**PYTHON BEGINNERS TUTORIAL**

**Getting started:**

This tutorial will majorly handle the practical aspects than theories. However, theories will be briefly explained on the fly. You are encouraged to read up from other sources like Wikipedia to know more. Before we go further, we first need to install Python and ***optionally*** a third party software.

**Installing Python**

Visit https://www.python.org/downloads/

Download the installation file based on the operating system you’re using and follow the instructions to properly install Python.

**Please Note:** we shall be using version 3+ for this course. Therefore, download and install the latest version you find.

We shall also be using **anaconda.** Anaconda is a much user friendly interface than the regular Python interface. To install anaconda, go to https://www.anaconda.com

At 360academia, we learn by doing. Please pay ettentio

**Introduction:**

Python is a high-level programming language developed by *Guido Van Rossum* in the late 1980s. It is presently administered by *Python Software Foundation*. It is an open source programming language and can be used in the development of various kinds of applications.

Being a high-level programming language, writing a program with Python is much like wring a regular English Language statement. These statements (programming codes) first need to be converted or translated to machine-readable byte codes, by the use of an ***interpreter***. Python can be used for creating games, web applications, mobile applications and also a very powerful tool for data-science (data analysis/analytics).

Python programming come really handy and recommended for individuals (beginners) that wish to dive into programming and/or analysis as their first programming language. The codes in Python are considerably shorter and easy to understand than most high-level programming languages.

**Programming with Python:**

**Python Syntax**

Python syntax refers to the rules that defines how a Python statement should be written as well as how the system should interpret these statement. Every programming language has its own syntax.

**Keywords**

Keywords are reserved words in a programming language that should not be used as *variables, constants, function names, or identifiers* in your code. Python also has its own Keywords and they are (arranged in alphabetical order as): *and, 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*.

**Python Identifiers**

An Identifier is a name given to a function, class, module, variable or object that you’ll be using in your program later on. These identifiers are defined by the programmer. Identifiers can be a combination of uppercase and lowercase alphabets, underscore and numbers majorly. Identifiers cannot begin with numbers (digits), but can contain numbers after an initial alphabet.

Some recommended ways of defining identifiers are:

myObject, my\_Class, myVariable\_10, var20, etc.

**PLEASE NOTE:** Python is case-sensitive. This means that uppercases and lowercase are seen differently. MyName, myName and MYNAME are all seen differently in Python. As such, they do not mean the same thing.

**Indentation:**

Indentation in Python programming works with four (4) spaces. For instance in a conditional statement:

if free\_learning:

print(“I am happy”)

On the second line of the code above, the space allowed for it to be a correct syntax are four (4) spaces.

**Comments:**

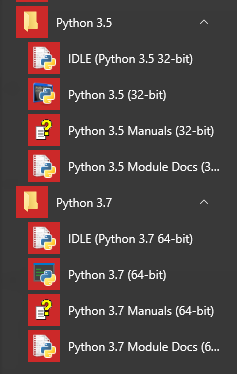
It is often useful and recommended to add comments to segments of your code, to make it easier to understand and perhaps remember what a block or line of code actually does. To add a comment, the line starts with a hash (#) symbol. This way, the program ignores that line when it is executing your program.

Please pay attention to the basic statements we have stated above, as we’ll encounter them further in our studies.

**The environment:**

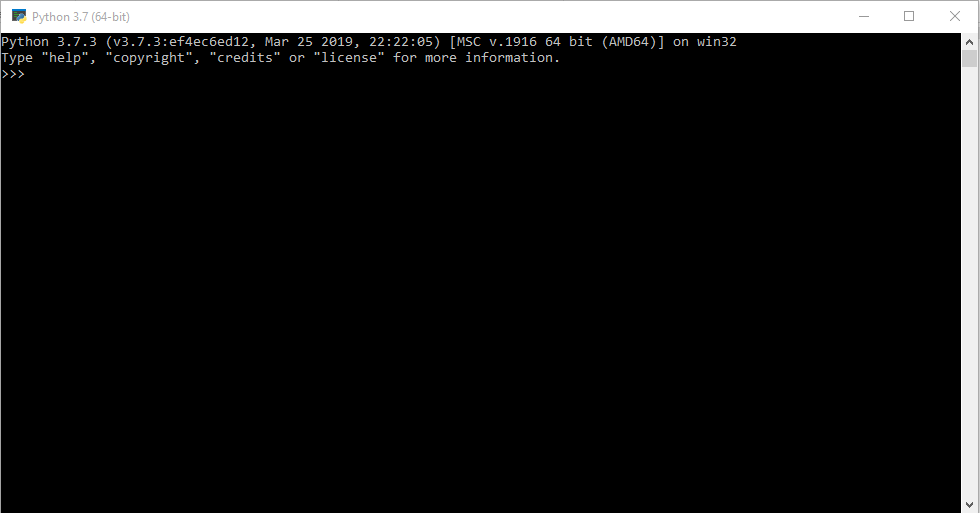
Run (open) the Python program you just installed from the **start menu > All programs (apps)**.

Depending on the version you installed, you should have a screen that look like this

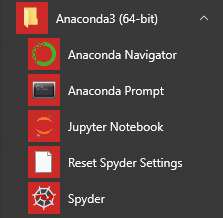


Open the file named “Python 0.0” (Please Note: 0.0 mean the version of python you downloaded and installed).

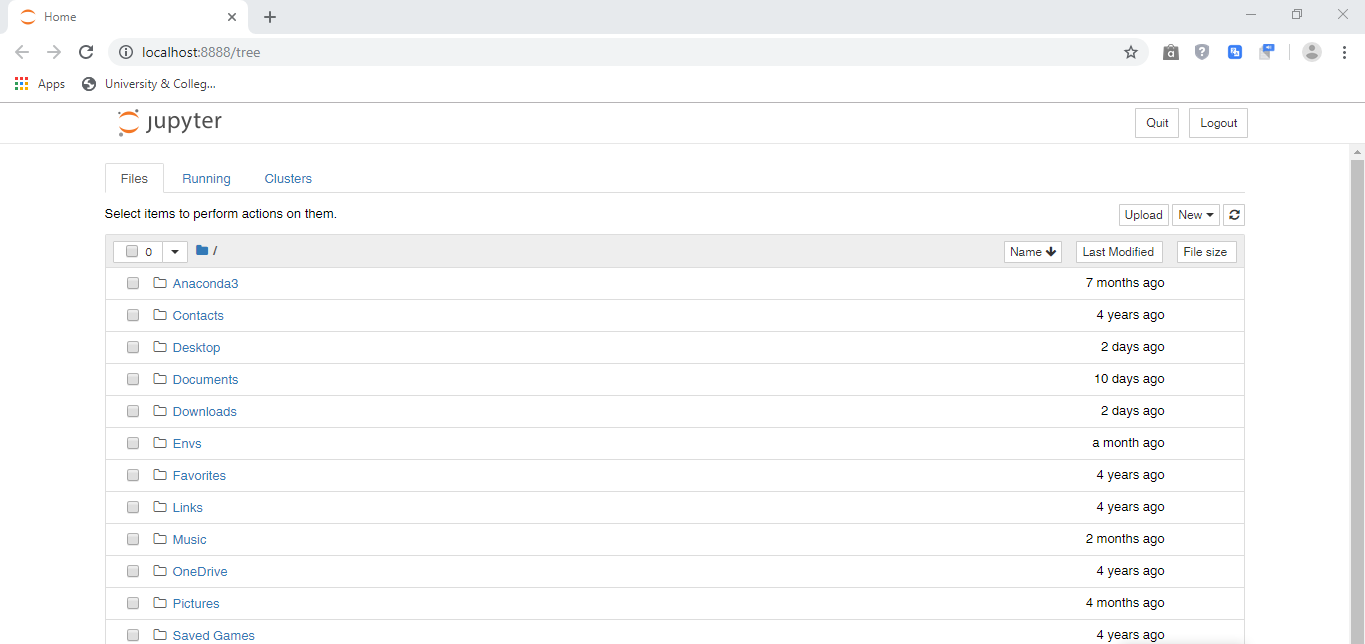
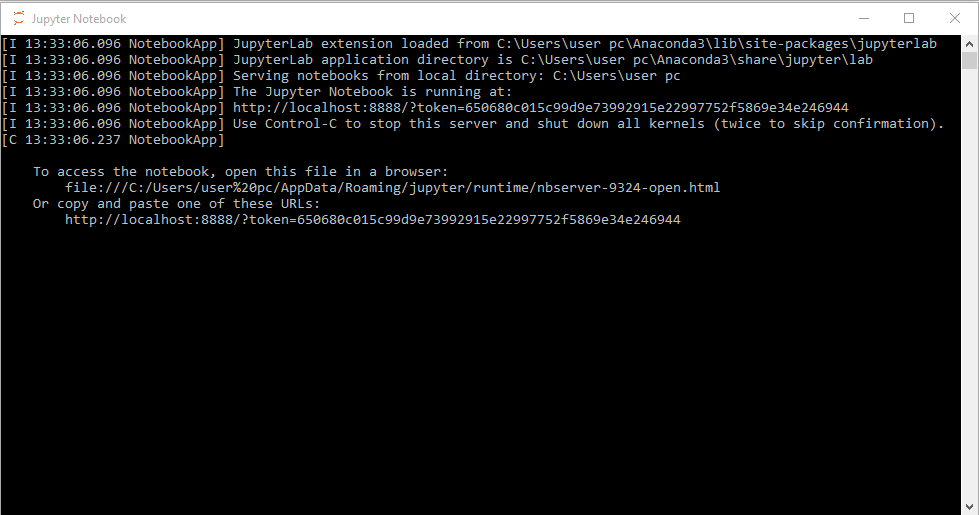
You should see a screen that looks like this:



If you decide to use anaconda (as we shall be using in this series):

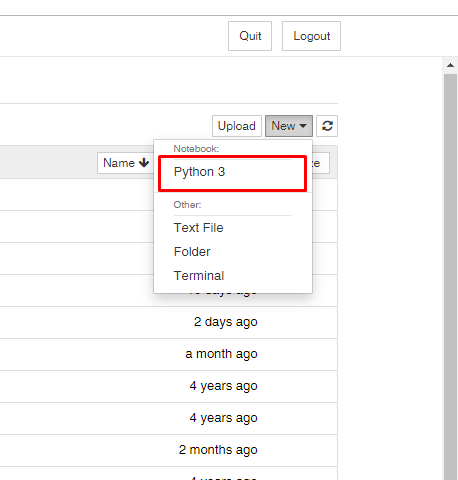


Open “Jupyter Notebook”, and you should have two applications opened. One is on a web browser and the other is a command line environment. The command line helps the web browser interface work. The command line and web browser opened should have features as shown in the image below:

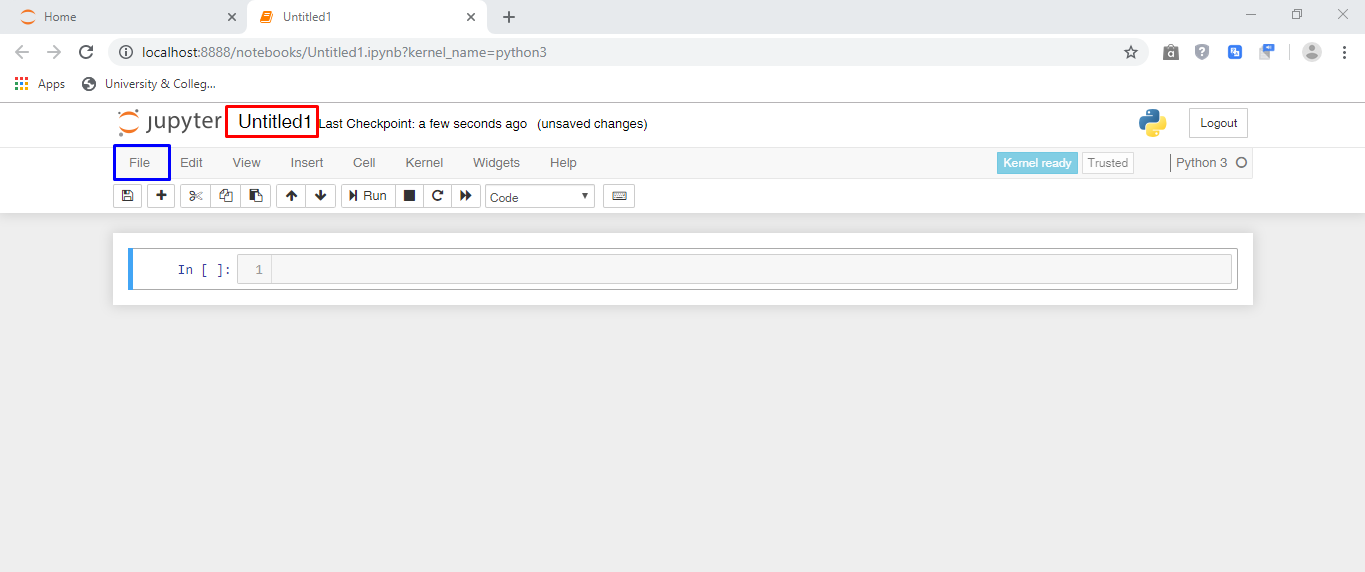


Locate a folder you would like to save your work into before moving to the next step. You can easily create a folder on your desktop and navigate from Jupyter Notebook to the folder you created.

Follow the step below to open an interface for python programming:



You should have an interface that looks like the image below:



Notice that the file name is “Untitled1”. You could either click on the filename (Untitled1) to change the name or Go to “file” and then hit “save” to store the file with any name of your choosing.

**Print statements:**

Print statements are used to print out results.

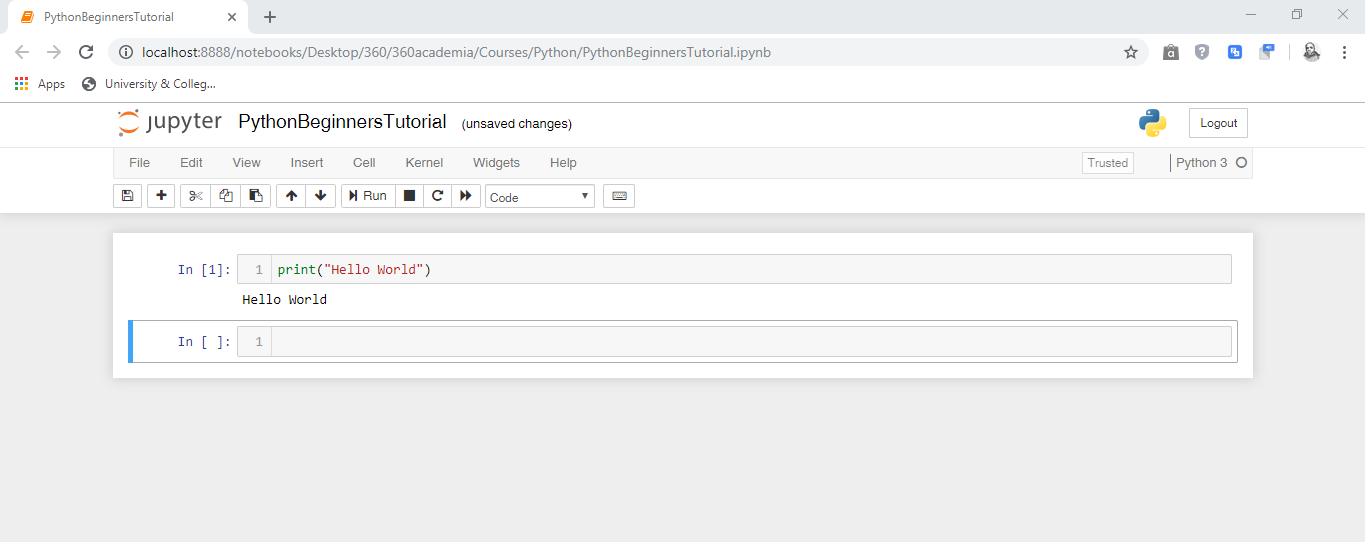
The format:

You first need to indicate that you want something printed out, then you say what you would like to print out. The compiler takes it up from there.

Type in the code below and hit enter

print(“Hello World”)

While on Jupyter Notebook, type the same code and click the “Run” button to see the output.



Congratulations!!! You have just written your first Python Program.

That was an illustration of how the print statement works, as well as checking to be sure that you have properly installed Python and Anaconda.

You could also type in multiple print statements in one block on the Jupyter Notebook and it will execute accordingly.

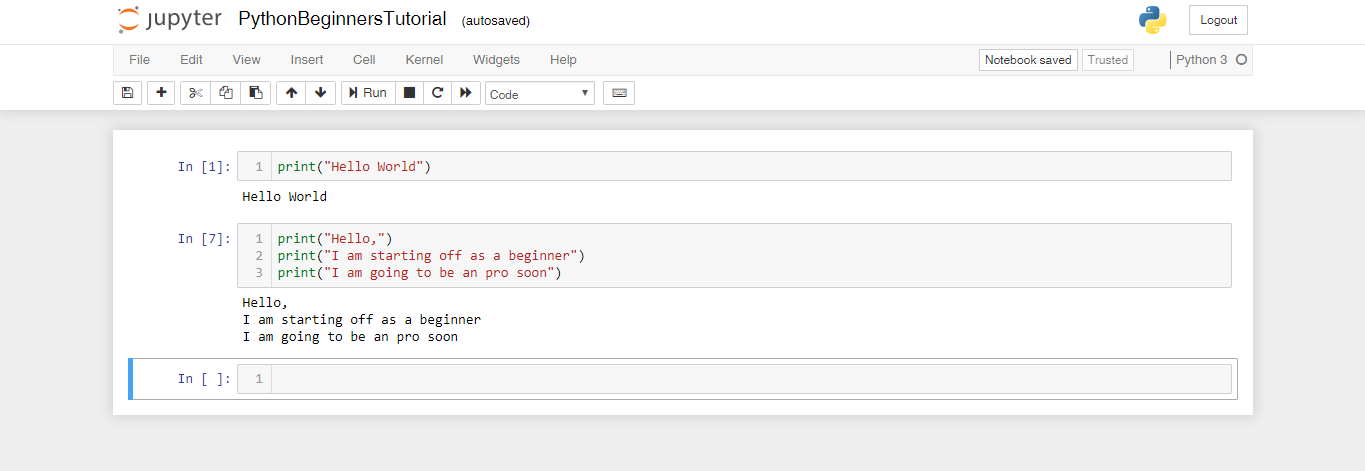
Take for instance the code below:

print(“Hello,”)

print(“I am starting off as a beginner”)

print(“I am going to be an pro soon”)

**PLEASE NOTE:** Copying and pasting codes could work faster and be more convenient. However, it is not recommended. It could slow down your learning speed. It is best practice to type out the codes, as it will help you understand it better.



**Variables and conditional statements:**

A variable can be likened to a container that stores items that you can access or change when you want to. Variables helps in pointing to a memory location used by a program. Variables can be assigned by using any of the formats we mentioned in the previous topic.

Let’s assign “a” the value of “1” (a = 1)

We have only assigned “a” a value of 1, but we haven’t used it or done anything with it yet.

Before you see the code snippet below, try printing out the value of “a” using the print statement we had earlier talked about.



