# Introduction to Python

Python in simple words is a High-Level Dynamic Programming Language which is interpreted. Guido Van Rossum, the father of Python had simple goals in mind when he was developing it, easy looking code, readable and open source. Python is ranked as the 3rd most prominent language followed by JavaScript and Java in a survey held in 2018 by Stack Overflow which serves proof to it being the most growing language.

Presented by Pregrad Class

# What is Python?

Python is currently my favorite and most preferred language to work on because of its *simplicity, powerful libraries, and readability*. You may be an old school coder or may be completely new to programming, Python is the best way to get started!

Python provides features listed below :

- Simplicity: Think less of the syntax of the language and more of the code.
- Open Source: A powerful language and it is free for everyone to use and alter as needed.
- Portability: Python code can be shared and it would work the same way it was intended to. Seamless and hassle-free.
- Being Embeddable & Extensible: Python can have snippets of other languages inside it to perform certain functions.
- Being Interpreted: The worries of large memory tasks and other heavy CPU tasks are taken care of by Python itself leaving you to worry only about coding.
- Huge amount of libraries: Data Science? Python has you covered. Web Development? Python still has you covered. Always.
- Object Orientation: Objects help breaking-down complex real-life problems into such that they can be coded and solved to obtain solutions.

## Jupyter Notebook

Jupyter Notebooks are a powerful way to write and iterate on your Python code for data analysis. Jupyter Notebook is built off of IPython and the Kernel runs the computations and communicates with the Jupyter Notebook front-end interface. Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. It is used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

## Why Learn Python?

Python's syntax is very easy to understand. The lines of code required for a task is less compared to other languages. Let me give you an example – If I have to print "Welcome To Edureka!" all I have to type:

```
print("Hi")
```

1. **Simple and easy to learn**

2. **Free and Open Source**

3. **Portable**

3. **Supports different programming paradigm**

3. **Extensible**

If you are wondering where you can use Python (Python Application), let me tell you that is where Python stands out.
It is advantageous over other programming languages because it is a:

- Syntax lite language
- Easy to use & learn
- Open-source language
- Dynamic Memory Allocation
- Extensive Support Libraries
- Desktop GUI applications
- Business applications
- Database Access
- Robust Web Application Development
- Supports Math & AI

## Applications of Python

Python is a general-purpose programming language that is used for a wide variety of applications. It is an interpreted, object-oriented, high-level programming language with a design philosophy emphasizing code readability  In this python tutorial, we think it is necessary that you understand its applications.

The language has an extensive standard library and many third-party modules. These libraries and modules can be used to develop desktop applications, web applications, and games. These applications have been explained briefly below:

- **Web and Internet Development**: Frameworks, micro-frameworks, content management systems, you name it – they've got it! Python offers a vast range of choices for web development. These options also include scraping tools like BeautifulSoup, and Twisted Python for asynchronous network programming and protocol support for HTML, XML, JSON, FTP, IMAP, etc.
- 
- **Scientific and Numeric:** You cannot imagine scientific computing without Python in the picture. Popular packages like numpy, pandas, Scipy, etc have made Python the #1 go-to language for scientific and numeric applications.
- 
- **Software Development:** Python has built a strong reputation among software developers for its immense support for build control, management, testing, etc. Some of the popular packages include SCons, Apache Gump, BuildBot, etc.
- 
- **Desktop GUIs**: Most default python distributions have the Tk library by default. You also have platform-specific tools like GTK+ and Microsoft Foundation Classes
- 
- **Business Applications:** Python is used to build ERP and e-commerce systems. Some popular application platforms and management systems include Odoo and Tryton

