**PYTHON FOR DATA SCIENCE&ML TRAINING**

**WEEK-1**

* **Environment setup**
* **Jupyter**
* **Numpy**
* **Pandas**

**PYTHON TRAINING**

**PYTHON**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas the other languages use punctuations. It has fewer syntactical constructions than other languages. It is a case sensitive in nature. Author wanted to name the programming language unique and short. Inspired by Monty Python’s Flying Circus, a BBC comedy series, he named it Python

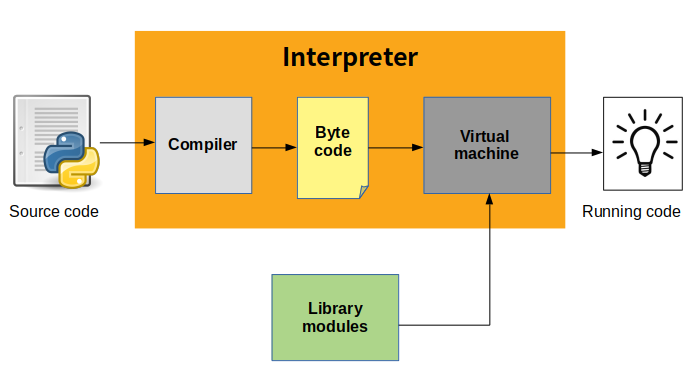
It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.



**Compiler:**

A compiler is a software that converts the source code to the object code. In other words, we can say that it converts the high-level language to machine /binary Language.



**Interpreter:** A program that can be analyse and execute a program line by line.

**For example:** print() is the right function and Print() is the wrong function to be used in the python programming.

**print(“Hello world!”)** here, we get an output>>> **Hello world!**

**DATA TYPES**

1. **Numeric**
2. **Sequence Type**
3. **Boolean**
4. **Set**
5. **Dictionary**

****

**Numeric:**

The numeric data type in Python represents the data that has a numeric value. A numeric value can be an integer, a floating number, or even a complex number. These values are defined as [Python int](https://www.geeksforgeeks.org/python-int-function/), [Python float](https://www.geeksforgeeks.org/float-in-python/), and [Python complex](https://www.geeksforgeeks.org/python-complex-function/) classes in [Python](https://www.geeksforgeeks.org/python-programming-language/).

* **Integers**– This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.
* **Float**– This value is represented by the float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
* **Complex Numbers** – A complex number is represented by a complex class. It is specified as *(real part) + (imaginary part)j*. For example – 2+3

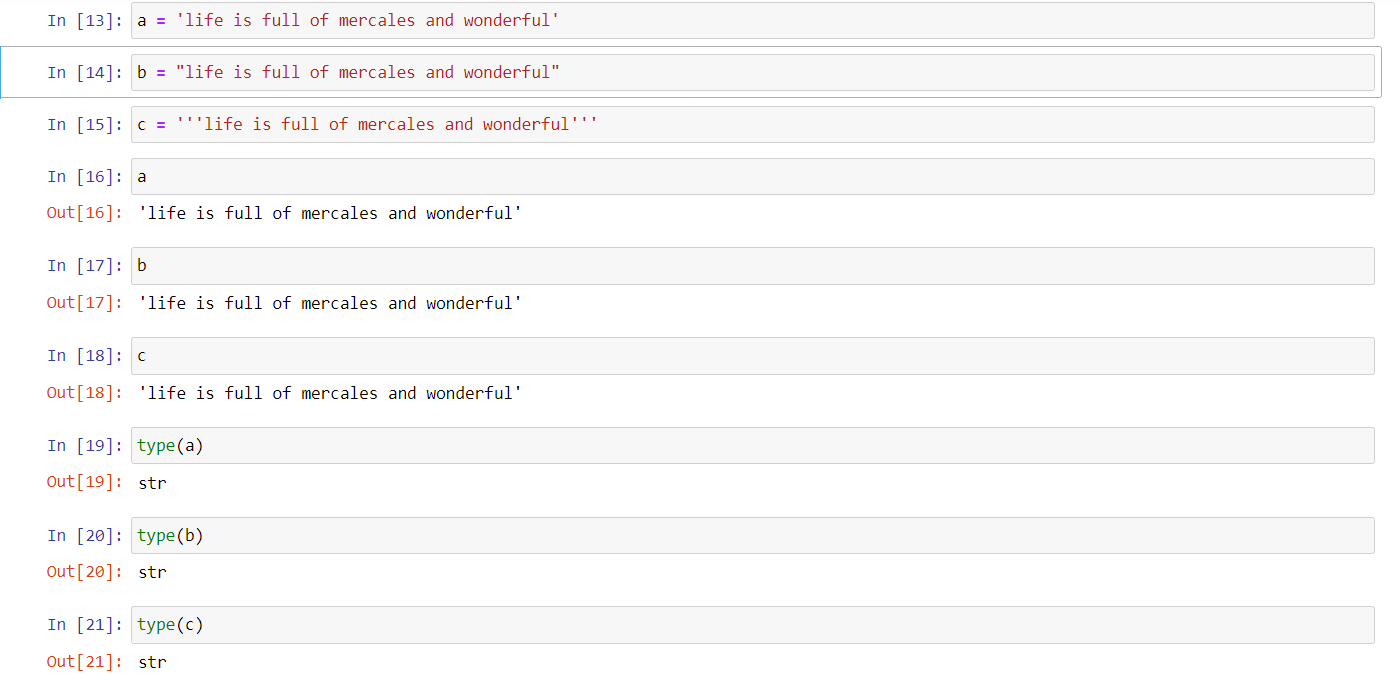
****

**Sequence type:**

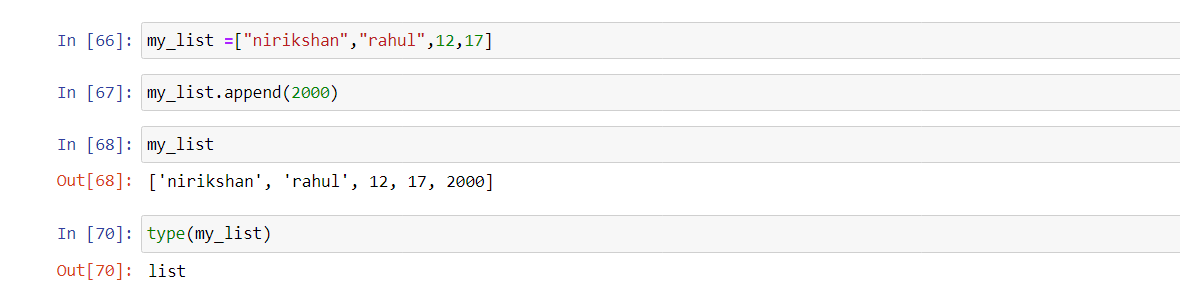
The sequence Data Type in Python is the ordered collection of similar or different data types. Sequences allow storing of multiple values in an organized and efficient fashion. There are several sequence types in Python

* [**String**](https://www.geeksforgeeks.org/python-data-types/#string)
* [**List**](https://www.geeksforgeeks.org/python-data-types/#list)
* [**Tuple**](https://www.geeksforgeeks.org/python-data-types/#tuple)
  + 1. **String:**

String in Python are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote, or triple-quote. In Python there is no character data type, a character is a string of length one. It is represented by str class.



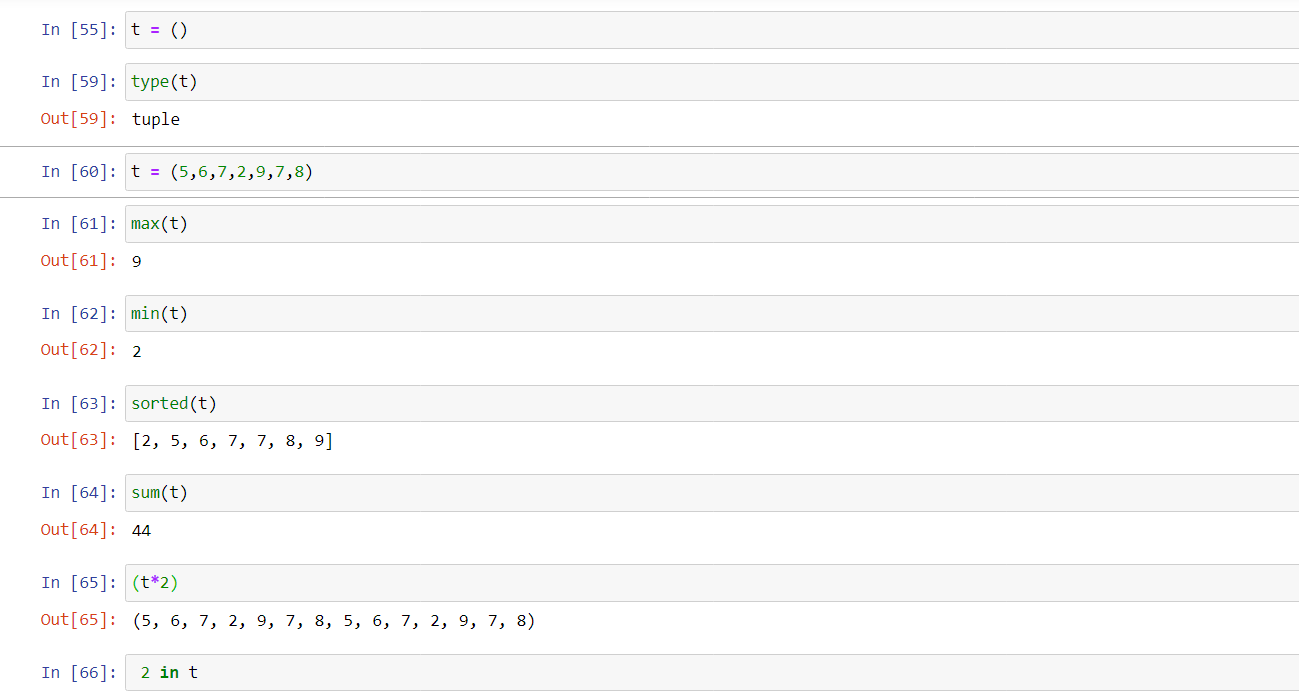
* + 1. **List:** List are just like arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

****

1. **Tuple**

**Tuple** is a collection of Python objects much like a list. The sequence of values stored in a tuple can be of any type, and they are indexed by integers.

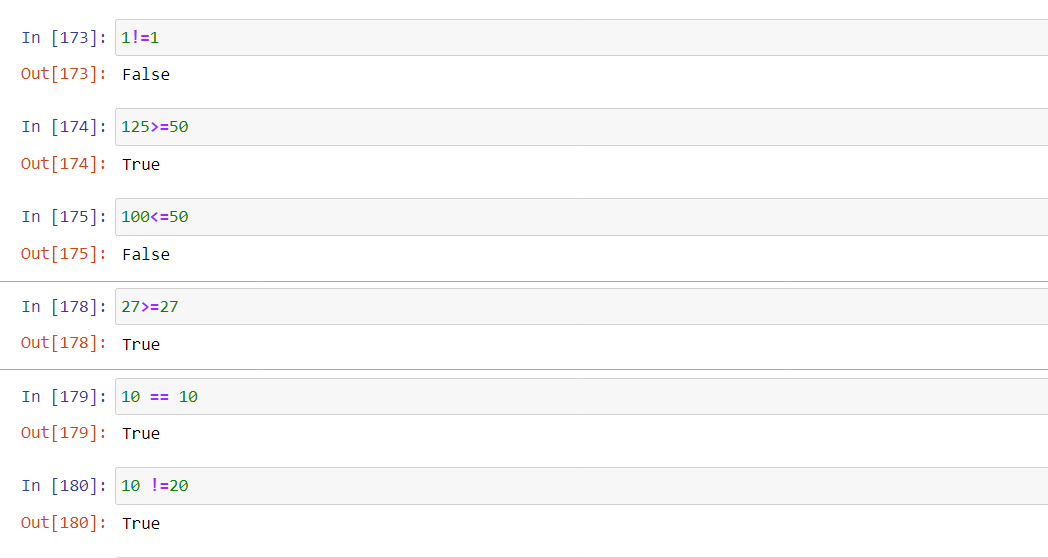
Values of a tuple are syntactically separated by ‘commas’. Although it is not necessary, it is more common to define a tuple by closing the sequence of values in parentheses. This helps in understanding the Python tuples more easily. A tuple is a sequence of immutable Python objects. Tuples are just like lists with the exception that tuples cannot be changed once declared. Tuples are usually faster than lists.



