**PYTHON CRASH COURSE**

Python is a general purpose, interactive, object oriented, versatile and flexible language. It is easy to learn python as it uses English keywords.

Python was developed by Guido van Rossum. Python has a vast list of good features. Some of them are listed below:

* Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners
* Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.
* Python language is freely available at [official](https://www.python.org/) websites. The source-code is also available. Therefore, it is open source
* Python supports object oriented language and concepts of classes and objects
* It can be easily integrated with languages like C, C++, JAVA etc.
* Graphical user interfaces can be developed using Python.
* Python has a large and broad library and provides rich set of module and functions for rapid application development

In this tutorial, we will be using python 3.

Before working with Python, we should prepare our Python environment.

**INSTALLATION GUIDE:**

Very first important thing is python itself. We should have python and an IDE(integrated development environment).

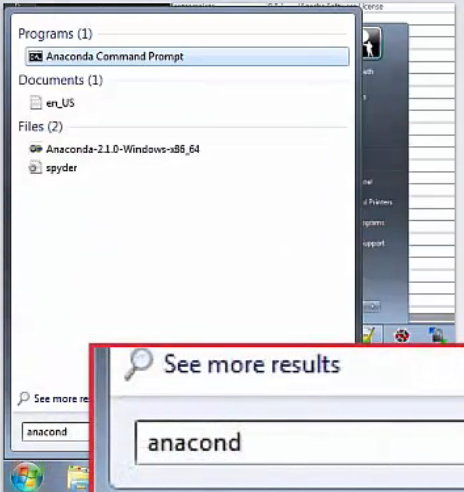
OR

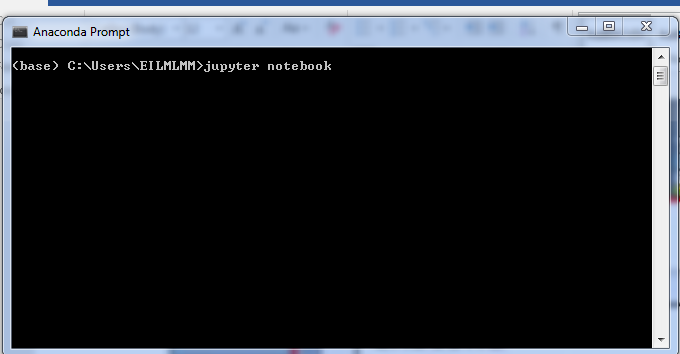
We will be using Jupiter notebook and latest version of python 3 through Anaconda distribution for this course.

Download Anaconda distribution from below website:

[**https://www.anaconda.com/download/**](https://www.anaconda.com/download/)

**STEPS:** Once anaconda distribution is installed, go to anaconda command Line and type jupyter notebook.

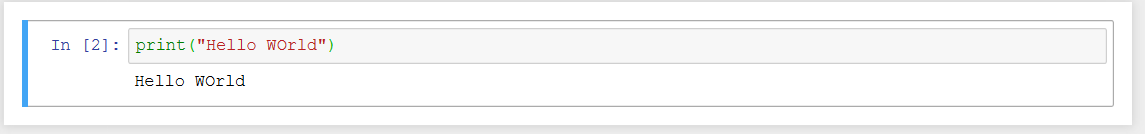




Jupyter notebook (previously called ipython) will open up in your browser.

Before we deep dive into problem solving, lets first learn the basics of python. Variables, Data structures, conditional loops, iteration all of these are fundamentals of any programing language.

As most of you have seen, “hello world” is the first program that we create in all programming language.



In python 3 we use parenthesis with print statement.

**VARIABLES:**

Reserved memory locations to store values. When we create a variable, we reserve some memory location.

In Python, we do not declare variable explicitly. Declaration happens automatically when we assign value to a variable.

Variables name cannot start with number and special symbols. Ex: 12counter

***Integer Variables:***



***Float Variable:***



***String Variable:***

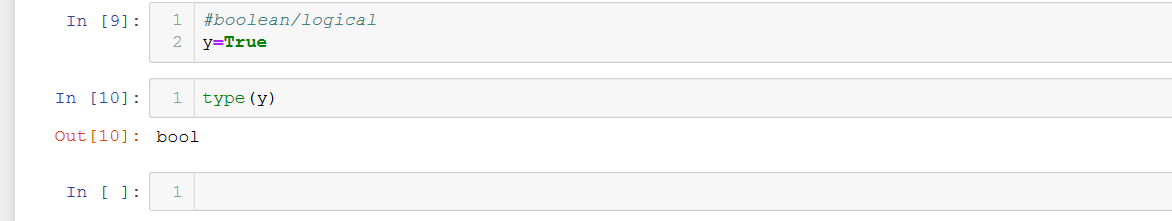
In python, we can use either single quotes or double quotes around the string.

Ex: “hello or ‘hello’, “2’ (a number represented as a string),’hello3’.



***Logical/Boolean variables:***

True and False



**IDENTIFIERS:**

Python Identifiers are user-defined names to represent a variable, function, class, module or any other object. If you assign some name to a programmable entity in Python, then it is nothing but technically called an identifier.

An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Hello and hello are two different identifiers in Python.

**KEYWORDS:**

Keywords are special words which are reserved and have a specific meaning. Python has a set of keywords that cannot be used as variables in programs.

All keywords in Python are case sensitive. So, you must be careful while using them in your code.

Some of the examples of Keywords are:

|  |
| --- |
| If break from del |
| False elif not as |
| True continue in import |

It’s a long list to remember all at once. The purpose of mentioning it here is only to give you an initial idea of the available keywords.

**OPERATORS IN PYTHON:**

Operators are symbols that are used to perform operations on operands. It returns result that can be used in application.

***Arithmetic Operators:***

Arithmetic Operators perform various arithmetic calculations like addition, subtraction, multiplication, division, %modulus, exponent, etc. There are various methods for arithmetic calculation in Python like you can use the eval function, declare variable & calculate, or call functions.

|  |  |
| --- | --- |
| // | Perform Floor division (gives integer value after division) |
| + | To perform addition |
| - | To perform subtraction |
| \* | To perform multiplication |
| / | To perform division |
| % | To return remainder after division(Modulus) |
| \*\* | Perform exponent (raise to power) |

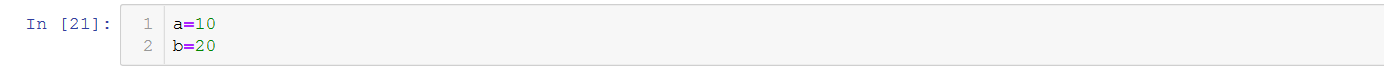




***Relational OPERATORS:***

These operators compare the values on either side of the operand and determine the relation between them.

|  |  |
| --- | --- |
| **Operators** | **Description** |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| == | Equal to |
| != | Not equal to |
| <> | Not equal to (similar to !=) |





***Assignment Operators:***

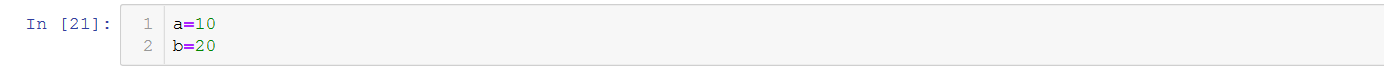
Python assignment operators are used for assigning the value of the right operand to the left operand.

|  |  |
| --- | --- |
| = | Assignment |
| /= | Divide and Assign |
| += | Add and assign |
| -= | Subtract and Assign |
| \*= | Multiply and assign |
| %= | Modulus and assign |
| \*\*= | Exponent and assign |
| //= | Floor division and assign |

***Logical Operators:***

Logical operators in Python are used for conditional statements are true or false. Logical operators in Python are AND, OR and NOT.

|  |  |
| --- | --- |
| **Operators** | **Description** |
| and | Logical AND (When both conditions are true output will be true) |
| or | Logical OR (If any one condition is true output will be true) |
| not | Logical NOT (Compliment the condition i.e., reverse) |



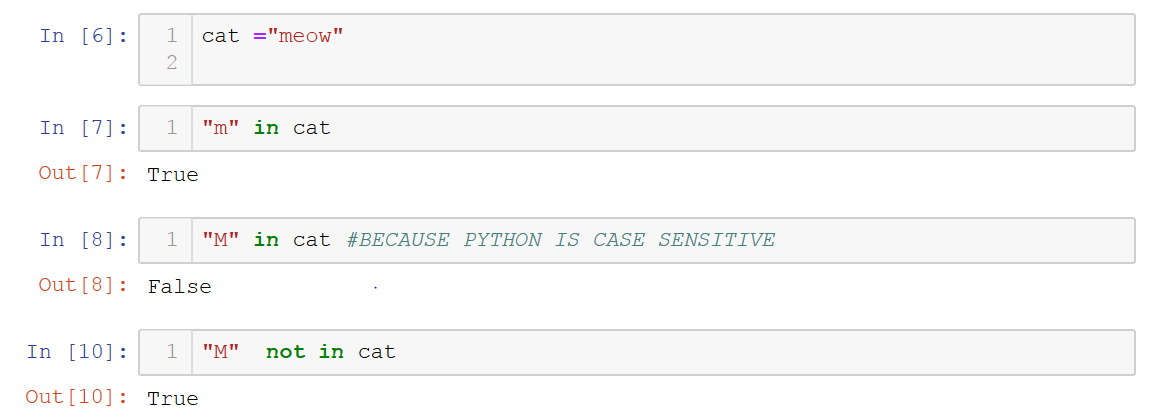




***Membership Operator:***

These operators test for membership in a sequence such as lists, strings or tuples. There are two membership operators that are used in Python. (in, not in). It gives the result based on the variable present in specified sequence or string

|  |  |
| --- | --- |
| **Operators** | **Description** |
| in | Returns true if a variable is in sequence of another variable, else false. |
| not in | Returns true if a variable is not in sequence of another variable, else false. |



***Identity Operators:***

To compare the memory location of two objects, Identity Operators are used. The two identify operators used in Python are (is, is not).

|  |  |
| --- | --- |
| **Operators** | **Description** |
| is | Returns true if identity of two operands are same, else false |
| is not | Returns true if identity of two operands are not same, else false. |

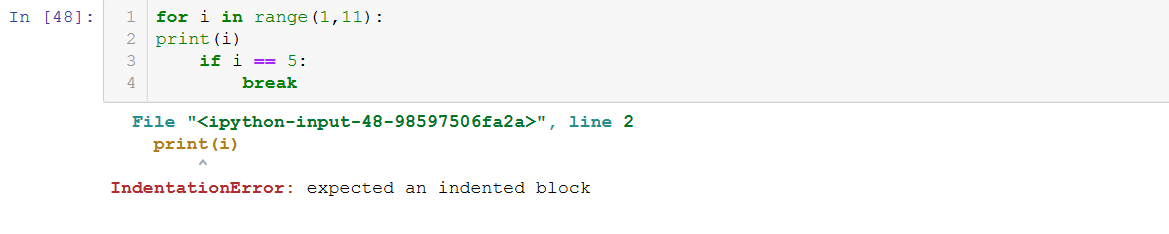
**IDENTATION:**

Indentation plays very important role in python. Blocks of code in python are denoted by indentation.