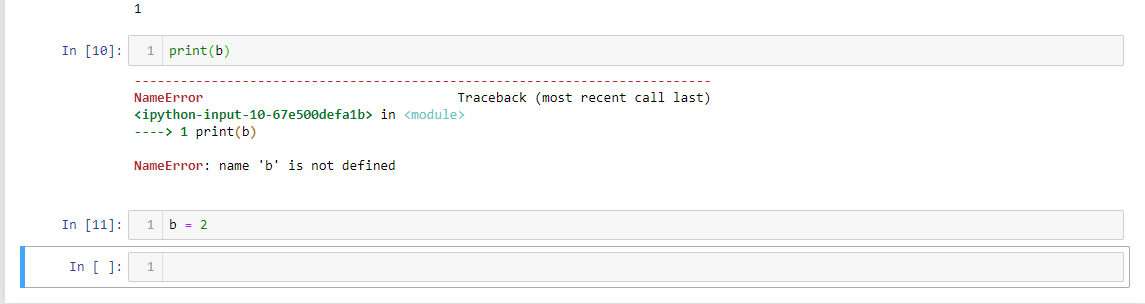
What if we run the code:

Print(b)

After which we assign b a value?

Let’s assume; b = 2

What do you think will happen? Guess what happens before you see the code snippet below.

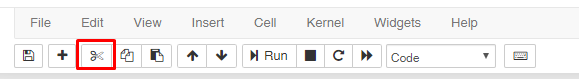


You are getting that error because you didn’t state (define) the value of “b” as at the time you printed it. Now assign ‘b’, go back to the block where you printed the value of b and run again.



Therefore, you have to always make the assignments or declarations before using it.

To delete a block of code, simply click on the block and then hit the scissors button.



What if you delete the block that says “b = 2” and then run the line that states “print(b)” again, what happens then?

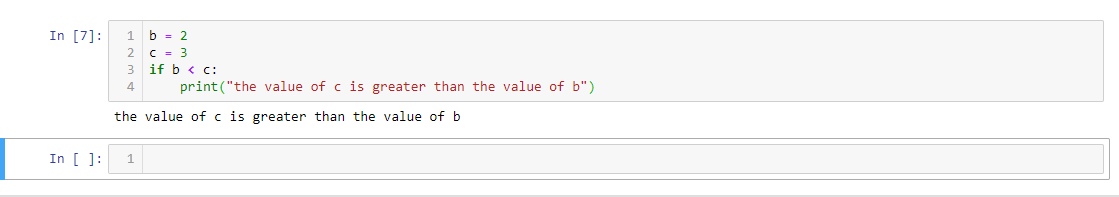
The code will run perfectly well. Because it remembers that “b” already was assigned a value from the last time you ran the program.



Let’s bring in some “if” (conditional) statements:

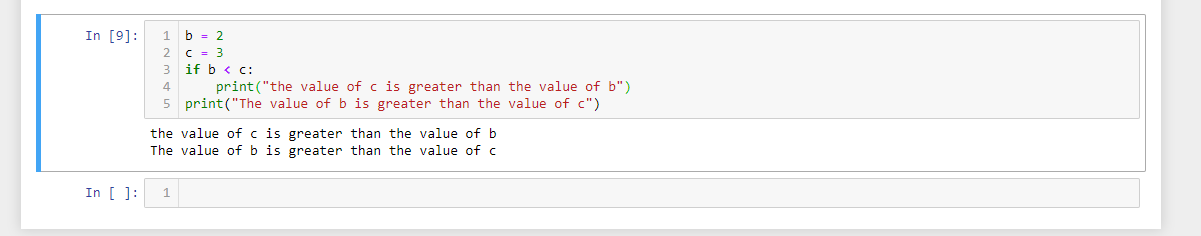
Conditional statements check for a (or some) conditions to be met before performing a particular action. For instance.

Consider the code below (remember what we said about indentation?):



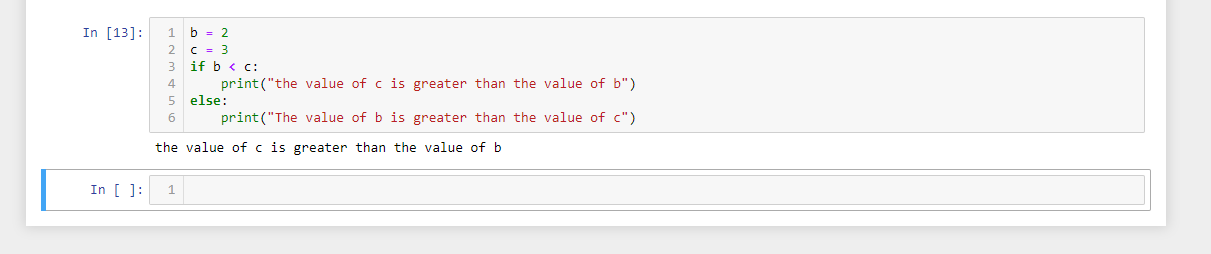
Looking at the variables assigned, the value of c is obviously greater than the value of b.

What if we add another print statement to the code?

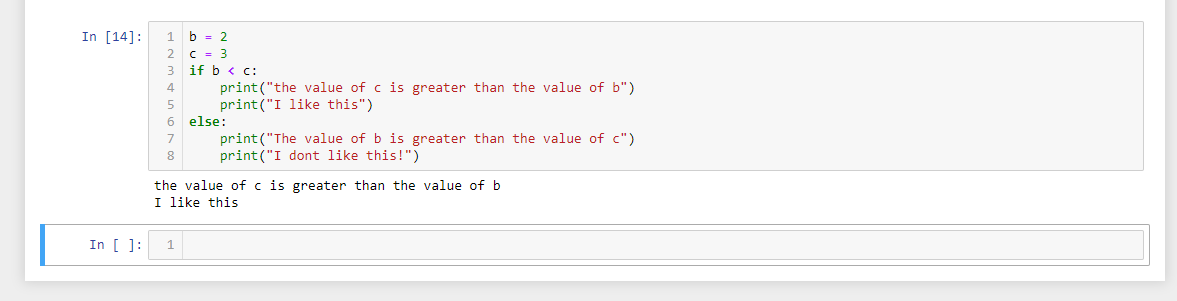


We get both lines because we didn’t inform the second “print statement” when it should be printed, and it isn’t dependent on any conditions at all.

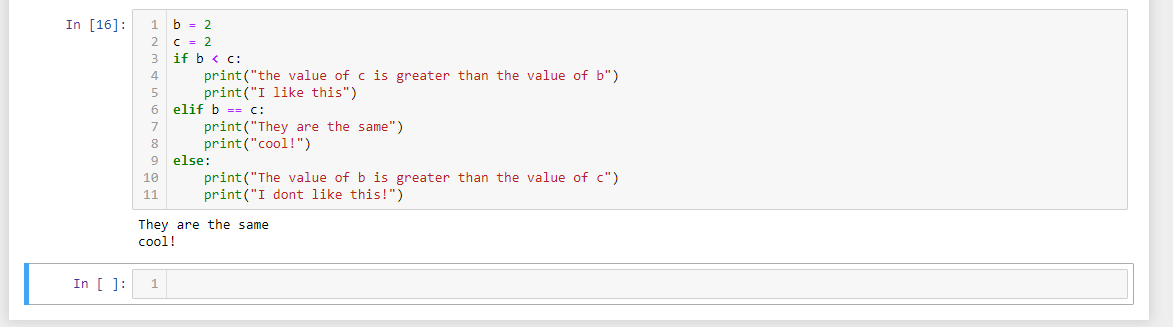
Let’s give it a condition and try again.



You can also have multiple print statements.



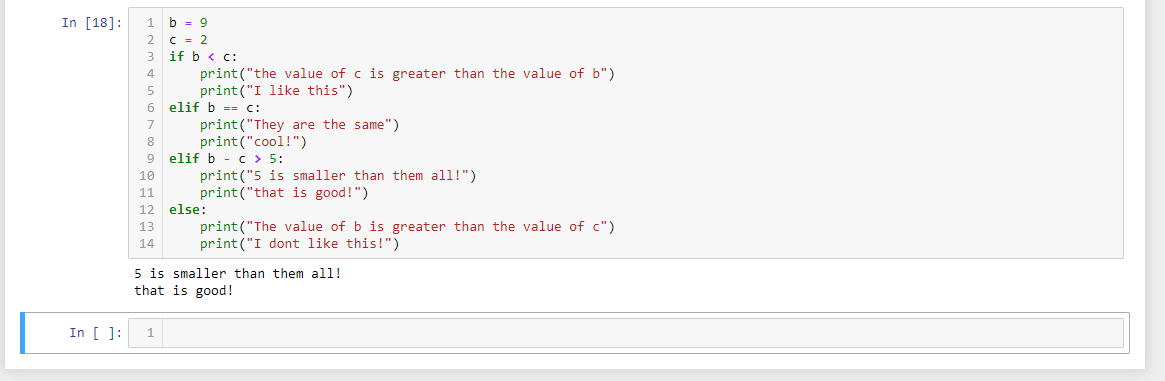
Let’s introduce an “else if” statement. An ‘if else’ statement in python is known as “elif”. When you have more than two conditions to check for, you use an else if statement.



Please note that the value of b and c are changed to help us see how the conditional statements work.

The single “=” sign assigns a value to a variable, while the double “==” sign indicates that the both values are the same.

We can further change the logic of a conditional statement as shown below:



We can have a conditional statement within another conditional statement as shown below.



**Please Note:** when you use “else”, it means that; anything else aside the “if” condition goes at this point.

To have a better understanding, spend more time on this model. Play with numbers to help you really understand how the conditional statement works.

Please note the indentation for the “if” statement within the “else” statement block, as well as the print statements contained therein (from line 6). It takes four spaces from the statement it belongs to. Proper indentation is part of Python programming. Without this, you’d get errors in your code.