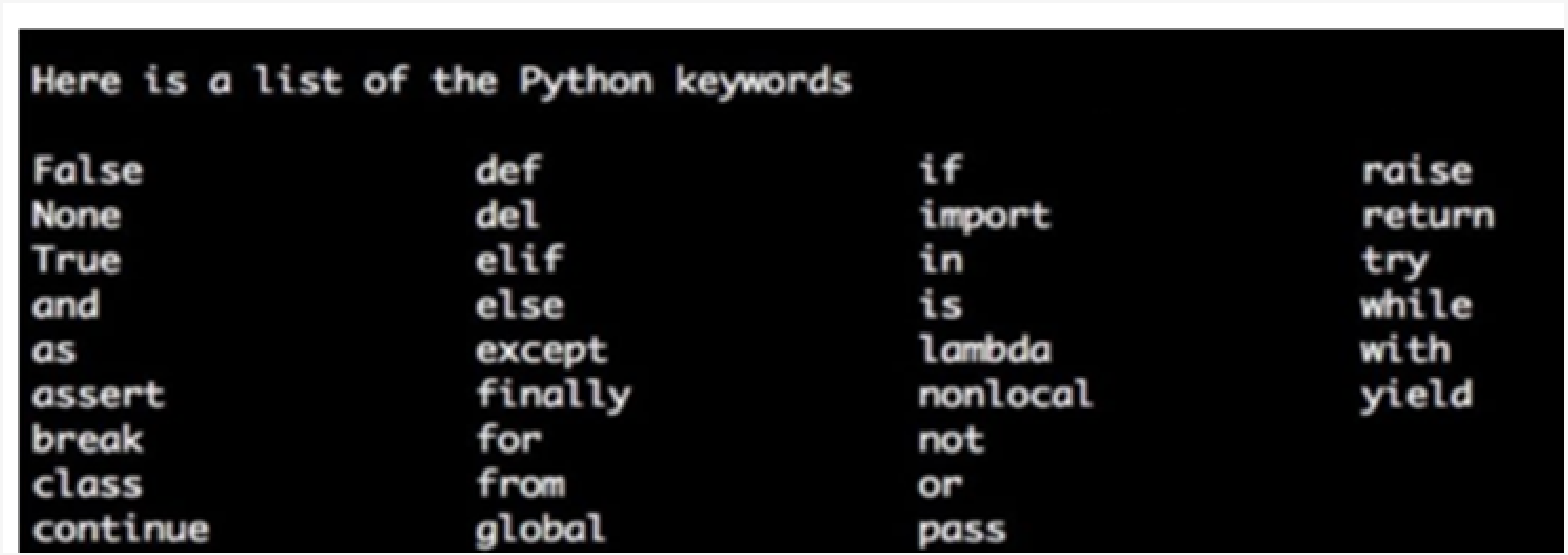
Let us start coding now!

# Python Basics

The basic concepts in any programming language are the foundation of any programmer, We will start with the most basic concept in python.

## Python Keywords

**Keywords are nothing but special names that are already present in python. We can use these keywords for specific functionality while writing a python program. Following is the list of all the keywords that we have in python:**



```
Here is a list of the Python keywords

False               def                 if                  raise
None                del                 import              return
True                elif                in                  try
and                 else                is                  while
as                  except              lambda              with
assert              finally             nonlocal            yield
break               for                 not
class               from                or
continue            global              pass
```

**Identifiers** are user defined names that we use to represent variables, classes, functions, modules etc.

## Python Popular libraries:

- **Tensorflow:** It is specially used for developing and training highly efficient Machine Learning and Deep Learning models, TensorFlow can also help you deploy these models to a host of platforms, such as a CPU, GPU(Graphic Processing unit), or a TPU(Tensor Processing Unit), with ease.
- **NumPy:** This library is utilized in adding support for large, multi-dimensional arrays and matrices, accompanied with a large collection of high-level mathematical functions to operate on these arrays.
- **Pandas:** It’s a software library specifically made for the Python programming language for the purpose of data analysis and manipulation.
- **Keras:** It is a software library which is open-source & allows a Python interface for artificial neural networks.
- **Matplotlib:** It is a library used for plotting in Python and its numerical mathematics extension NumPy.