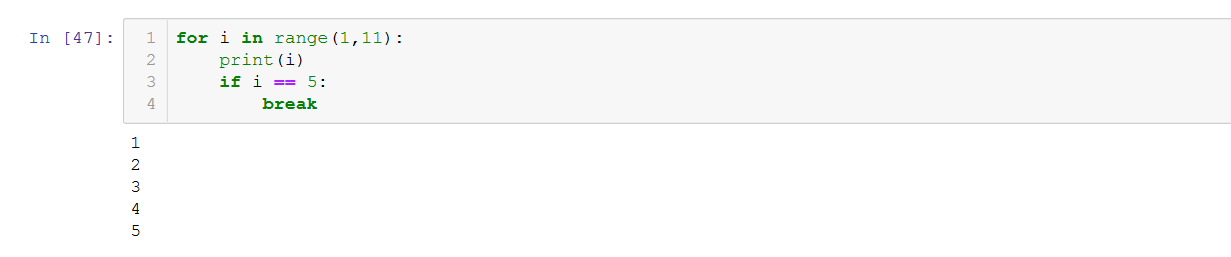
The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.

Generally, four whitespaces are used for indentation and is preferred over tabs. Here is an example.

Python interpreter will give indentation error.



Correct indentation:



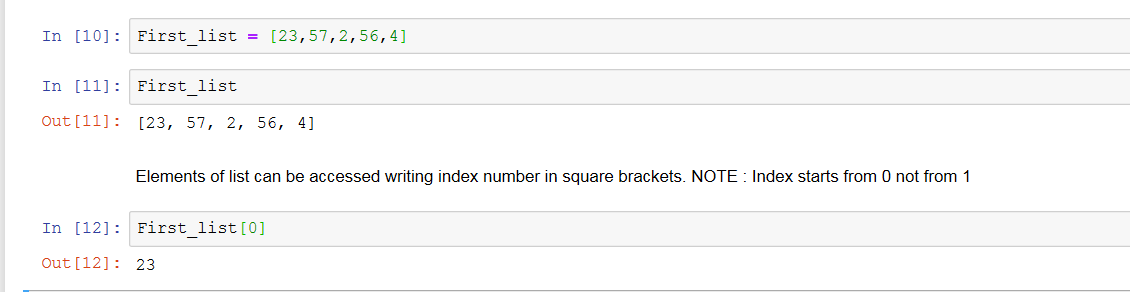
**DATA STRUCTURES IN PYTHON:**

Below are some data structures, which are used in Python. You should be familiar with them to use them as appropriate.

* List
* Strings
* Tuple
* Numbers
* Dictionary

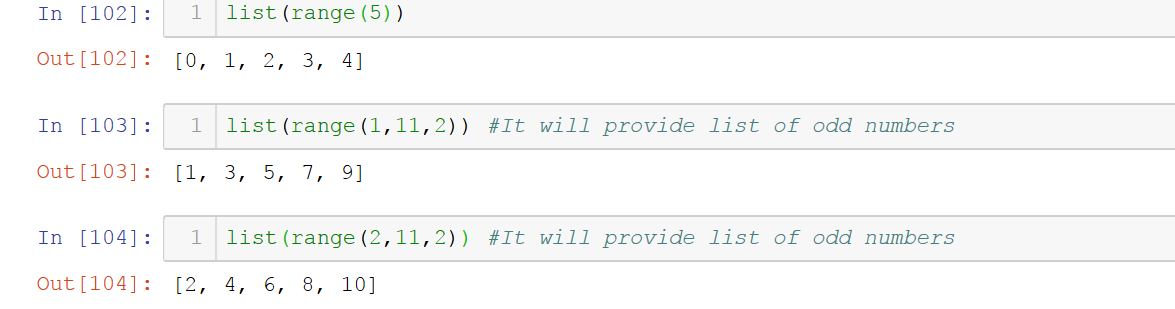
**LISTS:**

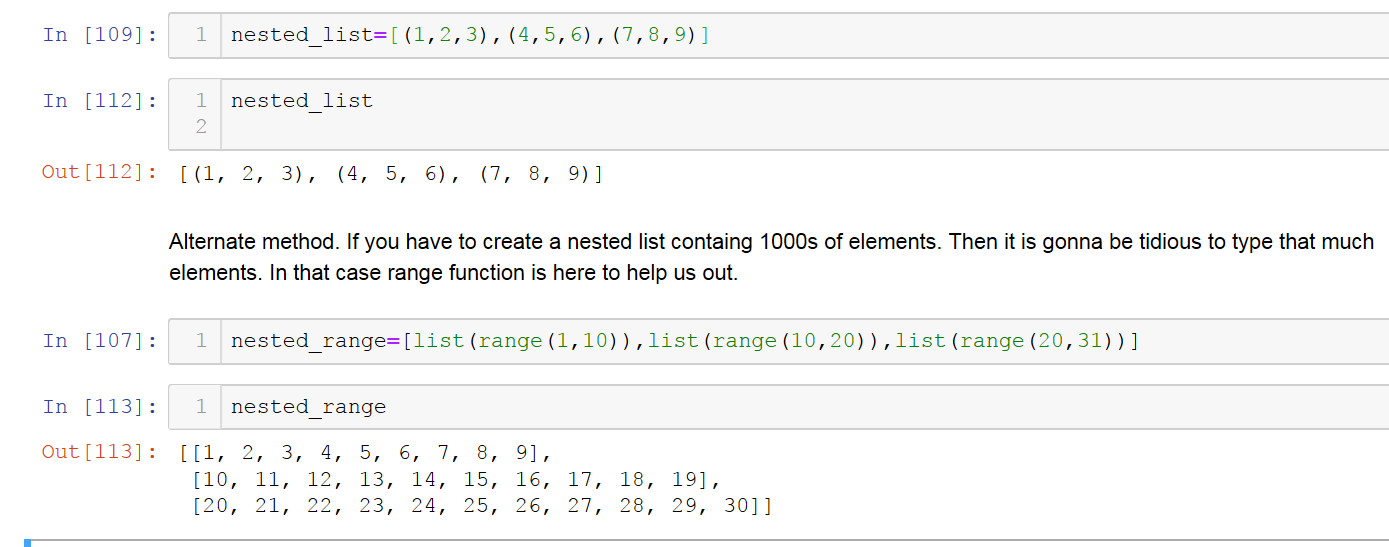
Lists are one of the most versatile data structure in Python. A list can simply be defined by writing a list of comma separated values in square brackets. Lists might contain items of different types, but usually the items all have the same type. Python lists are mutable and individual elements of a list can be changed. Items within list can be accessed through index. List can contain list within it.



***Range function:***

Range generates a specified list of numbers.





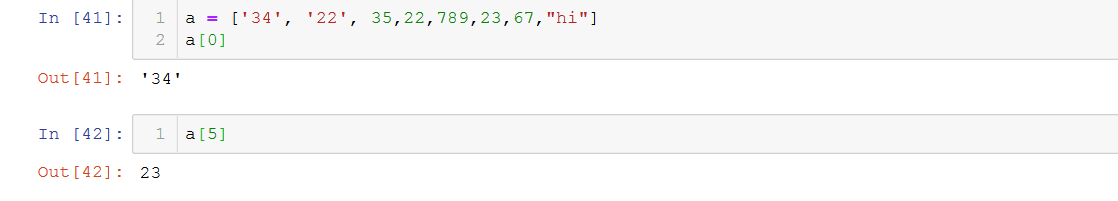
***Slicing:***

To get a sequence of elements from a list we use slicing.

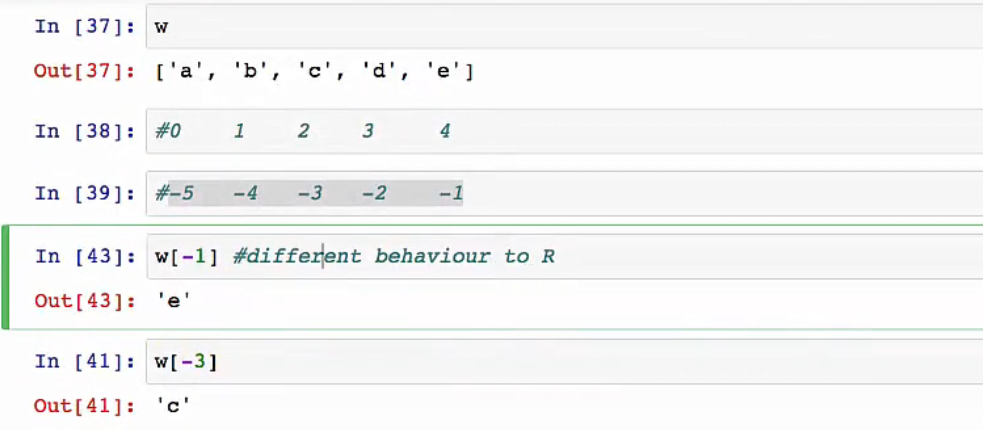


***Indexing:***

Reassigning of elements in list is also possible using index notation. Indexing starts from 0.