Let’s carry out an actual calculation. Assuming we are to check if an individual is overweight or not. Assuming we have the formula as: bmi = (individuals\_weight / (individuals\_height\_in\_miters2)).

“Raised to the power of” is written as double asterisks “\*\*” in most programming languages including Python.



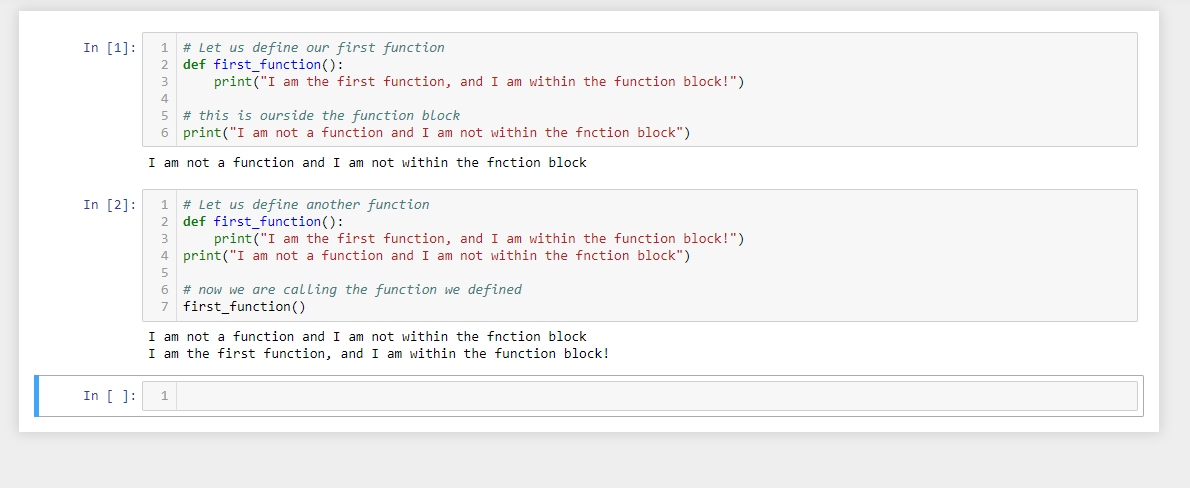
Also notice that we introduced the use of comments in our codes. These comments are ignored when the program runs.

**Milestone self-assessment**

Try calculating the area of a circle or other mathematical equations all by yourself. Don’t expect to get it right the first time. If you do, congratulations. If you don’t, look through your code, you’d definitely find the error.

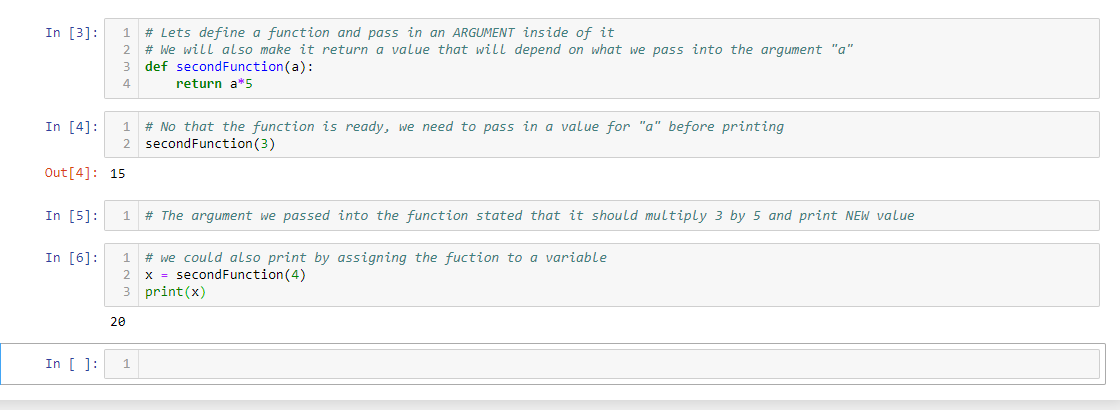
**FUNCTIONS**

Functions are a collection of programming instructions. Functions first need to be defined and called upon before the can be useful. See example below:

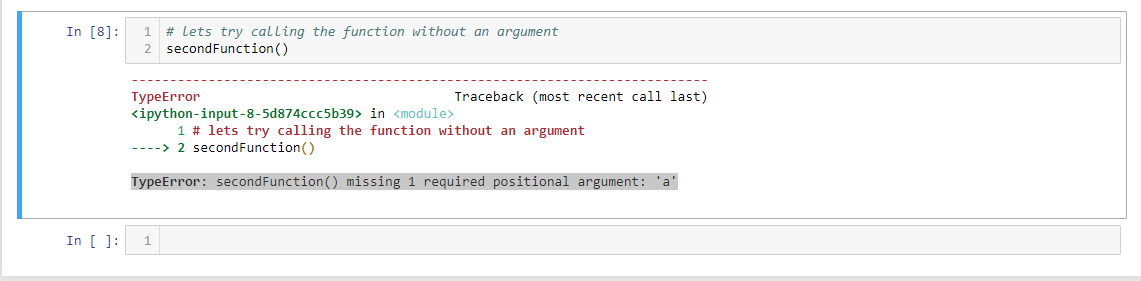


Notice that the first block didn’t execute the function. That was because all we did was define the function. We did not actually call it or put it to use it until the second block.

Functions can perform tasks before being returned. For instance; we could have a mathematical equation within a function and make a calculation each time the mathematical equation (function) is called. See the illustration below:



Note what happens when we try to execute the query without inputting an argument

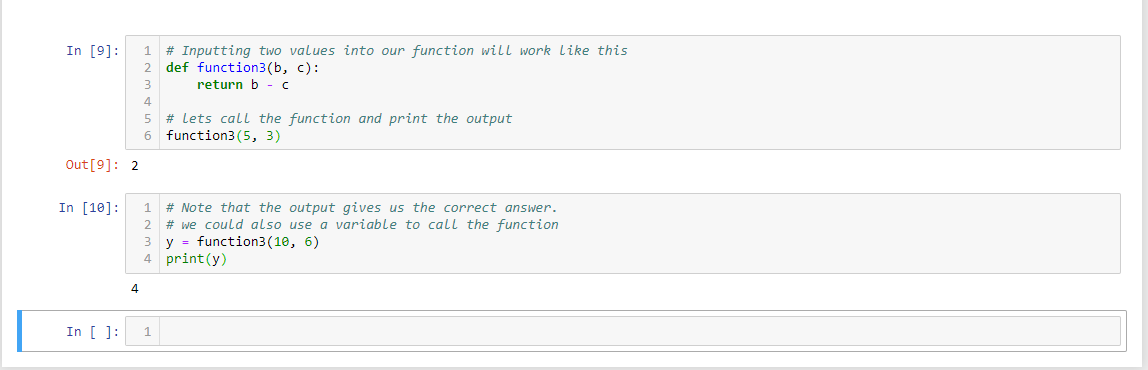


The error we get says:

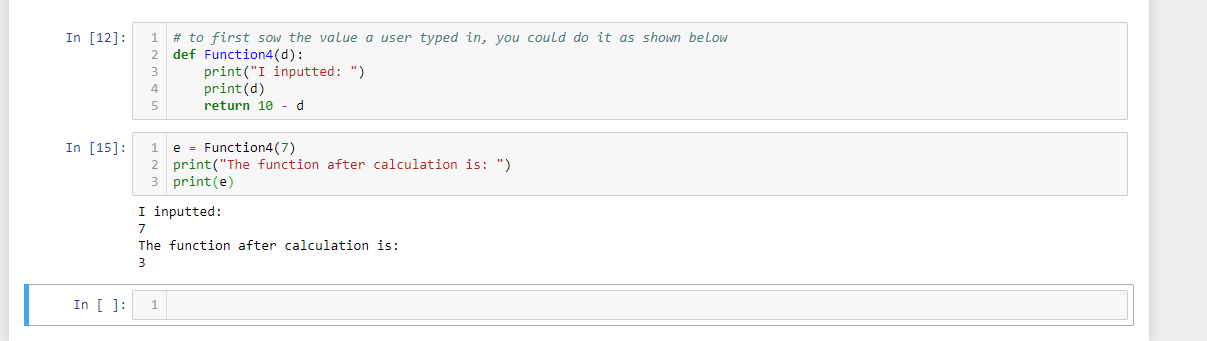
**TypeError**: secondFunction() missing 1 required positional argument: 'a'

Which obviously means we can’t perform that action without inputting a value to replace position “a” in the function. This value we are passing is called an “argument”.

What if we wanted to pass in two (arguments) into our function?