# Python Comments

programmer-coherent statements, that describe what a block of code means. They get very useful when you are writing large codes. It’s practically inhuman to remember the names of every variable when you have a hundred-page program or so. Therefore, making use of comments will make it very easy for you, or someone else to read as well as modify the code.

## Python Block Comments

A block comment in python is written with the same indentation as the code, it is used to explain the code. A block comment looks something like the one written in the example below.

# Python Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
In Python you don’t need to declare variables before using it, unlike other languages like Java, C etc.

## Assigning values to a variable

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables. Consider the below example:

## Python Global Variable

In python, GLOBAL Keyword can be used to access/modify the variables out of the current scope.

## Data Types In Python

Python supports various data types, these data types define the operations possible on the variables and the storage method. In this section of the Python tutorial, we'll go through some of the commonly used data types in Python.
Let's discuss each of these in detail. In this Python tutorial, we'll start with 'STRINGS' data type.

- **Python Strings :** Strings are among the most popular data types in Python. We can create them simply by enclosing characters in quotes. Python treats single and double quotes in exactly the same fashion. Consider the example below:

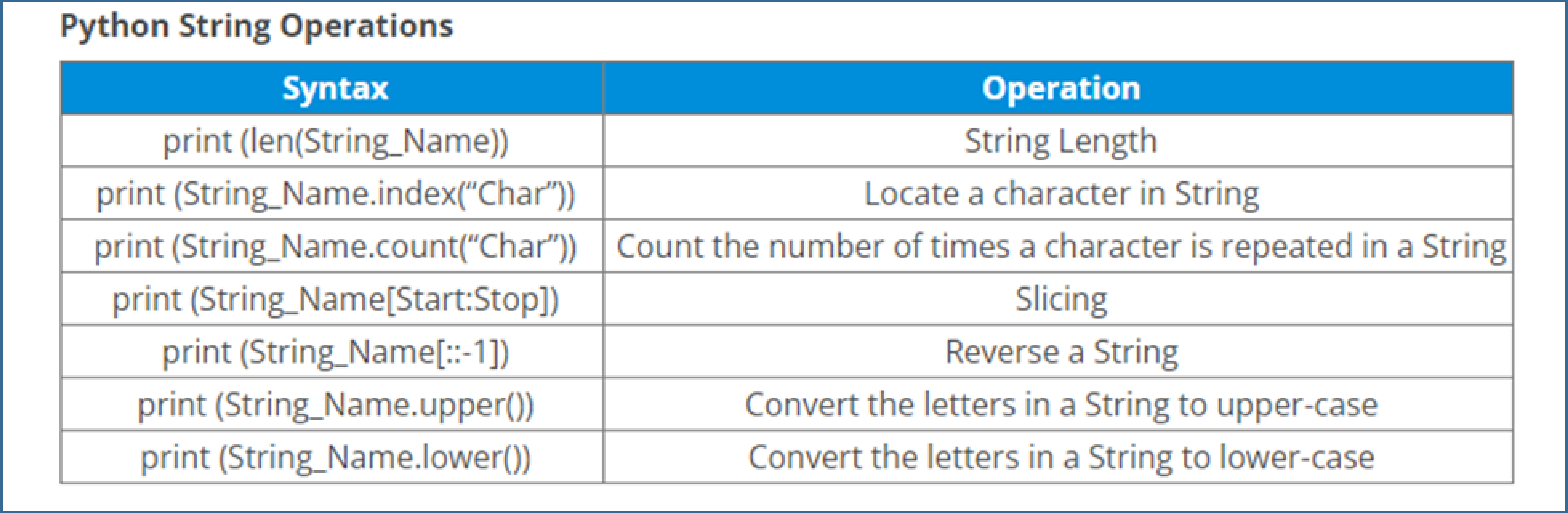
**Python String Methods :**
Some of the string methods used in python are written below:

- strip()
- Count()
- split()
- translate()
- index()
- format()
- find()
- center()
- join()

**Python Numbers:**
Just as expected Numeric data types store numeric values. They are immutable data types, this means that you cannot change its value. Python supports three different Numeric data types:

**Python Integer**
It holds all the integer values i.e. all the positive and negative whole numbers, example – 10.

**Python Float**
It holds the real numbers and are represented by decimal and sometimes even scientific notations with E or e indicating the power of 10 (2.5e2 = 2.5 x 102 = 250), example – 10.24.