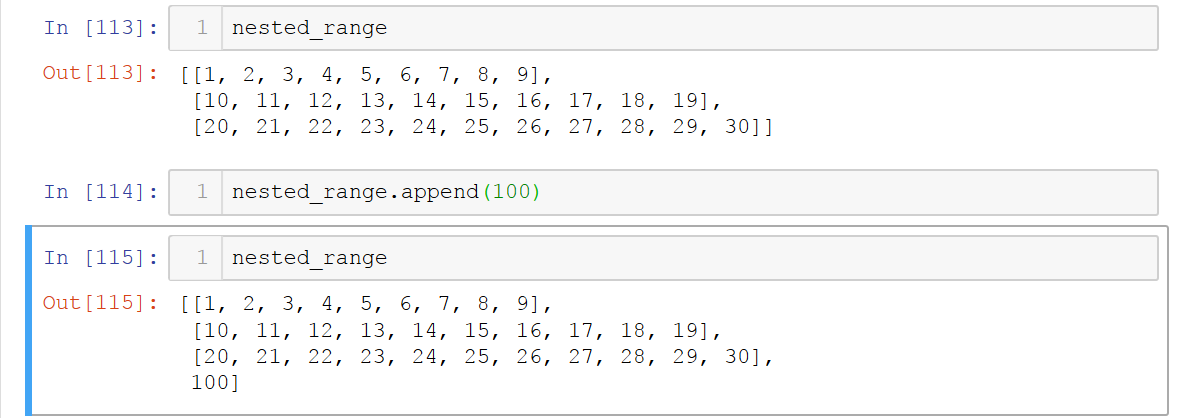


***Negative Indexing:***



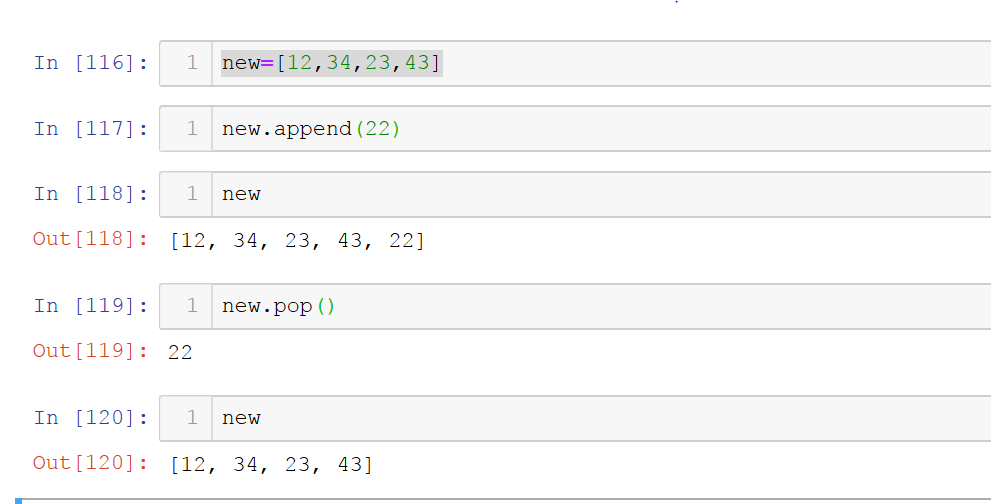
***Append function:***

It will add the element at the end of the list



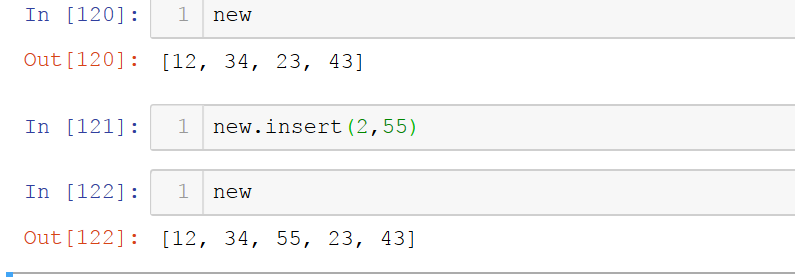
***POP function:***

It will remove the final element from the list



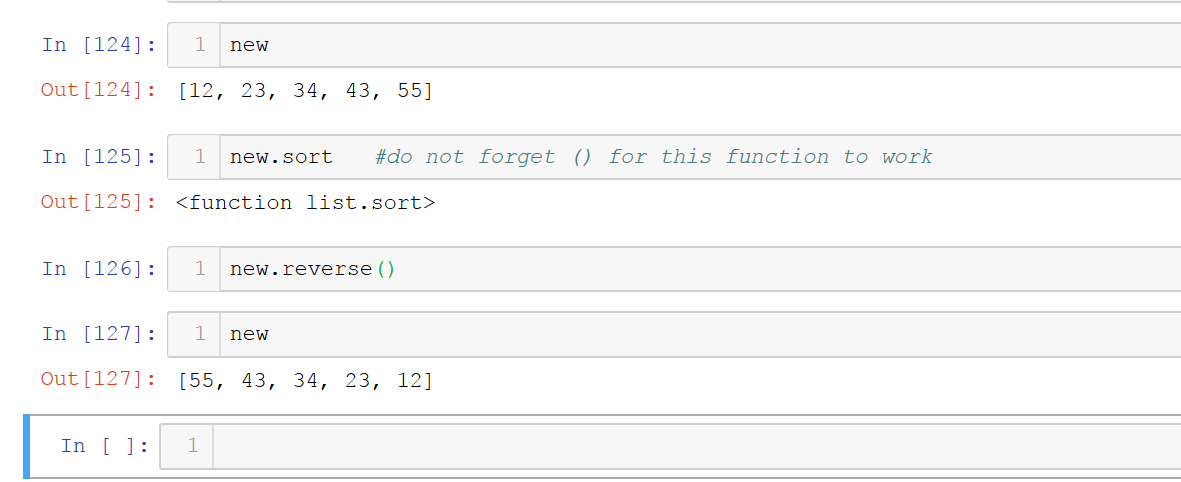
***Insert Function:***

It will allow us to put an element at a specific location that we specify.

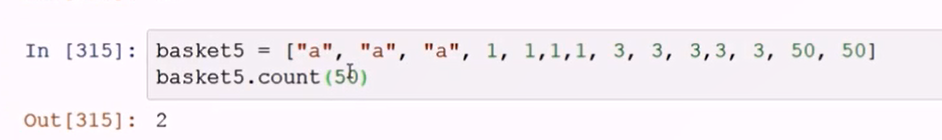


***Sort and Reverse Function:***

Arrange the elements from lower to higher(sort) and higher to lower sequence(reverse).



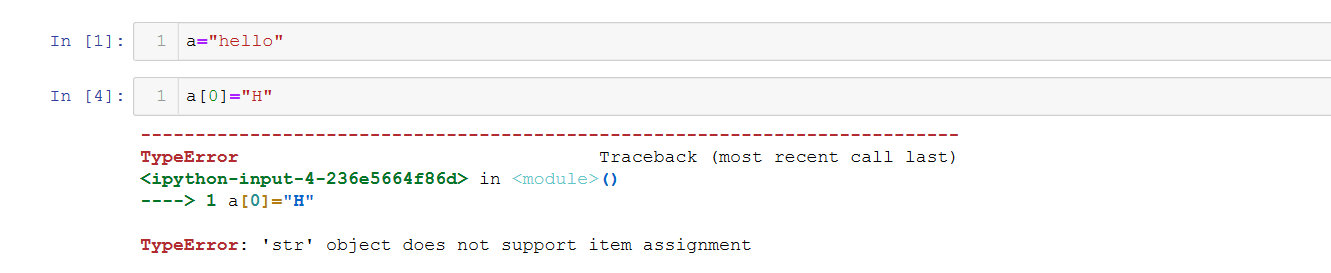
***Count:***



**STRINGS:**

Strings are defined with single inverted comma (‘) and double inverted commas (“). Strings are immutable.

Strings are sequence of elements and we can grab each character from its index number.





**Indexing:**

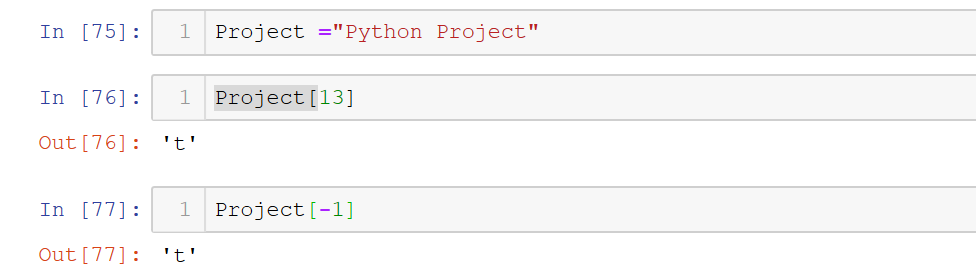
In Python strings each character is given an index from zero (at the beginning) to the length minus one (at the end).

For the string "Python", the indexes break down like this:

P y t h o n

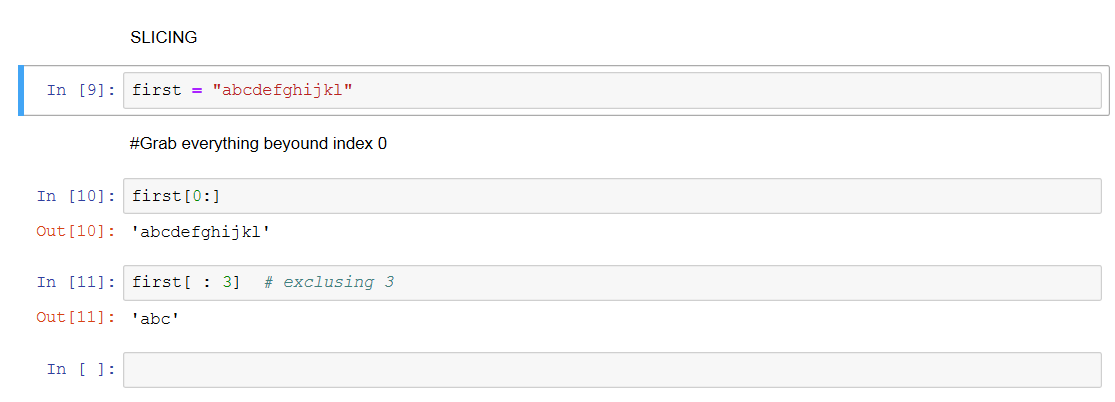
0 1 2 3 4 5

In addition, Python supports negative indexes, in which case it counts from the end. So the last character can be indexed with -1, the second to last with -2, etc.:



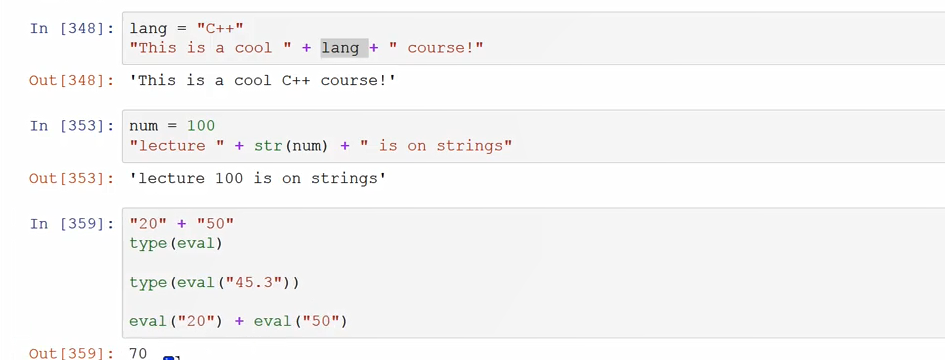
**Slicing in string:**

To get a sequence of elements from a string we use slicing.



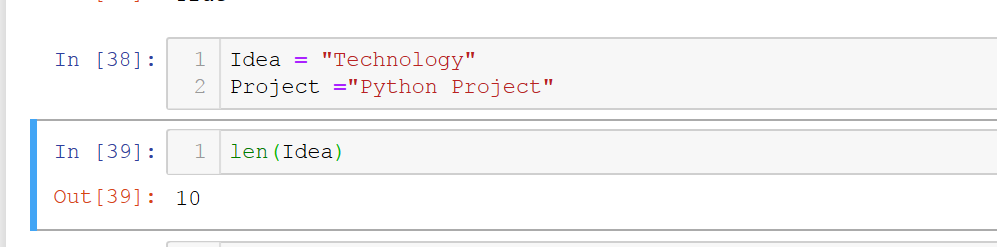
**String Concatenation:**

To add two string, we can perform String concatenation

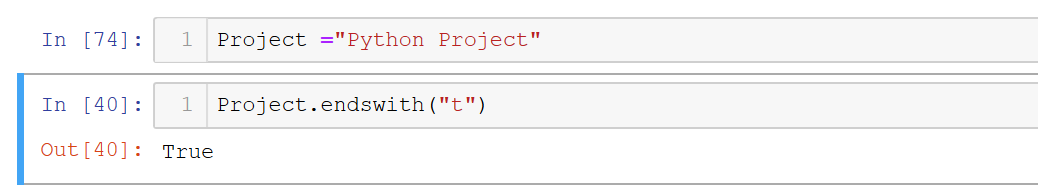


Len Function is a build in function. Which counts the numbers of elements in a list, string and dictionary.

