AI

LAB TASK

NAME: Zaid Asghar Virk

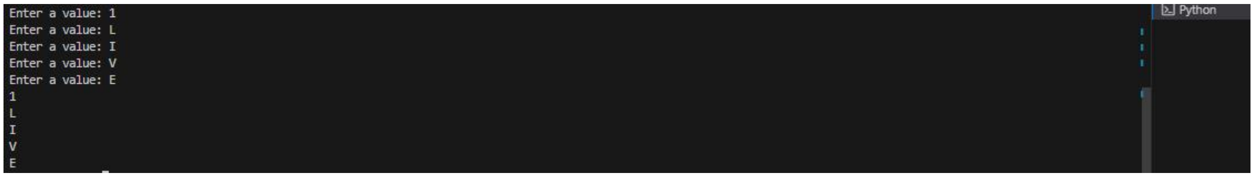
REG NO: FA21-BCS-136

Activity 1

```python
#empty list
my_list = []

for i in range(5):
    value = input("Enter a value: ")
    my_list.append(value)
```

```python
# Displaying
for value in my_list:
    print(value)
```

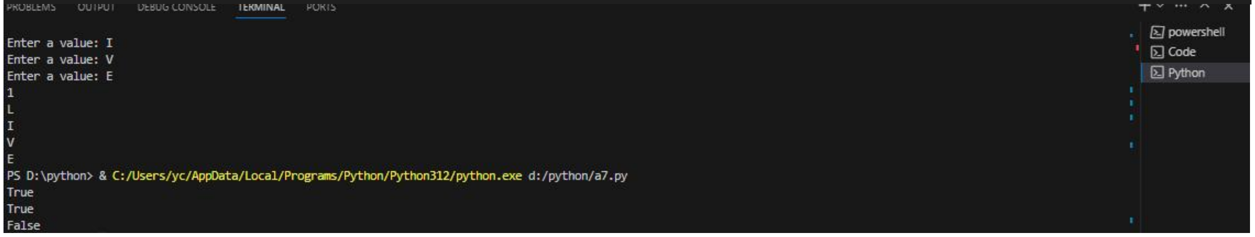
OUTPUT:



```
Enter a value: 1
Enter a value: L
Enter a value: I
Enter a value: V
Enter a value: E
1
L
I
V
E
```

Activity 7:

```python
def is_palindrome(s):
    s = s.lower().replace(' ', '')
    return s == s[::-1]
print(is_palindrome('dad'))  #True
print(is_palindrome('aibohphobia'))  #True
print(is_palindrome('Hello'))  #False
```



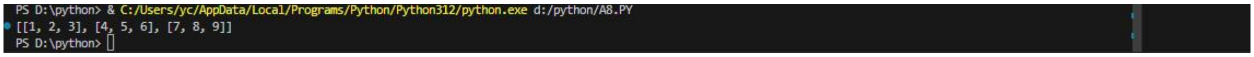
```
Enter a value: I
Enter a value: V
Enter a value: E
1
L
I
V
E
PS D:\python> & C:/Users/yc/AppData/Local/Programs/Python/Python312/python.exe d:/python/a7.py
True
True
False
```

Activity 8:

```
def matrix_multiply(a, b):
    result = [[0 for _ in range(len(b[0]))] for _ in range(len(a))]
    for i in range(len(a)):
        for j in range(len(b[0])):
            for k in range(len(b)):
                result[i][j] += a[i][k] * b[k][j]
    return result

a = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
b = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
c = matrix_multiply(a, b)
print(c)
```

Output:



```
PS D:\python> & C:/Users/yc/AppData/Local/Programs/Python/Python312/python.exe d:/python/A8.PY
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
PS D:\python> 
```

Activity 9:

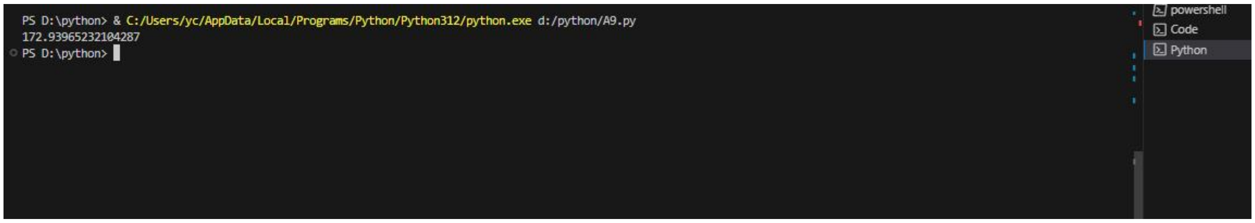
```
import math

def calculate_perimeter(coordinates):
    if not isinstance(coordinates, list) or not all(isinstance(coord, tuple) and
len(coord) == 2 for coord in coordinates):
        raise ValueError("Invalid input: coordinates must be a list of tuples (x,
y).")
```

```
    perimeter = sum(
        math.sqrt((coordinates[i][0] - coordinates[i - 1][0])**2 +
(coordinates[i][1] - coordinates[i - 1][1])**2)
        for i in range(1, len(coordinates))
    ) + math.sqrt((coordinates[0][0] - coordinates[-1][0])**2 + (coordinates[0][1]
- coordinates[-1][1])**2)
```

```
    return perimeter
```

```
coordinates = [(1, 20), (35, 78), (2, 2), (0, 0)]
perimeter = calculate_perimeter(coordinates)
print(perimeter)
```

Output:

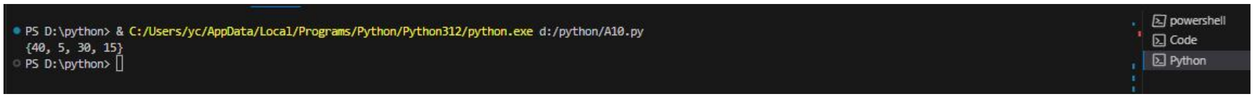


```
PS D:\python> & C:/Users/yc/AppData/Local/Programs/Python/Python312/python.exe d:/python/A9.py
172.93965232104287
PS D:\python> 
```

Activity 10:

```
def symmetric_difference(set1, set2):
  difference = set()

  for element in set1:
    if element not in set2:
      difference.add(element)

  for element in set2:
    if element not in set1:
      difference.add(element)

  return difference

set1 = {5, 10, 15, 20}
set2 = {10, 20, 30, 40}
symmetric_difference_set = symmetric_difference(set1, set2)
print(symmetric_difference_set)
```

Output:



```
PS D:\python> & C:/Users/yc/AppData/Local/Programs/Python/Python312/python.exe d:/python/A10.py
{40, 5, 30, 15}
PS D:\python> 
```

Activity 11:

```
directory = {}

def add_entry(full_name, contact_number):
  directory[full_name] = contact_number
```

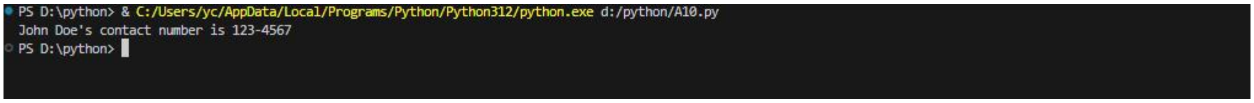
```
def find_entry(full_name):
  if full_name in directory:
    return directory[full_name]
  else:
    return None
```

```
add_entry(("John", "Doe"), "123-4567")
add_entry(("Jane", "Doe"), "234-5678")
```

```
search_full_name = ("John", "Doe")
contact_number = find_entry(search_full_name)
```

```
if contact_number:
  print(f"{search_full_name[0]} {search_full_name[1]}'s contact number is
{contact_number}")
else:
  print(f"{search_full_name[0]} {search_full_name[1]} is not in the directory")
```