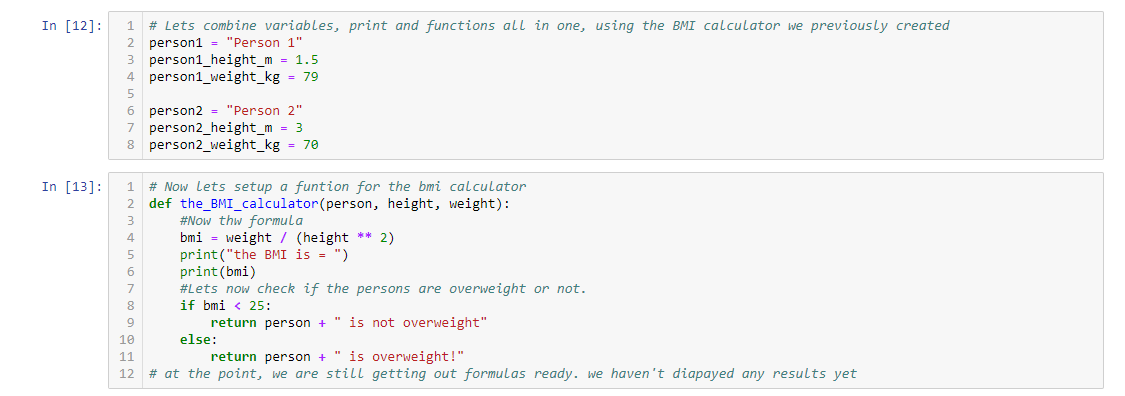
We could still play around the function by display the initial values before the computed values as shown below:



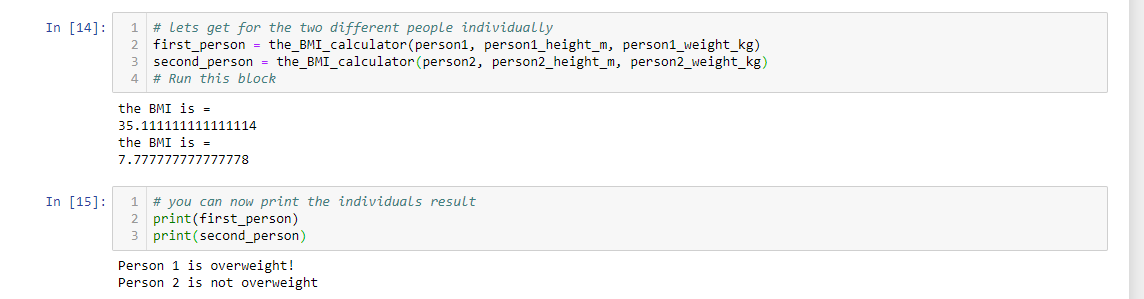
Recall the BMI calculator we previously created? Let’s modify it for two people and try using variables and functions to calculate them. We are also going to be introducing ***concatenation***.

When you have two stings to be displayed, you can display them both by concatenation (simply adding a plus “+” sign to separate them both).

First we would declare the variables and then make use of them in a function after which we will get the outputs.



Now that our variables and functions are all set, let’s print out the results.



You should have an output like this.

**Self-assessment**

With all you’ve learnt so far, write a program that converts Celsius to Fahrenheit.

Use the formula: (C \* (9/5)) + 32 = F

**DATA TYPES**

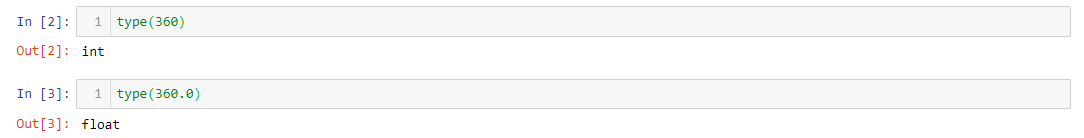
Data types specify how a data is to be handled. Python handles several data types that includes strings, Numbers, Boolean, list, date and time. These various data types are handled separately.

For instance; while + is used on strings for concatenation, it is used on numbers for addition. Python offers a way of identifying data types. The function “***type()***” is used to know what data type is currently being worked on. Let’s see examples of these data types:

**Strings (str):** Strings are basically text.



**Numbers:** Numbers are as the name implies. Numbers are segmented basically into float (float) and integer (int). Integers can’t contain decimals while floats are basically decimals.



Booleans (bool): This is a data type with only two values, which are True or False.

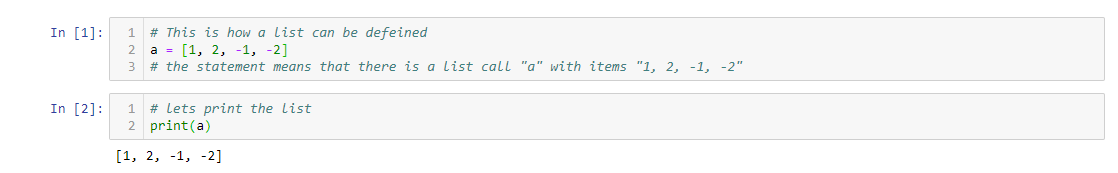


**LISTS DATA TYPE**

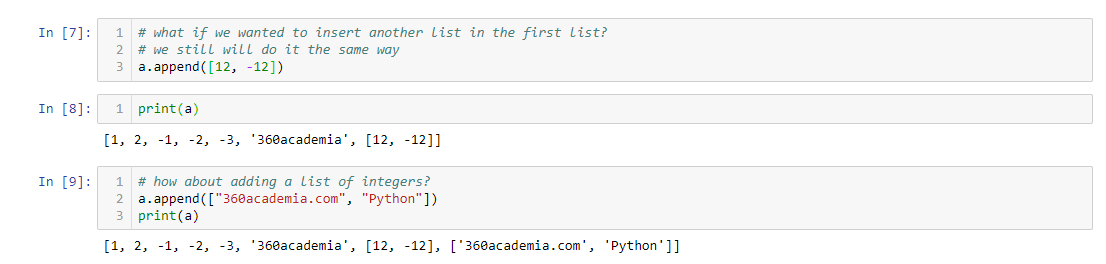
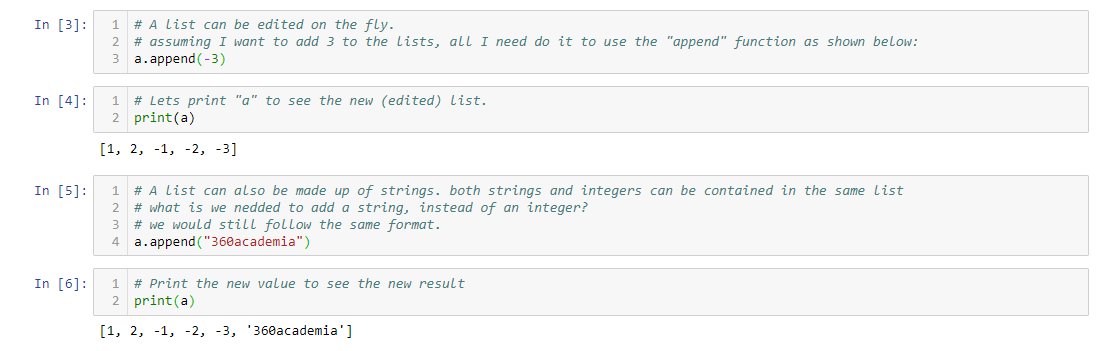
A list is a data type. It is used to store information of any type, kind and size. Assuming you need to buy a lot of items from a supermarket. You will need to store these items in a list of some kind.

Lists are usually numbered, and its numbering doesn’t start from 1, it starts from 0. If you have a list of 5 items, it wouldn’t be numbered from 1 through to 5, it would be number from 0 to 4 (which makes: 0, 1, 2, 3, 4).

Lets see some examples of how lists are applied and used.



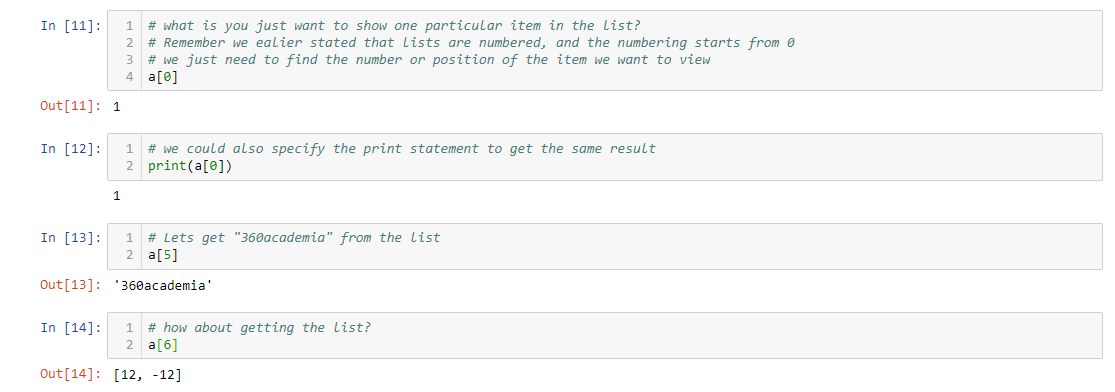
Adding items to already existing lists (***append***): The predefine function known as append helps this process.



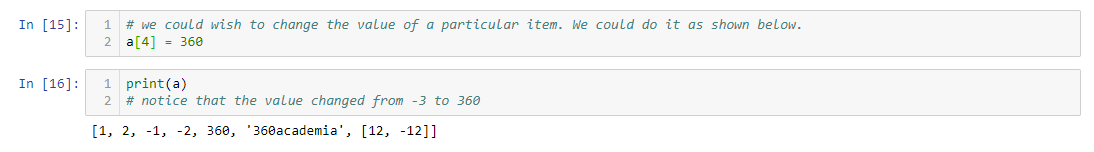
How about deleting or removing items from a list (pop)?