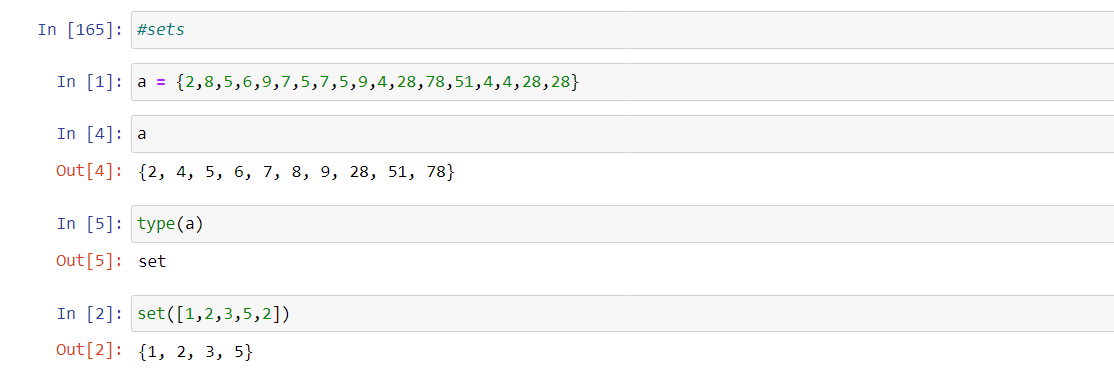
**BOOLEAN:**

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.



**Set:** A Set in [Python programming](https://www.geeksforgeeks.org/python-programming-language/) is an unordered collection data type that is iterable, mutable and has no duplicate elements.

****

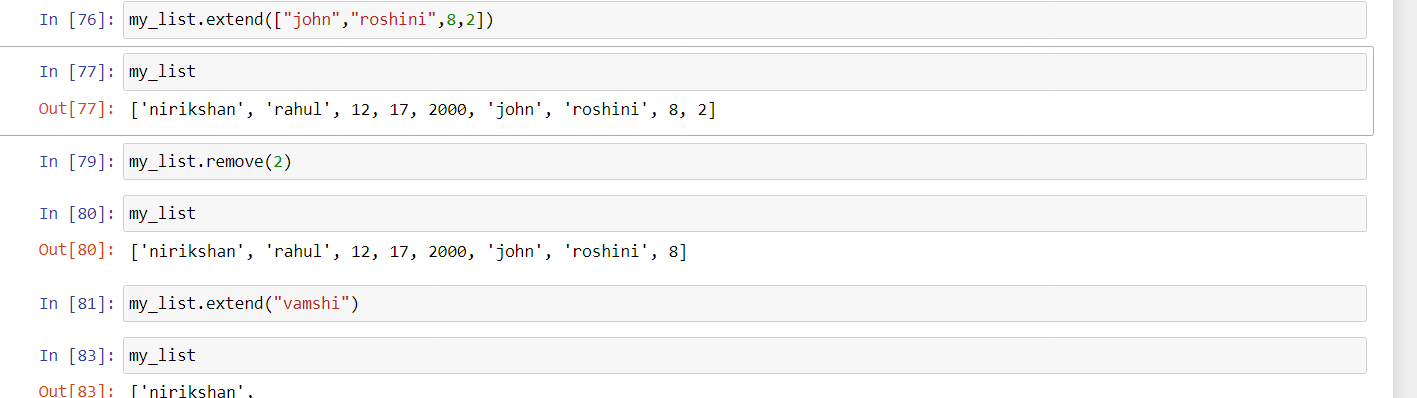
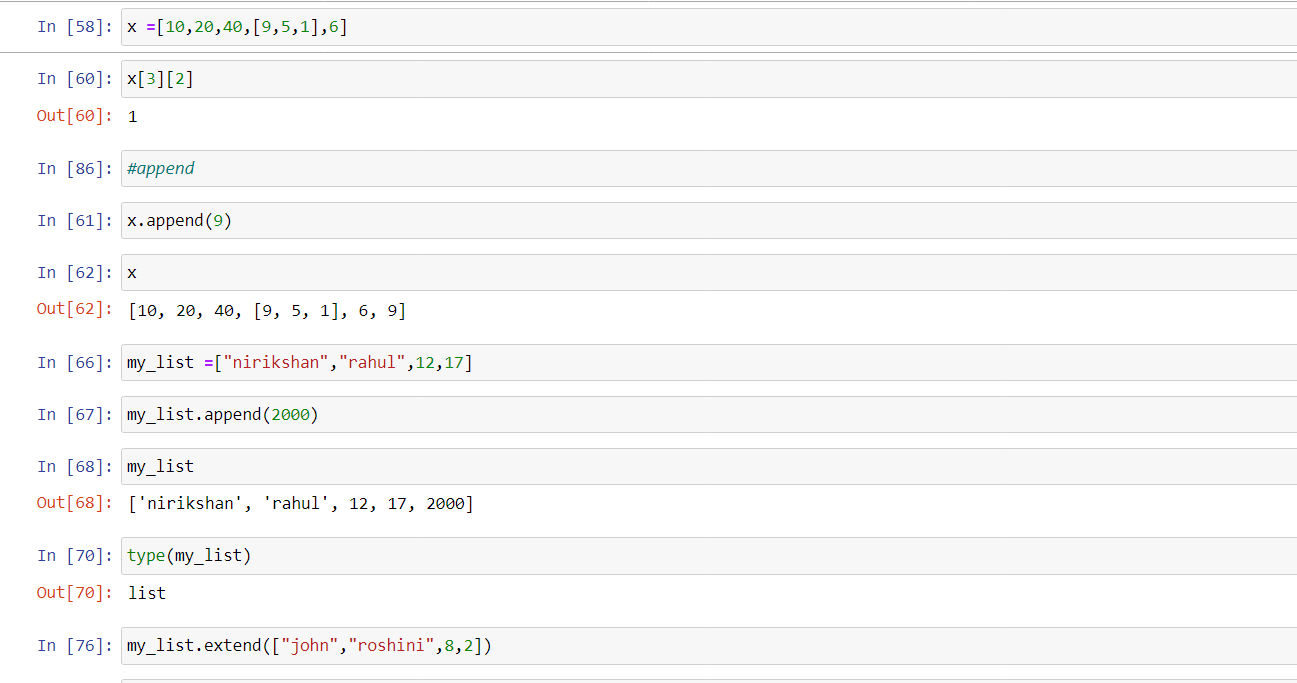
**Dictionary:**

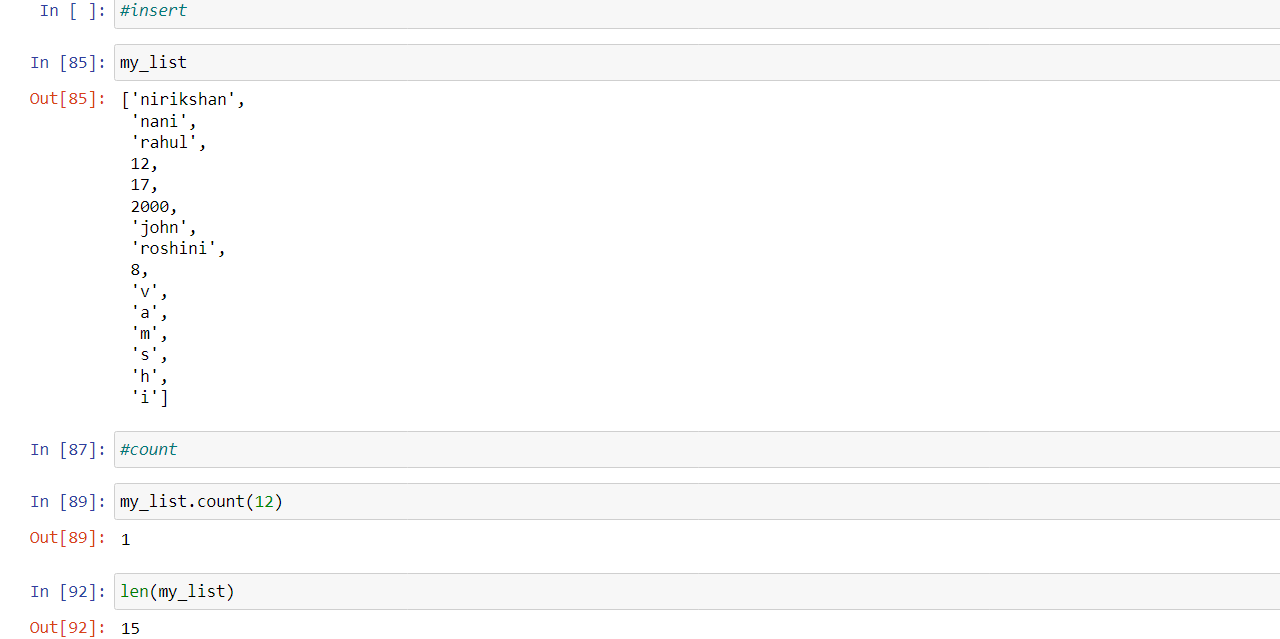
**A dictionary in Python** is a data structure that stores the value in value:key pairs.

