Introduction to PYTHON

Module 1 / Lecture-2

By: Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,

GLA University, Mathura

Topics

- input() and print() function
- Python Identifiers
- Python Statement, Indentation and Comments
- Quotation in Python
- Python Comments
- Docstring in Python
- Python Variables

input() function

>>>name **= input(“** Enter a value**”)**

Enter a value

- Characters within quotes are called strings.
- This particular use of a string, for requesting input from the user, is called a **prompt**.
- The input function displays the string on the screen to prompt the user for input.

Enter a value: Hello world

NOTE: *The **input function** always **returns** what the user enters as a **string** of characters.*

>>>**P** = **int(input(“**Enter a value **”))**

Enter a value 10

Basics of PYTHON

```
>>>print("Hello”)
Hello
>>>a=10
>>>print(a)
10
```

Basics of PYTHON

```
>>> name = input("What is your name?: ")
What is your name?:  Rossum

>>> print("Hello", name)
 Hello Rossum
```

- In Python,
  - **input()** is used to request and get information from the user, and
  - **print()** is used to display information on the screen.

Exercise

Print the following message on the console. Consider Name as a variable.

Hello I am *NAME* student of GLA University

Solution:

```
>>>name = input('enter  your name')
>>>print('Hello I am ', name, 'student of GLA university')
```

Python Identifiers

- A name used to identify
  - variable
  - function
  - class
  - module
  - object
- It starts with a letter
  - A to Z or
  - a to z or
  - an underscore (_)
- And it followed by zero or more letters, underscores and digits.

Naming conventions for Python identifiers

- **Class names** start with an **uppercase letter**.
- All other identifiers start with a lowercase letter.
- An identifier starting with **two leading underscores** and with **two trailing underscores,** indicates it is a language-defined special name.

Python Statement

Multi-line statement:

- We can make a statement extend over multiple lines with the line continuation character (\).

Example:

```
a = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

- This is **explicit line continuation**.

- Line continuation is implied inside parentheses ( ), brackets [ ] and braces { }.

Example:
```
a = (1 + 2 + 3 +
     4 + 5 + 6 +
     7 + 8 + 9)
```

Example:
```
colors = ['red',
          'blue',
           'green']
```

- We could also put multiple statements in a single line using semicolons

Example:

- a = 1; b = 2; c = 3

Python Indentation

- To define a block of code.
- *A code block (body of a **function, loop** etc.) starts with indentation and ends with the first un-indented line.*
- The amount of indentation is up to you, but
- It must be consistent throughout that block.
- Generally **four whitespaces** are used for indentation.

- **Example 1**

```
for i in range(1,11):
    print(i)
    if i == 5:
        break
```

- **Example 2**

```
if True:
    print('Hello')
a = 5
```

Quotation in Python

- Python accepts
  - **single (')**,
  - **double (") and**
  - **triple (''' or """) quotes**

  to denote string literals,
- The triple quotes are used to span the string across multiple lines.

For example:

>>>word = **'word'**

>>>sentence = **"This is a sentence."**

>>>paragraph = **"""This is a paragraph. It is made up of multiple lines and sentences."""**

Python Comments

- Hash (#) symbol or **“””multiline comment“””””** to start writing a comment.
- It extends up to the newline character.
- Python Interpreter ignores comment.

```
>>>#This is a comment
>>>#print out Hello
>>>print('Hello')
```

Doc-string in Python

- Documentation string.
- The first statement in a module, function, class, or method definition.
- We must write what a function/class does in the docstring.
- Triple quotes are used while writing docstrings.

Example:

```
>>> def testfunction( num ):
        """Function to the double vale"""
        return 2*num

>>>print(testfunction.__doc__)
Function to the double vale
```

Keywords

and	**as**	assert	**break**	**class**	continue
def	**del**	**elif**	**else**	**except**	exec
finally	**for**	from	global	**if**	**import**
in	**is**	lambda	nonlocal	**not**	**or**
pass	**raise**	**return**	**try**	**while**	**with** yield
True	**False**	None			

Python Variables

- **Variables:** a name that is assigned to a value

Ex. A = 50

- We don't need to declare a variable before using it.

- In Python, we simply assign a value to a variable and it will exist.

- This is handled internally according to the type of value we assign to the variable.

Python Variables

- It produces a unique number identifying a specific value (object) in memory.

```
>>>a=10
>>>b=10
>>>id(a)
1593988992
>>>id(b)
1593988992
>>>a=a+1
>>>id(a)
1593989008
```