Learn Python

PythonProgrammingLanguage.com

September 7, 2016

1 Introduction

1.1 What is Python?

Python is a computer programming language that lets you work more quickly than other programming languages. Experienced programmers in any other language can pick up Python very quickly, and beginners find the clean syntax and indentation structure easy to learn. This tutorial will help you to become a python developer.

1.2 The Python Interpreter

To get started, you will need the Python interpreter or a Python IDE. You can download Python interpreter here: **https://www.python.org/downloads/** The Python interpreter is a command line program:

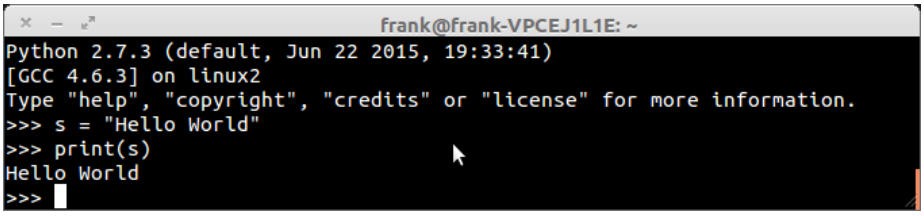

Figure 1: Python interpreter in the terminal.

Typically you would run a program using: **python programName.py** where programName.py is a text file that contains the code. Python programs are simply a collection of text files. If you want something more sophisticated than notepad for editing, you will need a Python IDE (recommend). A Python IDE will make programming Python easier.

1.3 Python IDE

A Python IDE is more than a rich text editor, it's a complete development environment. An IDE generally supports listing all program files, syntax highlighting and other features. There are a few Python IDEs you could choose from:

- PyDev
- Komodo Edit
- PyCharm
- Atom with Script plugin

Using one of these Python IDEs makes programming easier than in say, notepad. It will automatically color the text like the example below:

program.py

```
1  import sys
2
3  print('Arguments: ' + str(len(sys.argv)))
4  print('List: ' + str(sys.argv))
5
6  if len(sys.argv) < 2:
7      print('Too few arguments, please specify a filename')
8  else:
9      filename = sys.argv[1]
10     print('Filename: ' + filename)
11
12
```

Figure 2: Python IDE.

2 Text input and output

2.1 Output

To output text to the screen you will need one line of code:

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

To write multiple lines, add the '\n' character:

```
print("Hello World\nThis is a message")
```

To print variables

```
x = 3
print(x)
```

To print multiple variables on one line

```
x = 2
y = 3
print("x = {}, y = {}".format(x,y))
```

2.2 Input

To get a text value:

```
name = input("Enter a name: ")
```

To get a whole number:

```
x = int(input("What is x? "))
```

To get a decimal number

```
x = float(input("Write a number"))
```

3 Variables

Variables in Python can hold text and numbers. For example:

```
x = 2
price = 2.5
word = 'Hello'
```

The variable names are on the left and the values on the right. Once a variable is assigned, it can be used in other places of the program.
In the example above, we have three variables: x, price and word. Variables may not contain spaces or special characters.
Text variables may be defined in 3 ways:

```
word = 'Hello'
word = "Hello"
word = '''Hello'''
```

The type depends on what you prefer. Once defined variables can be replaced or modified:

```
x = 2

# increase x by one
x = x + 1

# replace x
x = 5
```

Python supports the operators +, -, / and * as well as brackets. Variables may be shown on the screen using the print statement.

```
x = 5
print(x)

y = 3 * x
print(y)

# more detailed output
print("x = " + str(x))
print("y = " + str(y))
```

The first output of the program above is simply the raw value of the variables. If you want to print a more detailed message like "x = 5", use the line 'print("x = " + str(x))'. This str() function converts the numeric variable to text.

4 If statements

Computer programs do not only execute instructions. Occasionally, a choice needs to be made. Such as a choice is based on a condition. Python has several conditional operators:

Operator	Meaning
$>$	Greater than
$<$	Smaller than
$==$	Equals
$!=$	Is not

Table 1: An example table.

Conditions are always combined with variables. A program can make a choice using the if keyword. For example:

```
x = int(input("Tell X"))

if x == 4:
    print('You guessed correctly!')

print('End of program.')
```

When you execute this program it will always print 'End of program', but the text 'You guessed correctly!' will only be printed if the variable x equals to four (see table above). Python can also execute a block of code if x does not equal to 4. The else keyword is used for that.

```
x = int(input("Tell X"))

if x == 4:
    print('You guessed correctly!')
else:
    print('Wrong guess')

print('End of program.')
```

5 For loops

To repeat code, the for keyword can be used. To execute a line of code 10 times we can do:

```
for i in range(1,11):
    print(i)
```

The last number (11) is not included. This will output the numbers 1 to 10. Python itself starts counting from 0, so this code will also work:

```
for i in range(0,10):
    print(i)
```

but will output 0 to 9.
Illustrated in this graphic:

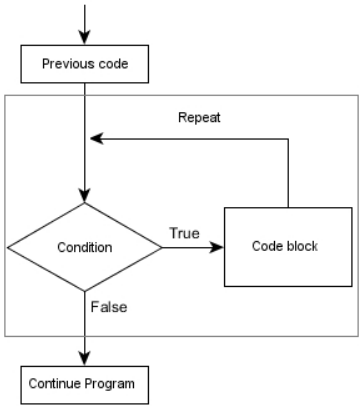

Figure 3: For loop

The code is repeated while the condition is True. In this case the condition is: i $<$ 10. Every iteration (round), the variable i is updated. Nested loops Loops can be combined:

```
for i in range(0,10):
    for j in range(0,10):
        print(i,' ',j)
```

In this case we have a multidimensional loops. It will iterate over the entire coordinate range (0,0) to (9,9).

