# DataType

Datatype represents the type of data stored into a variable or memory.

Type of Data type:-

* Built-in Data type
* User Defined Data type

# Built-in Datatype

These datatypes are provided by Python Language.

* None Type
* Numeric Types
* Sequences
* Sets
* Mappings

# User Defined Data type

* Array
* Class
* Module

# None Type

None datatype represents an object that doesn’t contain any value.

Following are the Numeric Data type:-

Int

Float Complex

Int – The int datatype represents an integer number. An integer number without any decimal point or fraction part. In Python, It is possible to store very large integer number as there is no limit for the size of an int datatype.

Ex:-

int type variable

20, 10, -50, -1002



y = 10

pin\_code = 564512 pin\_code = 564512



int value

Float – The float data type represents floating point numbers. A floating point number is a number that contains a decimal point.

Ex:-

25.56, 10.5, -45.69, -0.8

price = 25.56

run\_rate = -0.8

value = 5.1e5



5.1 × 105

float type variable

run\_rate=



-0.8

float value



**5.1e5**

It’s scientific notation where e or E represents exponentiation which represents the power of 10

Complex – A complex number is a number that is written in the form of a + bj or a + bJ. Where,

a = Real Part of the number

complex type variable

b = Imaginary part of the number



j or J = Square root value of -1



a and b may contain integer or float number.

complex number

Ex:- 5+7j, 0.8+2j

com = 5+7j

com = 5+7j

# Bool type

The bool datatype represents boolean value True or False. Python internally represents True as 1 and False as 0.

Ex:- True, False True + True = 2 True – False = 1

Following are sequence type:-

String List Tuple Range

String – String represents group of characters. Strings are enclosed in double quotes or single quotes.

Ex:- “Hello”, “Nikhil”, ‘Rahul’

String type variable

str1 = “Nikhil”

str1 = “Nikhil”



str1 = “Nikhil”



String

List – A list represents a group of elements. A list can store different types of elements which can be modified. Lists are dynamic which means size is not fixed. Lists are represented using square bracket [ ].

Ex:- data = [10, 20, -50, 21.3, ‘Hello’]

data[1] = 40

data

data

|  |  |
| --- | --- |
| [-5] | 10 |
| [-4] | 20 |
| [-3] | -50 |
| [-2] | 21.3 |
| [-1] | Hello |

|  |  |
| --- | --- |
| [0] | 10 |
| [1] | 20 |
| [2] | -50 |
| [3] | 21.3 |
| [4] | Hello |



Tuple – A tuple contains a group of elements which can be different types. It is similar to List but Tuples are read-only which means we cannot modify its element. Tuples are represented using parentheses ( ).

Ex:- data = (10, 20, -50, 21.3, ‘Hello’)

data[1] = 40

data

data

|  |  |
| --- | --- |
| [-5] | 10 |
| [-4] | 20 |
| [-3] | -50 |
| [-2] | 21.3 |
| [-1] | Hello |

|  |  |
| --- | --- |
| [0] | 10 |
| [1] | 20 |
| [2] | -50 |
| [3] | 21.3 |
| [4] | Hello |



Range – Range represents a sequence of numbers. The numbers in the range are not modifiable.

Ex:- rg = range(5) 0 1 2 3 4

rg = range(10, 20, 2) 10 12 14 16 18

rg

rg

rg



|  |  |
| --- | --- |
| [0] | 0 |
| [1] | 1 |
| [2] | 2 |
| [3] | 3 |
| [4] | 4 |

|  |  |
| --- | --- |
| [-5] | 0 |
| [-4] | 1 |
| [-3] | 2 |
| [-2] | 3 |
| [-1] | 4 |

|  |  |
| --- | --- |
| [0] | 10 |
| [1] | 12 |
| [2] | 14 |
| [3] | 16 |
| [4] | 18 |

# Set Type

A set is an unordered collection of elements much like a set in mathematics.

The order of elements is not maintained in the sets. It means the elements may not

appear in the same order as they are entered into the set.

A set does not accept duplicate elements.

Sets are unordered so we cannot access its element using index.

Sets are represented using curly brackets { }.

Ex:-

data = {10, 20, 30, “welcome”, “Raj”, 40}

data = {10, 20, 30, “welcome”, “Raj”, 40, 10, 20}

data[0] = 10

# Mapping Type/ dict / Dictionary

A map represents a group of elements in the form of key value pairs. Ex:-

data = {101: ‘Rahul’, 102: ‘Raj’, 103: ‘Sonam’ }

data = {‘rahul’:2000, ‘raj’:3000, ‘sonam’:8000, }

data data

|  |  |
| --- | --- |
| [101] | Rahul |
| [102] | Raj |
| [103] | Sonam |

|  |  |
| --- | --- |
| [‘rahul’] | 2000 |
| [‘raj’] | 3000 |
| [‘sonam’] | 8000 |



# Character

There is no concept of char data type in Python to represent individual character.