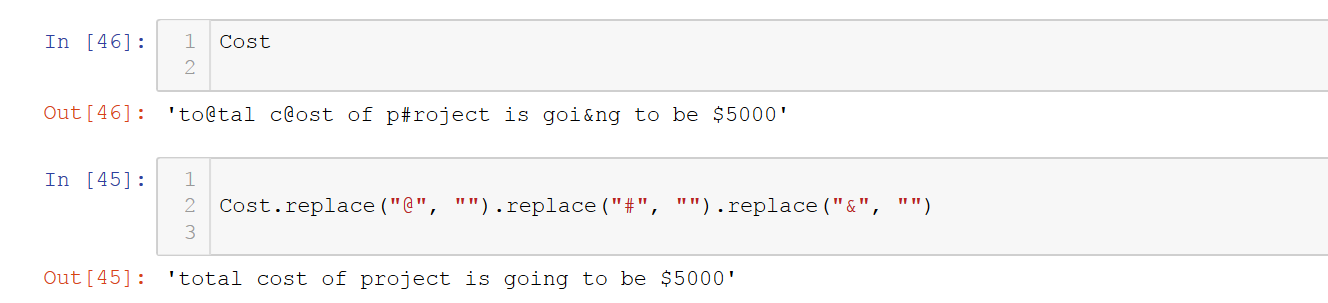
NOTE: Len starts counting elements from 1 whereas indexing starts from 0.



***ends with*** is built in function, it helps us to check if the particular string ends with the specified element or not.



***Replace function***



***Strides:***

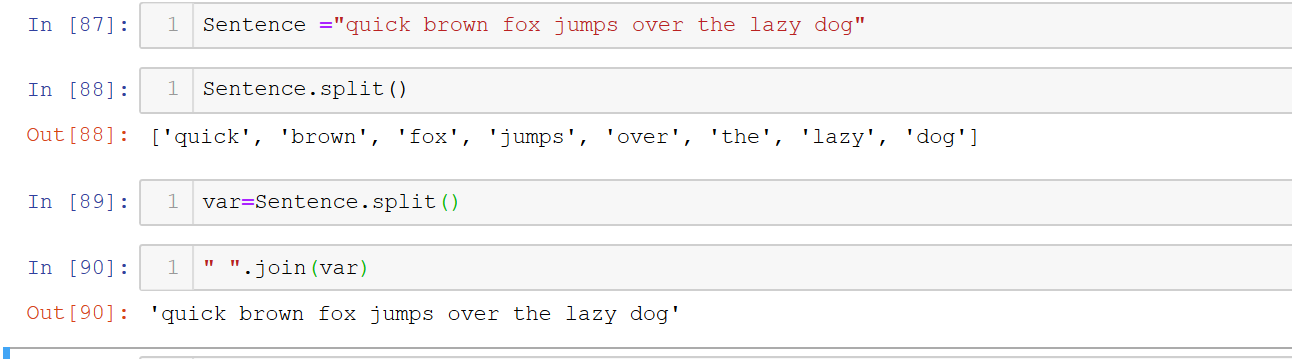
It Slice the string with gap of specified elements.

Ex: Var[2:8:2]

Slice the string beginning from index 2 to 8 with a jump of 2 elements.

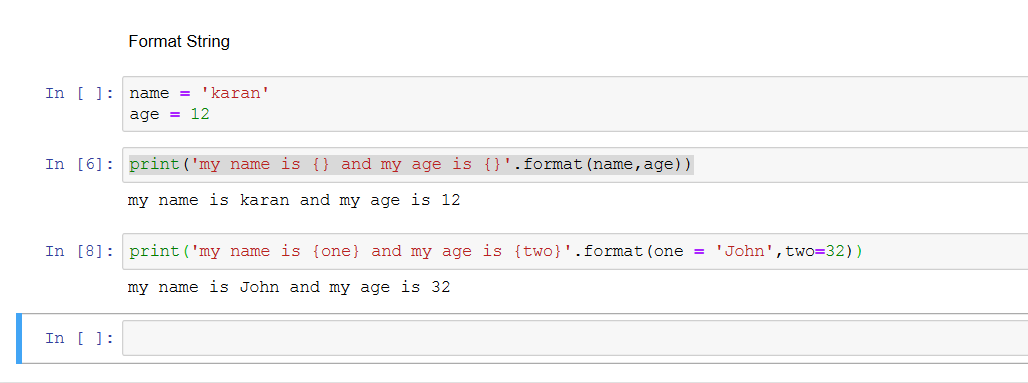


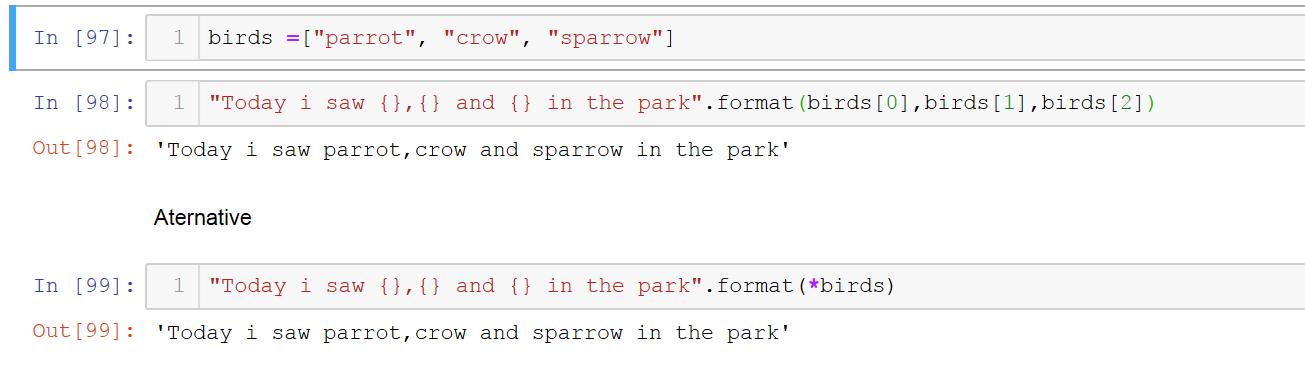
***Split and join function:***



***Format strings:***

Built-in python function and method





***EVAL Function:***

**eval**() evaluates the passed string as a **Python** expression and returns the result. For example, **eval**("1 + 1") interprets and executes the expression "1 + 1" and returns the result (2).



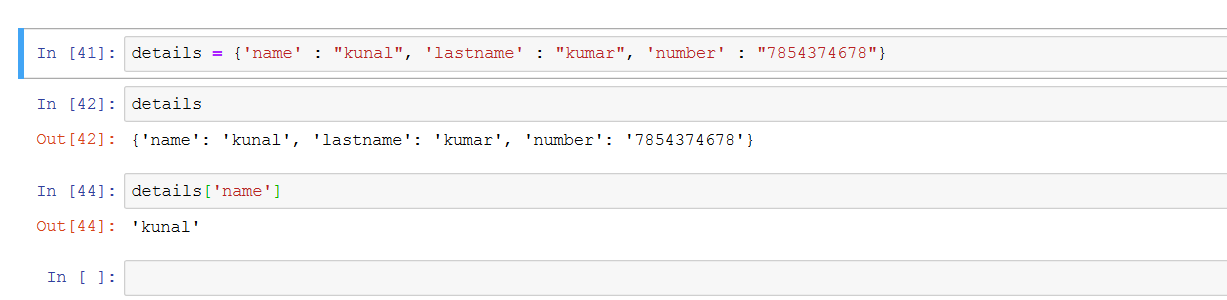
**DICTIONARY:**

Dictionary is unordered set of key-value pair. Pair of braces creates an empty dictionary.

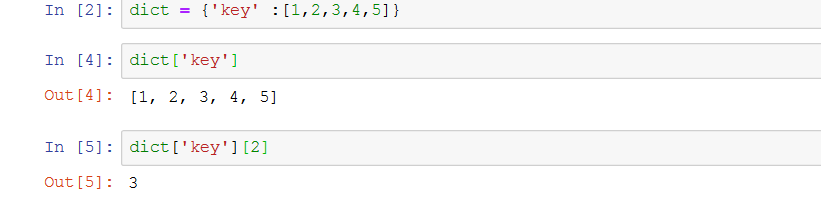
We cannot access elements of using index notation.

We use key corresponding to that value element to access elements.