6 While loop

Sometimes we are not sure how long sometimes will take. For example, how long a car driver would be adding petrol. A program could be (pseudocode):

```
while petrol_filling:
   increase price
   show price
   add petrol
```

Python has the while keyword. Lets build an extended guess the number game:

```
x = 0
while x != 5:
   x = int(input("Guess a number:"))

   if x != 5:
      print("Incorrect choice")

print("Correct")
```

This will keep asking a number until the number is guessed. This is defined in the line: while x != 5, or in English, "while x is not equal to five, execute".

Illustrated: Illustrated in this graphic:

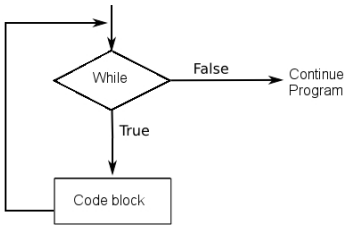

Figure 4: While loop

The program keeps repeating the code block while the condition (x!=5) is True. Once x equals 5, the program continues.

If a condition in a while loop is never met, it could cause the program to run forever or to freeze/crash.

7 Functions

To repeat lines of code, you can use a function. A function has a unique distinct name in the program. Once you call a function it will execute one or more lines of codes, which we will call a code block.

For example, we could have the Pythagoras function. In case your math is a little rusty,

```
a^2 + b^2 = c^2. Thus, c = sqrt( a^2 + b^2).
```

In code we could write that as:

```
import math

def pythagoras(a,b):
    value = math.sqrt(a *a + b*b)
    print(value)

pythagoras(3,3)
```

We call the function with parameters a=3 and b =3 on the last line. A function can be called several times with varying parameters. There is no limit to the number of function calls.

The def keyword tells Python we define a function. Always use four spaces to indent the code block, using another number of spaces will throw a syntax error.

It is also possible to store the output of a function in a variable. To do so, we use the keyword return.

```
import math

def pythagoras(a,b):
    value = math.sqrt(a*a + b*b)
    return value


result = pythagoras(3,3)
print(result)
```

The function pythagoras is called with a=3 and b=3. The program execution continues in the function pythagoras. The output of math.sqrt(a*a + b*b) is stored in the function variable value. This is returned and the output is stored in the variable result. Finally, we print variable result to the screen.

7.1 Lists

Python supports collections known as lists. A list is defined using square brackets. Let us define as simple list:

```
c = [5,2,10,48,32,16,49,10,11,32,64,55,34,45,41,23,26,27,72,18]
```

In this case we have a list defined by the variable c. We defined a purely random set of numbers above. Visual representation: Illustrated in this graphic:

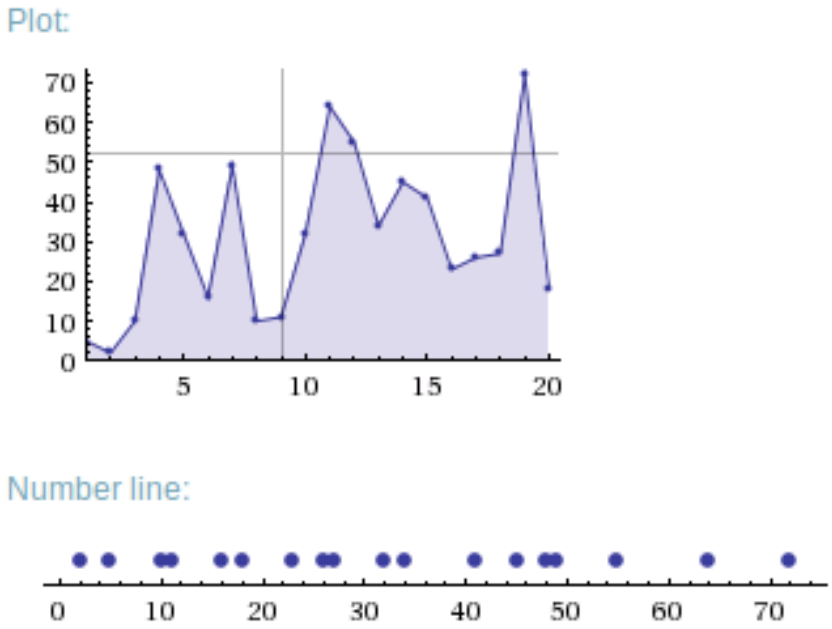

Figure 5: List

To access individual elements, we use the same brackets. To print the first element (Python starts counting from zero):

```
print(c[0])
```

To print the second element

```
print(c[1])
```

To print the last element, you can count from the back using the minus sign.

```
print(c[-1])
```

You can get the length of the list using the len function. Example code:

```
c = [5,2,10,48,32,16,49,10,11,32,64,55,34,45,41,23,26,27,72,18]
print(len(c))
```

8 Learn more

Congratulations! You learned the absolute basics.
Visit **http://pythonprogramminglanguage.com/** to learn more