EAST WEST UNIVERSITY

Lab Report-02

Course Title: Artificial Intelligence

Course Code: CSE 366

Semester: Fall 2021

Section No: 01

Submitted By

Name: Syeda Tamanna Sheme

ID:2018-2-60-010

Submitted To

Md Al-Imran

Lecturer

Department of Computer Science & Engineering

Date of Submission: 18 November, 2021

Lab Number: 02

Theory

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV Show called ëMonty Pythonís Flying Circusí and not after Python-the snake.

Python 3.0 was released in 2008. Although this version is supposed to be backward incompatibles, later on many of its important features have been backported to be compatible with version 2.7.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable.

Characteristics of Python

Following are important characteristics of python −

- It supports functional and structured programming methods as well as OOP.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is one of the most widely used language over the web.

Lists, Tuples, Set, NumPy, and Matplotlib

List, Tuple, Set, and Dictionary are the data structures in python that are used to store and organize the data in an efficient manner.

Lists: are just like dynamic sized arrays, declared in other languages (vector in C++ and ArrayList in Java). Lists need not be homogeneous always which makes it a most powerful tool in Python.

Tuple: A Tuple is a collection of Python objects separated by commas. In someways a tuple is similar to a list in terms of indexing, nested objects and repetition but a tuple is immutable unlike lists that are mutable.

Set: A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements. Python's set class represents the mathematical notion of a set.

Dictionary: in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key value pair. Key value is provided in the dictionary to make it more optimized.

NumPy: "Numeric Python" or "Numerical Python" is known as NumPy. NumPy is an opensource module of Python. It provides precompiled functions for mathematical and numerical routines. Therefore, in this lab we will learn the most effective way to handle arrays in Python.

Matplotlib: is a library for making plots of arrays in Python. Although Matplotlib is written primarily in pure Python, it makes heavy use of NumPy and other extension code to provide good performance even for large arrays.

Codes

##Strings in Python

```
# 1. Strings in Python
print("Hello, World!")
print('My name is Tamanna.')
print("Hello Tamanna !!")
print('!!!!12345!!!!')
print('This is CSE 225 First Lab Report!!!!')
print('Python Basics Lab')
```

Comment: Strings in python are surrounded by either single quotation marks, or double quotation marks. **'hello'** is the same as **"hello".**

We can display a string literal with the **print()** function.

Output

```
Hello, World!
My name is Tamanna.
Hello Tamanna!!
!!!!12345!!!!
This is CSE 225 Ist Lab Report!!!!
Python Basics Lab
```

Code

```
#Strings in Python

print('python' + 'is' + 'easy')
print('python' + 'is' + 'fantastic')
print('JAVA' *2)
print('2' *5)
print('3' *30)
print('100' + 'TAKA')
```

Comment: We can add two or more string using **(+)** operation. Also, we can do **multiply (*) operation** and multiply a string.

Output

```
pythoniseasy
pythonisfantastic
JAVAJAVA
22222
333333333333333333333333333333
100TAKA
```

Code

```
# Variable with Strings in Python

a = 'x' print(a)

b ='FOOTBALL'
print(b)

c = 'd' p = c print(p)

n = 'I am Tamanna.'
print(n)

name = 'EAST WEST UNIVERSITY' name =
'COMPUTER SCIENCE AND ENGINEERING'
print(name)

message1 = 'WELCOME TO'
message2 = 'CSE 225 LAB CLASS'
print (message1 + message2)
```

Comment: Assigning a string to a variable is done with the **variable name** followed by an **equal sign** and the string. If we have **two variables** in same name then the **last variable** will print.

Output

```
x
FOOTBALL
d
I am Tamanna.
COMPUTER SCIENCE AND ENGINEERING
WELCOME TOCSE 225 LAB CLASS
```

Code

```python
# Taking Input from USERS in Python.

print('Hello there! What is your name?')
Name = input()
print('Nice to meet you' + Name + '!')
print('How old are you,' + Name + '?')
age = input()
print('So, your age is ' + age)
```

Comment: Python allows for user input. That means we are able to ask the user for input. Python uses the **input()** method.

Output

```
Hello there! What is your name?
Tamanna
Nice to meet you Tamanna!
How old are you, Tamanna?
20
So, your age is 20
```

```
#  String length

   a =' jackpot'
   print(len(a))

#String traversal

for a_name in ["Joe", "Amy", "Brad",
"Angelina", "Zuki", "Thandi", "Paris"]:
invitation = "Hi " + a_name + ".  Please
come to my party on Saturday!"
print(invitation)

a = "STRING"
i = 0 while i < len(a):
    c = a[i]
print(c)
    i = i + 1
```

Comment: To get the length of a string, use the **len()** function.