## Python String Operations

| Syntax | Operation |
|---|---|
| print (len(String_Name)) | String Length |
| print (String_Name.index(“Char”)) | Locate a character in String |
| print (String_Name.count(“Char”)) | Count the number of times a character is repeated in a String |
| print (String_Name[Start:Stop]) | Slicing |
| print (String_Name[::-1]) | Reverse a String |
| print (String_Name.upper()) | Convert the letters in a String to upper-case |
| print (String_Name.lower()) | Convert the letters in a String to lower-case |

## Python Complex

These are of the form a + bj, where a and b are floats and J represents the square root of -1 (which is an imaginary number), example – 10+6j.

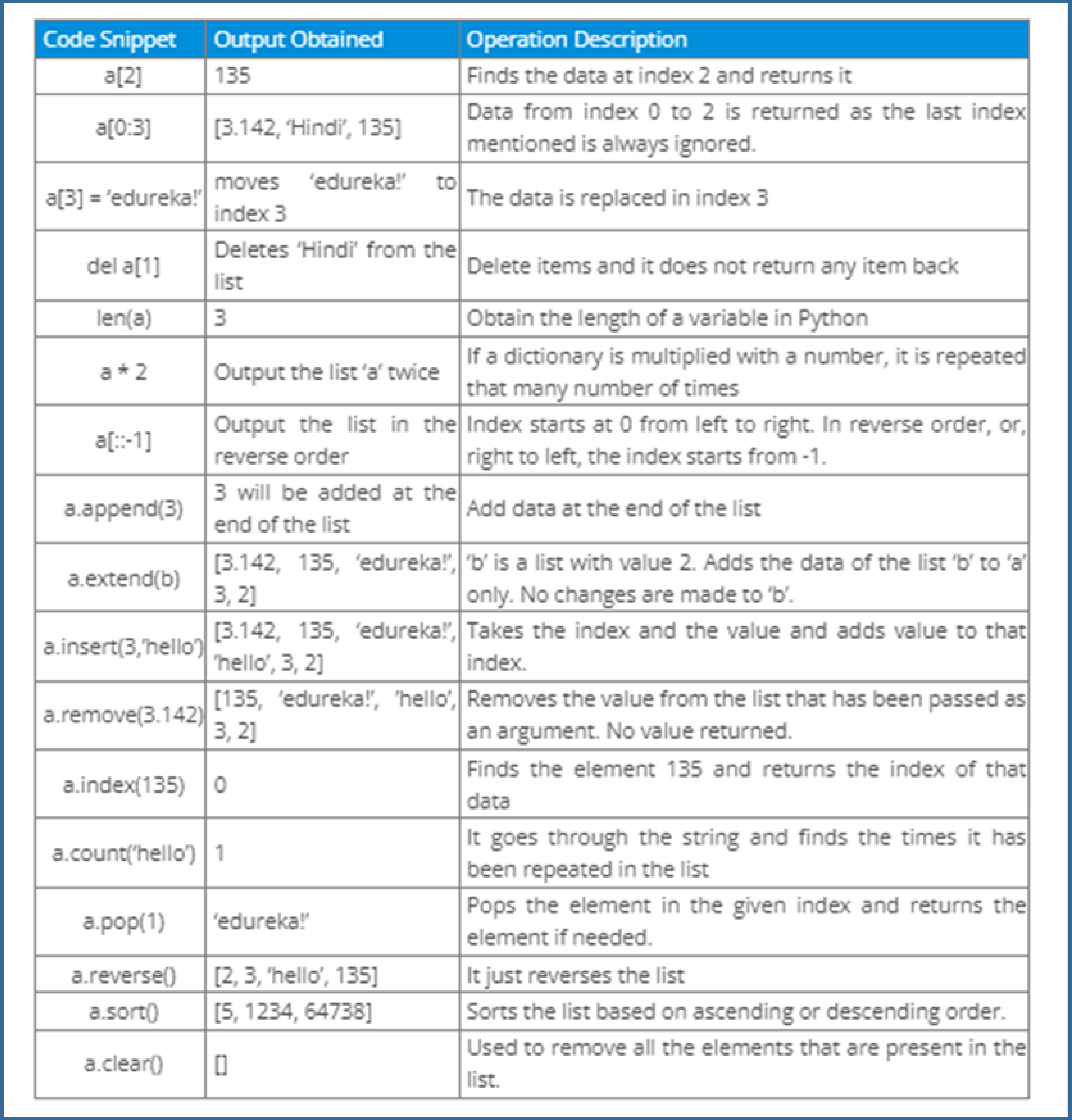
## Python Boolean

These are the decisive data types, they only return categorical value, i.e true or false.
Now you can even perform type conversion. For example, you can convert the integer value to a float value and vice-versa.
Ex —

## Lists

List in simple words can be thought of as arrays that exist in other languages but with the exception that they can have heterogeneous elements in them, i.e, different data types in the same list. Lists are mutable, meaning that you can change the data that is available in them.

For those of you who do not know what an array is, you can understand it by imagining a Rack that can hold data in the way you need it to. You can later access the data by calling the position in which it has been stored which is called as Index in a programming language.



| Code Snippet | Output Obtained | Operation Description |
|---|---|---|
| a[2] | 135 | Finds the data at index 2 and returns it |
| a[0:3] | [3.142, 'Hindi', 135] | Data from index 0 to 2 is returned as the last index mentioned is always ignored. |
| a[3] = 'edureka!' | moves 'edureka!' to index 3 | The data is replaced in index 3 |
| del a[1] | Deletes 'Hindi' from the list | Delete items and it does not return any item back |
| len(a) | 3 | Obtain the length of a variable in Python |