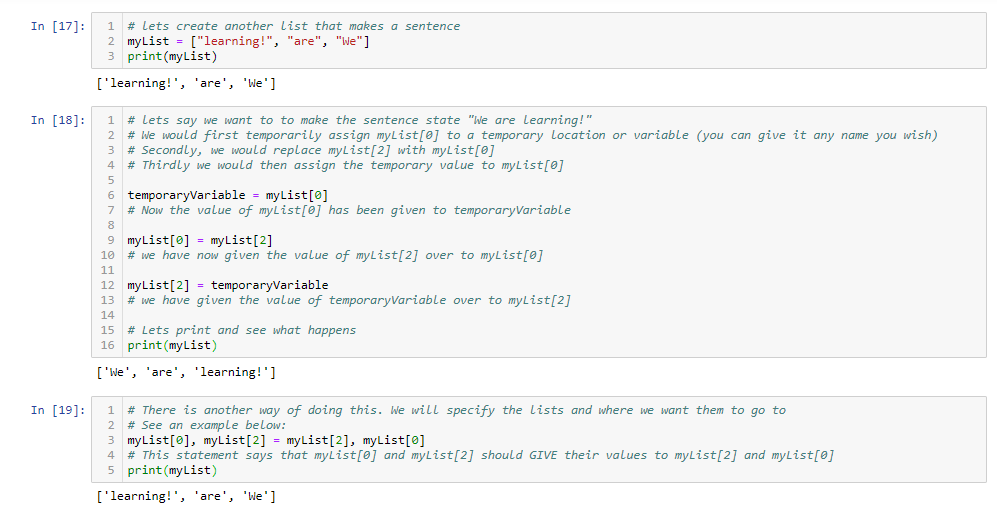
What if all we need is to see a particular item in a list?



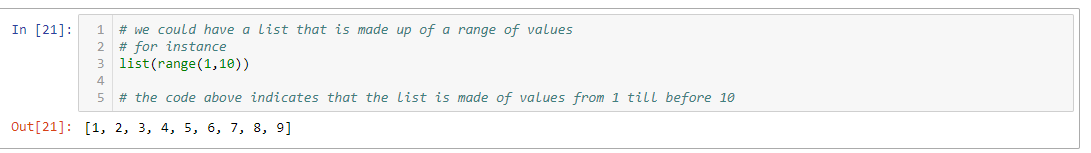
How about replacing a value in a list?



How about exchanging the value of a cell items?



We could have a list generated for use as shown below:



Try out something by yourself. Make a list for yourself as carry out the things we have just discussed.

**LOOP**

A loop is a programming construct that allows repetitive processing of a sequence of statements. There are basically two kinds of this loop statements which are for loop and while loop.

**FOR LOOP**

The general syntax:

for variable in list:

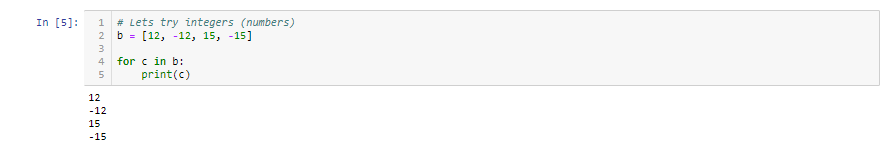
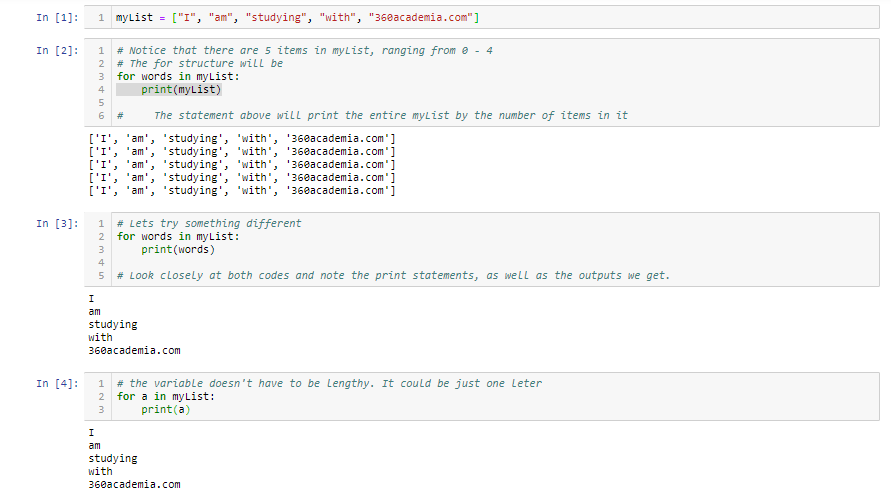
statements

else:

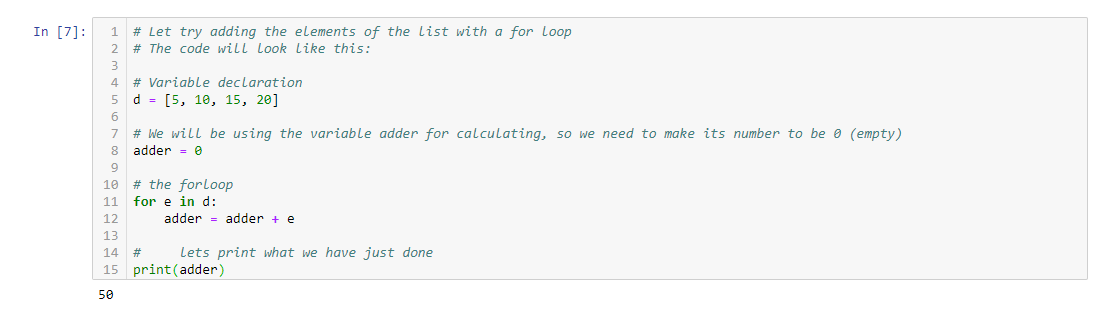
statements

Make no mistakes; loops and “if” statements look alike, but they are not the same things. Also pay attention to the right indentations when programming with python.

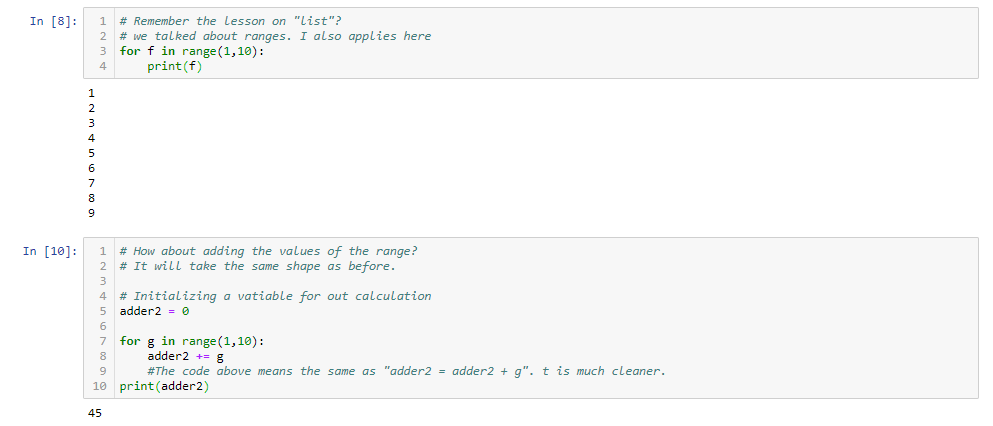
Let us work on some illustrations;



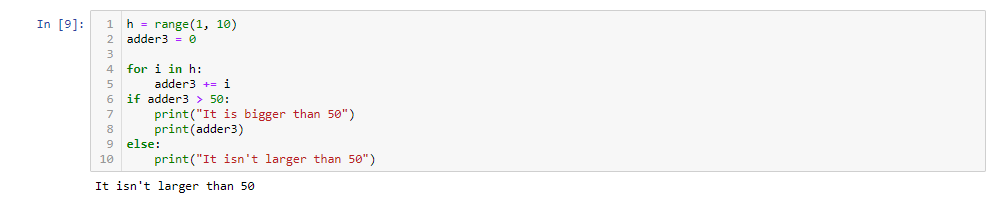
Let us try adding the values in list “b”.



A for loop with a range



How about having an “if else” statement to display the output?



Re try everything we just did here. It will help your understanding of for loops.

While loop

In Python, ‘while loop’ carries out a target statement repeatedly while a condition is true. When the case is no longer true, the control stops looping.

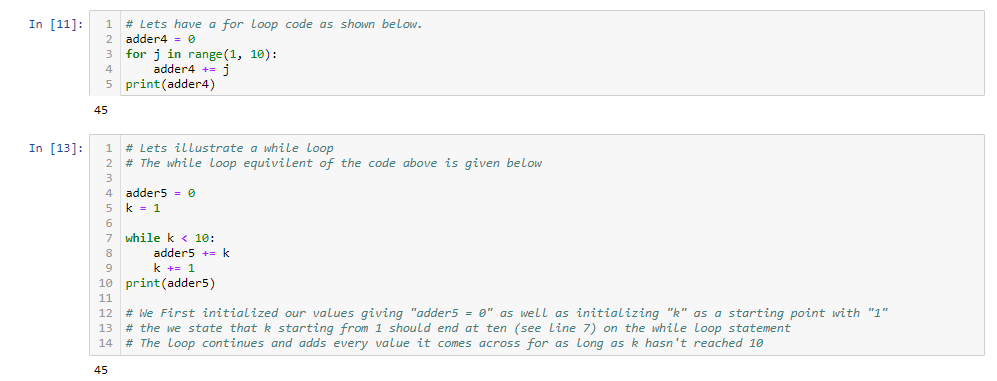
The ‘while loop’ syntax is:

while condition

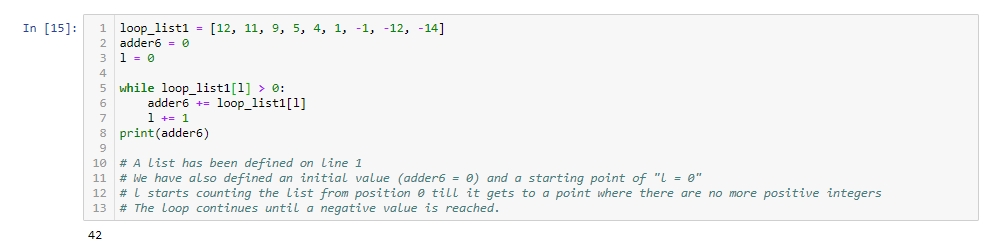
statement

See some illustrations below

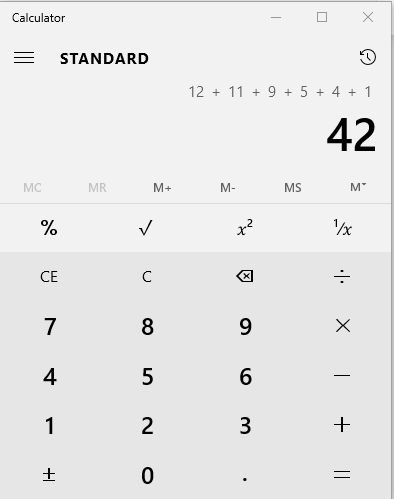
Lets translate a “for loop” code to a “while loop” code:



Lets now look at an illustration that makes while loops very important:



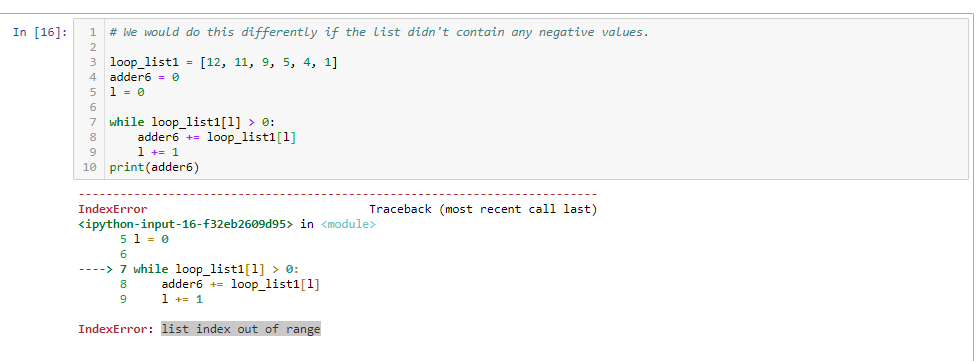
Lets add up the numbers with a calculator to see if our program works correctly.



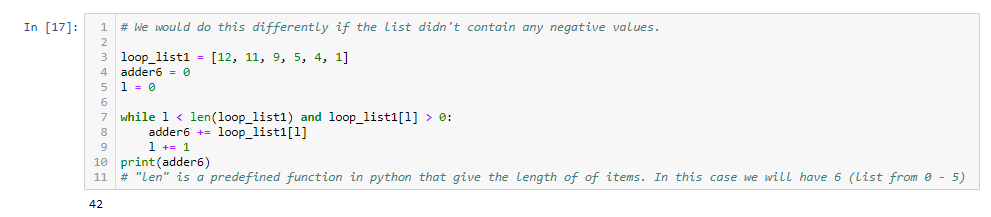
Our code works fine and we get the required result.

While loops come really handy in situations like the one we just stated above.

What if we didn’t have negative values? Maybe we just wanted to add up everything in the loop, and the loop didn’t contain any negative values.



Note the error we get. It is telling us that it went through the whole list and could fine even one negative value. We could fix this by asking it to just add-up the whole items within the range.

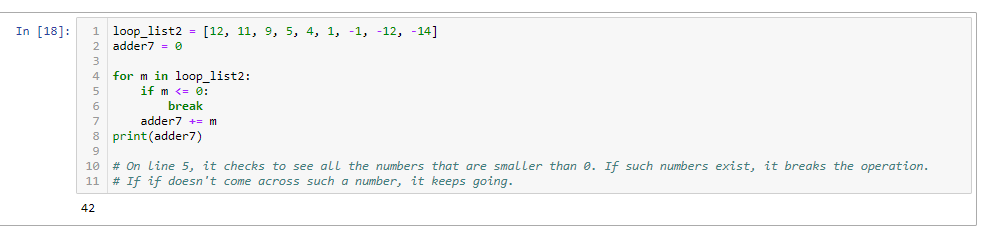


Note that we get the same value as what we got from the previous block.

For Loop Review:

Would it be possible to add-up positive numbers alone using a “for loop”?

It is very much possible. Let’s see an illustration.



Also note that we get the same output as the rest.

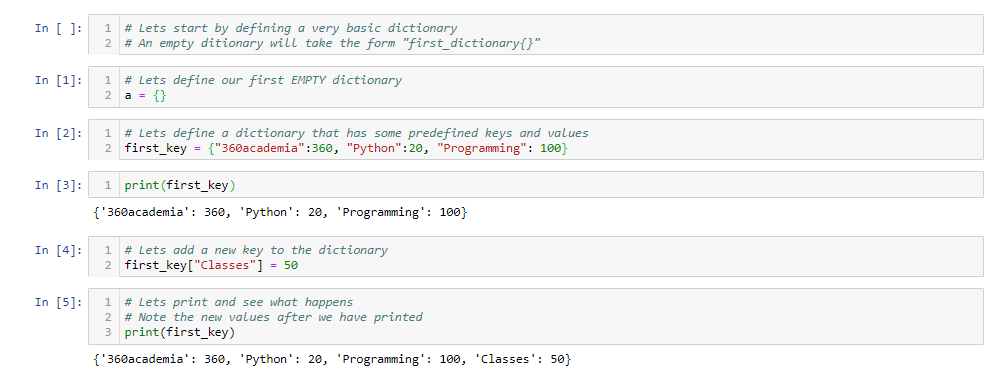
**DICTIONARY**

A dictionary is like a list but instead of looking up an index to access values, you will be using a unique key. Dictionary values can be anything nonetheless, the keys must be an immutable data type. Based on the syntax, a colon separates a key from its value and all are enclosed in curly braces.

A dictionary syntax is:

d = {first\_key : a, second\_key : 2, third\_key : abc}

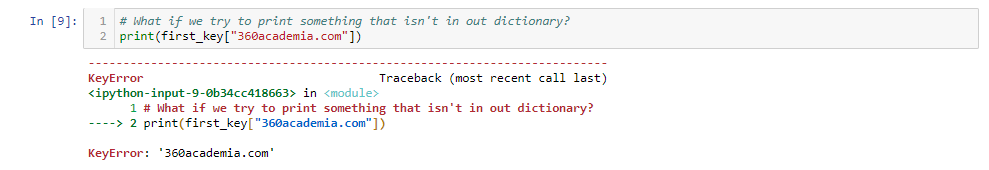
Lets see some illustrations:



Lets Print specific values from our dictionary:

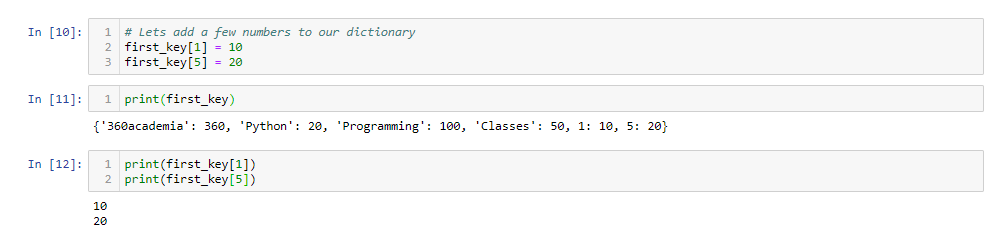


How about printing a key that doesn’t exist?



Since it doesn’t exist, we get an error.

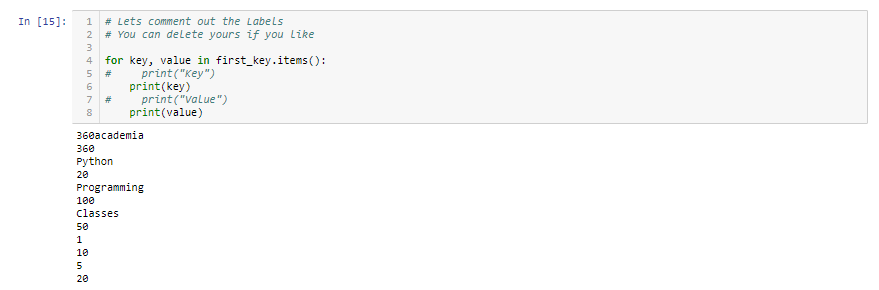
Lets add numbers to our dictionary.



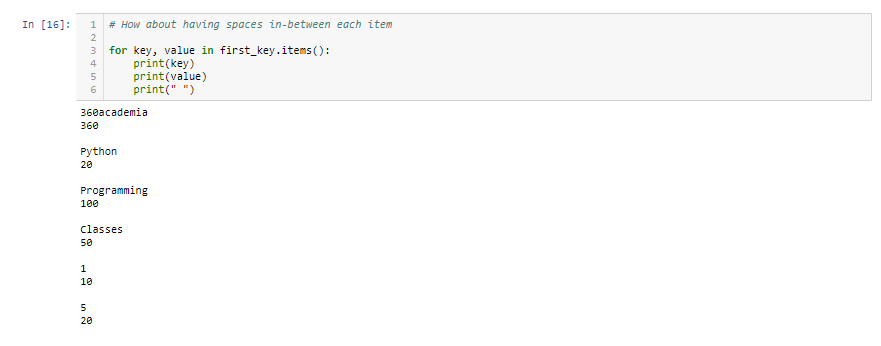
Lets print out everything that is within out dictionary.



Lets minimize what we can see. Comment out the irrelevant sprint statements.



Let us try creating spaces in-between each item:



Lets try adding the labels and spaces.