****

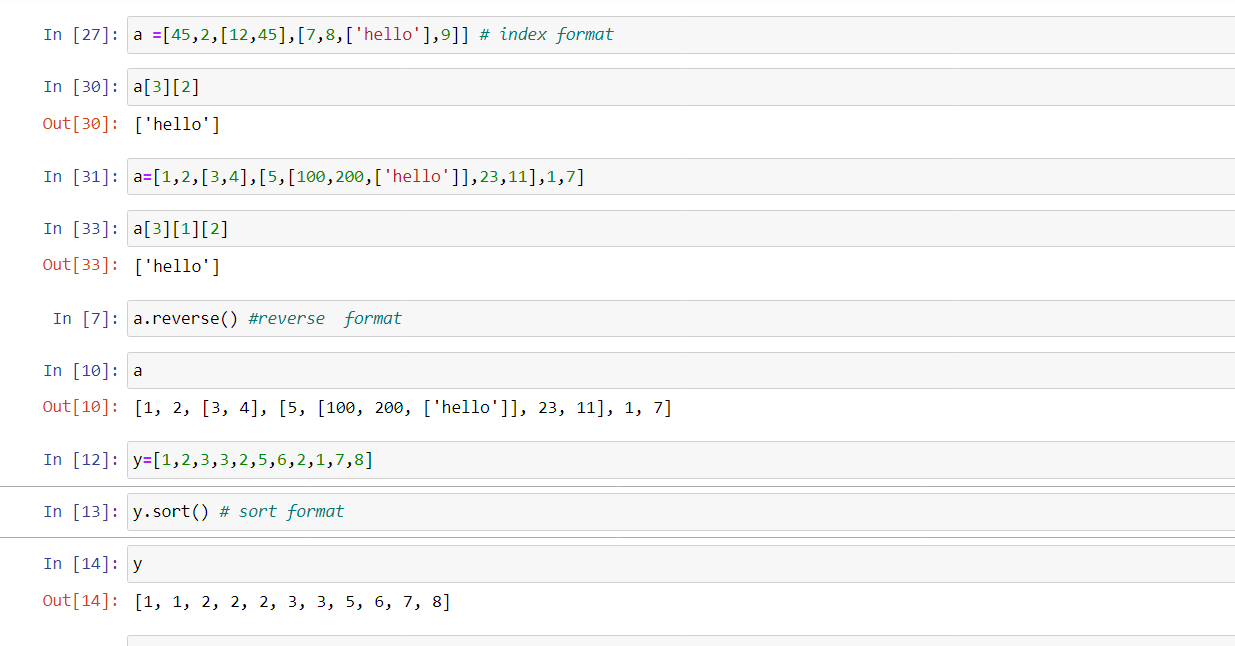
|  |  |
| --- | --- |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

**LIST METHODS**

****

****

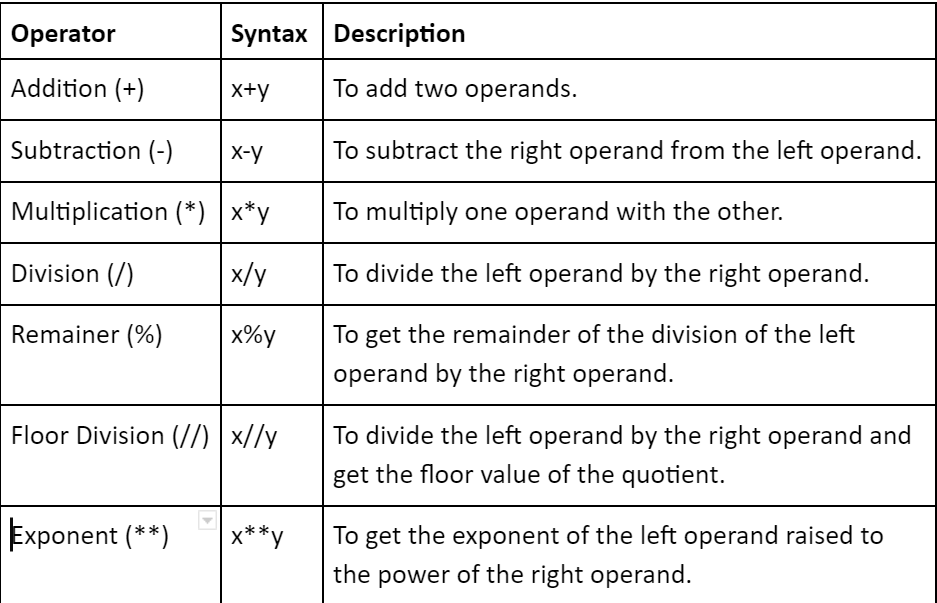
****

****

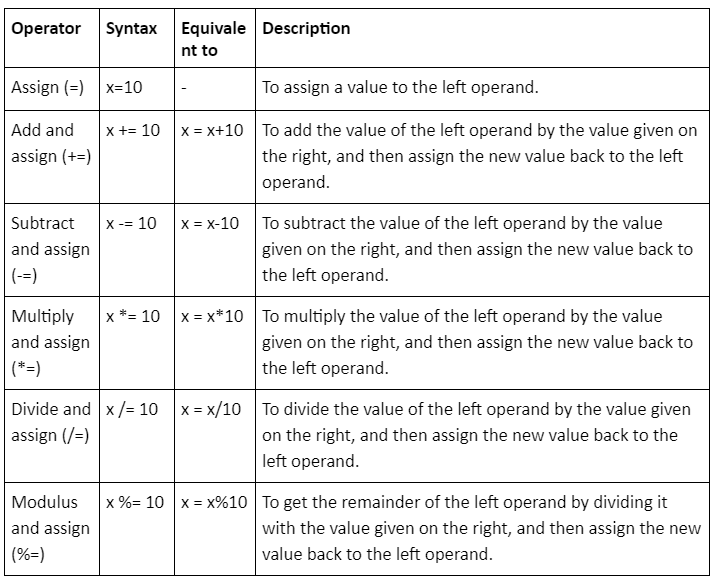
**PYTHON OPERATORS**

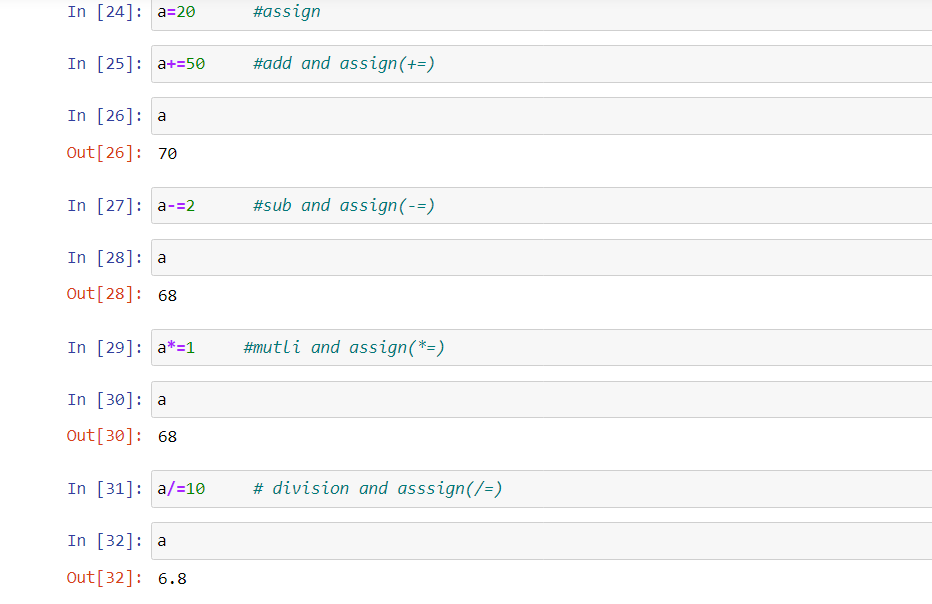
1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Logical operators
5. Identity operators
6. Membership operators
7. Bitwise operators

**Arithmetic operators:**

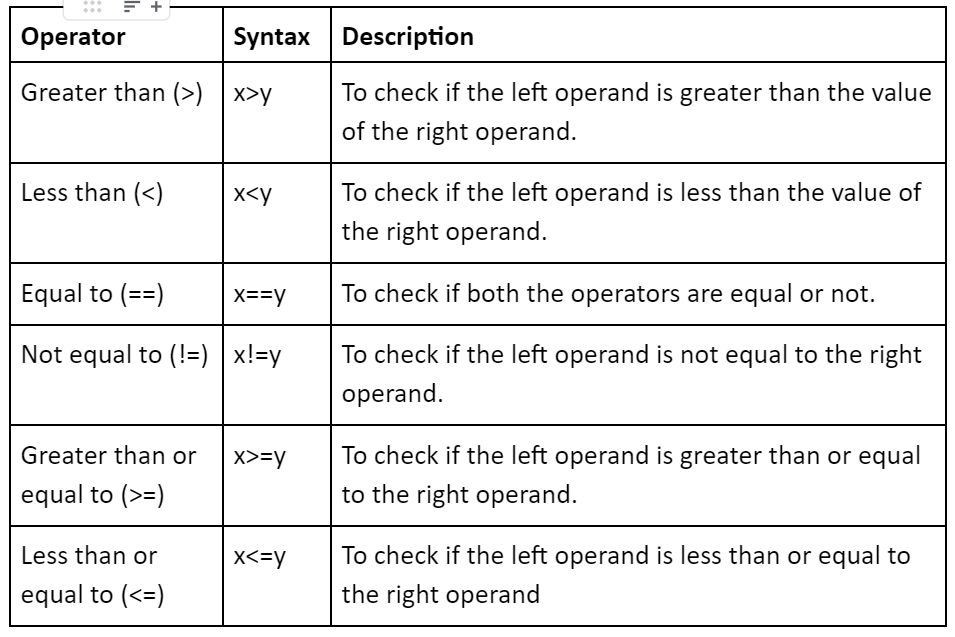


****

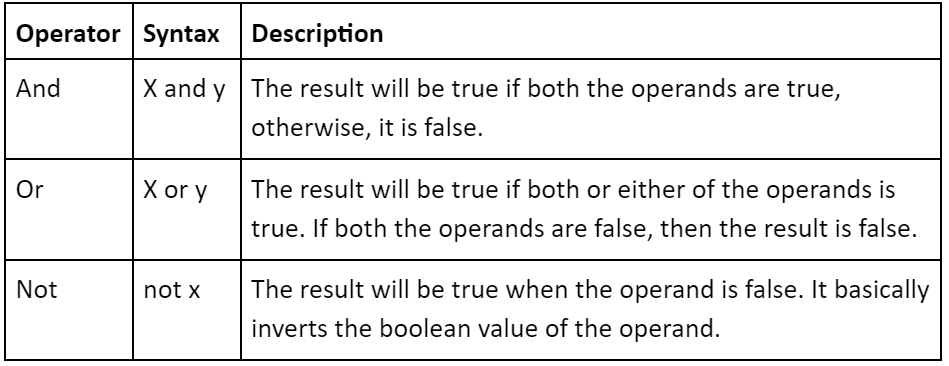
**Assignment operators:**

****

**Comparison operators:**Comparison operators are used to compare two values:

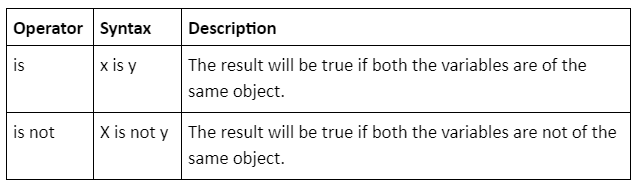


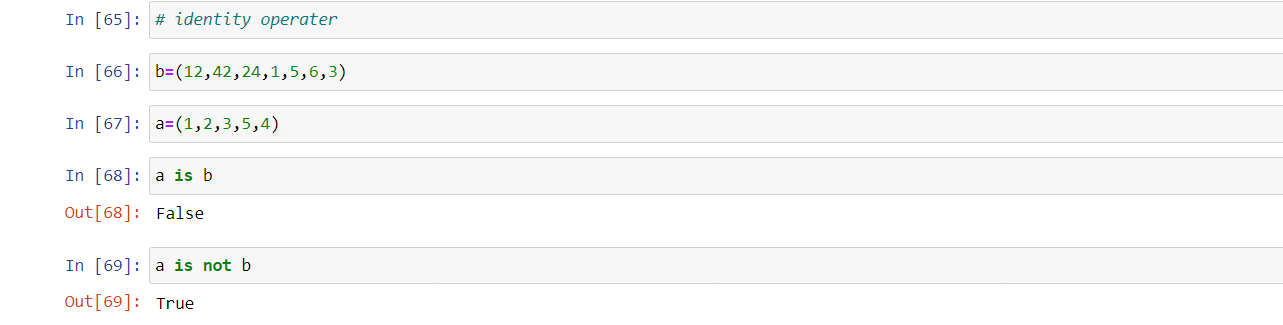
****

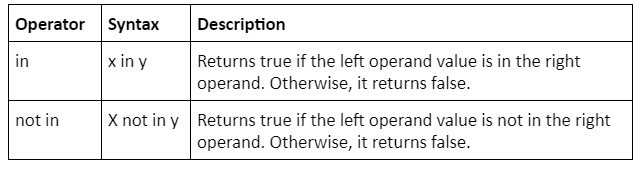
**Logical operators:** ****

****

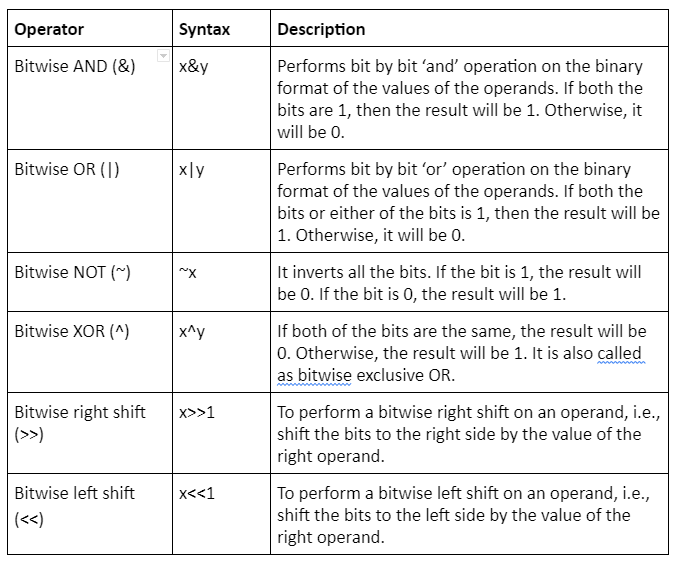
**Identity operators:**To check if two operands share the same memory location or not, we can use the identity operators.It doesn't mean that the two variables are the same, but they are of the same object.These come under the special operators’ category in python.