| a * 2 | Output the list 'a' twice | If a dictionary is multiplied with a number, it is repeated that many number of times |
| a[::-1] | Output the list in the reverse order | Index starts at 0 from left to right. In reverse order, or, right to left, the index starts from -1. |
| a.append(3) | 3 will be added at the end of the list | Add data at the end of the list |
| a.extend(b) | [3.142, 135, 'edureka!', 3, 2] | 'b' is a list with value 2. Adds the data of the list 'b' to 'a' only. No changes are made to 'b'. |
| a.insert(3,'hello') | [3.142, 135, 'edureka!', 'hello', 3, 2] | Takes the index and the value and adds value to that index. |
| a.remove(3.142) | [135, 'edureka!', 'hello', 3, 2] | Removes the value from the list that has been passed as an argument. No value returned. |
| a.index(135) | 0 | Finds the element 135 and returns the index of that data |
| a.count('hello') | 1 | It goes through the string and finds the times it has been repeated in the list |
| a.pop(1) | 'edureka!' | Pops the element in the given index and returns the element if needed. |
| a.reverse() | [2, 3, 'hello', 135] | It just reverses the list |
| a.sort() | [5, 1234, 64738] | Sorts the list based on ascending or descending order. |
| a.clear() | [] | Used to remove all the elements that are present in the list. |

# Python Tuples

A Tuple is a sequence of immutable Python objects. Tuples are sequences, just like Lists. We will see the differences between tuples and lists in this python tutorial.

Tuples cannot be changed unlike lists
Tuples use parentheses, whereas lists use square brackets.
Now you must be thinking why Tuples when we have Lists?
So the simple answer would be, Tuples are faster than Lists. If you're defining a constant set of values that you just want to iterate, then use Tuple instead of a List.
Guys, all Tuple operations are similar to Lists, but you cannot update, delete or add an element to a Tuple.

Now, stop being lazy and don't expect me to show all those operations, try it yourself.
Now that you have understood the various list functions, let's move over to understanding Tuples in Python Basics.