**Features of Python OR Advantages of Python**

**Simple and easy to learn:**

* Python is a simple programming language. When we read Python program, we can feel like reading English statements.
* The syntaxes are very simple and only 30+ keywords are available.
* When compared with other languages, we can write programs with very less number of lines. Hence more readability and simplicity.
* We can reduce development and cost of the project.

**Freeware and Open Source:**

* We can use Python software without any license and it is freeware.
* Its source code is open, so that we can we can customize based on our requirement. E.g.: Jython is customized version of Python to work with Java Applications.

**High Level Programming language:**

* Python is high level programming language and hence it is programmer friendly language. Being a programmer we are not required to concentrate low level activities like memory management and security etc..

**Platform Independent:**

* Once we write a Python program, it can run on any platform without rewriting once again. Internally PVM is responsible to convert into machine understandable form.

**Portability:**

* Python programs are portable. I.e. we can migrate from one platform to another platform very easily. Python programs will provide same results on any platform.

**Dynamically Typed:**

* In Python we are not required to declare type for variables. Whenever we are assigning the value, based on value, type will be allocated automatically. Hence Python is considered as dynamically typed language.

**But Java, C etc. are Statically Typed Languages because we have to provide type at the beginning only.**

**This dynamic typing nature will provide more flexibility to the programmer.**

**Both Procedure Oriented and Object Oriented:**

* Python language supports both Procedure oriented (like C, Pascal etc.) and object oriented (like C++, Java) features. Hence we can get benefits of both like security and reusability etc.

**Interpreted:**

* We are not required to compile Python programs explicitly. Internally Python interpreter will take care that compilation.

**If compilation fails interpreter raised syntax errors. Once compilation success then PVM (Python Virtual Machine) is**

**responsible to execute.**

**Extensible:**

* We can use other language programs in Python. The main advantages of these approaches are:

**We can use already existing legacy non-Python code**

**We can improve performance of the application**

**Embedded:**

* We can use Python programs in any other language programs.
* i.e we can embedd Python programs anywhere.

**Extensive Library:**

* Python has a rich inbuilt library.
* Being a programmer we can use this library directly and we are not responsible to implement the functionality.

**Limitations of Python/ Disadvantages of Python:-**

* Performance wise not up to the mark because it is interpreted language.
* Not using for mobile Applications.
* CONVERSION- Most of the languages has strong syntax. Python lack in this. This results in difficulty while translating a program into another programming language.

**Flavors of Python:**

**CPython:**

* It is the standard flavor of Python. It can be used to work with C lanugage Applications

**Jython or JPython:**

* It is for Java Applications. It can run on JVM

**IronPython:**

* It is for C#.Net platform

**PyPy:**

* The main advantage of PyPy is performance will be improved because JIT compiler is available inside PVM.

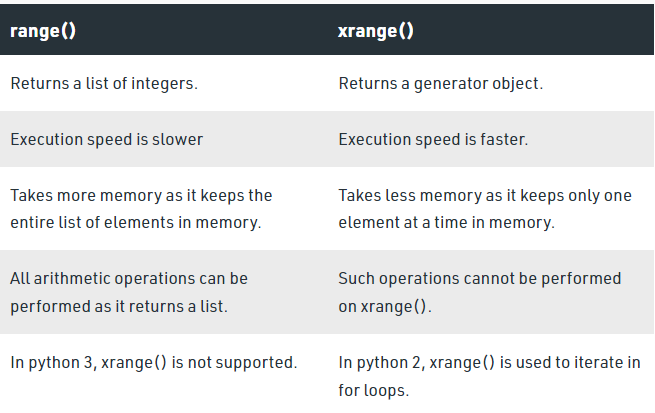
**RubyPython**

* For Ruby Platforms

**AnacondaPython**

* It is specially designed for handling large volume of data processing**.**

**range & xrange difference ---**



## How is memory managed in Python?

Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager. The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching

It is important to understand that the management of the Python heap is performed by the interpreter itself and that the user has no control over it, even if they regularly manipulate object pointers to memory blocks inside that heap. The allocation of heap space for Python objects and other internal buffers is performed on demand by the Python memory manager through the Python/C API functions

To avoid memory corruption, extension writers should never try to operate on Python objects with the functions exported by the C library: **malloc()**, **calloc()**, **realloc()** and **free()**.

* Memory management in python is managed by **Python private heap space**. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.
* The allocation of heap space for Python objects is done by Python’s memory manager. The core API gives access to some tools for the programmer to code.
* Python also has an **inbuilt garbage collector**, which recycles all the unused memory and so that it can be made available to the heap space.