Lab Report 0: Introduction to Python

Course: EEEE 536.60L1 and EEEE 636.60L1 – Biorobotics/Cybernetics

Group Members: Youssief Khalifa(yak9700), Majd Abo Kalam(mma7566)

Date: 30-1-2025

---

1) Manifesto

Both team mbmers actively worked in every section of this lab report. The structure of which collaboration was very efficient and led to high standards for all tasks being completed. We made sure that the lab questions and tasks were divided equally between both of us so that there is fairness and efficiency. The responsibility for specific set of exercises and sections of the report was taken by each of the team member. This also includes the coding parts we divided the work 5 by 5, Youssief Khalifa took the first 5 set of exercise questions and Majd Abo Kalam did the remaining 5.

---

2) Introduction

The lab introduced us to the basic elements of Python: printing of output, introduction and use of variables, loops, conditional statements, functions, and object-oriented programming. In doing such exercises, students became fluent in Python syntax, and learn to work with such data structures as lists, tuples, and dictionaries. This lab also covers file handling and exception management, which are used in real-world programming tasks.

---

3) Theory

A) As we may know python is a high-level, interpretive computer language with a high level of readability and simplicity. Python is a language that is broadly utilized in scientific computation, data analysis, and artificial intelligence. In the laboratory, following theoretical constructs have been emphasized:

I.

- **Variables and Data Types**: Python accommodates a range of types, such as integers, float, strings, and boolean types.
- **Control Constructs:** Python employs conditional statements (if, elif, else) and loop constructs (while, for) for controlling a program's flow.
- **Functions:** Python functions are blocks of codes that can be reused and have specific operations performed in them. Python functions can have an argument and yield a value.
- **Object-Oriented Programming (OOP):** Python adheres to OOP constructs, and one can develop classes and objects through Python.
- **Data Structures:** Python employs lists, tuples, and dictionaries for storing and manipulating groups of information.

II. While this lab only focused on basic python programming, the root concepts are extremely essential for applications in biorobotics/cybernetics. For Example:

- Data Analysis: With the processing and analyzing physiological signals, such as ECG, EMG, EEG data, data analysis is increasingly performed using Python.
- Programming: Python is a programming language that can be used for tasks such as programming data collection or signal processing.
- Embedded Systems: Python is ideal to use to program robotic systems and prosthetic devices control systems.

---

4) Methodology

How did you build the sensor data collection?

I did not work with the actual sensor data in this lab. I was able to understand the language of basics by writing Python codes in Visual Studio code (VS Code). As we were working with Python for the first time, there was no working with hardware sensors or data collection processes. The aim was to use a development environment where I could practice writing and executing Python scripts.

How were the data cleaned and analyzed?

As this lab introduced users to python they could not work with a real world dataset or sensor data that would need to be cleaned or preprocessed. But I was practicing fundamental coding concepts that are usually part of data analysis workflow like variables, variables, looping, if, functions, classes, etc and data structures (lists, tuples, dictionaries).

Preprocessing Techniques: Despite not being able to work on any real world datasets, I utilized basic programming techniques that would help in performing data preprocessing tasks. For example:

- How to filter and control the program flow.
- Closely matches the loops to iterate through sequence like iterated data points in dataset.
- MODIFY data using text based format.

Feature Extraction Techniques (if applicable): No feature extraction took place in this lab as focus was on Python fundamentals than Data science or Machine learning. I did practice working with lists and dictionaries which are used to store and manipulate structured data in later stages of data analysis.

3. **Machine Learning (if applicable):** This introductory lab did not go into machine learning. The essence of the usage was to learn the basics of python syntax, work with various data types and be familiar with the core programming concepts that goes as the primal ground of data science and AI.

---

5) Results

A)

Exercise 1: Hello, World!

```
print("Hello, World!")
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
hello world
```

Explanation:

This is the simplest Python program. The text or output is displayed or displayed to the console by using the print() function. Here, when used in this case, it will print "Hello, World!". Typically, this is the first exercise when learning any programming language because it teaches the syntax of outputting text.