## Tuples

Tuples in Python are the same as lists. Just one thing to remember, tuples are immutable. That means that once you have declared the tuple, you cannot add, delete or update the tuple. Simple as that. This makes tuples much faster than Lists since they are constant values.
Operations are similar to Lists but the ones where updating, deleting, adding is involved, those operations won’t work. Tuples in Python are written a=() or a=tuple() where ‘a’ is the name of the tuple.

```
a = ('List', 'Dictionary', 'Tuple', 'Integer', 'Float')
print(a)
Output = (‘List’, ‘Dictionary’, ‘Tuple’, ‘Integer’, ‘Float’)
```

That basically wraps up most of the things that are needed for tuples as you would use them only in cases when you want a list that has a constant value, hence you use tuples. Let us move over to Dictionaries in Python Basics.

## Python Sets

A Set is an unordered collection of items. Every element is unique.
A Set is created by placing all the items (elements) inside curly braces {}, separated by a comma.
Ex -- set_name = {1,2,3,4,5}

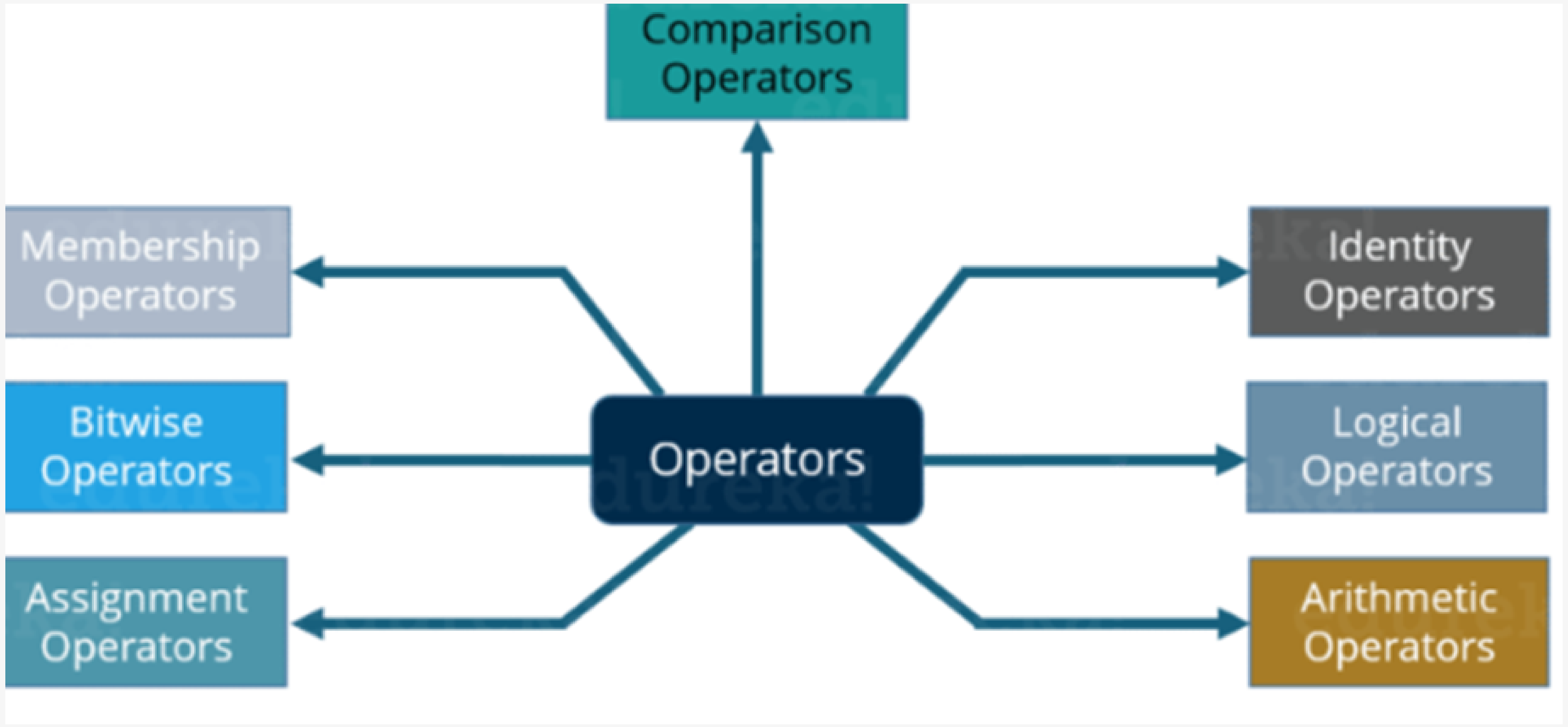
## Python Dictionary

Any python tutorial is worthless without a proper explanation of dictionaries. Now let me explain you Dictionaries with an example.
I am guessing you guys know about Aadhaar Card. For those of you who don’t know what it is, it is nothing but a unique ID which has been given to all Indian citizen. So for every Aadhaar number, there is a name and few other details attached.
Now you can consider the Aadhaar number as a ‘Key’ and the person’s detail as the ‘Value’ attached to that Key.
Dictionaries contains these ‘Key Value’ pairs enclosed within curly braces and Keys and values are separated with ‘:’. Consider the below example:

Dictionary Methods
clear()
copy()
values()
update()
fromkeys()
get()
items()
keys()
pop()
popitem()

## Operators in Python:

Operators are the constructs which can manipulate the values of the operands. Consider the expression 2 + 3 = 5, here 2 and 3 are operands and + is called operator.
Python supports the following types of Operators:



**Comparison Operators:**
These Operators compare the values on either sides of them and decide the relation among them. Assume A = 10 and B = 20.

