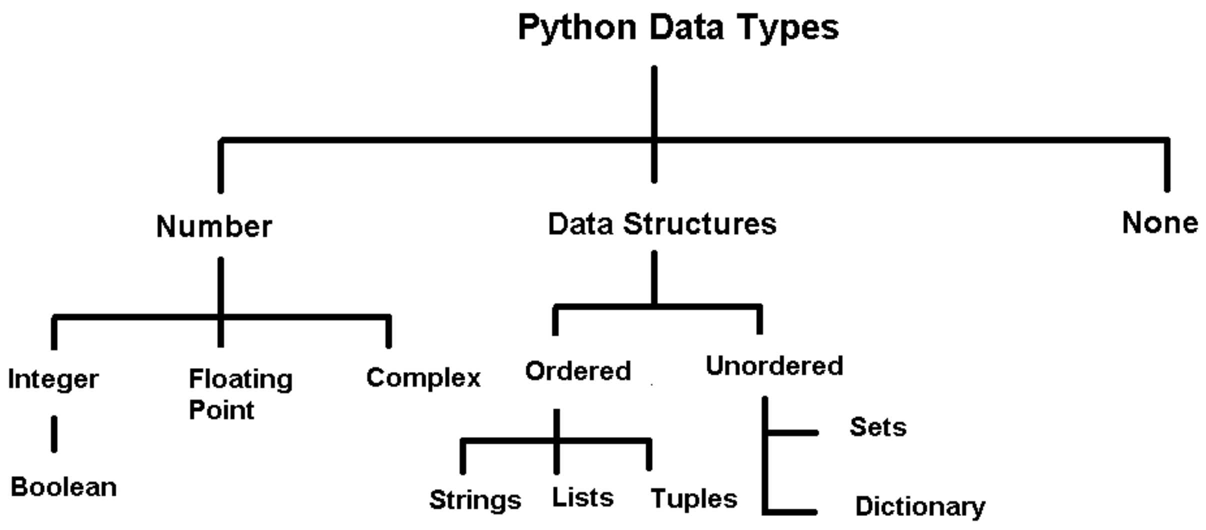
11 DEC 23| DAY - 4 | PYTHON
4/100
#100DAYSOFDATA SCIENCE
PYTHON | NUMPY | PANDAS | MATPLOTLIB | SEABORN | POWERBI | SQL | MACHINE LEARNING | & MANY MORE

DataTypes

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python, data types are actually classes and variables are instances (object) of these classes.



1. Numeric Datatypes

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.

1. Integers

This value is represented by int class. It contains positive or negative whole numbers.

```python
a = 10
print(a,type(a))
```

```
10 <class 'int'>
```

2. 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.

```python
b = 10.0
print(b,type(b))
```

```
10.0 <class 'float'>
```

3. Complex

Complex number is represented by a complex class. It is specified as (real part) + (imaginary part)j.

```python
c = 4+7j
print(c,type(c))
```

```
(4+7j) <class 'complex'>
```

2. Boolean Datatypes

Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). It is denoted by the class bool.

```python
d = True
print(d,type(d))
```

```
True <class 'bool'>
```

3. Sequential Datatypes

1. List

1. Lists in Python are denoted by square brackets [].
2. Lists allow accessing elements by their index.
3. Duplicate elements are allowed in lists.
4. Lists are mutable and can be modified after creation.
5. Lists can store multiple data types.

```
a = [] #Entpy list
for i in range(10):
    a.append(i)
print(a)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing in list

- Forward indexing starts form 0, 1, 2, 3, 4....
- Backward indexing starts form −1, −2, −3..., where −1 is the last element in the list

```
my_list = [10, 3.14, "Hello", True, [1, 2, 3], 2, 3, 4, 5, 7, 8]
my_list[2] ##getting element using forward indexing
my_list[-5] ##getting element using backward indexing
type(my_list)
```

```
list
```

Adding or assigning values in list

```
my_list[-3] = 78.7
my_list[-1] = 100+9j ##backward indexing start from -1
my_list[0] = 7   ##forward indexing starts from 0
my_list[6] = False
my_list
```