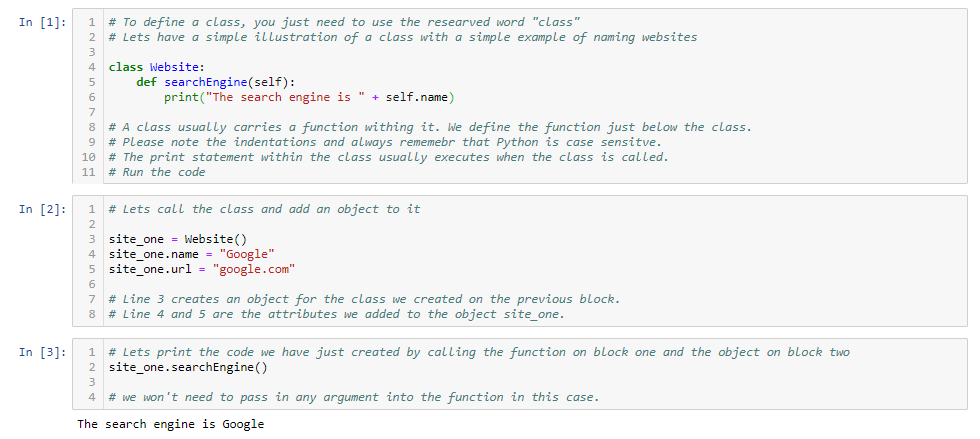


Classes and Objects

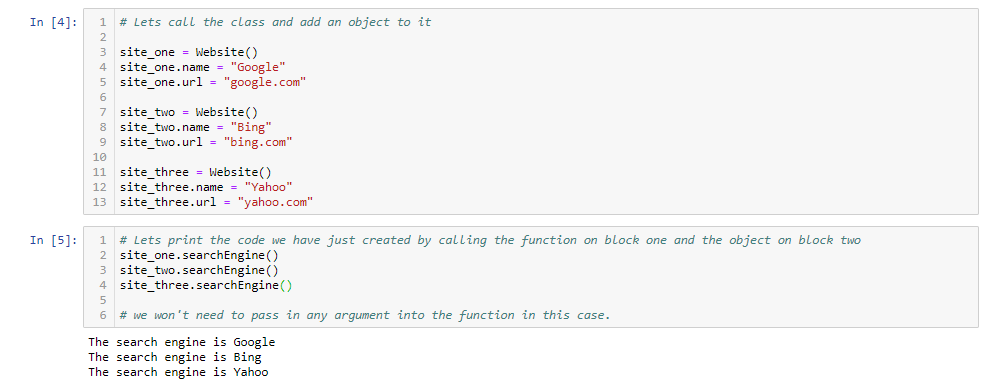
Python is an object-oriented programming language, which means that it manipulates and works with data structures called objects. Objects can be anything that could be named in Python. A class can be explained simply as a way of organizing, managing, and creating objects with the same attributes and methods.

Please Note: you need to understand the concept of functions before going through this segment. Please look through the previous lessons for the topic on functions.

Let’s see a little illustration.

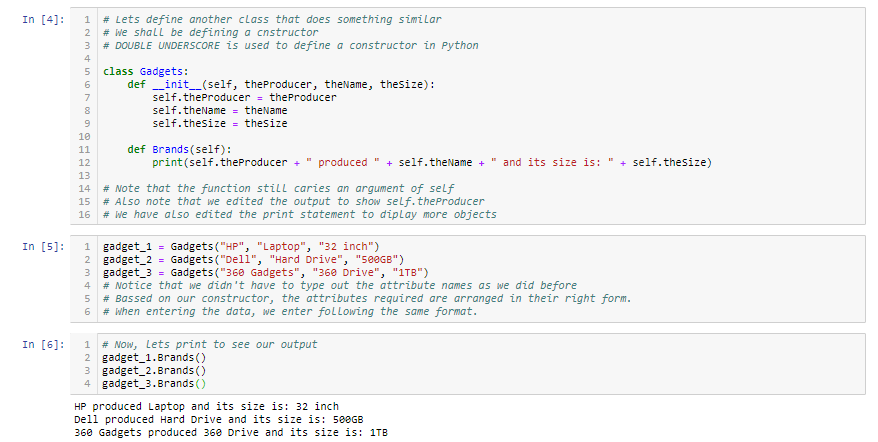


Let’s edit our code a little and two more objects and also print them.



This isn’t the best practice for classes. The use of a ***constructor*** is most recommended by us.

Let’s see an illustration.



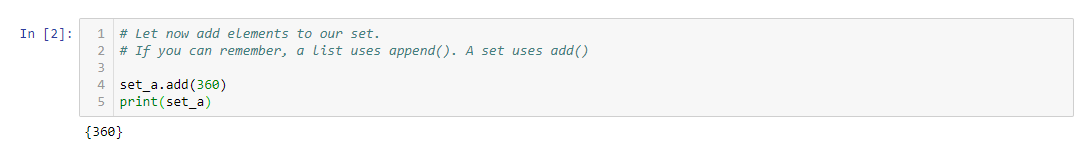
Set

A set is a type of data that stores a set of things. The items a set store MUST be unique. That is to say that a set will not store two things that are the same. Assuming you have a set of items, “1, 2, 3” and you wish to add “2”, the set won’t add it because it already exists. Let’s see some illustrations.

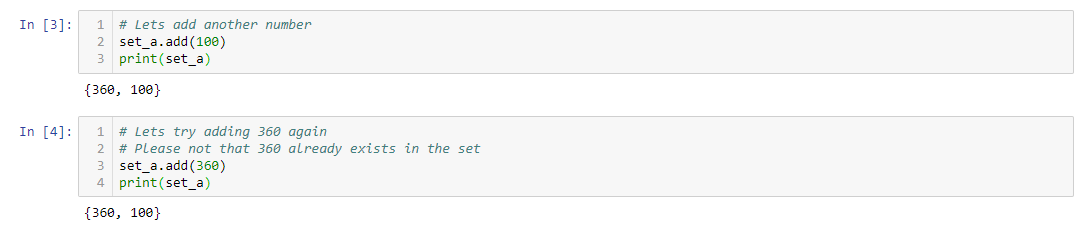
Lets start off by defining a set.



Lets add an item to the set



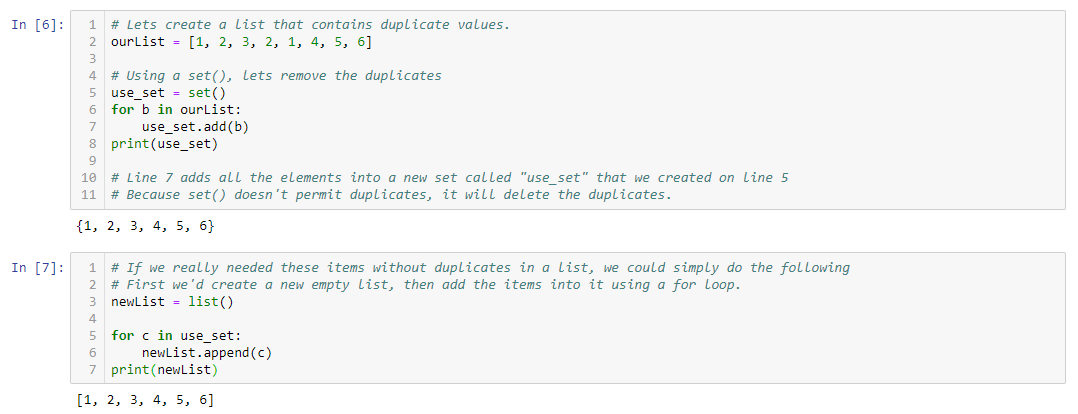
Lets try adding an item that already exists.



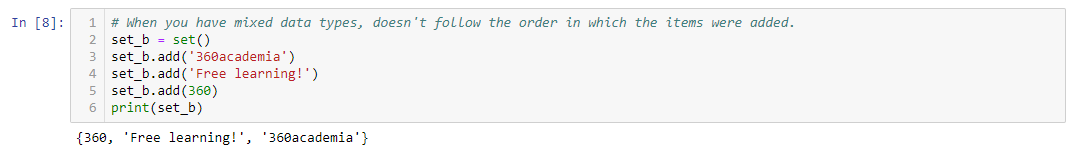
Let’s iterate over the elements in the set:



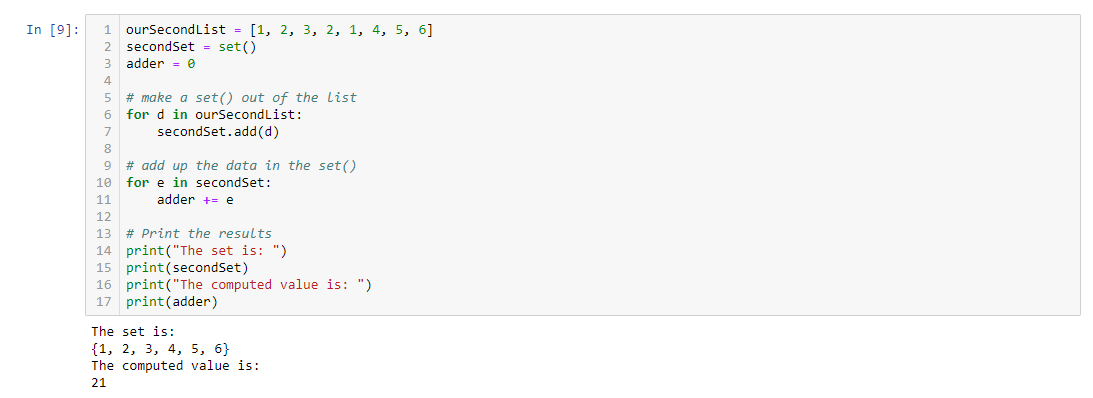
A set comes in very handy and useful for removing duplicates in a list. Let’s make a set with duplicate items, remove the duplicates and also making a new list with the created set.



We could add various data types in one set.



Let’s add up the data in a list, without the duplicates.



Boolean

Boolean operations have just two possible values, which are True or False, and not both at the same time. Comparisons in Python can only generate one of the two possible responses: True or False.

Let’s see a few examples:

