Data types in Python

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes. Some of the datatypes in Python are:-

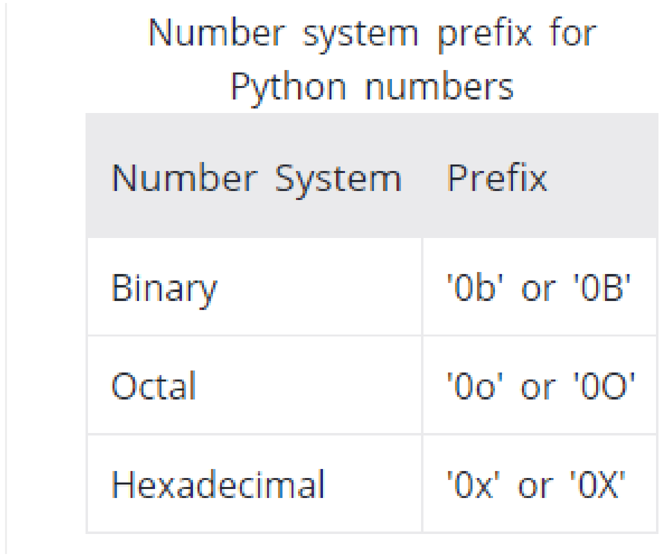
- Python Numbers
- Python Strings
- Python List
- Python Tuple
- Python Dictionary

Number Data Type in Python

- Python supports integers, floating point numbers and complex numbers. They are defined as int, float and complex class in Python.
- Integers and floating points are separated by the presence or absence of a decimal point. 5 is integer whereas 5.0 is a floating point number.
- Complex numbers are written in the form, x + yj, where x is the real part and y is the imaginary part.
-

Number Data Type in Python

- We can use the type() function to know which class a variable or a value belongs to and isinstance() function to check if it belongs to a particular class.



Number system prefix for Python numbers

Number System	Prefix
Binary	'0b' or '0B'
Octal	'0o' or '0O'
Hexadecimal	'0x' or '0X'

Python List

- List is an ordered sequence of items. All the items in a list do not need to be of the same type.
- **List** is a collection which is ordered and changeable. It allows duplicate members.
- In Python programming, a list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.
- It can have any number of items and they may be of different types (integer, float, string etc.).
- Lists are mutable, and hence, they can be altered even after their creation.
- We can use the index operator [] to access an item in a list. Index starts from 0. So, a list having 5 elements will have index from 0 to 4.

Python List

e.g.

```
# empty list
my_list = []
# list of integers
my_list = [1, 2, 3]
# list with mixed datatypes
my_list = [1, "Hello", 3.4]
```

Python List

- Also, a list can even have another list as an item. This is called nested list.

e.g.

```
# nested list
my_list = ["mouse", [8, 4, 6], ['a']]
```

Adding elements to a List

- **Using append():**
  - Elements can be added to the List by using built-in **append()** function.
  - **Only one element** at a time can be added to the list by using append() method, for addition of multiple elements with the append() method, loops are used.

Adding elements to a List

- **Using insert() method:**
  - append() method only works for addition of elements at the end of the List, ***for addition of element at the desired position, insert()*** method is used.
  - Unlike append() which takes only one argument, insert() method requires two arguments(position, value).

Adding elements to a List

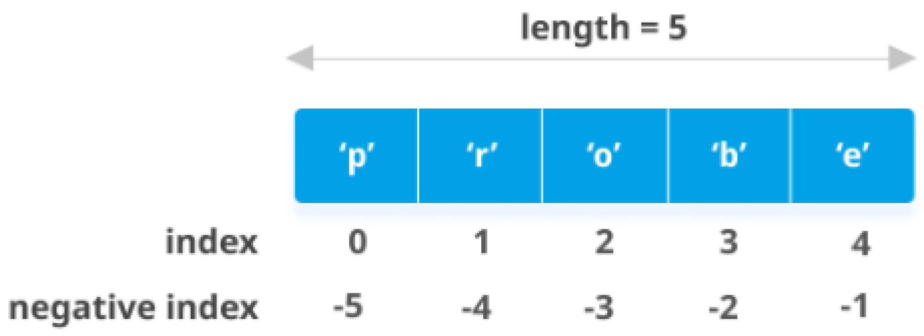
- **Using extend() method:**
  - **extend()**, this method is used to add multiple elements at the same time at the end of the list.

Access elements from a list

- We can use the index operator [] to access an item in a list. Index starts from 0. So, a list having 5 elements will have index from 0 to 4.
- Trying to access an element other that this will raise an IndexError. The index must be an integer. We can't use float or other types, this will result into TypeError.
- Nested list are accessed using nested indexing.

Access elements from a list

- Negative Indexing:
  - In Python, negative sequence indexes represent positions from the end of the array.
  - Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second last item etc
- E.g. my_list = ['p','r','o','b','e']



Removing elements from the List

- **Using remove() method:**
  - Elements can be removed from the List by using built-in **remove()** function but an Error arises if element doesn’t exist in the set.
  - Remove() method only removes one element at a time, to remove range of elements, iterator is used. remove() method is used to remove the given item.
  - If a list contains duplicate elements, the remove() method only removes the first matching element.
  - The syntax of the remove() method is:

    list.remove(element)

Removing elements from the List

- The syntax of the pop() method is:

  list.pop(index)

- The pop() method takes a single argument (index).
- The argument passed to the method is optional. If not passed, the default index **-1** is passed as an argument (index of the last item).

Removing elements from the List

- **del[a : b]** :-
  - This method **deletes all the elements in range** starting from index ‘a’ till ‘b’ mentioned in arguments.

```
a=[1,2,3,4,5]
del a[0:3]
print(a)
```

Slicing of a List

- To print a specific range of elements from the list, we use Slice operation. Slice operation is performed on Lists with the use of colon(:).
- To print elements from beginning to a range use [:Index],
- to print elements from end use [:-Index],
- to print elements from specific Index till the end use [Index:],
- to print elements within a range, use [Start Index:End Index] and
- to print whole List with the use of slicing operation, use [:].
- Further, to print whole List in reverse order, use [::-1].

List Methods in Python

- **len()** :- This function returns the **length** of list.
- **min()** :- This function returns the minimum element of list.
- **max()** :- This function returns the maximum element of list.

```
print (len(list))

print (min(list))

print (max(list))
```

List Methods in Python

- **count()** :- This function counts the **number of occurrences** of elements in list.

print (lis.count(3))

- **sum() :** Calculates sum of all the elements of List.
  Syntax: sum(List)
- **length:**Calculates total length of List.
  Syntax:len(list_name)