Output



```
PS D:\python> & C:/Users/yc/AppData/Local/Programs/Python/Python312/python.exe d:/python/A10.py
John Doe's contact number is 123-4567
PS D:\python> 
```

PART 2

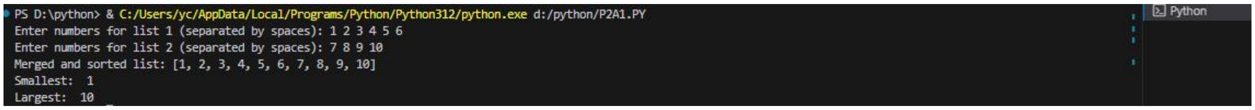
Activity 1 & 2

```python
def get_user_input(list_name):
  while True:
      numbers = input(f"Enter numbers for {list_name} (separated by spaces): ")
      num_list = [int(num) for num in numbers.split()]
      return num_list

def merge_and_sort_lists(list1, list2):

  merged_list = list1 + list2
  merged_list.sort()
  return merged_list
```

```python
list1 = get_user_input("list 1")
list2 = get_user_input("list 2")
```

```python
merged_list = merge_and_sort_lists(list1, list2)
print("Merged and sorted list:", merged_list)
print("Smallest: ", merged_list[0])
print("Largest: ", merged_list[-1])
```

Output:



```
PS D:\python> & C:/Users/yc/AppData/Local/Programs/Python/Python312/python.exe d:/python/P2A1.PY
Enter numbers for list 1 (separated by spaces): 1 2 3 4 5 6
Enter numbers for list 2 (separated by spaces): 7 8 9 10
Merged and sorted list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Smallest:  1
Largest:  10
```

Assignment

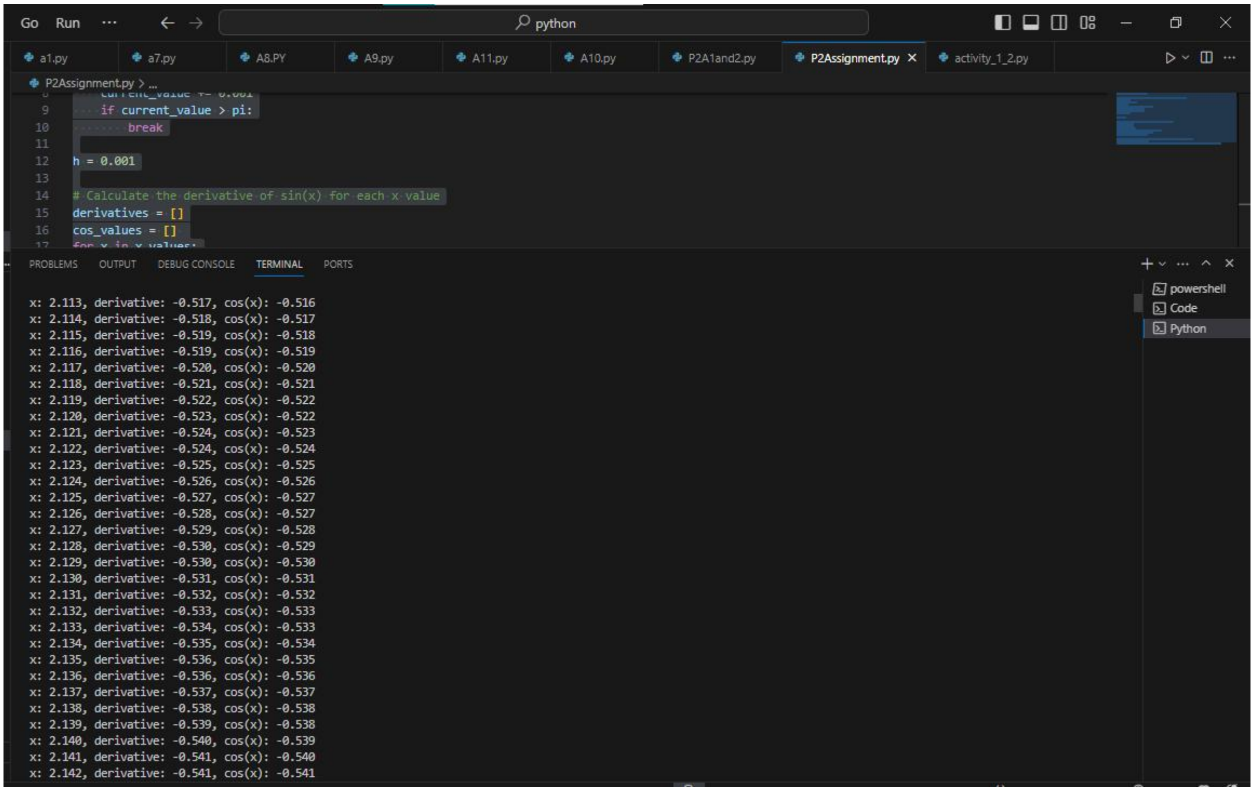
```
from math import cos, pi, sin

# Create a list of x values from -pi to pi, incrementing by 0.001
x_values = []
current_value = -pi
while True:
    x_values.append(current_value)
    current_value += 0.001
    if current_value > pi:
        break
```

```
h = 0.001
```

```
# Calculate the derivative of sin(x) for each x value
derivatives = []
cos_values = []
for x in x_values:
    derivative = (sin(x + h) - sin(x)) / h
    derivatives.append(derivative)
    cos_values.append(cos(x))
```

```
# Print each x value with its corresponding derivative and cos(x) value
for i in range(len(x_values)):
    print(f"x: {x_values[i]:.3f}, derivative: {derivatives[i]:.3f}, cos(x):
{cos_values[i]:.3f}")
```

output



```python
    current_value += 0.001
    if current_value > pi:
        break

h = 0.001

# Calculate the derivative of sin(x) for each x value
derivatives = []
cos_values = []
for x in x_values:
```

```
x: 2.113, derivative: -0.517, cos(x): -0.516
x: 2.114, derivative: -0.518, cos(x): -0.517
x: 2.115, derivative: -0.519, cos(x): -0.518
x: 2.116, derivative: -0.519, cos(x): -0.519
x: 2.117, derivative: -0.520, cos(x): -0.520
x: 2.118, derivative: -0.521, cos(x): -0.521
x: 2.119, derivative: -0.522, cos(x): -0.522
x: 2.120, derivative: -0.523, cos(x): -0.522
x: 2.121, derivative: -0.524, cos(x): -0.523
x: 2.122, derivative: -0.524, cos(x): -0.524
x: 2.123, derivative: -0.525, cos(x): -0.525
x: 2.124, derivative: -0.526, cos(x): -0.526
x: 2.125, derivative: -0.527, cos(x): -0.527
x: 2.126, derivative: -0.528, cos(x): -0.527
x: 2.127, derivative: -0.529, cos(x): -0.528
x: 2.128, derivative: -0.530, cos(x): -0.529
x: 2.129, derivative: -0.530, cos(x): -0.530
x: 2.130, derivative: -0.531, cos(x): -0.531
x: 2.131, derivative: -0.532, cos(x): -0.532
x: 2.132, derivative: -0.533, cos(x): -0.533
x: 2.133, derivative: -0.534, cos(x): -0.533
x: 2.134, derivative: -0.535, cos(x): -0.534
x: 2.135, derivative: -0.536, cos(x): -0.535
x: 2.136, derivative: -0.536, cos(x): -0.536
x: 2.137, derivative: -0.537, cos(x): -0.537
x: 2.138, derivative: -0.538, cos(x): -0.538
x: 2.139, derivative: -0.539, cos(x): -0.538
x: 2.140, derivative: -0.540, cos(x): -0.539
x: 2.141, derivative: -0.541, cos(x): -0.540
x: 2.142, derivative: -0.541, cos(x): -0.541
```