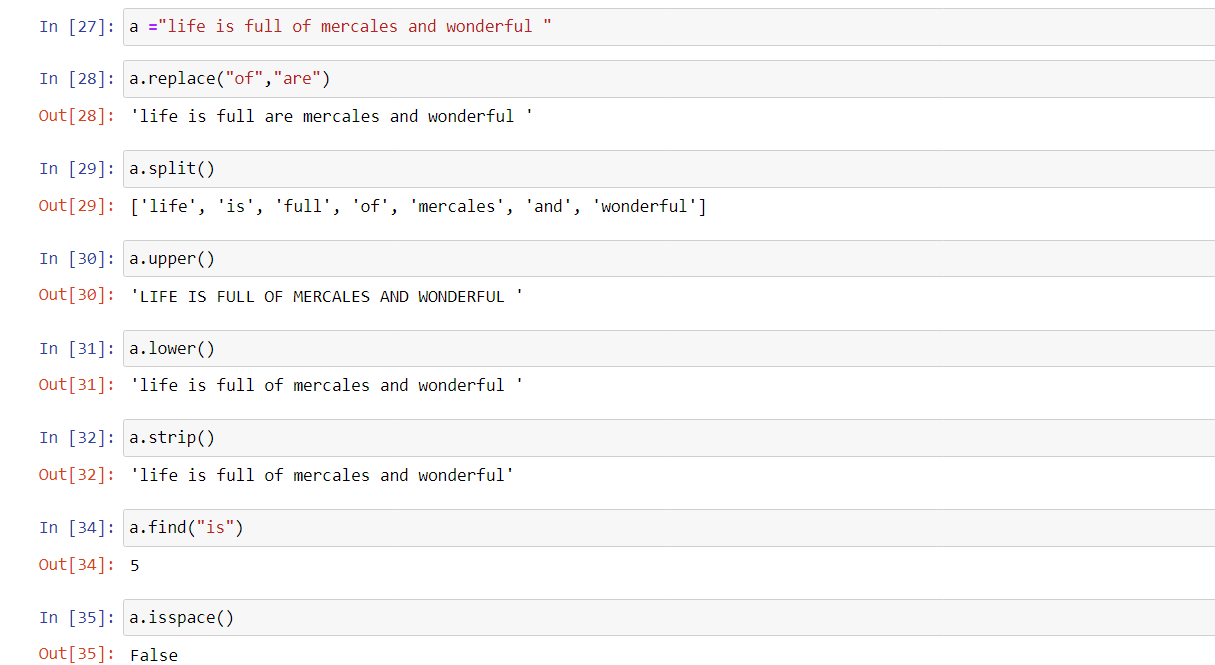
**Membership operators:** ****

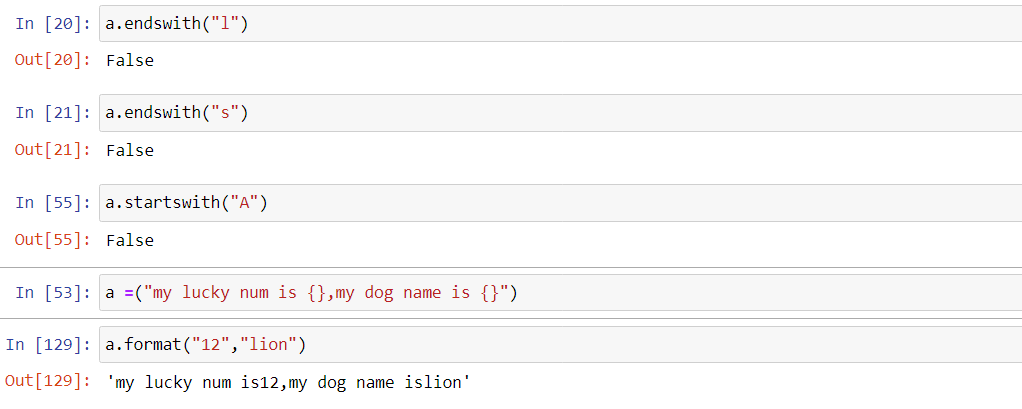
**Bitwise operators:**

 ****

**STRINGS METHODS:**Here are some of the most common string methods. A method is like a function, but it runs "on" an object. If the variable s is a string, then the code s.lower() runs the lower() method on that string object and returns the result (this idea of a method running on an object is one of the basic ideas that make up Object Oriented Programming, OOP). Here are some of the most common string methods:

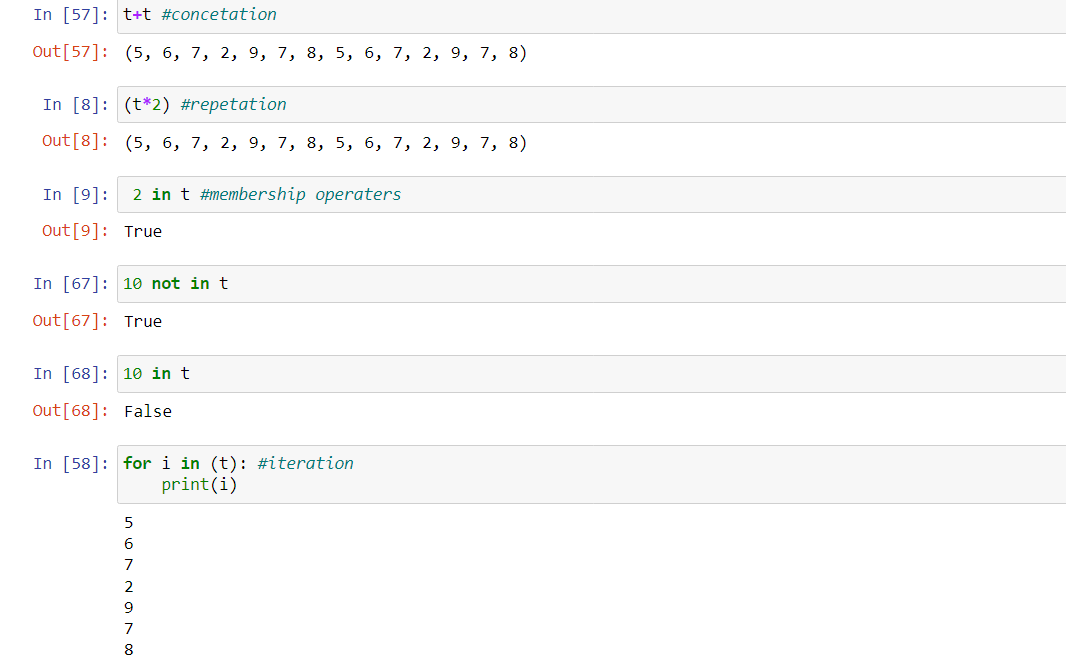
* s.lower(), s.upper() -- returns the lowercase or uppercase version of the string
* s.strip() -- returns a string with whitespace removed from the start and end
* s.isalpha()/s.isdigit()/s.isspace()... -- tests if all the string chars are in the various character classes
* s.startswith('other'), s.endswith('other') -- tests if the string starts or ends with the given other string
* s.find('other') -- searches for the given other string (not a regular expression) within s, and returns the first index where it begins or -1 if not found
* s.replace('old', 'new') -- returns a string where all occurrences of 'old' have been replaced by 'new'
* s.split('delim') -- returns a list of substrings separated by the given delimiter. The delimiter is not a regular expression, it's just text. 'aaa,bbb,ccc'.split(',') -> ['aaa', 'bbb', 'ccc']. As a convenient special case s.split() (with no arguments) splits on all whitespace chars.

s.join(list) -- opposite of split(), joins the elements in the given list together using the string as the delimiter. e.g. '---'.join(['aaa', 'bbb', 'ccc']) -> aaa---bbb---ccc  
  


