - Jhonatan Parada

# This could be done:
# [import lab8_3] or [from lab8_3 import hello],
# but the function call [hello()] in lab8_3
# should be removed.

def hello(): print('Hello World')

def helloNum(n):
    par_err = '[n] parameter must be and integer'
    if not isinstance(n, int):
        print(par_err)
        return

    for i in range(n):
        hello()

helloNum(3)
```

```
@jhonatanparada499 ➜/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_4.py
Hello World
Hello World
Hello World
@jhonatanparada499 ➜/workspaces/ET574 (main) $ 
```



```python
# lab8_5.py - Jhonatan Parada
from random import randint

def main():
    prompt = 'Enter a text: '
    text = input(prompt).title()
    n = randint(1, 10)

    print(f'text = {text}')
    print(f'n = {n}')
    print(f'message(text, n) will print the following:')
    message(text, n)

def message(p1,p2):
    for i in range(p2):
        print(p1)

main()
```

```
@jhonatanparada499 ➜/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_5.py
Enter a text: looking good ahead to spring
text = Looking Good Ahead To Spring
n = 3
message(text, n) will print the following:
Looking Good Ahead To Spring
Looking Good Ahead To Spring
Looking Good Ahead To Spring
@jhonatanparada499 ➜/workspaces/ET574 (main) $ 
```



```python
# lab8_6.py - Jhonatan Parada
from random import randint

def main():
    n = randint(1, 10)
    numList = list(range(1, n + 1))
    print(middle(numList))

def middle(l):
    if not isinstance(l, list): return

    msg_1 = 'No changes made to list.'
    msg_2 = 'List length'
    lst_len = len(l)
    new_lst = l.copy()

    if lst_len <= 1: print(msg_1)
    else: new_lst = new_lst[1:-1]

    print(f'{msg_2} = {lst_len}', l, sep='\n')
    return new_lst

main()
```

```
@jhonatanparada499 →/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_6.py
List length = 7
[1, 2, 3, 4, 5, 6, 7]
[2, 3, 4, 5, 6]
@jhonatanparada499 →/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_6.py
No changes made to list.
List length = 1
[1]
[1]
@jhonatanparada499 →/workspaces/ET574 (main) $ python labs/Jhonatan_LAB8/lab8_6.py
List length = 2
[1, 2]
[]
@jhonatanparada499 →/workspaces/ET574 (main) $ 
```

2. On this lab, while doing lab8_1.py, I realized that using the function [eval] was a more elegant or readable form to convert a string to an int or float data type, unlike I did in the previous lab, where I used a nested try statement to check and convert the string to either integer of float.

On this lab, I also learned that one can reuse code from other files, such as functions. One of the advantages of this is that one can write more organized and readable code, but it might become harder to deal and manage too many dependencies, especially when those dependencies also have other dependencies, it all can become a recursive game all the sudden.