For strings this means that we would like to process one character at a time. Often we start at the beginning, select each character in turn, do something to it, and continue until the end. This pattern of processing is called a **traversal**. We can use **for or while** for doing traversal operation.

Output

```
8
Hi Joe.  Please come to my party on Saturday!
Hi Amy.  Please come to my party on Saturday!
Hi Brad.  Please come to my party on Saturday!
Hi Angelina.  Please come to my party on Saturday!
Hi Zuki.  Please come to my party on Saturday!
Hi Thandi.  Please come to my party on Saturday!
Hi Paris.  Please come to my party on Saturday!
S T R I N G
```

# String Slicing

```
b = "Hello, World!"
print(b[2:5])

b = "Hello, World!"
print(b[:5])

b = "Hello, World!"
print(b[2:])

b = "Hello, World!"
print(b[-5:-2])
```

Comment: You can return a range of characters by using the **slice** syntax.

Specify the **start index** and the **end index**, separated by **a colon,** to return a part of the string.

By **leaving out the start index,** the range will start at the first character.

Use **negative indexes** to start the slice from the **end of the string.**

Output

```
llo
Hello
llo, World!
Orl
```

# 'in' operator, looping and counting

```
txt = "The best things in life are free!"
print("free" in txt)

txt = "The best things in life are free!" if "free"
in txt:   print("Yes, 'free' is present.")

txt = "The best things in life are free!"
print("expensive" not in txt)

txt = "The best things in life are free!" if
"expensive" not in txt:
  print("Yes, 'expensive' is NOT present.")
```

Comment:

Returns **True** if a sequence with the specified value is **present in the object-(in)**

Returns **True** if a sequence with the specified value is **not present** in the object- (**not in**)

Output

```
# “in” operator-output
True
Yes, 'free' is present.
True
Yes, 'expensive' is NOT present.
```

#looping – While loop

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

y = 8 while y <= 10:
print(y)     y = y+1

m = 11
while m <= 20:
print(m)     m = m + 2
 z = 0
while z <= 20:
   print('PYTHON LAB TASK')
   z = z + 3
```

Comment: Python has two primitive loop commands-**(1)** **while** loops **(2)** **for** loops

While Loop: With the **while** loop we can execute a set of statements as long as a condition is true.

The **while** loop requires relevant variables to be ready, in this example we need to define an indexing variable, **i,** which we set to **1.**

Output:

```
#  While loop output
0  1  2  3  4  8  9  10  11  13  15  17  19
PYTHON LAB TASK
PYTHON LAB TASK
PYTHON LAB TASK
PYTHON LAB TASK
PYTHON LAB TASK
PYTHON LAB TASK
PYTHON LAB TASK
```

```
##looping – For loop

for i in range(1,4):
        print('LOCKDOWN')

for i in range(1,4,2):
print('SHUTDOWN')

for i in range(5,-1,-1):
        print(i)

for i in range(1,10,4):
        print('GOOD MORNING')
```

Comment: A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the **for** keyword in other programming languages, and works more like **an iterator method** as found in other object-orientated programming languages.

With the **for loop** we can execute a set of statements, once for each item in a list, tuple, set etc.

The **for loop** does not require an **indexing variable** to set beforehand.

Output

```
LOCKDOWN
LOCKDOWN
LOCKDOWN
SHUTDOWN
SHUTDOWN
5  4  3  2  1  0
GOOD MORNING
GOOD MORNING
GOOD MORNING
```

Code

```
# USE OF INFINITE LOOP AND BREAK STATEMENT

while True:
    print('Please enter your name: ')
name = input()
 if name == 'SUPERMAN':
        break
    print('THANK YOU')
```

Comment: With the **break** statement we can stop the loop before it has looped through all the items.

Output

```
#  LOOP AND BREAK STATEMENT OUTPUT

Please enter your name:

SHEME

THANK YOU

Please enter your name:

SUPERMAN
```

Code

```
# conditional Statements in Python

print('Enter your command: ')
robot_move = input()

if robot_move == 'front':
print('Moving Front')

elif robot_move == 'back':
 print('Moving Back')

else:
print('Stand Still')
```

Comment: Python supports the usual logical conditions from mathematics:

- Equals: **a == b**
- Not Equals: **a != b**
- Less than: **a < b**
- Less than or equal to: **a <= b**
- Greater than: **a > b**
- Greater than or equal to: **a >= b**

These conditions can be used in several ways, most commonly in **"if statements"** and loops.

Output