****

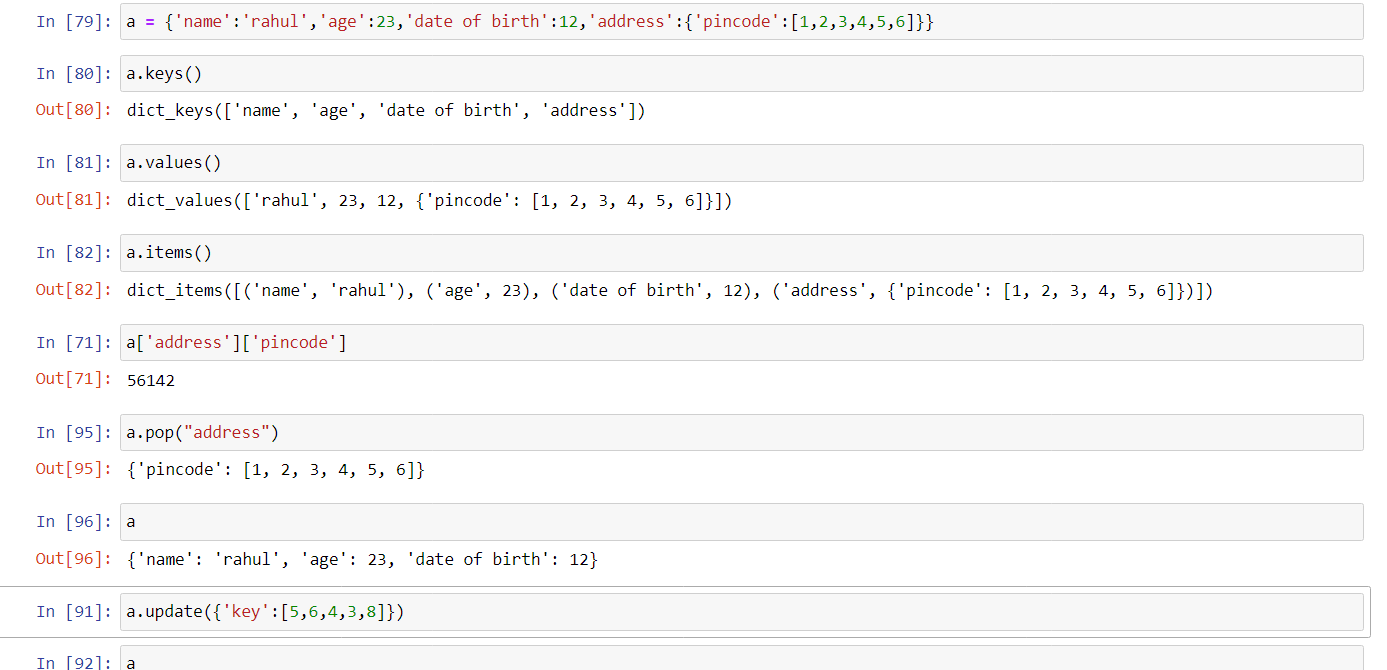
**Tuple operations**

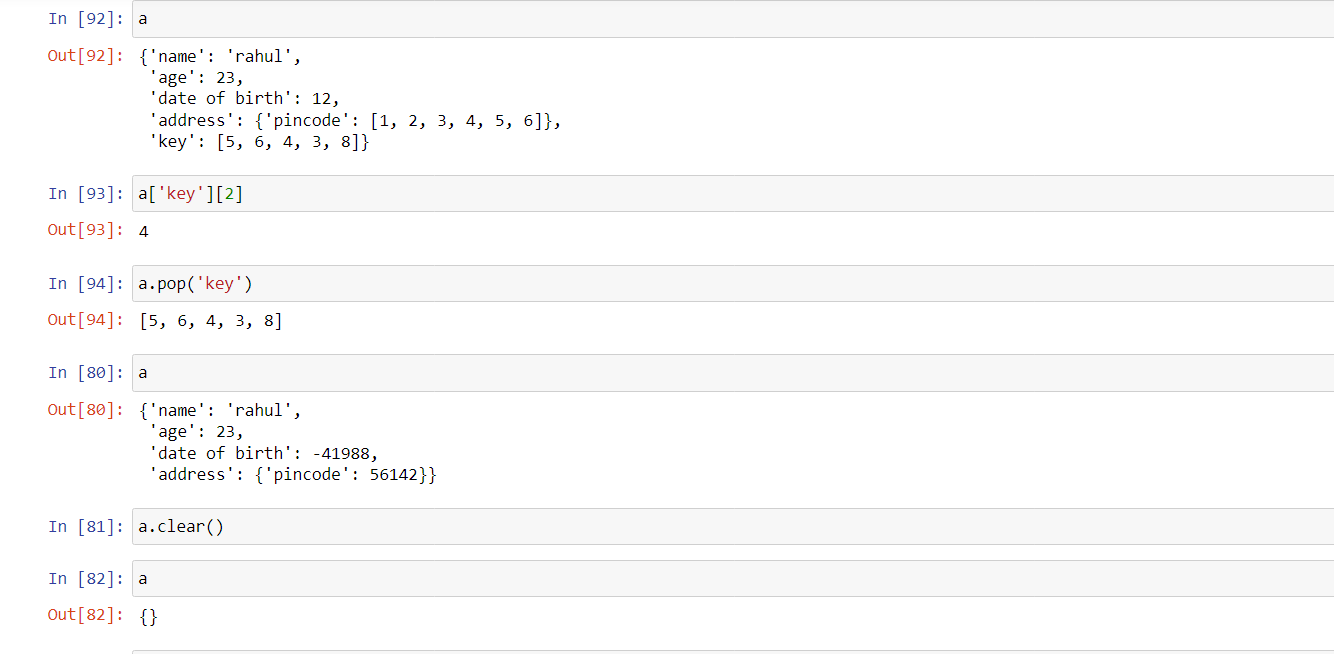
|  |  |  |
| --- | --- | --- |
| **Operation** | **Operator** |  |
| Concatenation | + |  |
| Repetition | \* |  |
| Iteration | for |  |
| Membership | in |  |

****

**Dictionary methods**

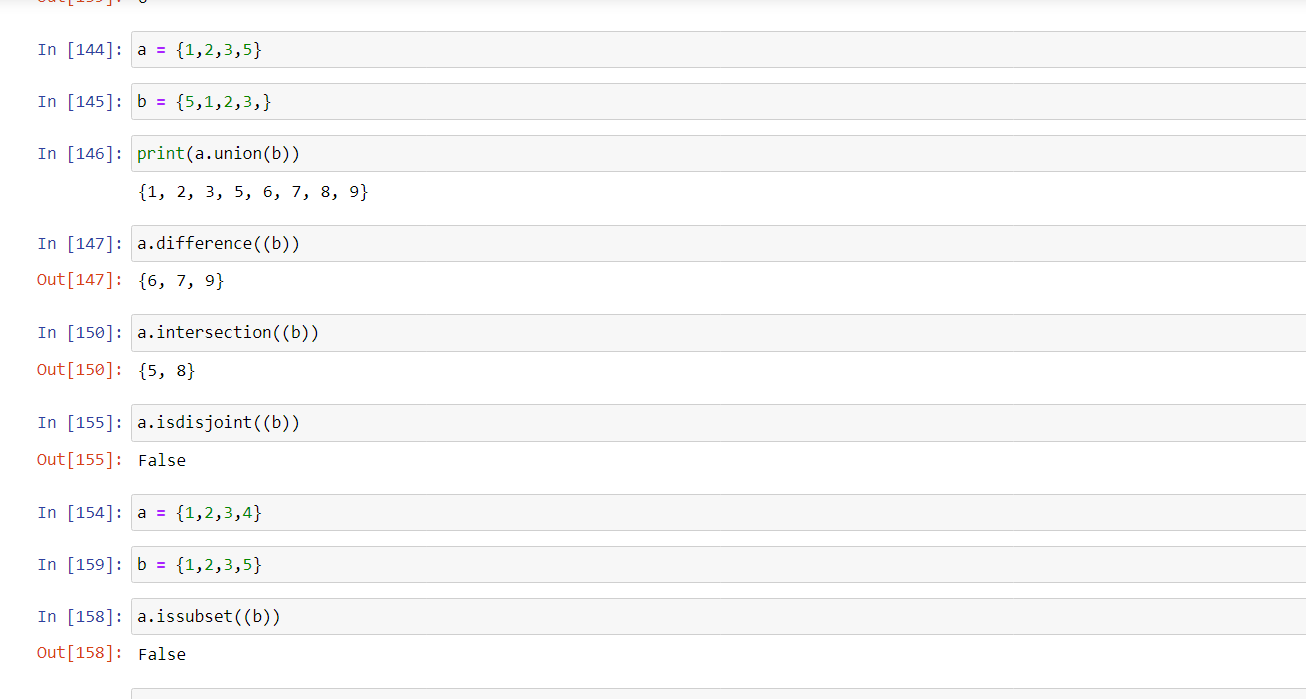
|  |  |
| --- | --- |
| **Method** | **Description** |
| [clear()](https://www.w3schools.com/python/ref_dictionary_clear.asp) | Removes all the elements from the dictionary |
| [copy()](https://www.w3schools.com/python/ref_dictionary_copy.asp) | Returns a copy of the dictionary |
| [fromkeys()](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | Returns a dictionary with the specified keys and value |
| [get()](https://www.w3schools.com/python/ref_dictionary_get.asp) | Returns the value of the specified key |
| [items()](https://www.w3schools.com/python/ref_dictionary_items.asp) | Returns a list containing a tuple for each key value pair |
| [keys()](https://www.w3schools.com/python/ref_dictionary_keys.asp) | Returns a list containing the dictionary's keys |
| [pop()](https://www.w3schools.com/python/ref_dictionary_pop.asp) | Removes the element with the specified key |
| [popitem()](https://www.w3schools.com/python/ref_dictionary_popitem.asp) | Removes the last inserted key-value pair |
| [setdefault()](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| [update()](https://www.w3schools.com/python/ref_dictionary_update.asp) | Updates the dictionary with the specified key-value pairs |
| [values()](https://www.w3schools.com/python/ref_dictionary_values.asp) | Returns a list of all the values in the dictionary |

****

****

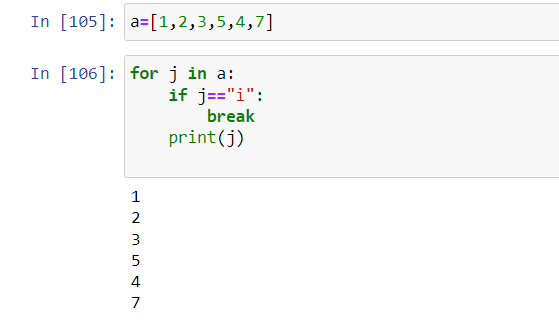
|  |  |  |
| --- | --- | --- |
| **Operation** | **Notation** | **Meaning** |
| Intersection | A ∩ B | all elements which are in both and |
| Union | A ∪ B | all elements which are in either or (or both) |
| Difference | A − B | all elements which are in but not in |
| Complement | (or) | all elements which are not in |

**SET OPERATIONS:**

****

**CONTROL STATEMENTS IN PYTHON**Control statements are an essential aspect of any programming language, including Python. Control statements in Python are used to manage the flow of execution of a program based on certain conditions.

Control statements in Python are a powerful tool for managing the flow of execution. They allow developers to make decisions based on specific conditions and modify the normal sequential flow of a program. By using control statements effectively, developers can write more efficient and effective code.

**Break:**  
 The break statement is used to terminate a loop, i.e., for loop, while loop, or nested loop. When a break statement executes inside a loop, it immediately terminates the loop and transfers control.  


**Continue**

when the condition specified in the if statement is true, the continue statement is executed, and the control is transferred to the next iteration of the loop. This means that the current iteration is skipped, and the loop continues with the next iteration. ****

# **Python If Else Statements – Conditional Statements**

If-Else statements in Python are part of conditional statements, which decide the control of code. As you can notice from the name If-Else, you can notice the code has two ways of directions. There are situations in real life when we need to make some decisions and based on these decisions, we decide what we should do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code. Conditional statements in Python languages decide the direction(Control Flow) of the flow of program execution.

**IF STATEMENT**

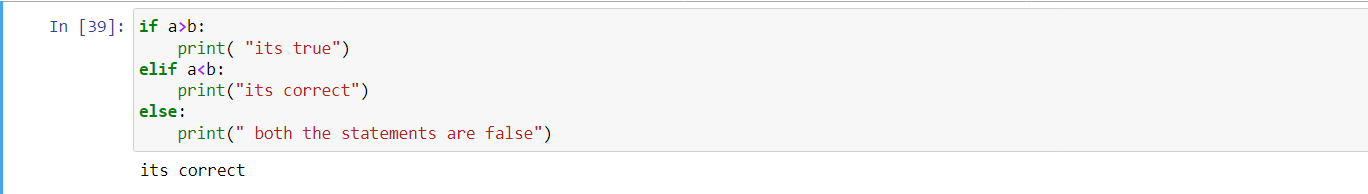
The[if statement](https://www.geeksforgeeks.org/python3-if-if-else-nested-if-if-elif-statements/) is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not.  the condition after evaluation will be either true or false. if the statement accepts boolean values – if the value is true then it will execute the block of statements below it otherwise not.

**IF ELSE**

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won’t. But if we want to do something else if the condition is false, we can use the else statement with the if statement to execute a block of code when the if condition is false.

**IF-ELIF-ELSE**Here, a user can decide among multiple options. The if statements are executed from the top down.As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final “else” statement will be executed.