**Assignment Operators:**
An Assignment Operator is the operator used to assign a new value to a variable.

# Flow Control and Conditioning in Python

You know that code runs sequentially in any language, but what if you want to break that flow such that you are able to add logic and repeat certain statements such that your code reduces and are able to obtain a solution with lesser and smarter code. After all, that is what coding is. Finding logic and solutions to problems and this can be done using loops in Python and conditional statements.
Conditional statements are executed only if a certain condition is met, else it is skipped ahead to where the condition is satisfied. Conditional statements in Python are the if, elif and else.
With conditional statements understood, let us move over to loops. You would have certain times when you would want to execute certain statements again and again to obtain a solution or you could apply some logic such that a certain similar kind of statements can be executed using only 2 to 3 lines of code. This is where you use loops in Python.
Loops can be divided into 2 kinds.
Finite: This kind of loop works until a certain condition is met
Infinite: This kind of loop works infinitely and does not stop ever.

Loops in Python or any other language have to test the condition and they can be done either before the statements or after the statements. They are called :

- **Pre-Test Loops**: Where the condition is tested first and statements are executed following that
- **Post Test Loops**: Where the statement is executed once at least and later the condition is checked.

You have 2 kinds of loops in Python:

- for
- while

Let us understand each of these loops with syntaxes and code snippets below.

**For Loops:** These loops are used to perform a certain set of statements for a given condition and continue until the condition has failed. You know the number of times that you need to execute the for loop.
This is how the for loops work in Python. Let us move ahead with the while loop in Python Basics.

**While Loops:** While loops are the same as the for loops with the exception that you may not know the ending condition. For loop conditions are known but the while loop conditions may not.