```
[7, 3.14, 'Hello', True, [1, 2, 3], 2, False, 4, 78.7, 7, (100+9j)]
```

The len() function returns the number of items in an objec

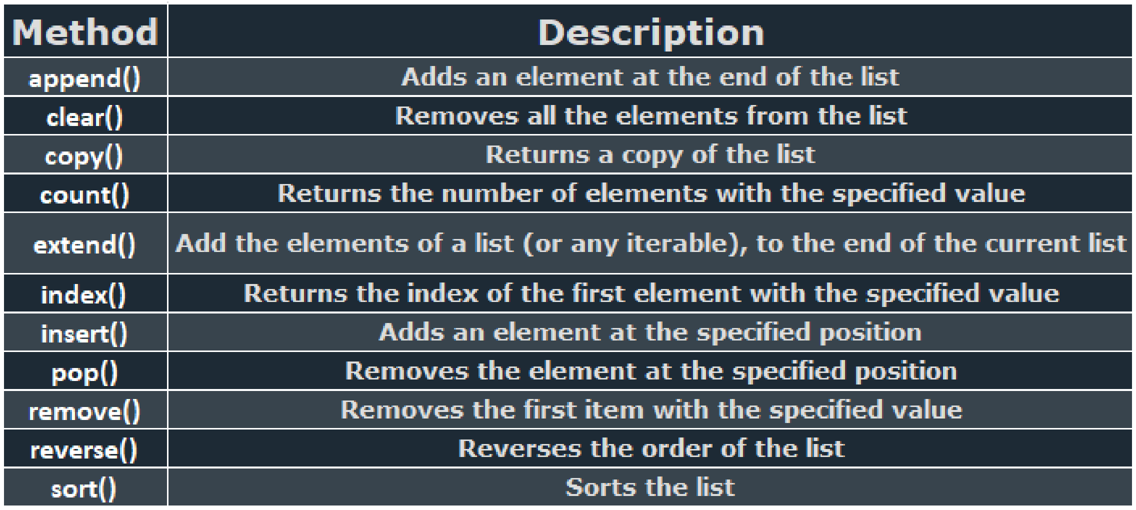
```
print(len(my_list)) ## Length or size of the list
```

```
11
```

Slicing

In Python, list slicing is a common practice and it is the most used technique for programmers to solve efficient problems start is the index of the list where slicing starts. stop is the index of the list where slicing ends. step allows you to select nth item within the range start to stop.

```
my_list[2:9]

['Hello', True, [1, 2, 3], 2, False, 4, 78.7]

my_list[-5:]

[False, 4, 78.7, 7, (100+9j)]

my_list[::-2]

[(100+9j), 78.7, False, [1, 2, 3], 'Hello', 7]

my_list[::3]

[7, True, False, 7]

my_list[:7:]

[7, 3.14, 'Hello', True, [1, 2, 3], 2, False]
```



Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

```
my_list.append(700)
my_list

[7, 3.14, 'Hello', True, [1, 2, 3], 2, False, 4, 78.7, 7, (100+9j),
700]

new_list = my_list.copy()
new_list

[7, 3.14, 'Hello', True, [1, 2, 3], 2, False, 4, 78.7, 7, (100+9j),
700]

my_list.count(7)
```

```
2
```

```python
dim = [54,89,12,247]
my_list.extend(dim)
my_list
```

```
[7,
 3.14,
 'Hello',
 True,
 [1, 2, 3],
 2,
 False,
 4,
 78.7,
 7,
 (100+9j),
 700,
 54,
 89,
 12,
 247]
```

```python
my_list.index(700)
```

```
11
```

```python
my_list.insert(0,77)
my_list
```

```
[77,
 7,
 3.14,
 'Hello',
 True,
 [1, 2, 3],
 2,
 False,
 4,
 78.7,
 7,
 (100+9j),
 700,
 54,
 89,
 12,
 247]
```

```python
my_list.pop(1)
```

```
7
```

```
my_list
```

```
[77,
 3.14,
 'Hello',
 True,
 [1, 2, 3],
 2,
 False,
 4,
 78.7,
 7,
 (100+9j),
 700,
 54,
 89,
 12,
 247]
```

```
my_list.remove(4)
my_list
```

```
[77,
 3.14,
 'Hello',
 True,
 [1, 2, 3],
 2,
 False,
 78.7,
 7,
 (100+9j),
 700,
 54,
 89,
 12,
 247]
```

```
sort = [0, 7, 95, 13, 94, 50, 2, 17, 28, 9]
sort.sort()
sort
```

```
[0, 2, 7, 9, 13, 17, 28, 50, 94, 95]
```

```
sort.reverse()
sort
```

```
[95, 94, 50, 28, 17, 13, 9, 7, 2, 0]
```

```
sort.clear()
sort
```

```
[]
```

```
A = [1,2,3,4,5]
B = [6,7,8,9,10]
d = A + B
d
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
print(sum(A)) #sum() is used to sum up the varible in list
print(sum(d))
```

```
15
55
```

Learning Progress:

- Numerical Datatypes: I have thoroughly grasped the concept of numerical data types in Python, including integers, floats, and complex numbers. I understand their operations and limitations.
- Boolean Datatypes: I have comprehensively explored boolean data types, True and False, and their application in conditional statements and logical operations.
- List Functions: I have mastered various functions associated with lists, including append(), clear(), copy(), pop(), reverse(), and others. I can utilize these functions to manipulate and manage lists effectively.
- Indexing: I have gained a solid understanding of both forward and backward indexing within lists. I can access and modify elements based on their position, including slicing techniques for extracting specific sub-lists.

Next Steps:

- Building Program 1: I am ready to develop a program that creates a list of multiples of 4 or 7 while removing any duplicate elements. I will utilize list functions and conditional statements to achieve this objective.
- Building Program 2: Next, I will build a program that sorts a list into two separate lists, one containing even numbers and the other containing odd numbers. I will utilize loop structures and conditional statements to perform this sorting operation.

Program 1