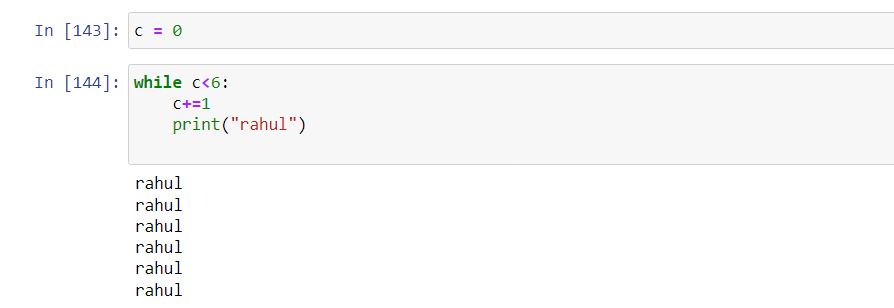
**LOOPING STATEMENTS IN PYTHON**

Python programming language provides two types of loops – **For loop** and **While loop** to handle looping requirements. Python provides three ways for executing the loops.

While all the ways provide similar basic functionality, they differ in their syntax and condition-checking time.

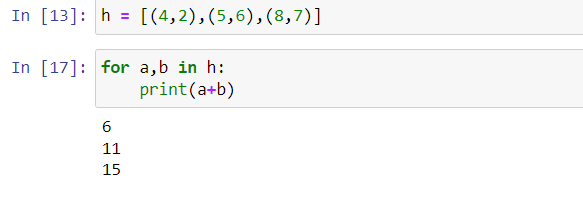
## ****WHILE LOOP IN PYTHON****

In [Python](https://www.geeksforgeeks.org/python-programming-language/), a [while loop](https://www.geeksforgeeks.org/python-while-loop/) is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.



## FOR LOOP IN PYTHON

[For loops](https://www.geeksforgeeks.org/python-for-loops/) are used for sequential traversal. We can use for loop to iterate lists, [tuples](https://www.geeksforgeeks.org/python-tuples/), strings and [dictionaries](https://www.geeksforgeeks.org/python-dictionary/) in Python.

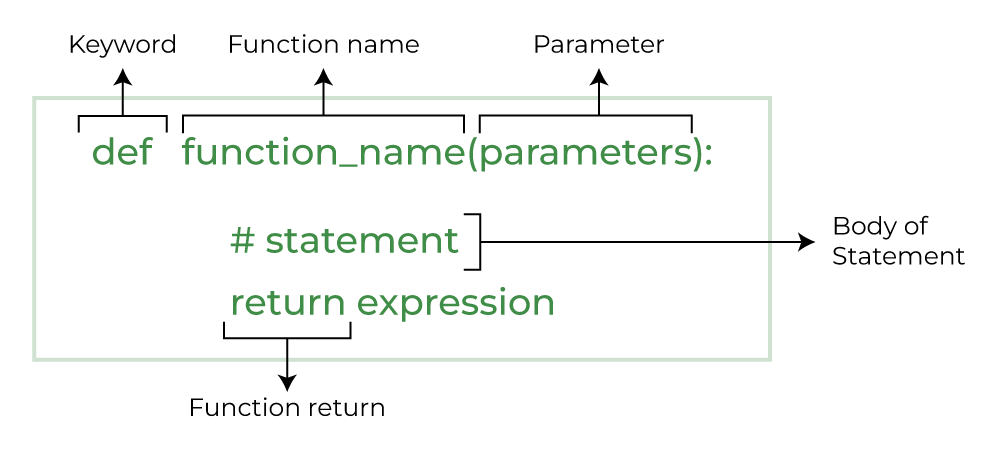


**PYTHON FUNCTIONS**

**Python Functions** is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

Some **Benefits of Using Functions**

* Increase Code Readability
* Increase Code Reusability



****

**MAP LAMBDA IN PYTHON**

A lambda expression is a way of creating a little function inline, without all the syntax of a def. Here is a lambda with a single n parameter, returning the parameter value doubled.

lambda n: n \* 2

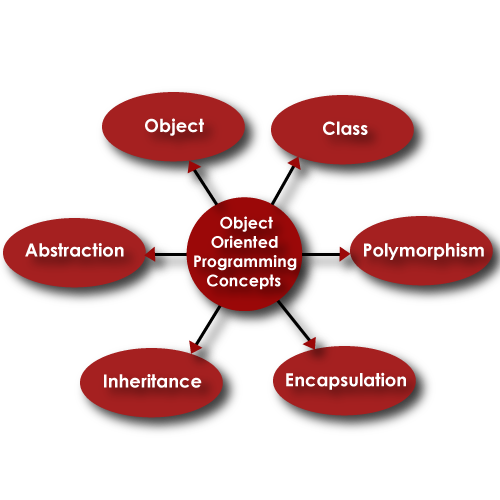
The code of the lambda is typically a single expression without variables or if-statements, and does not use "return". Lambda is perfect where you have a short computation to write inline. Many programs have some sub-part which can be solved very compactly this way. For longer code, def is better. To work with map(), the lambda should have one parameter in, representing one element from the source list. Choose a suitable name for the parameter, like n for a list of numbers, s for a list of strings. The result of map() is an "iterable" map object which mostly works like a list, but it does not print.



**OOPS CONCEPT IN PYTHON**

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In [Python](https://www.javatpoint.com/python-tutorial), we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.



* Class
* Object
* Inheritance
* Polymorphism
* Data Abstraction
* Encapsulation

## Class

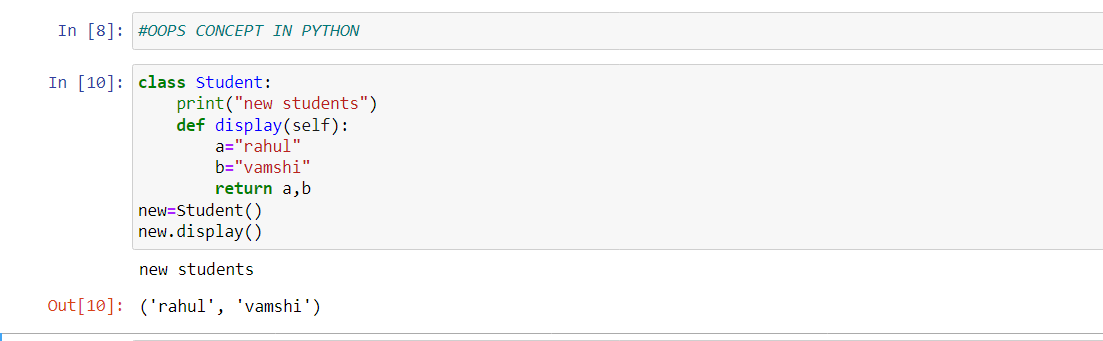
The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

## Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute \_\_doc\_\_, which returns the docstring defined in the function source code.

When we define a class, it needs to create an object to allocate the memory. Consider the following example.



**Inheritance**

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

**Polymorphism**

Polymorphism contains two words "poly" and "morphs". Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

## Encapsulation

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

**Data Abstraction**

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation.

Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

**NUMERICAL PYTHON**

**NUMPY**

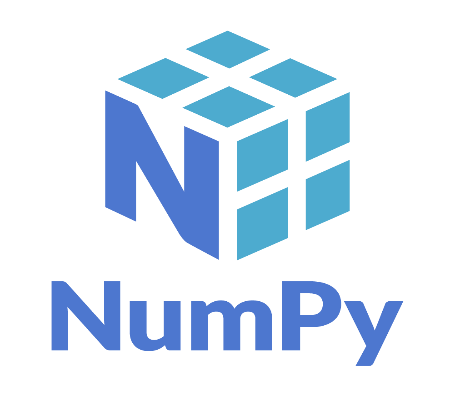
Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with python. It is open-source software.

**Features of Numpy**

NumPy has various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy in Python can also be used as an efficient multi-dimensional container of generic data. Arbitrary data types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

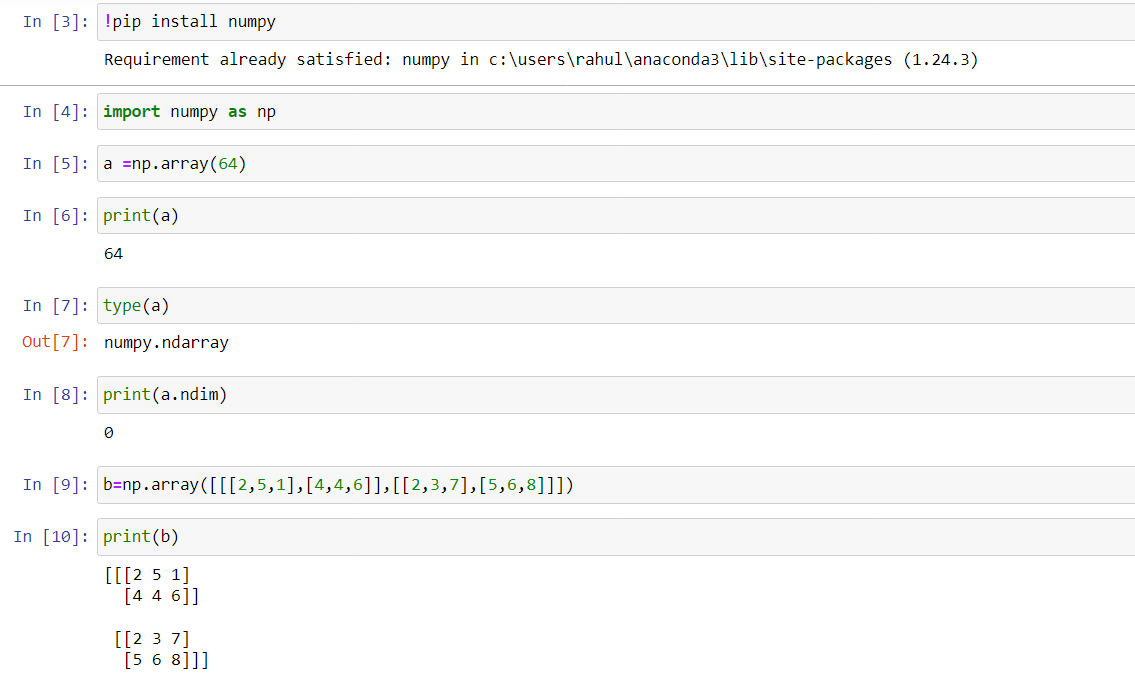


**Array in Numpy**

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array.A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as **ndarray**. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.

**Creating a Numpy Array**

Arrays in Numpy can be created by multiple ways, with various number of Ranks, defining the size of the Array. Arrays can also be created with the use of various data types such as lists, tuples, etc. The type of the resultant array is deduced from the type of the elements in the sequences.

****

# **Basics of NumPy Arrays**

Numpy stands for Numerical Python. It is a Python library used for working with an array. In Python, we use the list for purpose of the array but it’s slow to process. NumPy array is a powerful N-dimensional array object and its use in linear algebra, Fourier transform, and random number capabilities. It provides an array object much faster than traditional Python lists.

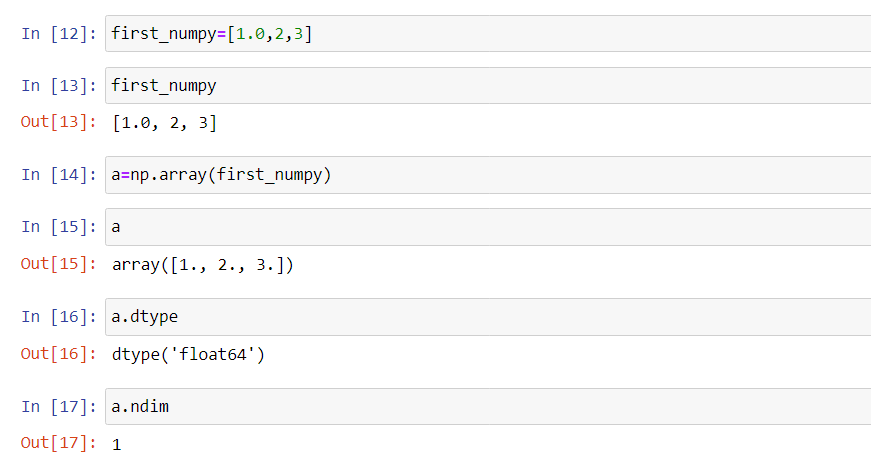
#### **Types of Array:**

1. One Dimensional Array
2. Multi-Dimensional Array

#### **One Dimensional Array:**

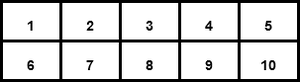
A one-dimensional array is a type of linear array.

