MACHINE LEARNING LAB I

Introduction to Python

MUNADI SIAL

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

Code of Ethics

- All students must come to the lab on time
- Students must remain attentive and avoid use of mobile phones
- Respect peers and faculty through speech and actions
- Students should not be sleeping during the lab
- Discussion unrelated to lab is NOT allowed
- Eatables are NOT allowed in the lab
- Late submission of lab reports will be subjected to penalty
- Copying of lab reports will NOT be tolerated
- Sharing of code is NOT permitted

Lab Evaluation

		PLO4 - CLO4	PLO4 - CLO4	PLO5 - CLO5	PLO8 - CLO6	PLO9 - CLO7
Name	Reg. No	Viva /Quiz / Lab Performance 5 Marks	Analysis of data in Lab Report 5 Marks	Modern Tool Usage 5 Marks	Ethics 5 Marks	Individual and Team Work 5 Marks

Python Language

- Python is an interpreted, open-source language with an emphasis on code readability and has a very large community support
- Consider the following comparison between C++ and Python codes:

Python	**C++**

`a = 10`
`b = "M"`
`c = 7.6`
`print(a,b,c)`
	`int a = 10;`
	`char b = "M";`
	`float c = 7.6;`
	`cout << a << " " << b << " " << c << endl;`

- The output of both programs is the same:

```
>> 10 M 7.6
```

- No semicolons are needed to end each line in Python
- It is not necessary to declare the data type of variable in Python
- The print() statement creates spaces and new line automatically

Python Scripts

- Python scripts are files that end in **.py** format and contain the code

- Let's write a simple python script and execute it via the terminal:

```
a = 10
b = "M"
c = 7.6
print(a,b,c)
```

- Once you have typed the above code, save the script as **test.py**
- Now open the terminal, go the directory (with cd) where you saved the script and execute it with the following command:

```
python test.py
```

Variables in Python

- To add a comment in code, use # in Python
- The data type of a variable can change in Python:

```
a = 10          # this will work
a = 5.5         # this will work
a = "house"     # this will work
```

- The equivalent code in C++ will give an error:

```
int a = 10;     // this will work
a = 5.5;        // error
a = "house";    // error
```

Data Types in Python

- Python contains several data types, some of which are given below:

```
x_string = "Hello World"
x_integer = 25
x_float = 25.1
x_boolean = True
x_complex = 2j
x_list = [1, 2, 3, 4, 5]
x_tuple = (1, 2, 3, 4, 5)
x_set = {1, 2, 3, 4, 5}
x_dictionary = {"name":"Ali", "age":31}
```

Arithmetic Operators

- The following code shows the arithmetic operators in Python:

```
a = 10
b = 3

add = a + b
sub = a - b
mul = a*b
div = a/b
quotient = a//b
remainder = a%b
power = a**b

print(add, sub, mul, div, quotient, remainder, power)
 >> 13 7 30 3.3333333333333335 3 1 1000
```

Logical Operators

- The following code shows the logical operators in Python:

```
a = True
b = False

print(a and b)
>> False

print(a or b)
>> True

print(not a)
>> False

print(not(a and b))
>> True

print(not(a or b))
>> False

print(not b)
>> True
```

Strings

- Python supports the string data type which is an array of characters

```
h = "Manipulator"
print(h)
```

Output:

```
Manipulator
```

- You can get individual characters with square brackets

```
print(h[0])
print(h[1])
print(h[8])
```

Output:

```
M
a
p
```

- You can get the number of characters with the **len()** function

```
g = len(h)
print(g)
```

Output:

```
11
```

Strings

- You can concatenate strings easily in python

```
c = "Computer"
d = "Vision"
e = c + d
print(e)
```

Output:

```
ComputerVision
```

```
f = c + " " + d
print(f)
```

Output:

```
Computer Vision
```

- You can check if a character is present in the string with the "in" keyword

```
print("t" in c)
print("s" in c)
```

Output:

```
True
False
```

User Input

- To get input from user, we use the input() function

```
v1 = input("Enter the first number: ")
print(v1)

v2 = input("Enter the second number: ")
print(v2)
```

Output:

```
Enter the first number: 200
200
Enter the second number: 100
100
```

User Input

- The **input()** function returns a string data which is stored in the variable
- If we add the previous two numbers, they will concatenate because they are strings, not numbers

```
print(v1 + v2)
```

Output: `200100`

- To compute the numeric sum, we need to convert the string data type to an int or float data type

```
v1 = int(v1)
v2 = int(v2)
print(v1 + v2)
```

Output: `300`

If…Else

- Python supports the IF…ELSE conditional statements which choose to execute statements depending if a condition is true or false
- The syntax for a IF statement is given as:

```
if <condition>:
   <statement_1>
   <statement_2>
```

- An example is given below:

```
a = 4
b = 3
if a > b:
   print("a is greater than b")
```

- The <condition> can be any of the following:

```
a==b     a!=b     a < b     a > b     a <= b     a >= b
```

If…Else

- The **else** keyword contains statements that execute when the condition is not met

```
a = 4
b = 3
if a > b:
   print("a is greater than b")
else:
   print("b is greater than a")
```

- Note that the colon symbol is part of the syntax
- To contain statements, they are indented with spaces
- The indents are mandatory in Python to define the block of statements

If…Else

- The **elif** keyword nests an if command inside an else statement
- It is the equivalent of **else if**

```
a = 4
b = 3
if a > b:
   print("a is greater than b")
elif (a < b):
   print("a is less than b")
else
   print("a is equal to b")
```

- Only one of the three print statements will be executed

Loops

- Python has two types of loops: WHILE and FOR
- The **while** loop has the following syntax:

```
while <condition>:
    <statement_1>
    <statement_2>
```

- The while loop checks the condition. If the condition is true, the statements are executed. After executing, the condition is rechecked. If it is true, it is executed again. This continues until the condition becomes false at which point the loop ends
- An example is given below

```
a = 0
while a < 5:
    print(a)
    a = a + 1
```

Output:

```
0
1
2
3
4
```

Loops

- The **break** keyword terminates the loop in which it is placed:

```
a = 0
b = 3
while a < 5:
    print(a)
    a = a + 1
    if a == b:
        break
```

Output:

```
0
1
2
```

Loops

- The **continue** keyword skips the current iteration of the loop:

```
a = 0
b = 3
while a < 5:
    a = a + 1
    if a == b:
        continue
    print(a)
```

Output:

```
1
2
4
5
```

Loops

- The **for** loop goes through a sequence of items (iterable object)
- We use the **range()** to create a sequence of numbers

```
for i in range(0,5):
    print(i)
```

Output:

```
0
1
2
3
4
```

```
for items in range(0,10,2):
    print(items)
```

Output:

```
0
2
4
6
8
```

Loops

- The **for** loop can also go through a sequence of characters
- The iterable object is a string variable in this case

```
for i in "PYTHON":
    print(i)
```

Output:

```
P
Y
T
H
O
N
```

Functions

- We can define and call functions in python

```
def my_function(a, b):                    # Function Definition
    out = a + b
    print(a, "+", b, "=", out)
    return out

value = my_function(3,4)                  # Function Call
print("value =", value)
```

Output:

```
3 + 4 = 7
value = 7
```

Lab Tasks

- Download the materials from LMS
- Perform the Lab Tasks given in the manual
- Convert the completed manual into .pdf and submit on LMS