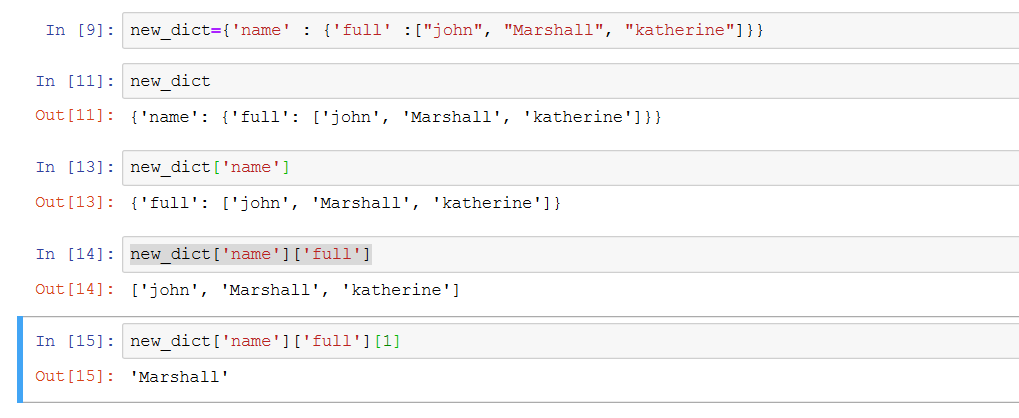


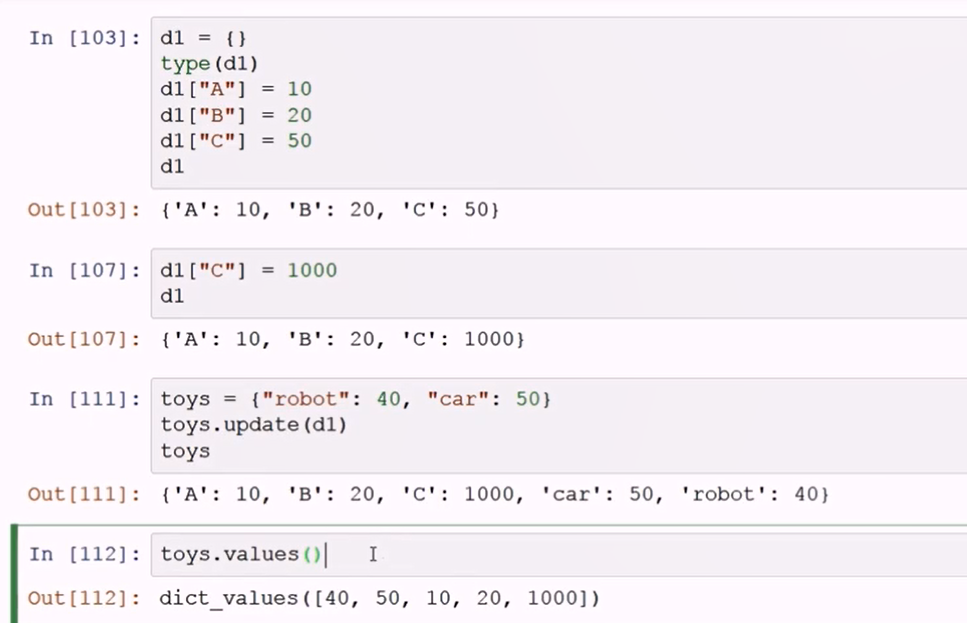


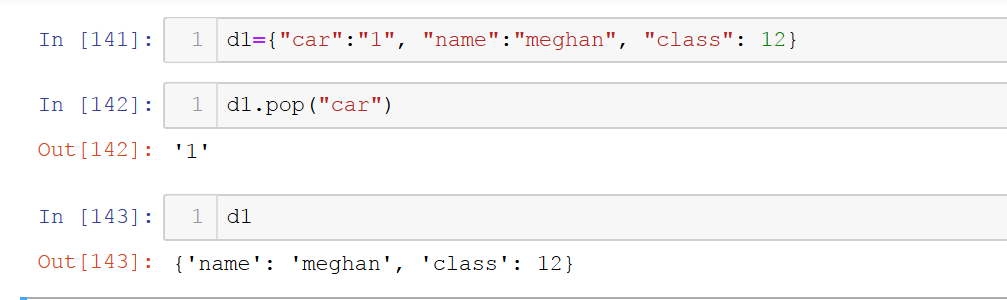
Dictionaries can take any items as their value.

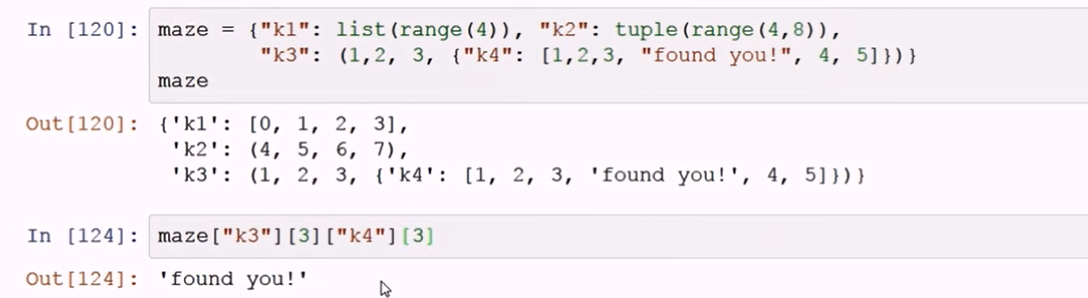


***Nested Dictionary:***





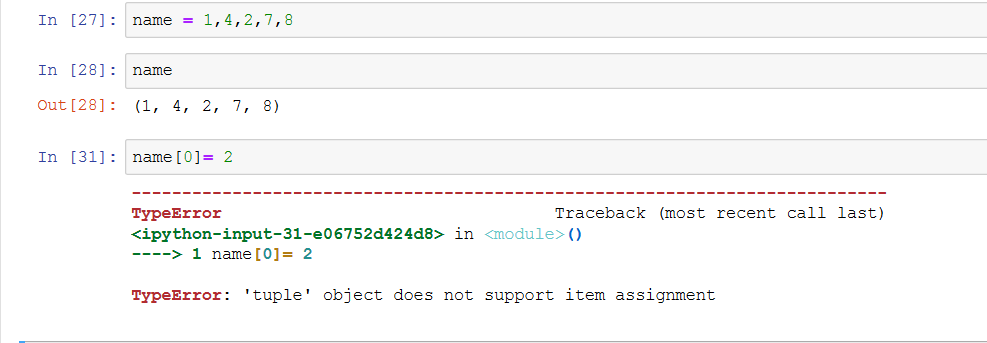


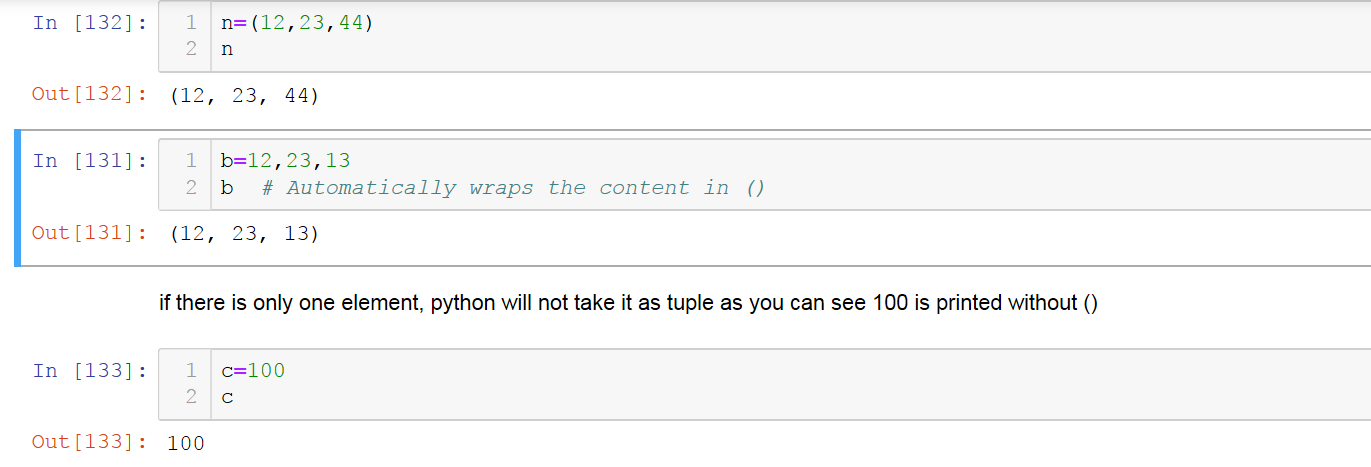


**TUPLES:**

Sequence of elements inside brackets (). Elements of Tuples are immutable.

Output will show in parenthesis. Tuples are immutable.



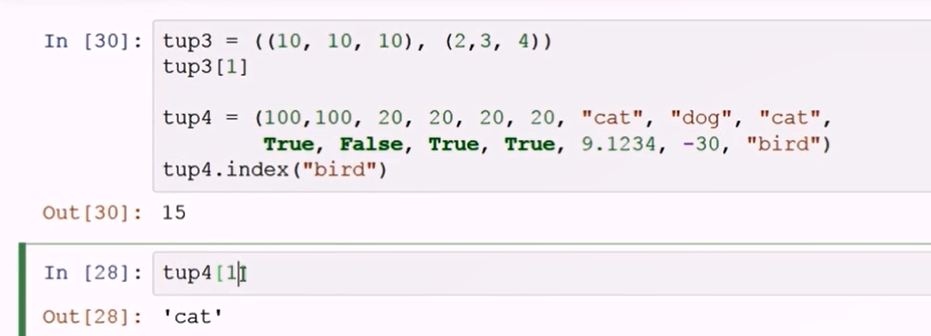


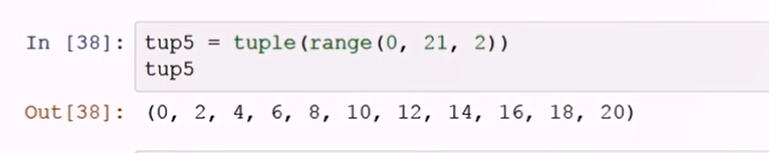
Tuple has only 2 functions: index and count.

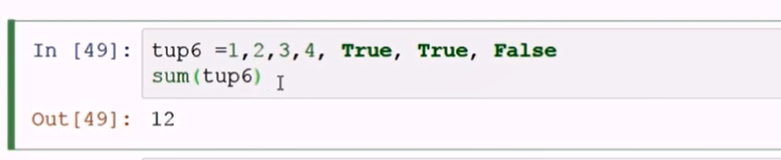


We have to use tuples when we are sure that user is not going to change the sequence of elements.

We can create nested tuples as well (just like we did previously in list)



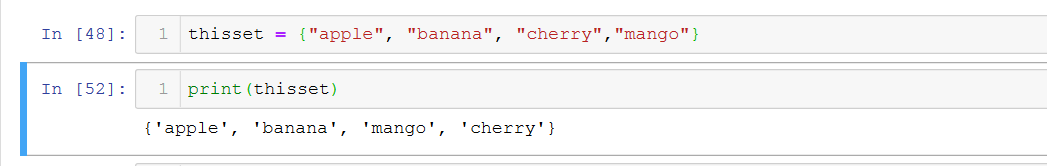




**SETS:**

Collection of unique elements inside curly brackets.

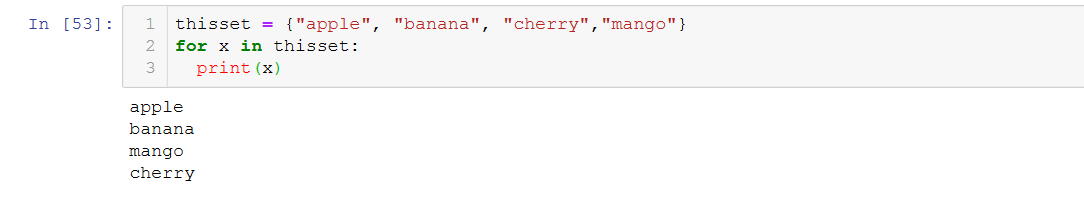
Sets are unordered, so the items will appear in a random order.

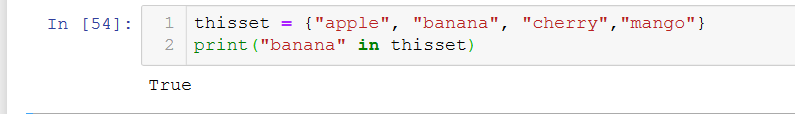


***Access Elements:***

You cannot access items in a set by referring to an index, since sets are unordered the items has no index.

But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using thein keyword.