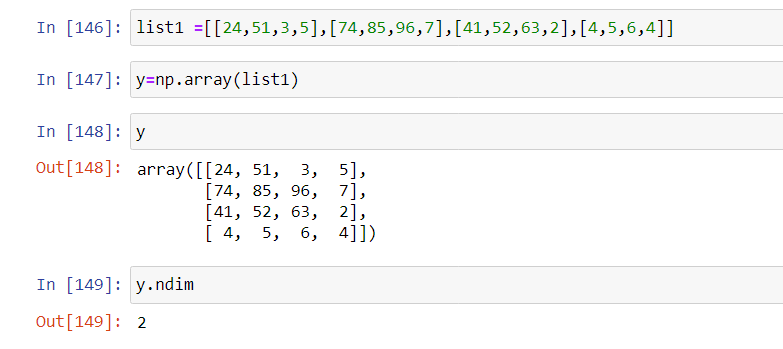
Lightbox

****

#### **Multi-Dimensional Array:**

Data in multidimensional arrays are stored in tabular form.





**Functions in python**

Here are some of the important NumPy functions in Python which every Data scientist.

**np.dtype**

This parameter specifies the data type of the returned array. It is an optional parameter and its default value is None. If dtype is set to None, the data type of the returned array will be the same as the input array.

**np.size()**

In Python, numpy.size() function count the number of elements along a given axis.

**np.arange():**

 This function is used to create an array with a range of values.

**np.zeros():**

 This function is used to create an array filled with zeros.

**np.ones():**

 This function is used to create an array filled with ones.

**np.linspace():**

 This function is used to create an array with a specified number of evenly spaced values.

**np.random.rand():**

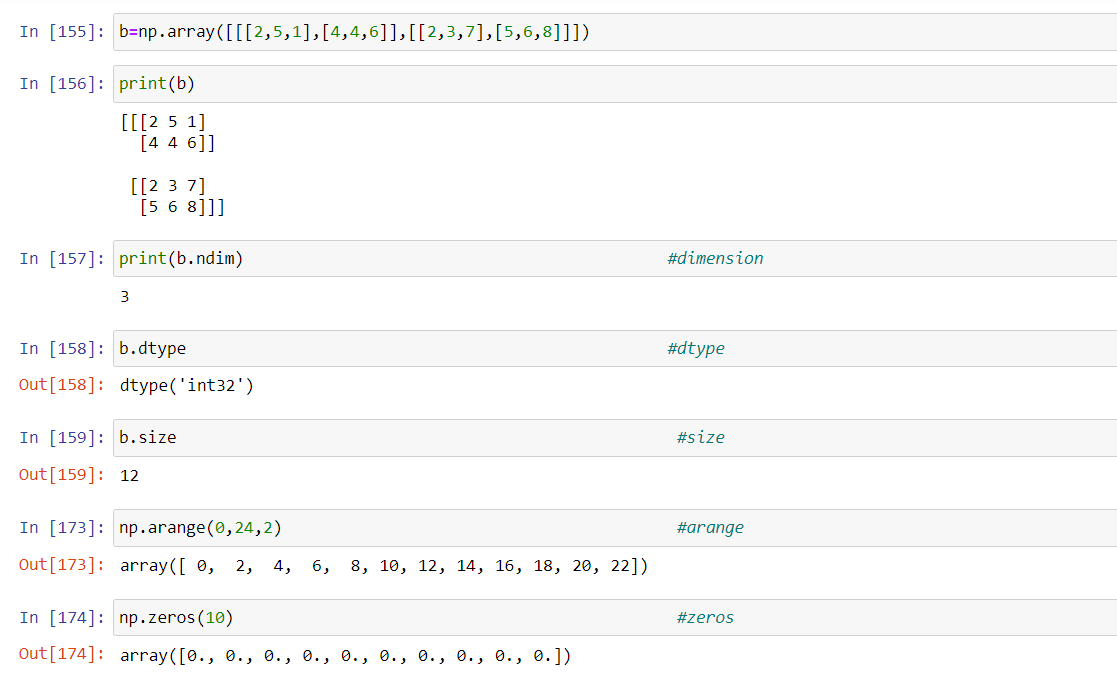
This function is used to create an array with random values between 0 and 1.

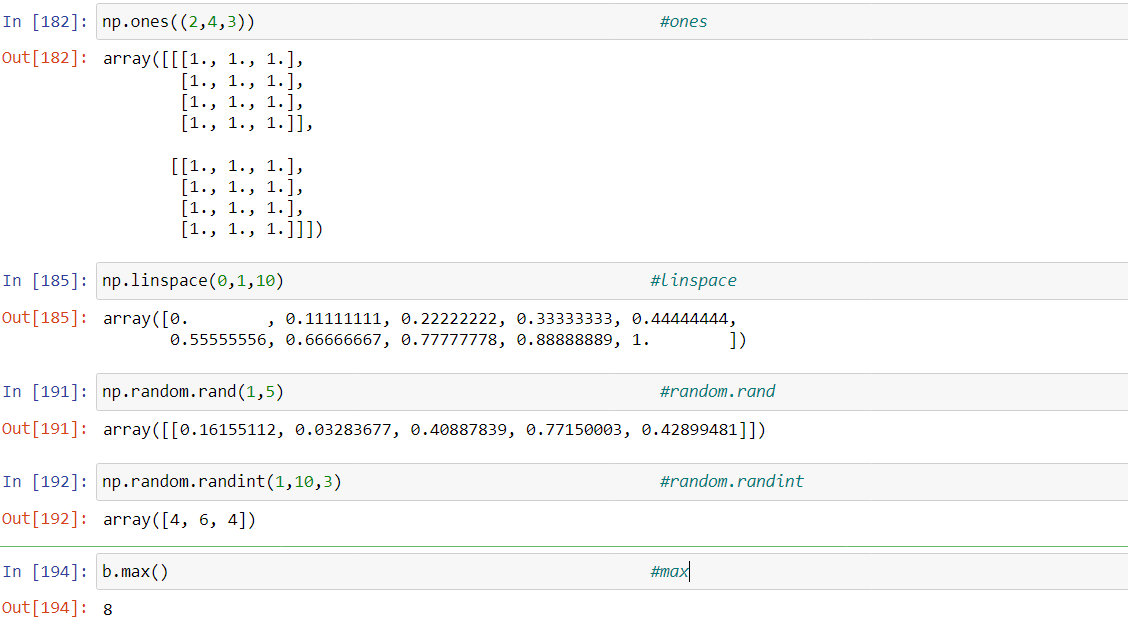
**np.random.randint():**

 This function is used to create an array with random integer values between a specified range.

**np.max():**

This function is used to find the maximum value in an array.





**argmax( ) and argmin( )**

functions are used to return the index of the maximum element along axis = 0 and axis = 1 , the elements from top to bottom along the column come under axis = 0 , whereas the elements from left to right along each row come under axis = 1

**np.reshape()**

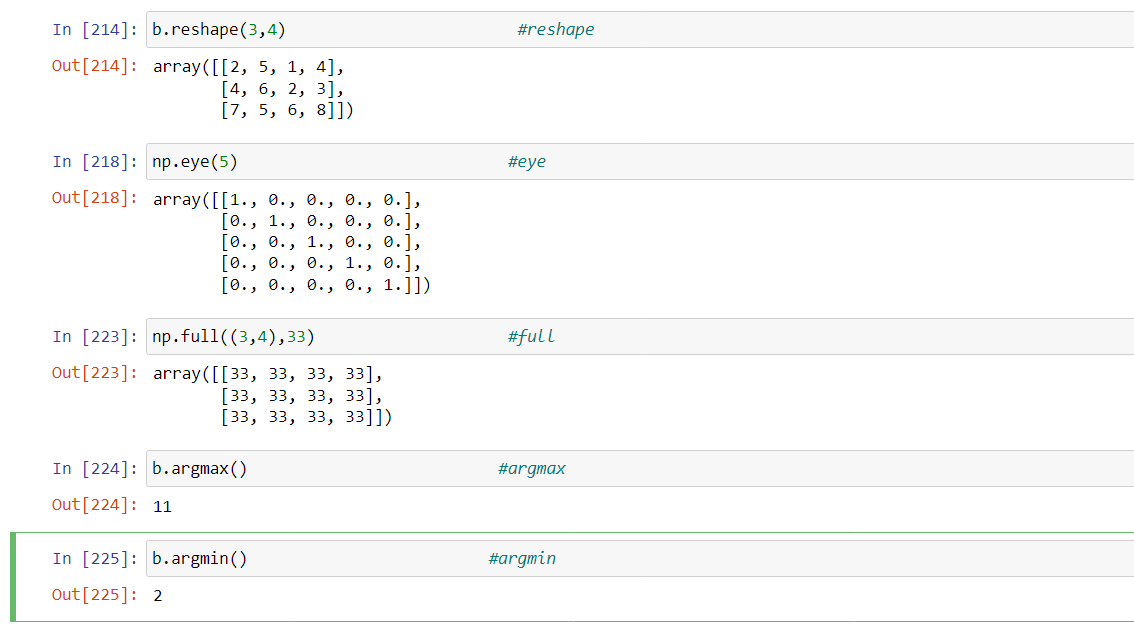
 Reshape is a method is used to reshape the matrix into required number of rows and columns. But, the total number of elements in the matrix should be equal to product of rows and columns.

**np.eye()**

Create a square N x N identity matrix (1’s on the diagonal and 0’s elsewhere)

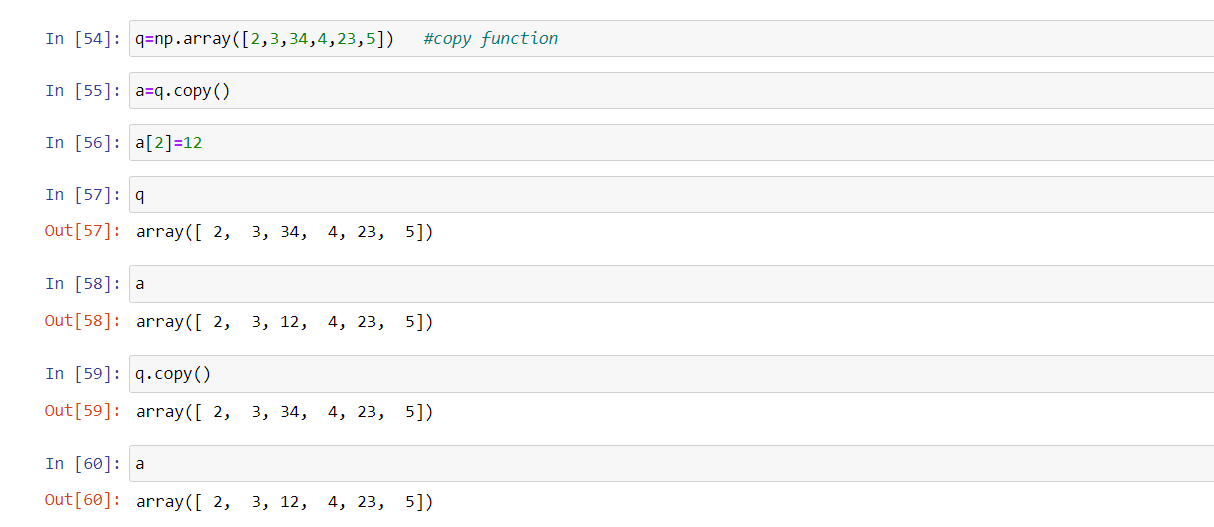
**np.full()**

**np.full(shape, fill\_value, dtype = None, order = ‘C’):**Return a new array with the same shape and type as a given array filled with a fill\_value.

