Faculty of Engineering and Technology

School of Computer Engineering and Technology

Cybersecurity and Forensics

Python Programming Lab Assignment No:01

Problem Statement:

Introduction to basic Python commands.

Aim:

To learn the basics of the python programming language and understand fundamental syntax and semantics of Python Programming

Objectives:

1. To learn the basics of the Python programming language.
2. To learn the Variable declaration, User input and output of the Python programming language

Theory:

- Introduction to Python
- Basic Commands in python:
    - Variable declaration: Assigning Values to Variables, Multiple Assignment
    - Arithmetic operations: Python Numbers, Python Strings
    - User input commands
- **Assigning Values to Variables**

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
print counter
print miles
print name
```

Here, 100, 1000.0 and "John" are the values assigned to *counter*, *miles*, and *name* variables, respectively. This produces the following result −

```
100
1000.0
John
```

- **Multiple Assignment**

Python allows you to assign a single value to several variables simultaneously.
For example −

```
a = b = c = 1
```

For example −

```
a,b,c = 1,2,"john"
```

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

- **Standard Data Types**

The data stored in memory can be of many types. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

- **Python Numbers**

Number data types store numeric values. Number objects are created when you assign a value to them. For example −

```
var1 = 1
var2 = 10
```

Python supports four different numerical types −

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)

Here are some examples of numbers −

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j

080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

- Python allows to use a lowercase l with long, but it is recommended that you use only an uppercase L to avoid confusion with the number 1. Python displays long integers with an uppercase L.
- A complex number consists of an ordered pair of real floating-point numbers denoted by x + yj, where x and y are the real numbers and j is the imaginary unit.

• Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
The *plus (+)* sign is the string concatenation operator and the asterisk (*) is the repetition operator. For example –

This will produce the following result:

```
str1 = 'Hello World!'

>>print str1         # Prints complete string
>> Hello World!

>>print str1[0]      # Prints first character of the string
>>H

>>print str1[2:5]    # Prints characters starting from 3rd to 4th
>>llo

>>print str1[2:]     # Prints string starting from 3rd character to end of the string
>> llo World!
```

```
>>print str1[:3]          #Prints from starting of the string till index '2'
>> Hel

>>print str1 * 2      # Prints string two times
>> Hello World!Hello World!

>>print str1 + "TEST" # Prints concatenated string
>> Hello World!TEST
```

- **User Input**

While programming, we might want to take the input from the user. Taking input is a way of interact with users, or get data to provide some result. In Python two built-in methods are provided to read the data from the keyboard, the **input()** and **raw_input()** function.

- **input():** Python provides a simple framework for getting user input in the form of the *input()* function. ***The input function reads a line from the console, converts it into a string, and returns it.*** When the *input()* function is called, **the program flow halts until the user enters some input**. The user then adds some information and presses the Enter key. The input function returns to the program with the entered string.

  - **Input Type**

The input function converts all information that it receives into the string format.
example 1:

```
a = input("Enter name: ")
print("You Entered Name:", a)
```

Output:

```
Enter name: John
You Entered Name: John
```

example2:

```
num = input("Enter a number: ")
print(You Entered Number:",num)
print(type(num))
```

Output:

```
Enter a number:10
You Entered Number:10
<class 'str'>
```

In the above example2, we have used the *input()* function to take input from the user and stored the user input in the *num* variable.It is important to note that the entered value **10** is a string, not a number. So, type(num) returns <class 'str'>.

○ Accept an integer input from the user

As input() function returns everything as a string, we need to perform some explicit **type-conversion** for accepting integers. Here, we will use the **int()** function, converted from string to integer .

For example:

```
num = int(input("Enter a number: "))
print("You Entered Number:",num)
```

Output:

```
Enter a number: 100
You Entered Number: 100
```

○ Accept a float input from the user

Similarly, we can get a float value using the **float()** function.

```
num = float(input("Enter a number: "))
print("You Entered Number:",num)
```

Output:

```
Enter a number: 6.6
You Entered Number: 6.6
```

○ Multiple input values in a single line

We can also ask for multiple values directly on a single line with only one call to the input() function. For example, let's get some information about a student from the user and store it in different variables. **split()**function helps in getting multiple inputs from users. It breaks the given input by the specified separator. If a separator is not provided then any white space is a separator. Generally, users use a split() method to split a Python string but one can use it in taking multiple inputs.

example:

```
name, age, score = input("Enter student's name, age and score separated by space:").split()
print("Student Name:", name)
print("Student Age:", age)
print("Student Score:", score)
```

Output:

```
Enter student's name, age and score separated by space: John 20 99

Student Name: John

Student Age: 20

Student Score: 99
```

- **raw_input() -** The raw_input function is used in Python's older version like Python 2.x. It takes the input from the keyboard and return as a string. The Python 2.x doesn't use much in the industry.

Platform: Python
Input: ---

Output: ----

Conclusion:

Understood the basic commands, statements and syntax of python programming language.

FAQs:

1. What are the features of python language?
2. Explain print () statement of python language.
3. How to write single line and multi-line comments in python programming language.