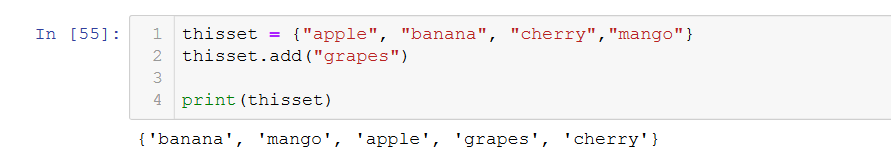


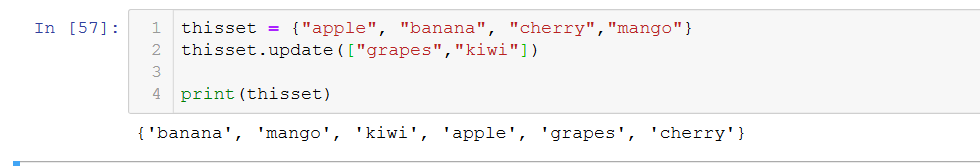
***Change Items:***

Once a set is created, you cannot change its items, but you can add new items.

To add one item to a set use the add () method.

To add more than one item to a set use the update () method.





**CONTROL FLOW:**

IF, ELSE, ELIF

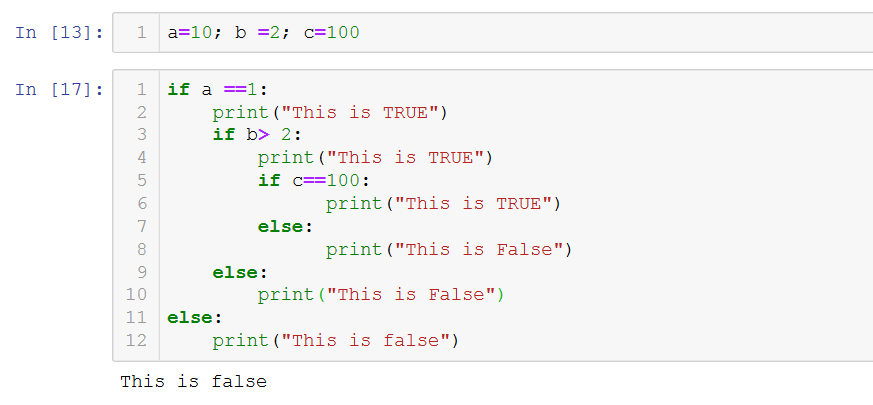
* Adds logic to our code.
* If statement performs a task if the condition is true.
* Else statement performs a default when the if statement’s condition is false.
* Elif (else if) works in between if and else statements.

**NOTE: Depends on Indentations**

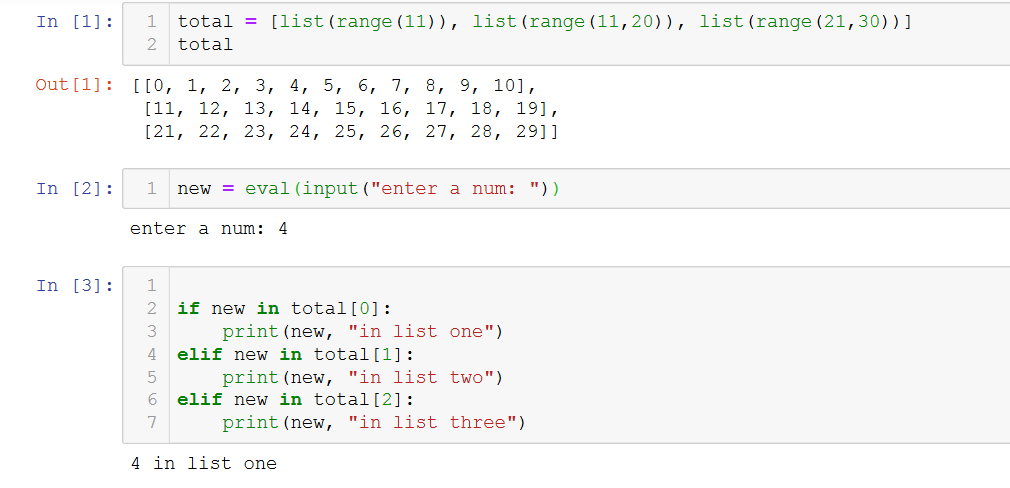
***If statement:***

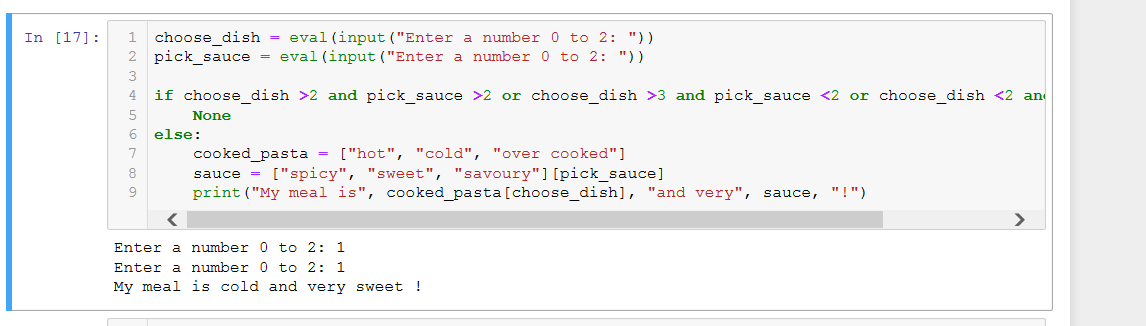
***If else statement:***

***Nested If else:***



***Multiple condition:***



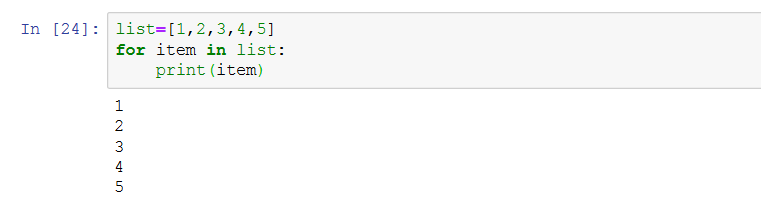


**LOOPS IN PYTHON:**

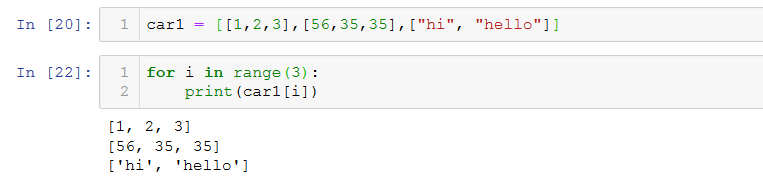
Loops can execute a block of code number of times until a certain condition is met. Their usage is common in programming.

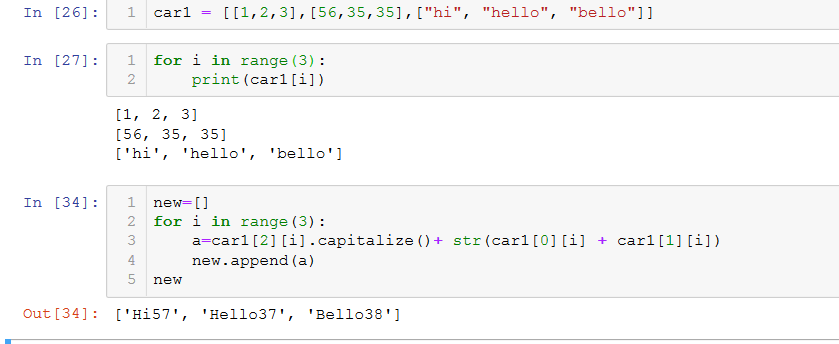
**FOR LOOPS:**

Allows you to iterate through a sequence (list, tuple or dictionary).

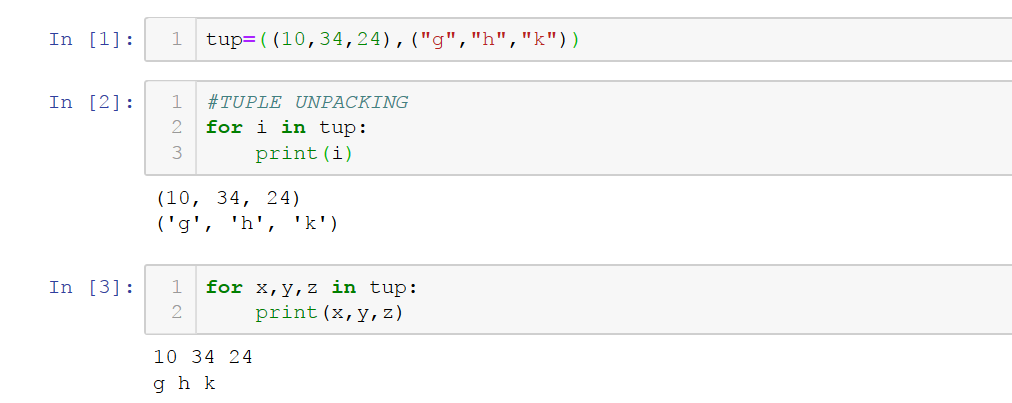


Here, item is temporary variable.

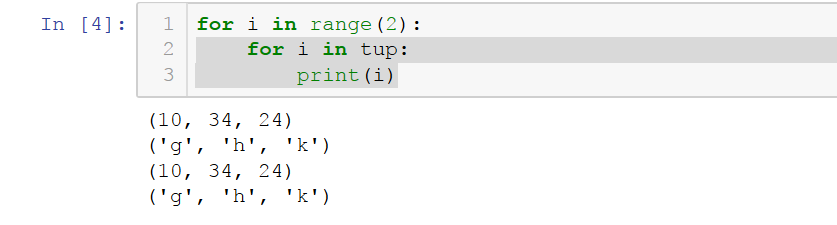




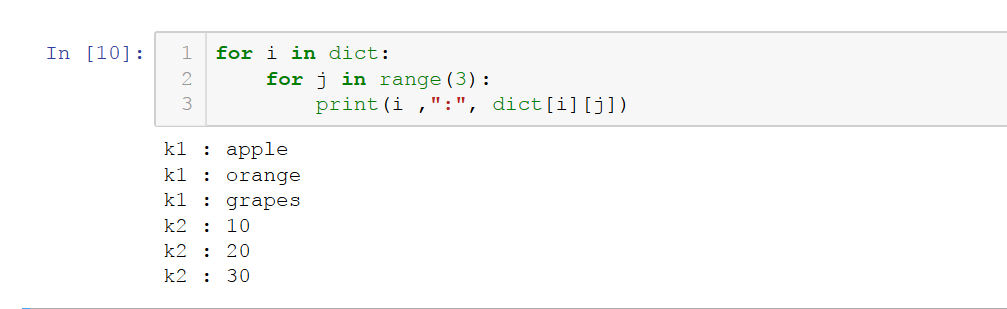
***TUPLE UNPACKING:***



**Nested For loop:**



dict = {“k1": ["apple", "orange”, “grapes"], "k2": ["10", "20","30"]}



**WHILE LOOP:**

Continue performs an action until the mentioned condition is met.