****

**Copy()**

Use np.copy() function to copy Python NumPy array (ndarray) to another array. This method takes the array you wanted to copy as an argument and returns an array copy of the given object. The copy owns the data and any changes made to the copy will not affect the original array. Alternatively, you can also try ndarray.copy() function.

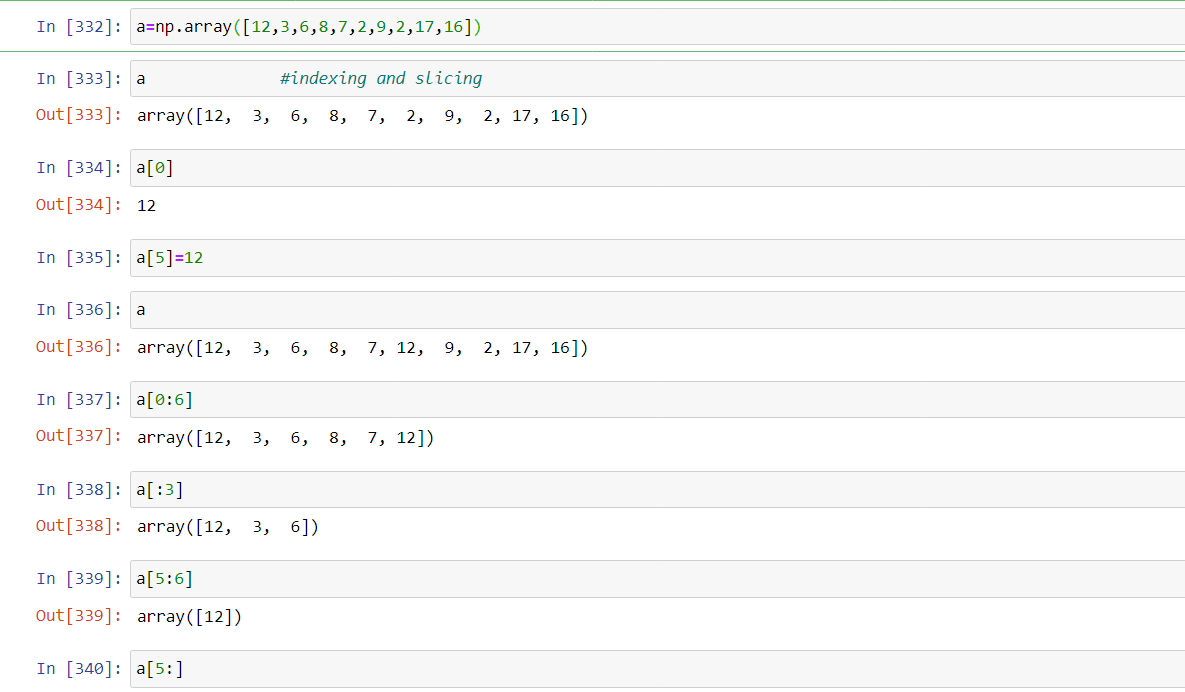
****

**Indexing and slicing**

Contents of ndarray object can be accessed and modified by indexing or slicing, just like Python's in-built container objects.

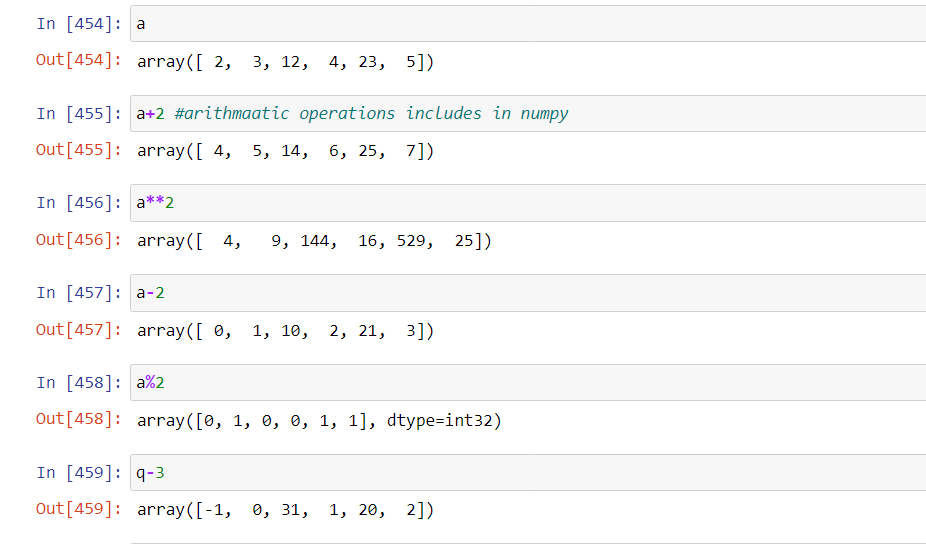
As mentioned earlier, items in ndarray object follows zero-based index. Three types of indexing methods are available − **field access, basic slicing** and **advanced indexing**.

Basic slicing is an extension of Python's basic concept of slicing to n dimensions. A Python slice object is constructed by giving **start, stop**, and **step** parameters to the built-in **slice** function. This slice object is passed to the array to extract a part of array.

****

# **Arithmetic Operations**

| **Function** | **Description** |
| --- | --- |
| Add | Add corresponding elements in arrays |
| Subtract | Subtract elements in second array from first array |
| Multiply | Multiply array elements |
| divide, floor\_divide | Divide or floor divide (truncating the remainder) |
| Power | Raise elements in first array to powers indicated in second array |

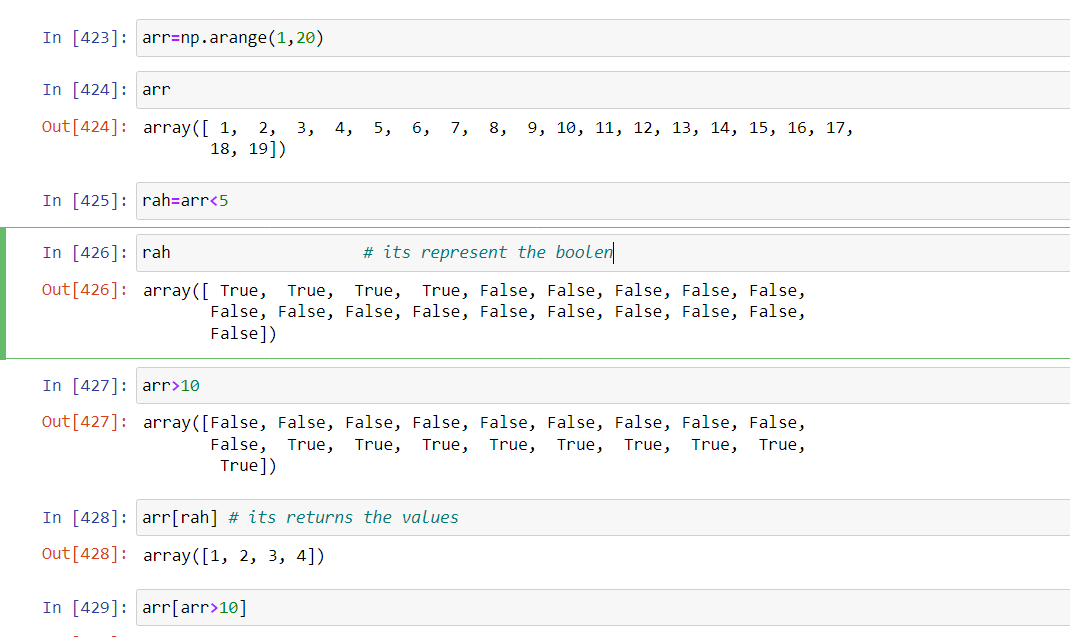
****

**Comparison operators in NumPy**

NumPy provides several comparison and logical operations that can be performed on NumPy arrays.

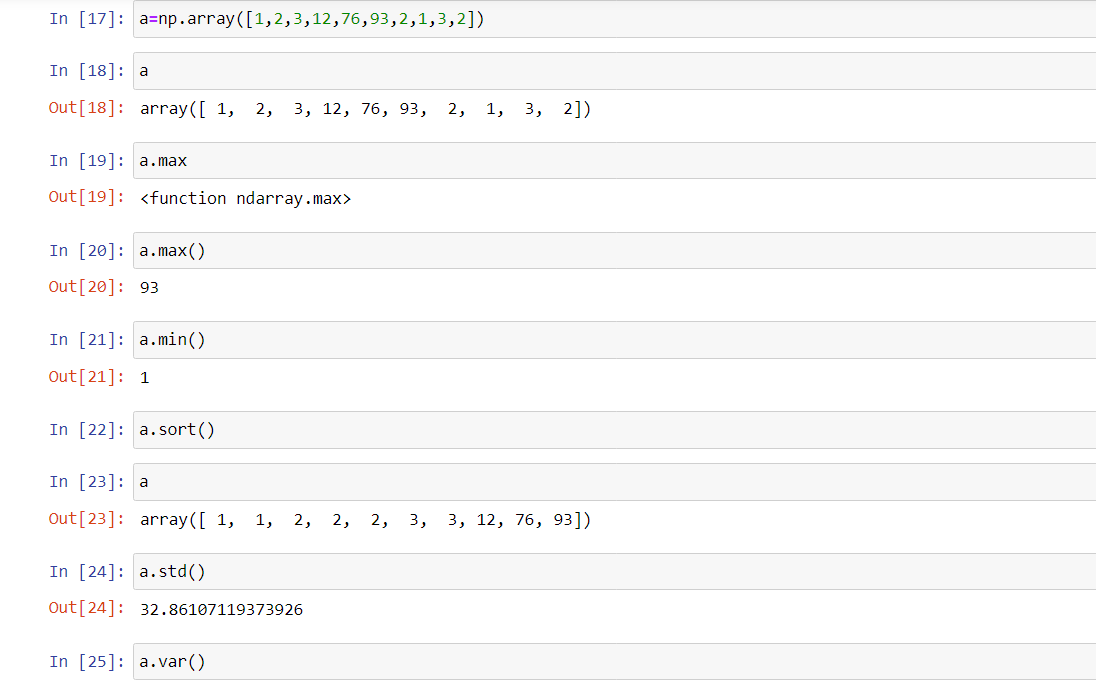
NumPy's comparison operators allow for element-wise comparison of two arrays.

Similarly, logical operators perform boolean algebra, which is a branch of algebra that deals with True and False statements.

****

**Functions**

|  |  |  |
| --- | --- | --- |
| np.sum | np.nansum | Compute sum of elements |
| np.prod | np.nanprod | Compute product of elements |
| np.mean | np.nanmean | Compute mean of elements |
| np.std | np.nanstd | Compute standard deviation |
| np.var | np.nanvar | Compute variance |
| np.min | np.nanmin | Find minimum value |
| np.max | np.nanmax | Find maximum value |

****

****