# **Introduction to Python**

Python was developed by Guido Van Rossum at the National Research Institute for Mathematics and Computer Science in Netherlands during 1985-1990. Python is derived from many other languages, including ABC, Modula-3,C,C++,Algol-68,SmallTalk,Unix shell and other scripting languages. Rossum was inspired by Monty Python's Flying Circus, a BBC comedy series and he wanted the name of his new language to be short, unique and mysterious. Hence he named it python. It is general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python source code is available under the GNU General Public License(GPL) and is now maintained by a core development team at the National Research Institute.

**Features of Python**

1. Simple and easy-to-learn
2. Interpreted and interactive
3. Object-oriented
4. Portable
5. Scalable
6. Extendable
7. Dynamic
8. GUI Programming and Database
9. Broad Standard Library

**Using Interactive Interpreter**

You can start python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window. Get into command line of python. For Unix/Linux, you can get into interactive mode by typing $python or python%. For windows/DOS it is c:>python.

Invoking the interpreter without passing a script file as a parameter brings up the following prompt-

$ python

python 2.7.10 (default, sep 27 2015, 18:11:38)

[GCI 5.1.1 20150422 (Red Hat 5.1.1-1)] on linux2

Type "help", "copyright", "credits" or "license" for more

information1

>>>

Type the following text at python prompt and press enter:

>>>print("programming in python")

The result will be as given below

programming in python

**1 - Variable and Assignments**

**a) Scripts From Command Line**

This method invokes the interpreter with a script parameter which begins the execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active. a python script can be executed at command line by invoking the interpreter on your application, as follows.

for Unix/Linux $python script.py or python% script.py

for windows/DOS c:>python script.py

**Identifiers**

A python identifier is a name used to identify a variable, function, class, module or any other object. Python is case sensitive and hence uppercase and lowercase letters are considered distinct. The following are rules for naming an identifier in python.

1. Identifier can be combination of letters in lowercase(a to z) or uppercase(A to Z) or digit(0-9) or an underscore(\_). For example total and Total is different.
2. Reserved keywords cannot be used as an identifier.
3. Identifiers cannot begin with a digit.
4. Special symbols like !,@,#, etc cannot be used in an identifier. For example sum@ is an invalid identifier.
5. Identifier can be of any length.

**Reserved Keywords**

These are keywords reserved by the programming language and prevent the user or the programmer from using it as it as an identifier in a program. There are 33 keywords in python 3.3. This number may vary with different versions. To retrieve the keywords in python the following code can be given at the prompt.

>>> import keyword

>>> print(keyword.kwlist)

['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

**Variables**

Variables are reserved memory locations to store values. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign(=) is used to assign values to variables. The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in that variable.

**Example program**

a = 100

b = 1000

print(a)

print(b)

**Output**

100

1000

**Comments in Python**

Comments are very important while writing a program. It describes what the source code has done. Comments are for programmers for better understanding of a program. In python , we use the hash(#) symbol to start writing a comment. All the character after the #and up to the end of the physical line are part of the comment.

**Example program:**

>>>#this is demo of comment

>>>print("hello")

For multiline comments use triple quotes, either ''' or """. the statements within the start and end of triple quotes are considered as comment.

**Example program:** Consider sample python script to produce sum of two variables

“””

within this quotes all are considered as comments

This program will print sum of two variable

a is entered first

b is entered next

c will produce sum of a and b

“””

a = input(“enter a value”)

b = input(“enter b value”)

c = a + b

print “sum of a and b is “, c

**Indentation in Python**

Most of the programming languages like c, c++ use braces{} to define a block of code. Python uses indentation. A code block starts with indentation and ends with the first unindented line. The amount of indentation can be decided by the programmer, but it ,must be consistent throughout the block. Generally four whitespaces are used for indentation.

**Example program**

if True:

print("correct") # 4 space gap given at begining

else:

print("Wrong") # same 4 space gap given

**Input, Output and Import Functions**

**Displaying the Output**

The function used to print output on a screen is print statement where you can pass zero or more expression separated by commas. The print function converts the expression you pass into a string and writes the result to standard output.

**Example program**

>>>a=2

>>>print("the value of a is:",a)

the value of a is 2

**Reading the Input**

Python 2.7 provides two built-in functions to read from standard input(keyboard). These functions are raw\_input and input.

input is used to get integer or float user inputs

raw\_input is used to get string or int or float user inputs

**Example program**

**1)** a = input("enter your age:")

print("your age is: ",a)

**output**

enter your name: 20

your age is: 20

**2)** a, b, c = input(“enter a, b and c values, each separated by comma character: \n”)

print “a+b+c is: ”, (a+b+c)

**output:**

enter a, b and c values, each separated by comma character:

10,20,30

a+b+c is: 60

3) name = raw\_input(“enter your name:”)

print “entered name is: %s” %name

**output:**

enter your name: python

entered name is: python

**Import function**

When the program grows bigger or when there are segments of code that is frequently used, it can be stored in different modules. A module is a file containing python definition and statements. Python modules have a file name and end with expression .py. Definition inside a module can be imported to another module or the interactive interpreter in python. We use the import keyword to do this .

