# Python

**Pip**

While working as a developer, you’ll find yourself wanting to use some functionality that is not included in your language’s standard library. For example, in a standard .py file or document you will only be able to work with the standard library tools packaged with the language.

However, as a language grows and is used for more purposes, developers build their own tools and package them for use by others. These are known as third party modules. Pip is the package manager we use in Python to install these modules. Installing third party modules is pretty simple, because pip does a great job at knowing which version of that package we need to match our Python version.

**Comments**

Comments are useful because they allow you to explain what your code is doing. Every language has a way of ensuring that some lines will not be executed at run time.

As a developer, one of the most important jobs is writing re-usable code. By explaining what our code does in comments, we make it easier for ourselves and others to edit our code later on. In addition, comments can be helpful for writing pseudo-code when you’re trying to work out a tough problem.

# commenting a single line

# we can even comment out code

# print "this will not print!"

print "read below for more on multi-line comments in python!"

copy

#this would execute

# This line and below would not execute

'''

Triple quotations allow us to comment across multiple lines as long as

the triple quoted comment is not the first thing in your file.

You can use double or single quotes!

'''

**Data Types**

Data type refers to how the computer knows to classify information. To determine data type, ask what category a value belongs to. Here’s a list of the data types that you will surely be using in building web applications.

There are several general classifications for data we’re interested in. **Primitive data types** are the basic building blocks of a language. Most languages have these in common. Here are the most common:

* **Boolean-** Assesses the truth value of something. It has only two values: True & False
* **Numbers-** Integers (whole numbers), floating point numbers (commonly known as decimal numbers), and complex numbers.
* **strings-** A text literal. Most pages in the web work with strings quite often.

**Composite types** are collections composted of the above primitive types**.**

* **Tuples-** A type of data that is immutable (can’t be modified after its creation) and can hold a group of values. **Tuples** can contain mixed data types.
* **Lists-** A type of data that is mutable and can hold a group of values. Usually meant to store a collection of related data.
* **Dictionaries-** A group of key-value pairs. Dictionary elements are indexed by unique keys which are used to access values.

## In Python, (almost) everything is an object. We will touch on this later when we get into Object Oriented Programming(OOP).

**Indentation & Line-Endings**

One of the most important aspects of Python is indentation. Python has no brackets, braces, or keywords to indicate the start and code itself. You’ll see that indenting starts a new code block and un-indenting ends that block. Don’t worry if these codes don’t make sense right now; we’ll go over function and if- statements later. Just take not of how the indentation looks.

Conditional Expression

age = 16

if age >= 18:

print "legal age"

elif age == 17 or age < 17:

print "you are so young!"

my\_list = [4, "dog", ['german', 'shepherd'], 'bitme']

for element in my\_list:

print element

for count in range(0,100):

print "looping -", count

count = 0

while count < 5:

print "looping -", count

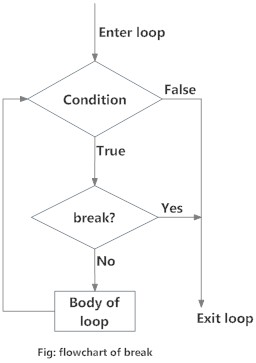
count += 1

x = [19,2,54,-2,7,12,98,32,10,-3,6]

x.sort

y,z=[i for i in x if i<0 ],[j for j in x if j>0]

print y,z //splits list at 0



**Functions**

A function is a named block of code that we can execute to perform a specific task. More simply, a function is a list of instructions that you can run at any time. If you find something that you seem to be using over and over again, it might be best to have a way to streamline the process. A function can optionally take in parameters, perform a series of instructions, and optionally return something afterwards. Here’s an example:

def add(a,b)

x = a + b

return x

result = add(3,5)

print result

Think of the function as a factory. If we were building a new car we would:

* Acquire raw materials (variables) needed for creating a car.
* Send the raw materials (invoke and pass arguments) to a car manufacturing plant (function)
* Do something (process) with the raw materials(parameters)
* Drive the car (function’s return value)

The factory has all the instructions to build a new car and will perform all the tasks. When you want a new car, all you have to do is call the factory to request a new car.

The advantages of using functions are:

* Reducing the duplication of code
* Breaking down complex problems into simpler pieces
* Improving clarity of code

**Syntax**

Pay attention to a few details. The **def** keyword signifies the declaration of a function. This indicates that **the following code is a function and assigns a name to that function, so we can call it later**. Parameters are information we input into a function, and appear inside the parenthesis that follow the function name.

**Function Parameters**

We define the input of functions using parameters. Like we’ve seen before, some functions do not have to take parameters. However, functions can optionally have one or more parameters.