```
Enter your command:

front

Moving Front

Enter your command:

back

Moving Back

Enter your command:

Cake

Stand Still
```

```
# Comparison operator
a = 21 b = 10
c = 0

if ( a == b ):    print ("Line 1 - a is equal to b")
else
   print ("Line 1 - a is not equal to b")
             if ( a != b ):
   print ("Line 2 - a is not equal to
b") else:    print ("Line 2 - a is
equal to b")

if not b == a:    print ("Line 3 - a is
not equal to b") else:    print ("Line
3 - a is equal to b")

if ( a < b ):
   print ("Line 4 - a is less than b")
else:    print ("Line 4 - a is not less
than b")

if ( a > b ):
   print ("Line 5 - a is greater than b")
else:    print ("Line 5 - a is not greater
than b")

a = 5; b = 20; if ( a <= b ):    print ("Line 6 - a is either
less than or equal to  b") else:    print ("Line 6 - a is
neither less than nor equal to  b")

if ( b >= a ):
   print ("Line 7 - b is either greater than  or equal to b")
else:    print ("Line 7 - b is neither greater than nor
equal to b")
```

Comment: Comparison operators are used to **compare two values.**

There are many comparison operators in python. Some of them are:

==, !=, <>, <, >, <=, >=

Output

```
Line 1 - a is not equal to b
Line 2 - a is not equal to b
Line 3 - a is not equal to b
Line 4 - a is not less than b
Line 5 - a is greater than b
Line 6 - a is either less than or equal to  b
Line 7 - b is either greater than  or equal to bCode
```

#String Parse

```
original_string = "ab_cd_ef"
 split_string = original_string.split("_")
print(split_string)

a_string = "ab_cd_ef"
parsed_string = a_string.split("_", 1)
print(parsed_string)
```

Comment: Parsing a string splits the string into substrings by a specified delimeter.

Call **str.split(sep)** to parse the **string str** by the **delimeter sep** into a list of strings.

Output

```
#String Parsing Output
['ab', 'cd', 'ef']
['ab', 'cd_ef']
```

#Lists

```
thislist = ["apple", "banana", "cherry"]
print(thislist)

thislist = ["apple", "banana", "cherry", "apple",
"cherry"]
print(thislist)

list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]

thislist = list(("apple", "banana", "cherry"))
print(thislist)

tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)

mytuple = ("apple", "banana", "cherry")
print(type(mytuple))
```

Comment: Lists are used to store **multiple items** in a **single variable**.

Lists are one of **4 built-in data** types in Python used to store collections of data, **the other 3 are Tuple, Set, and Dictionary**, all with different qualities and usage.

Lists are created using square brackets.

List items are **ordered, changeable, and allow duplicate** values.

List items are indexed, the first item has **index [0],** the second item has **index [1] etc.**

Output

```
#Lists Output
['apple', 'banana', 'cherry']
['apple', 'banana', 'cherry', 'apple', 'cherry']
['apple', 'banana', 'cherry']
<class 'tuple'>
```

#Operation

```
a = 21 b = 10 c = 0

c = a + b
print ("Line 1 - Value of c is ", c)

c *= a
print ("Line 3 - Value of c is ", c)

c /= a
print ("Line 4 - Value of c is ", c)

c **= a
print ("Line 6 - Value of c is ", c)

c //= a
print ("Line 7 - Value of c is ", c)
```

Comment: Operators are the constructs which can manipulate the value of operands.

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Output

```
#Operation Output
Line 1 - Value of c is  31
Line 3 - Value of c is  1092
Line 4 - Value of c is  52.0
Line 6 - Value of c is  2097152
Line 7 - Value of c is  99864
```

More Example:

Code

```
print("This is CSE366-ARTIFICIAL INTELLIGENCE.\
      Fall 2021section 1.\
      Total students is 36.")

a = 10
print(a)
print(type(a))

b = 10.10
print(b)
print(type(b))

c = "Painkiller"
print(c)
print(type(c))

f = 0B10
print(f)


g = 0X10
print(g)

h = 0B10
print(f)

i = 0O10
print(i)
```

Output

```
This is CSE366-ARTIFICIAL INTELLIGENCE.
Fall 2021section 1.
Total students is 36.
10
<class 'int'>
10.1
<class 'float'>
Painkiller
<class 'str'>
2
16
2
8
```

More Example:

Code:

```
import numpy

a = 2 ** 52 <= 2**56 //10
print(a)

season = "What is coming"
length = len (season)
print(length)
print(type(length))
last = season[length-3]
print(last)
middle = season[length-8]
print(middle)

for char in season:
    print(char)


ch = 0
while (ch < length):
    print(season[ch])
    ch = ch+1

print(season[6:])
print(season[:4])
#without space
string = "Good Morning. This is 366.";
count = 0;
for i in range(0, len(string)):
    if(string[i]!= ' '):
        count = count + 1;

print("Total number of characters in a string: " + str(count));

#with space
string = "Good Morning. This is 366.";
count = 0;
for i in range(0, len(string)):

        count = count + 1;

print("Total number of characters in a string: " + str(count));

new = season.upper()
print(new)

cricket = "banglasesh lost the game"
print(cricket.strip)
```

```python
a = [12,10,34,90,30]
print(a)

mixed = ['sakib', 'sheme', 10.79, 20]

a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")


a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")

loc = "bank"
if loc =="auto shop":
    print("welocome to the auto shop")

elif loc == "bank":
    print("welcome to the bank")
else:
    print("where are u?")
```

Output

```
True    14    <class 'int'>

I   s  W  h  a  t

 I  s  c  o  m  I  n  g W h a t  I  s   c  o

m I n g s

coming

What

Total number of characters in a string: 22

Total number of characters in a string: 26

WHAT IS COMING

<built-in method strip of str object at 0x0000024A78E3D080>

[12, 10, 34, 90, 30]

a and b are equal

a is greater than b

welcome to the bank
```

Exercise

#1. Write a program in Python to find the root of a quadratic equation.

```
import math

a = 1
b = 5
c = 6
d = (b**2) - (4*a*c)

# find two solutions
sol1 = (-b-cmath.sqrt(d))/(2*a)
sol2 = (-b+cmath.sqrt(d))/(2*a)

print('The solution are {0} and {1}'.format(sol1,sol2))
```

Output

```
The solution are (-3+0j) and (-2+0j)
```

#2. Write code to perform grade computation

```
marks = float(input("Enter your marks in Computer Science: "))

if marks > 90:
   print("Grade: A+")
elif marks >= 80 and marks <= 90:
   print("Grade: A+")
elif marks >= 70 and marks < 80:
   print("Grade: A")
elif marks >= 60 and marks < 70:
   print("Grade: B+")
elif marks >= 50 and marks < 60:
   print("Grade: B")
elif marks >= 40 and marks < 50:
   print("Grade: C")
else:
   print("Grade: F")
```

Output

```
Enter your marks in Computer Science: 90
Grade: A+
```

```
"""3. Given two numeric lists or tuples x_vals and y_vals of equal
length, compute their inner product uzing zip(). Additionally count
the number of even number in 0 to 99. Furthermore given pairs =
((4, 5), (6,7), (8,9)) count the number of pairs (x,y) such that a and b
are odd."""

# Part 1
x_vals , y_vals = (1,2,3),(4,5,6)
sum = 0
for x,y in zip(x_vals,y_vals):
    sum = sum + x*y
print("The inner product is:", sum)

# Part 2
print(len([n for n in range(0,100) if n%2==0]))

# Part 3
pairs = ((4,5),(6,7),(8,9))
count = 0
for i in pairs:
    if i[0]%2 == 0 and i[1]%2 == 0:
        count = count + 1
print(count)
```

Output

```
The inner product is: 32

50

0
```

Results

After doing the lab task and lab work we are able to learn -Strings in Python, String length and String traversal, String Slicing, ‘in’ operator, looping and counting, Comparison operator, String Methods, parsing, Lists and operation, Variables, expressions, and statements Conditional Executions, Functions, Loops. Now we are able to do programs related to these topics. After doing the lab task, I did some problems related to my task. Now we are able to solve many problems.

Discussion

Python is a general-purpose, versatile and popular programming language. It’s great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer’s stack as it can be used for everything from web development to software development and data science applications.

This lab task is a great introduction to both fundamental programming concepts and the Python programming language. Python 3 is the most up-to-date version of the language with many improvements made to increase the efficiency and simplicity of the code that we write. Dayby-day, python new version are realising and all the new versions have new features and these new features are better than previous one. So, finally I can say that python will be more user friendly in future.