UNIT 2

Python Hello World Program

"Hello world" is often considered the inaugural phrase uttered by programmers. Whether they are novices or seasoned professionals, everyone with coding experience has embarked on their journey with the timeless task of printing "Hello World." In this blog post, we will explore various methods to print hello world in Python.

"Hello world" Using print() in Python

To print hello world in Python, we will utilize the print() function available in Python. The print() method is employed to display Python objects as strings through the standard output.

Following is the syntax for the Python print function:

Syntax: print("string")

Where String: "Hello world" will be the string here.

- Python

```
print("Hello World")
```

Output

```
Hello World
```

Printing “Hello World” Using sys

The sys module in Python offers a number of functions and variables that are used to manage various elements of the Python runtime environment, and we’ll be using it to print strings in this method. Due to its accessibility to variables and functions that have a strong relationship with the interpreter, it enables working on the interpreter.

Stdout is used to show output to the screen console using sys.std.write().

Syntax: sys.stdout.write(“string”)

Program Input and the raw_input() Built-in Function

The easiest way to obtain user input from the command-line is with the raw_input() built-in function. It reads from standard input and assigns the string value to the variable you designate. You can use the int() built-in function (Python versions older than 1.5 will have to use the string.atoi() function) to convert any numeric input string to an integer representation.

```
>>> user = raw_input('Enter login name: ')
Enter login name: root
>>> print 'Your login is:', user
Your login is: root
```

The above example was strictly for text input. A numeric string input (with conversion to a real integer) example follows below:

```
>>> num = raw_input('Now enter a number: ') Now enter a number: 1024 ...
```

Variables and Assignment

When programming, it is useful to be able to store information in variables. A **variable** is a string of characters and numbers associated with a piece of information. The **assignment operator**, denoted by the "=" symbol, is the operator that is used to assign values to variables in Python. The line x=1 takes the known value, 1, and **assigns** that value to the variable with name "x". After executing this line, this number will be stored into this variable. Until the value is changed or the variable deleted, the character x behaves like the value 1.

```
x = 1
x
1
```

Strings, lists, and tuples

Strings in Python:

A string is a sequence of characters that can be a combination of letters, numbers, and special characters. It can be declared in python by using single quotes, double quotes, or even triple quotes. These quotes are not a part of a string, they define only starting and ending of the string. Strings are immutable, i.e., they cannot be changed. Each element of the string can be accessed using indexing or slicing operations.

```
# Assigning string to a variable
```

```
a = 'This is a string'

print (a)

b = "This is a string"

print (b)

c= '''This is a string'''

print (c)
```

Output:

```
This is a string
This is a string
This is a string
```

Lists in Python:

Lists are one of the most powerful data structures in python. Lists are sequenced data types. In Python, an empty list is created using list() function. They are just like the arrays declared in other languages. But the most powerful thing is that list need not be always homogeneous. A single list can contain strings, integers, as well as other objects. Lists can also be used for implementing stacks and queues. Lists are mutable, i.e., they can be altered once declared. The elements of list can be accessed using indexing and slicing operations.

```
# Declaring a list

L = [1, "a" , "string" , 1+2]
```

```
print L

#Adding an element in the list

L.append(6)

print L

#Deleting last element from a list

L.pop()

print (L)

#Displaying Second element of the list

print (L[1])
```

The output is:

```
[1, 'a', 'string', 3]
[1, 'a', 'string', 3, 6]
[1, 'a', 'string', 3]
a
```

Tuples in Python: A tuple is a sequence of immutable Python objects. Tuples are just like lists with the exception that tuples cannot be changed once declared. Tuples are usually faster than lists.

```
tup = (1, "a", "string", 1+2)
```

```
print(tup)
print(tup[1])
```

The output is :

```
(1, 'a', 'string', 3)
a
```