Once the while statement is no longer true, the while loop ends.

***Syntax:***

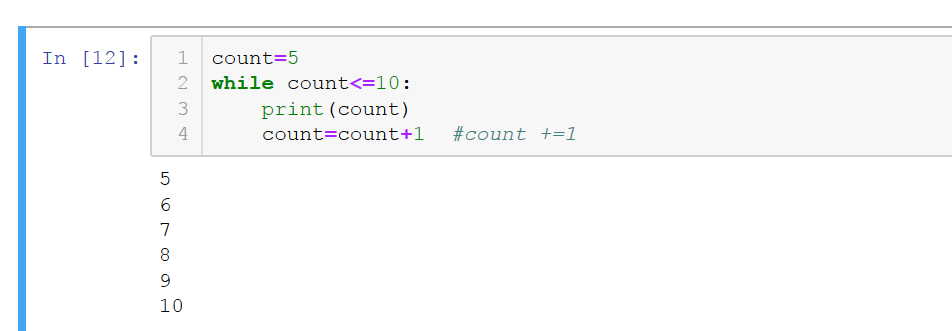
*While(condition):*

*Executable code1*

*Executable code2*

*Executable code3*

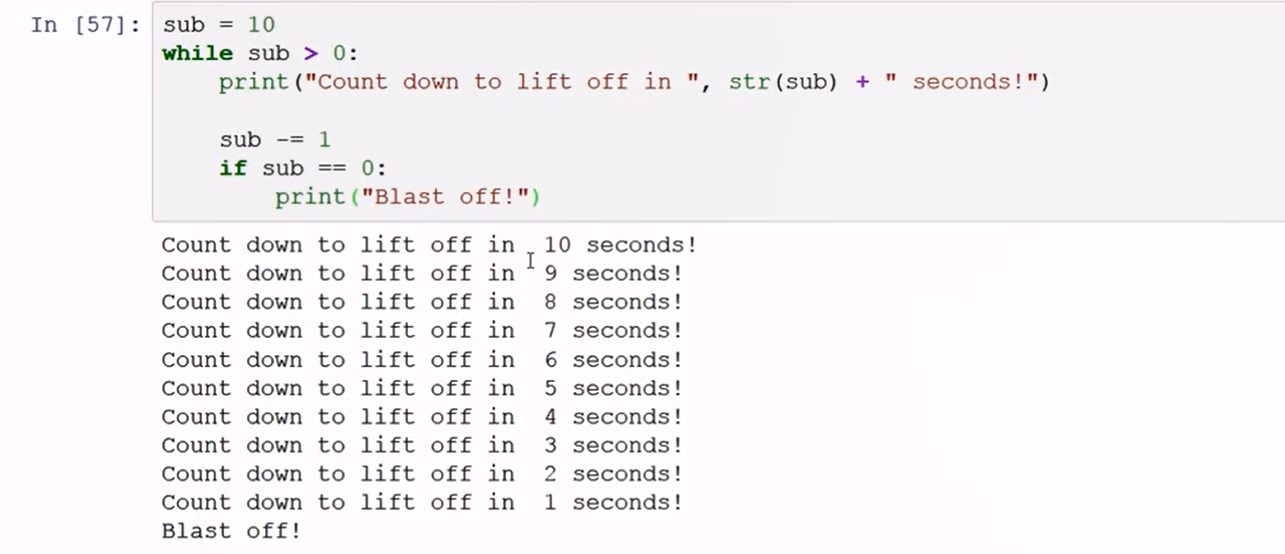
When while loop executes it will check the *condition* first, if condition is met true it will execute the code1 and code 2 statements, it will again go back to the first line of code to check the condition and execute the code 1 and code2 until the condition is false. Once condition is false it will come out of while loop and execute the Executable code3.

Count = count+1 in the below code prevents the infinite loop. 

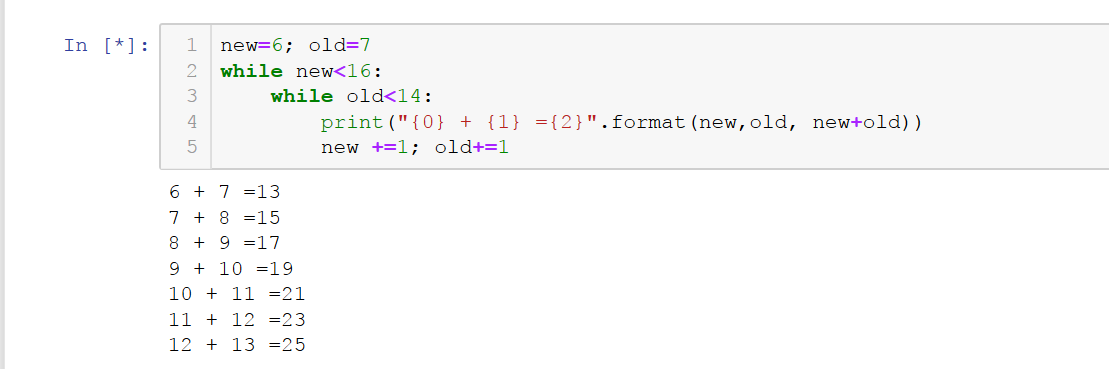
Below code will fall into an infinite loop as the value of counter is always going to be 5 and <=10.

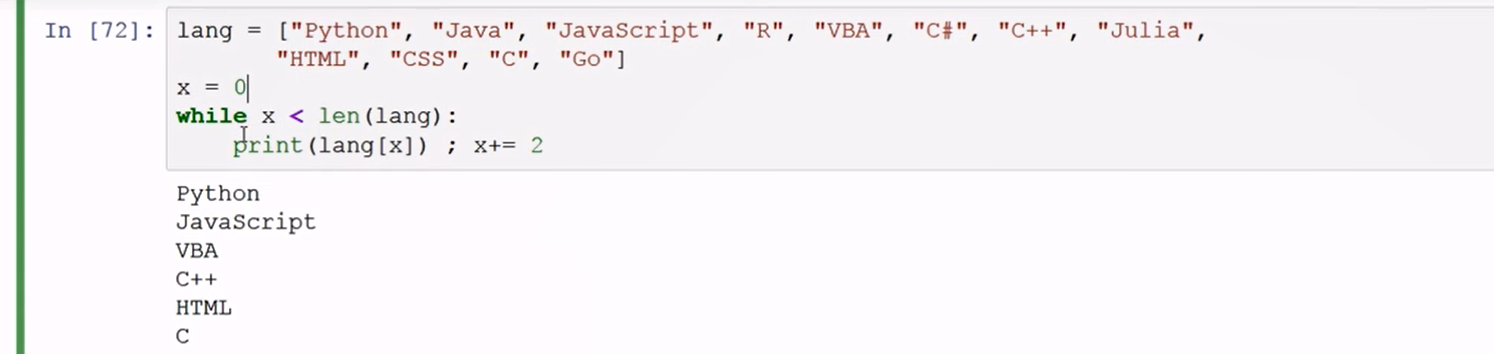


**While and if combined:**



**Nested While Loop:**





**Round is an inbuilt function.**

**RANGE FUNCTION:**

Generator of numerical value.

Range (0,5)

**LIST COMPREHENSSION:**



num=[item\*\*2 for item in x]

Syntax:

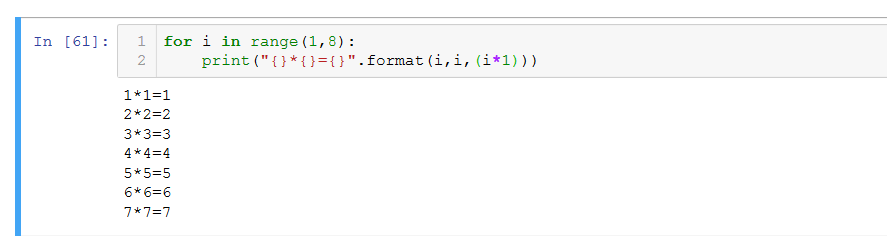
For i in [python iterable]:

Expression(i)

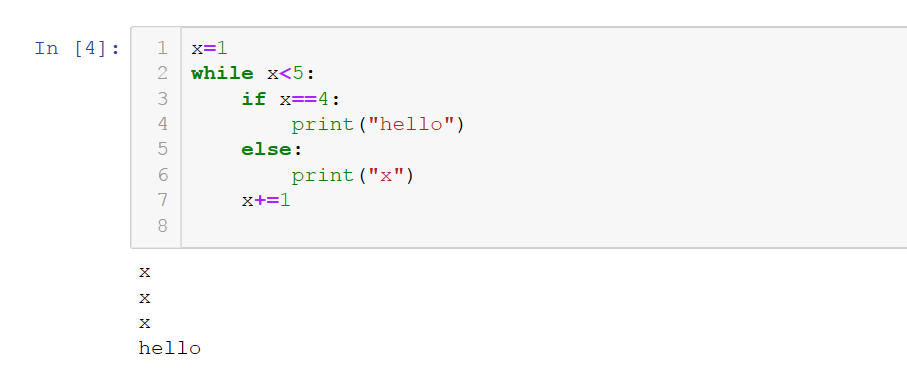
Python iterable can be a list, tuple or advanced data structure.

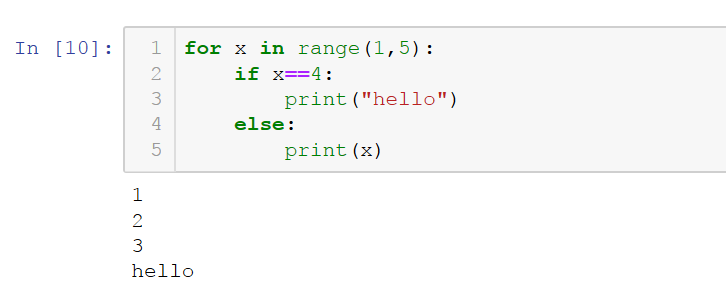
Fir i in range(1:10)

Print(i)



**CONTROL FLOW IN LOOPS:**





**Try x=0**

**While x<len(num example**

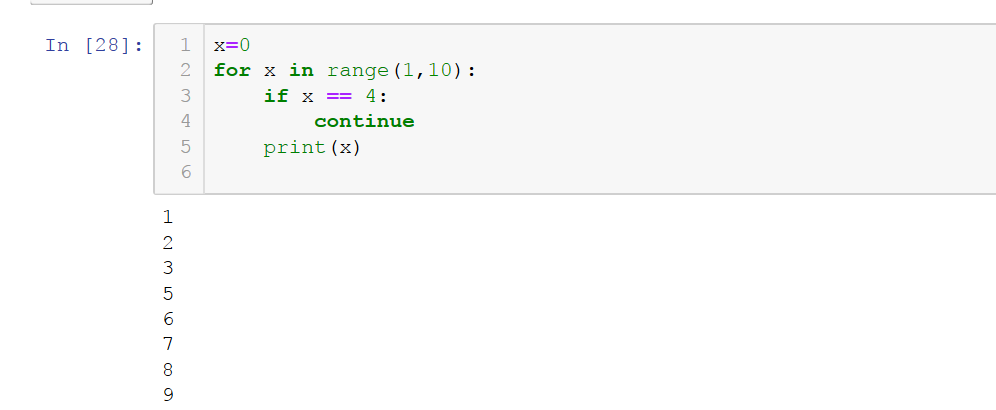
**BREAK, PASS AND CONTINUE:**

Used in Loop control.