Exercise 2: Variables and Data Types

```
name = "Alice"
  age = 25
  is_student = True

  print(name, age, is_student)
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
Alice 25 True
```

Explanation:

This is an exercise of how to declare and use variables in python. Data is stored in variables and each variable has its data type:

- Value of 'Alice' is stored in string str whose name is name.
- Age is an integer value (int) with the value of 25.
- On the other hand the is_student is a boolean (bool) and will remain True.
- Values of these variables are printed using print() function, isolated in the next line by spaces.

Exercise 3: Taking User Input

```
name = input("Enter your name: ")
print("Hello, " + name + "!")
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
Enter your name: Youssief Khalifa
Hello, Youssief Khalifa!
```

Explanation:
With regard to this exercise, the input() function facilitates the ability of the program to take input from the keyboard. The variable name receives input from the user asking for his or her name. Then, the program concatenates the input with the string 'Hello, ' and prints the result. Let's say the user inputs "Alice" and the output will be "Hello, Alice".

Exercise 4: Conditional Statements

```
four = 4
five = 5
six = 6
```

a) IF statements :

```
if five >= four:
    print ('five is bigger than four')
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
five is bigger than four
```

```
if five < four:
    print ('five is bigger than four')
else:
    print ("That can't be true")
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
That can't be true
```

```
if five < four:
    print ('five is bigger than four')

elif six > 5:
    print ('6 is bigger than 5')
```

```
else:
   print ("That can't be true")
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
6 is bigger than 5
```

Explanation:

- The above exercise shows you how to control the flow of the program using conditional statements (if, elif, else) in case of specific conditions defined.
- First of all we can present first if statement: Is five or bigger than or equal four? Since this is indeed true, it prints "five is bigger than four".
- The second if statement checks if five is less than four. Since this is not true, the else part of the conditional executes and prints 'That can't be true'.
- The third if-elif-else statement has multiple conditions to be checked. As six > 5 is true, it prints "6 is bigger than 5".

Exercise 4: Conditional Statements (WHILE Statements)

```
counter = 0

while counter <= 5:
   print(counter)
   counter += 1
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
0
1
2
3
4
5
```

Explanation:

- This code epeatedly loop returns the code block until the set condition is true. In this example:
- A variable counter is initialized to 0 initially.
- In the while loop, counter is less than or equal to 5. The body of the loop runs if the condition is true.
- Inside the loop: We print the current value of counter using print(counter)
- In counter += 1, counter is incremented by 1.
- The loop runs until the counter is 6, and since counter <= 5 is false, the loop ends.

Exercise 4: Conditional Statements (FOR LOOP Statements)

```
for number in range(6):
```

```
print(number)
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
0
1
2
3
4
5
```

Explanation:

- An iteration is done in a for loop where we loop over a sequence, such as a list, tuple or range. In this example:
- In the range(6) function the sequence of numbers are 0 to 5 inclusive of 0 but not 6.
- The loop body is executed once for each value, and the variable number is assigned each number in the sequence.
- It then prints the current value of number inside the loop by print(number).

Exercise 5: Functions

```
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))
```

```
yousef@Yousefs-MacBook-Pro-2 ~ % /usr/local/bin/python3 /Users/yousef/Desktop/Cybernetics.py
Hello, Alice!
```

Explanation:
This tutorial introduces functions, reusable blocks of code, which are so useful that Java makes heavy use of them. greet() takes one parameter of name then returns a formatted string “Welcome [name]”. The result is then outputted from the print() function by the result of calling greet("Alice"), or "Hello, Alice!"