# File Handling with Python

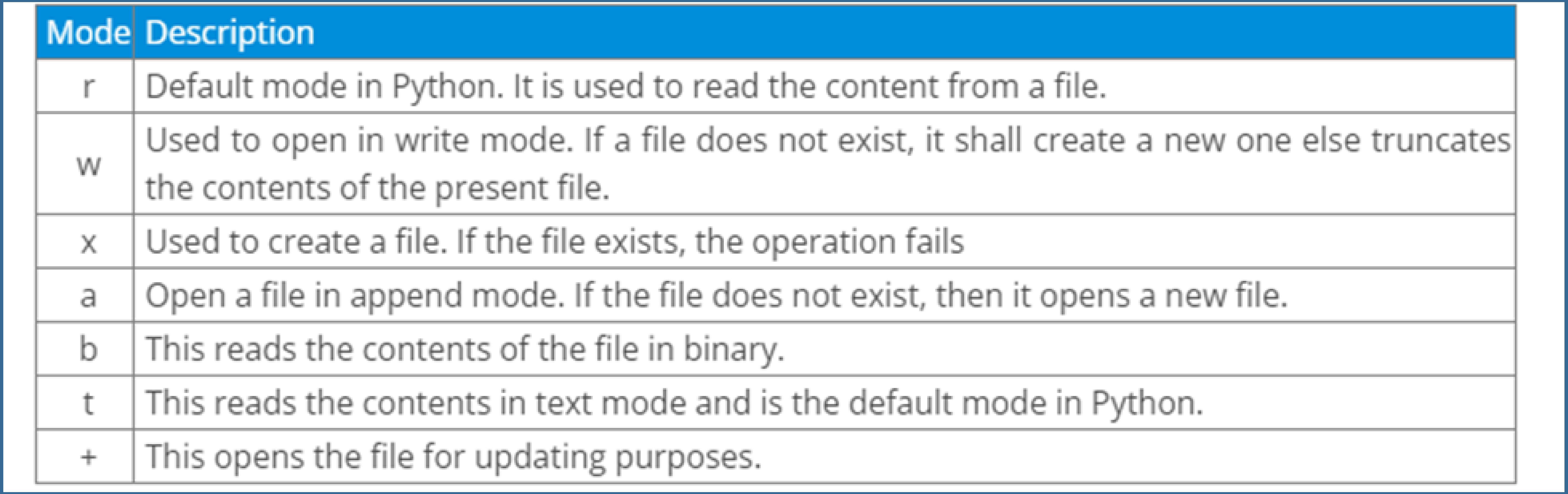
Python has built-in functions that you can use to work with files such as reading and writing data from or to a file. A file object is returned when a file is called using the open() function and then you can do the operations on it such as read, write, modify and so on.

If you would like to know about file handling in detail, you can go through the complete tutorial- File handling in Python.

The flow of working with files is as follows :

- Open the file using the open() function
- Perform operations on the file object
- Close the file using the close() function to avoid any damage to be done with the file

```
File_object = open('filename','r')
```



| Mode | Description |
|---|---|
| r | Default mode in Python. It is used to read the content from a file. |
| w | Used to open in write mode. If a file does not exist, it shall create a new one else truncates the contents of the present file. |
| x | Used to create a file. If the file exists, the operation fails |
| a | Open a file in append mode. If the file does not exist, then it opens a new file. |
| b | This reads the contents of the file in binary. |
| t | This reads the contents in text mode and is the default mode in Python. |
| + | This opens the file for updating purposes. |

## Object Oriented Programming Structures

Older programming languages were structured such that data could be accessed by any module of the code. This could lead to potential security issues that led developers to move over to Object-Oriented Programming that could help us emulate real-world examples into code such that better solutions could be obtained.

There are 4 concepts of OOPS which are important to understand.

They are:

- **Inheritance:** Inheritance allows us to derive attributes and methods from the parent class and modify them as per the requirement. The simplest example can be for a car where the structure of a car is described and this class can be derived to describe sports cars, sedans and so on.
- **Encapsulation:** Encapsulation is binding data and objects together such that other objects and classes do not access the data. Python has private, protected and public types whose names suggest what they do. Python uses '_' or '__' to specify private or protected keywords.
- **Polymorphism:** This allows us to have a common interface for various types of data that it takes. You can have similar function names with differing data passed to them.
- **Abstraction**: Abstraction can be used to simplify complex reality by modeling classes appropriate to the problem.

# Thank you

Have a fun learning with us!