Break stops a loop at a certain point.

Pass prevents a code being run at all.

Continue runs a loop at a certain point

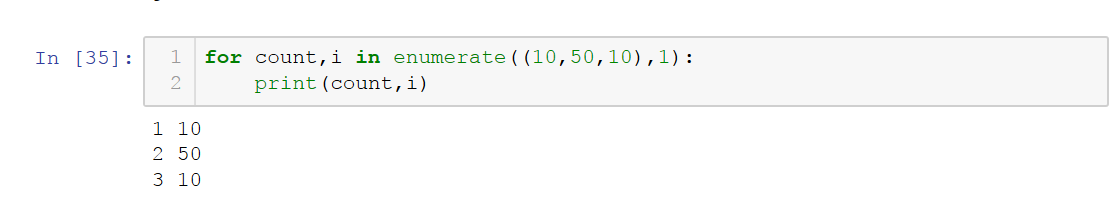


**ZIP AND ENUMERATE:**

**Zip : Can combine numbers from two or more lists into nested tuples.**

**Enumerate:**

**Can generate an integer number next to each output.**



**FUNCTIONS:**

* Allows us to not consensually type the code again and again.
* Reduces the amount of time.
* Before defining name of function use Def in the starting. Ex- def square(arg1, arg2)
* Def square (arg1, arg2):

Return(arg1+arg2)

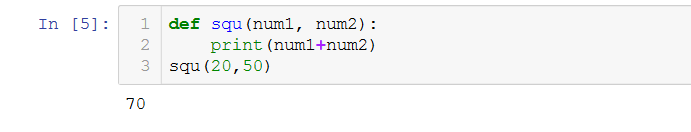
Num = square(20,50)

Default values

* Def square (arg1=20, arg2=50):

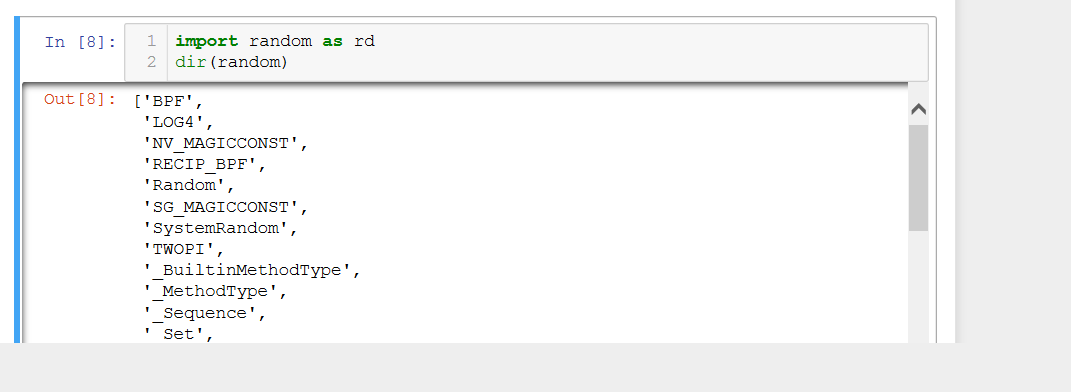
Return(arg1+arg2)

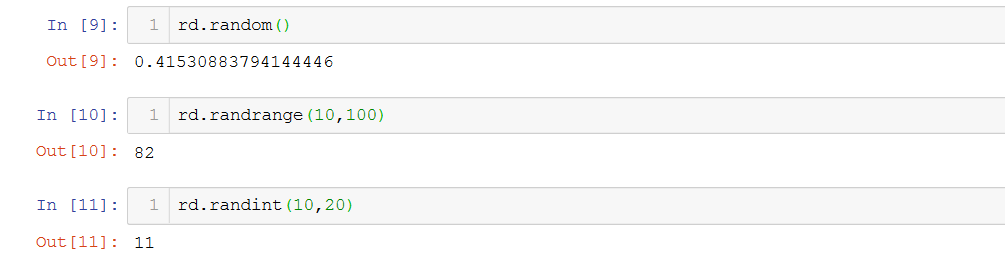
Num = square ()

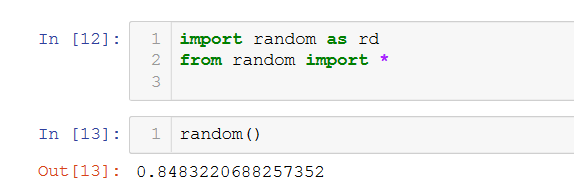


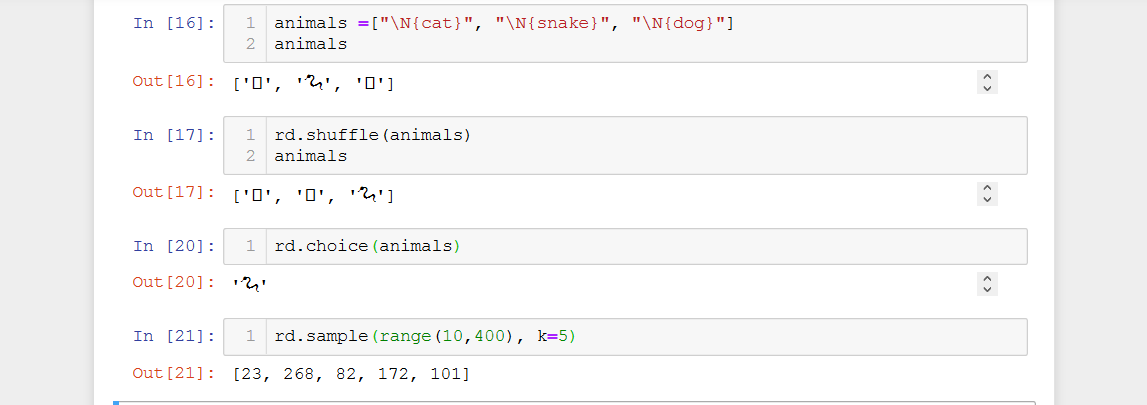
**RANDOM MODULE:**

* Requiring Importing (default package)
* Generate Random Values









BUILT-IN FUNCTIONS:

Python interpreter has a number of functions that are always available for use.

Example:

Print()

min()

max()

abs()

help(max)

HANDLING ERRORS:

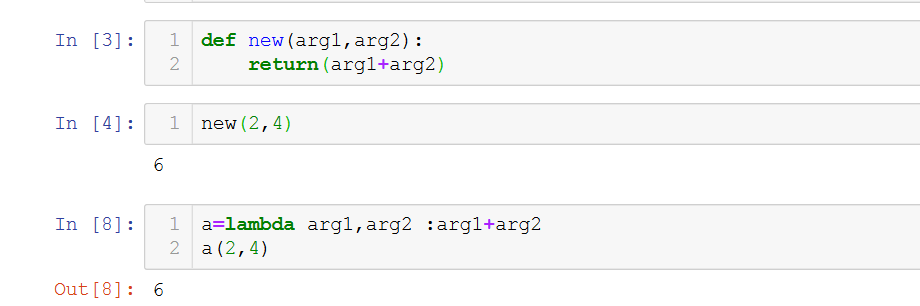
Lamda is an anonymous function, meaning it does not require a def name.

Does not require a return.

Lamdas are single line expression.

Can only be used as a substitute for basic functions.

Can-not add a docstring.



Map and Filter Function:

TRY, EXCEPT AND FINALLY:

Python will output different types of error notifications:

TypeError

NameError

AttributeError

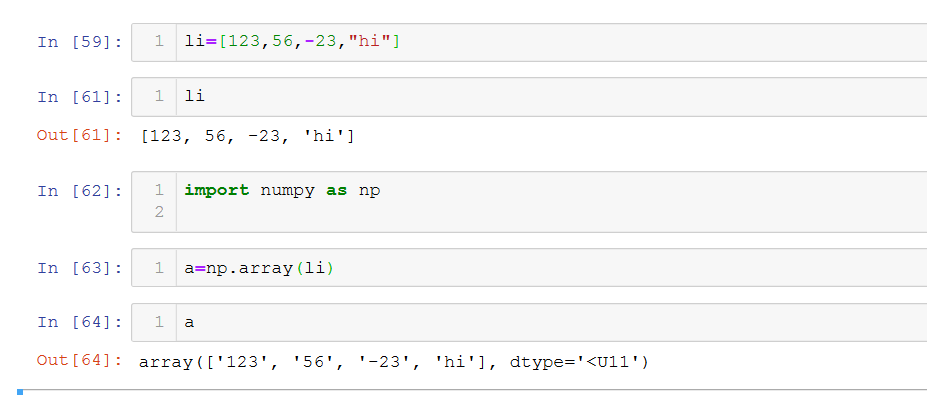
IndentationError

SyntaxError

Working with Text File:

**Arrays in Python:**

**Contains same types of data.**



**Slicing Arrays:**

**When we are slicing an arrays, we are working with array not with the copy.**

**If you want to copy:**

**B = a.copy**

**While slicing an array, you do not create a copy of array.**

**LIBRARIES IN PYTHON:**

There are many libraries with predefined which we can directly import into our code and make our life easy.

**Most popular python data science libraries.**

**Numpy –**

It is a Linear algebra library for python. library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

If you have anaconda, install numpy by going to you your terminal or command prompt and type:

Conda numpy install

If you do not have anaconda

Pip install numpy

SciPy

Pandas

Seaborn

SciKit-Learn

Matplotlib

Plotly

PySpark

**MATRICES:**

**Representation of table of data.**

**A[3,2]**

**0 1 2 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **23** | **43** | **34** | **445** |
| **1** | **56** | **6** | **99** | **3** |
| **2** | **8** | **0** | **0** | **2** |
| **3** | **4** | **1** | **3** | **0** |

**A[0,:] . 0 1 2 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **23** | **43** | **34** | **445** |
| **1** | **56** | **6** | **99** | **3** |
| **2** | **8** | **0** | **0** | **2** |
| **3** | **4** | **1** | **3** | **0** |

**A[:,1]**

**0 1 2 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **23** | **43** | **34** | **445** |
| **1** | **56** | **6** | **99** | **3** |
| **2** | **8** | **0** | **0** | **2** |
| **3** | **4** | **1** | **3** | **0** |

**Ways to create Matrix:**

**Np.reshape(\*,\*,”C”) DEFAULT**

C= Rows will be populated first.

**Np.reshape(\*,\*,”F”)**

F= Columns will be populated first(F stands for Fortran language)

**Np.array():**

Combined lists into matrices row by row