Exercise 6: Functions with Arguments

```
import random

def make_fun(sentence):
    '''
    This function makes fun of you.
    '''
    mocked_sentence = ''
    for letter in sentence:
        rand = random.randint(1,50)
        if rand % 3:
            mocked_sentence += letter
```

```
        else:
            mocked_sentence += letter.upper()

    return mocked_sentence

fardin_intro = 'Hi guys, This is a python workshop and I will be teaching you how to code in python today'

make_fun(fardin_intro)
```

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\AI expo\final proj\vehicleImages> & "C:/Users/ghena abo kalam/anaconda3/envs/python/python.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/test.py"
HI GUys, This Is A pythON workshop aNd I will be teAchINg YoU HOw to cODe in PytHon todAy
```

Explanation:

This exercise shows a function that modifies a string. The sentence that is given as input to the make_fun() function goes through the randomizer within in rhyme() and some random letters are capitalized based on a random number generator. The random.randint(1, 50) produces a random integer from 1 to 50, while the condition in the if statement is if the number is divisible by 3, capitalizing the letter. The final result is a fun, combined case of the input sentence.

Exercise 7: Classes

```
# Basic class definition.
class Square:
    def __init__(self,breadth,width):
        self.breadth = breadth
        self.width = width

    def area(self):
        return self.width * self.breadth

    def perimeter(self):
        return 2*(self.width) + 2*(self.breadth)

my_square = Square(2,4)
my_square.breadth
my_square.width
my_square.area()
my_square.perimeter()
```

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\AI expo\final proj\vehicleImages> & "C:/Users/ghena abo kalam/anaconda3/envs/python/python.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/test.py"
Breadth = 2
Width = 4
Area = 8
Perimeter = 12
```

Explanation:

This exercise teaches the implementation of the object oriented programming (OOP) in Python. We define a class Square which has __init__ method (constructor) to initialize the breadth and width attributes on the object. Additionally, it also contains two methods of the class.

- Area is a function to calculate the area of the square.
- The perimeter of the square is calculated by perimeter().
- Then, the code defines an instance of the Square class with breadth = 2, widen = 4, and calls the methods to calculate and print the area, and perimeter.

Exercise 8: Lists

```
list_a = [1,2,3]
list_b = [int_a,float_a,string_a,list_a]

list_b
#Lists are mutable
list_b + list_a
# Accesing an item in the list
list_b[3]
```

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\AI expo\final proj\vehicleImages> & "C:/Users/ghena abo kalam/anaconda3/envs/python/python.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/test.py"
[1, 2.0, 'My name is majd', [1, 2, 3], 1, 2, 3]
[1, 2, 3]
```

Explanation:
With our list_a list we have a list of integers [1, 2, 3] and list_b list, which contain a slight mix of variables and list_a. list_b plus list_a is made, but list_b[3] gets the fourth element of list_b and it is list_a.

Exercise 9: Tuples

```
tuple_a = (1,2,3,4,5,'a')
# Unpacking the tuple
one,two,*three_and_more = tuple_a
one
two
three_and_more
```

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\AI expo\final proj\vehicleImages> & "C:/Users/ghena abo kalam/anaconda3/envs/python/python.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/test.py"
1
2
[3, 4, 5, 'a']
```

Explanation:
Tuples are ordered, immutable collections of items that are introduced in this exercise. The tuple_a has a mix of integer and string. In this case, line one, two, *three_and_more = tuple_a

is unpacking the tuple assigning first two to one and two and the rest of the values to three_and_more as a list.

Exercise 10: Dictionaries

```
dict_a = {
        'EVERYTHING':'NOTHING',
        100:'Still nothing :P',
        'again':[0,0,0,0,-float('inf')]
    }
#Access one
dict_a['EVERYTHING']

#Try again
dict_a[100]

#Again
dict_a['again']
```

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\AI expo\final proj\vehicleImages> & "C:/Users/ghena abo kalam/anaconda3/envs/python/python.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/test.py"
NOTHING
Still nothing :P
[0, 0, 0, 0, -inf]
```

Explanation:

Dictionaries are a type of collections of key value pairs, and this exercise demonstrates the same. A string, another string and a list of string is associated to each key. These print() statements call and display the values of keys.

B)

We decided to include all the screenshots as we felt like that all figures and codes were necessary to include.

6) References

GeeksforGeeks. (2023). Python Programming Language.
https://www.geeksforgeeks.org/python-programming-language/

Real Python. (2023). Object-Oriented Programming (OOP) in Python.
https://realpython.com/python3-object-oriented-programming/

W3Schools. (2023). Python Tutorial.

https://www.w3schools.com/python/