**Example program**

>>>import math

>>>math.pi

3.141592653589793

**Operators in Python**

**1) Arithmetic operators:**

**+, -, \*,/,//,%,\*\*,-**

+,-,\*,/ works as expected from their algebra class notations.

// is called floor division and it truncates the floating point value while preserving the data type.

% is called modulus and gives reminder of division. It works right to left.(?)

\*\* is exponential.

**2) Conditional operators**

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

These operators result in Boolean True or False value. They are good for using in conditions

**3) Membership operators**

in, not in simply checks if the value is in a particular sequence or range and return True and False accordingly.

**Variable as Storage unit.**

Since variables hold reference to data for manipulation. Popular database operations like Insert, Update, Delete should also be possible on variable.

To something like insert or assign or define:

**Syntax is :** *variable = value*

*E.g. >>>x=5*

To update a variable value:

**Syntax is :** variable = new \_value

E.g. >>>x=10

After new value assignment, the variables are pointer is pointed to new value. Since one variable can point to only one location at a time. The value old value is garbage collected.

To delete a variable:

**Syntax is :** *del variablename*

*E.g. >>>del x*

**Unique property of the = (assignment) operator**

It right to left hence , in variable = value , value is considered first and the identification is value based.

**Variable Nomenclature**

1. Variable name must start with [\_a-z, A-Z] then followed by [\_a-z, A-Z, 0-9]

2. Variable names are case sensitive

3. Keywords cannot be used as Python Variable names

>>>import keyword

>>>keyword.kwlist

['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

Examples of valid and invalid variable name.

\_a=3 #valid

Apple = “apple” #valid

0time = 0 #in-invalid

**Variable and Data Types**

number = 100 # An integer assignment

miles = 1000.0 # A floating point

name = "John" # A string

x = [1,2,”users”] # A List

y = (“username”, “password”) # A Tuple

dict ={'name':'john','code':6734,'dept':'sales'} # A dictionary

**Python Variable properties**

**1. Implicitly defined**

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

**2. Strongly typed**

Python is strongly typed as the interpreter keeps track of all variables types.Compiler here cannot can tell which type a variable refers to.

**Type() function**

type(object) -> the object's type

>>>x=”name”

>>>type(x)

<type 'str'>

**Multiple assignment**

Ex: >>>*a=b=c=1*

*>>>a,b,c= 1,2,”name*

**Python default variable ‘\_’**

‘\_’ is a variable auto defined by Python and it stores last calculation. It is like ANS button on calculator.

>>> 20 + 30

50

>>> 50 + \_

*100*

## **The str(), int(), and float() Functions**

## If you want to concatenate an integer such as 29 with a string to pass to print(), you’ll need to get the value '29', which is the string form of 29. The str() function can be passed an integer value and will evaluate to a string value version of it, as follows:

>>> **str(29)**

'29'

>>> **print('I am ' + str(29) + ' years old.')**

I am 29 years old.

Because str(29) evaluates to '29', the expression 'I am ' + str(29) + ' years old.' evaluates to 'I am ' + '29' + ' years old.', which in turn evaluates to 'I am 29 years old.'. This is the value that is passed to the print() function.

The str(), int(), and float() functions will evaluate to the string, integer, and floating-point forms of the value you pass, respectively. Try converting some values in the interactive shell with these functions, and watch what happens.

>>> **str(0)**

'0'

>>> **str(-3.14)**

'-3.14'

>>> **int('42')**

42

>>> **int('-99')**

-99

>>> **int(1.25)**

1

>>> **int(1.99)**

1

>>> **float('3.14')**

3.149

>>> **float(10)**

10.0

The previous examples call the str(), int(), and float() functions and pass them values of the other data types to obtain a string, integer, or floating-point form of those values.

The str() function is handy when you have an integer or float that you want to concatenate to a string. The int() function is also helpful if you have a number as a string value that you want to use in some mathematics. For example, the input() function always returns a string, even if the user enters a number. Enter **spam = input()** into the interactive shell and enter **101** when it waits for your text.

>>> **spam = input()**

101

>>> **spam**

'101'

The value stored inside spam isn’t the integer 101 but the string '101'. If you want to do math using the value in spam, use the int() function to get the integer form of spam and then store this as the new value in spam.

>>> **spam = int(spam)**

>>> **spam**

101

Now you should be able to treat the spam variable as an integer instead of a string.