```
a = [100,200,300,200,400,500,700,900.5,768,0,210,-78,200,500]
n = len(a)
multi = []
result = []
for i in a:
    i = int(i)
    if i > 0:
        i = int(i)
        if i % 4 == 0 or i % 7 == 0:
            multi.append(i)

for i in a:
    if i not in result:
        result.append(i)


print(result)
print(multi)
```

```
[100, 200, 300, 400, 500, 700, 900.5, 768, 0, 210, -78]
[100, 200, 300, 200, 400, 500, 700, 900, 768, 210, 200, 500]
```

Program 2

```
a = [11,23,45,67,88,94,100,108.5,108.50,107.8,78,10]
even = []
odd = []
for i in a:
    if i % 2 == 0:
        even.append(i)
    else:
        odd.append(i)
print(even)
print(odd)
```

```
[88, 94, 100, 78, 10]
[11, 23, 45, 67, 108.5, 108.5, 107.8]
```

2. Set

1. Sets in Python are denoted by curly brackets {}.
2. Set is unhashable.
3. Sets supports multiple deatatypes(except mutable like list).
4. Set does not alow duplicate elements.
5. Sets are unordered.
6. Indexing is not support in sets.

```
a = {1,2,3,True,4,2,3,False,4,'zahid',4.5,('hi','hello',77),87j}
type(a)
```

```
set
```

```
b = list(a) #typecasting to list
print(b)
print(type(b))
```

```
[False, 1, 2, 3, 4, 4.5, 87j, ('hi', 'hello', 77), 'zahid']
<class 'list'>
```

```
c = {'zahid', 'hi', 78, 77}
c
```

```
{77, 78, 'hi', 'zahid'}
```

```
c.add('hey') ##It is use to add single element in the set and it will sort the set ascending
c
```

```
{77, 78, 'hey', 'hi', 'zahid'}
```

```
c.update(['hello', 'how', 'are', 75]) #It is is use to add multiple elements in the set
c
```

```
{75, 77, 78, 'are', 'hello', 'hey', 'hi', 'how', 'zahid'}
```

```
c.remove('hello') #It will throw errors if the element is not present in the set
c
```

```
{75, 77, 78, 'are', 'hey', 'hi', 'how', 'zahid'}
```

```
c.discard(107) #It will work same as remove but if the elemnt is not present in the set it wil not throw an error
```

```
c.pop() #It wil randomly pop element from the set
```

```
'hey'
```

```
c
```

```
{75, 77, 78, 'are', 'hi', 'how', 'zahid'}
```

```
c.clear()
c
```

```
set()
```

Union, Intersection and Difference

```
a = {1,2,3,4,5}
b = {4,5,6,7,8}
```

```
c = a.union(b) # c = a | b (Union is also denoted by |)
c
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
d = a.intersection(b) # c = a % b (Intersection is also denoted by &)
d

{4, 5}

e = a - b #Difference is denoted by -
e #eg is a - b then it will subtract the common values of b with a and gives output

{1, 2, 3}
```

Learning Progress:

- Set Operations: I have successfully grasped the concept of set operations in Python, including union, intersection, and difference. I understand how to utilize these operations to combine, compare, and manipulate sets effectively.
- Unions: I have mastered the use of the | operator to find the union of two sets, resulting in a new set containing all unique elements from both original sets.
- Intersections: I have explored the application of the & operator to find the intersection of two sets, resulting in a new set containing only the elements that exist in both original sets.
- Differences: I have learned to leverage the - operator to find the difference of two sets, resulting in a new set containing the elements present in the first set but not in the second.

Next Steps:

Program Development: I am ready to build a program where I am given a list of colors. My objective is to create a new list containing the rainbow colors from the original list whether the element are in lower or upper case. I will utilize set operations to efficiently achieve this.

Program 3

```
a = 
['yellow','blue','Green','Green','violet','indigo','blue','yellow','saffron','pitch','pink']
rainbow_color = 
['yellow','blue','Green','Red','violet','indigo','Orange']
rainbow_lower = []
result = []
for i in rainbow_color:
    i = i.lower()
    rainbow_lower.append(i)

for i in a:
    if i.lower() in rainbow_lower:
        result.append(i)
```

```
result = set(result)

print(result)

{'indigo', 'blue', 'yellow', 'violet', 'Green'}
```