>>> **spam \* 10 / 5**

202.0

Note that if you pass a value to int() that it cannot evaluate as an integer, Python will display an error message.

>>> **int('99.99')**

Traceback (most recent call last):

File "<pyshell#18>", line 1, in <module>

int('99.99')

ValueError: invalid literal for int() with base 10: '99.99'

>>> **int('twelve')**

Traceback (most recent call last):

File "<pyshell#19>", line 1, in <module>

int('twelve')

ValueError: invalid literal for int() with base 10: 'twelve'

The int() function is also useful if you need to round a floating-point number down. If you want to round a floating-point number up, just add 1 to it afterward.

>>> **int(7.7)**

7

>>> **int(7.7) + 1**

8

In your program, you used the int() and str() functions in the last three lines to get a value of the appropriate data type for the code.

print('What is your age?') # ask for their age

myAge = input()

print('You will be ' + str(int(myAge) + 1) + ' in a year.')

The myAge variable contains the value returned from input(). Because the input() function always returns a string (even if the user typed in a number), you can use the int(myAge) code to return an integer value of the string in myAge. This integer value is then added to 1 in the expressionint(myAge) + 1.

The result of this addition is passed to the str() function: str(int(myAge) + 1). The string value returned is then concatenated with the strings 'You will be ' and ' in a year.' to evaluate to one large string value. This large string is finally passed to print() to be displayed on the screen.

## **The len() Function**

## You can pass the len() function a string value (or a variable containing a string), and the function evaluates to the integer value of the number of characters in that string.

print('The length of your name is:')

print(len(myName))

Enter the following into the interactive shell to try this:

>>> **len('hello')**

5

>>> **len('My very energetic monster just scarfed nachos.')**

46

>>> **len('')**

0

**id() function:**

This is used to determine the id of a variable. It is value based and not variable name based.

>>>x=5

>>>id(x)

140566819039352

>>>y=6

>>>hex(id(y))

'0x163c080'

>>>z=5

>>>id(z)

'0x164d6d0'

**Membership Operators**

Python's membership operators test for membership in a sequence, such as strings, lists, and tuples. There are two membership operators supported by python which are described in the following table.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| in | Evaluates to true if the variable on either side of the operator point to the same object and false otherwise |
| not in | Evaluates to true if it does not finds a variable in the specifies sequence and false otherwise. |

**Identity Operator**

Identity operators compare the memory locations of two objects. There are two identity operators as shown in the following table

|  |  |
| --- | --- |
| **Operator** | **Description** |
| is | Evaluates to true if the variables on either side of the operator point to the same objects and false otherwise |
| is not | Evaluates to false if the variables on either side of the operator point to the same objects and true otherwise |

**Questions**

1) Assign the value 10 to a, and value 20 to b. Assign the value of a + b to variable c.

2) Which of the following will produce the right result

a) 2 + 5 produces '25'

b) '2' + 5 produces '25'

c) '2' + '5' produces '25'

d) 2 + '5' produces '25'

3) Compute the area and perimeter of a circle with radius = 3

4) Declare a string literal by enclosing the literal using single, double or triple quotes. Triple quote allows the literal to span multiple lines.

>>> spam = 'mail'

>>> contents = """This is Line 1

... and this is Line 2

... and this is Line 3

... """

>>> spam

'mail'

>>> contents

'This is Line 1\nand this is Line 2\nand this is Line 3\n'

>>> 'How're you?' *# Syntax error for same quote*

**SyntaxError**: invalid syntax

>>> "How're you?" *# Using different quote*

"How're you?"

>>> 'How**\'**re you?' *# Escape quote with \*

"How're you?"

>>> foobar

'"No, thanks, Mom," I said, "I don**'**t know how long it will take."'

5) Certain mathematical operations such as addition and multiplication can be used on the String data type. Study the examples given below to see how it works

>>> 5 + 6 *# Adding numbers*

11

>>> '5' + '6' *# Adding strings*

'56'

>>> 5 \* 6 *# Multiplying numbers*

30

>>> '5' \* 6 *# Multiplying string with number*

*'555555'*

6) Assign strings 'Hello' and 'World!' to single variable a

**Sol:**

a = "Hello"+"World!"

# repeating string 5 times

b = "HelloWorld!"\*5

7)You can make use of the len(x) function to find out the number of characters in a string.

>>> greetings = "Hello World"

>>> len(greetings) *# get the length of string*

11

>>> greetings[0] *# get the 1st character*

'H'

8) What is the value of x?

>> x = 1 == 1 and 1 != 0 or 1 > 0

9)Extract each word from variable 'greetings' and assign to

# variables 'first', 'middle' and 'last'.

greetings = "How are you"

first = greetings[ :3]

middle = greetings[ 4:7]

last = greetings[ 8:]