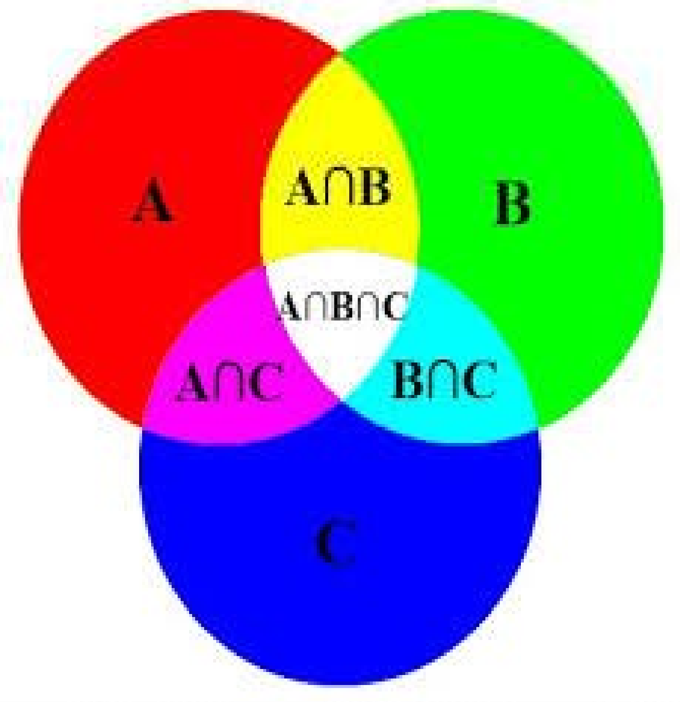
Python Collections (containers or Arrays)

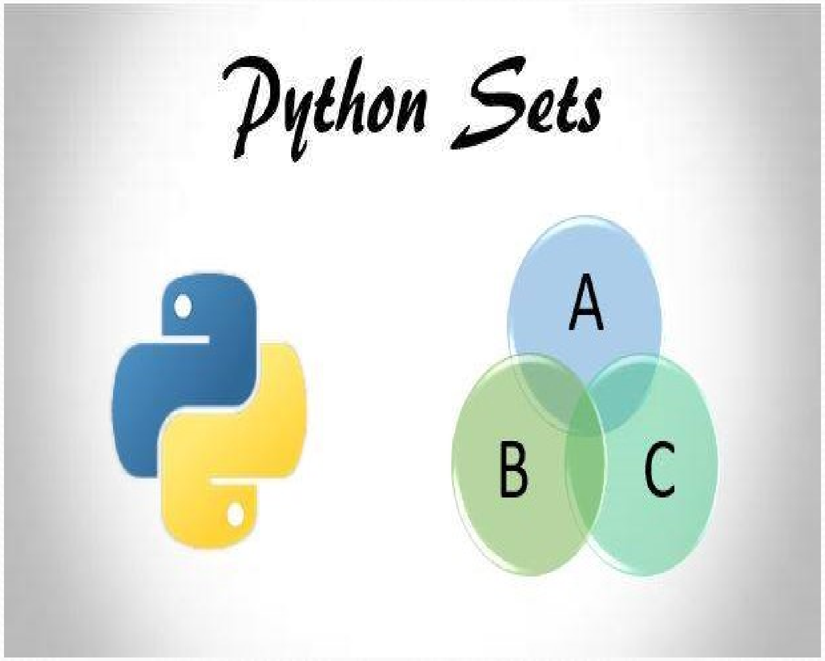
There are four collection data types in the Python programming language, They are

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

Sets in Python





Sets in Python

- It is a data structure
- Has number of elements
- **Homogeneous or heterogeneous**
- **Elements are unique (duplicates are not allowed)**
- To make a set use { }

```
s = { 10, 20,30, 40, 50 }
print(s)
```

- **There is no particular order**

List vs Set

list : a

ith element : a[i]

next element : a[i + 1]

previous element : a[i - 1]

is a sequence

concept of position for each element

```
# set is an iterable
s = { 10, 20,3}
for i in s :
        print(i, end =  " ")
print()
```

set :

no sequence and hence no indexing

no concept of an element in a particular position

represents a finite set of math

Creating set

```
s = set()  #empty set
s={10,20,30}
s={10,} #set with one element
a=[50,60,70]
s=set(a)  # set with three elements
st="Bangalore"
s=set(st) # set with 8 elements
s = set("mississippi")
print(s) # {'s', 'i', 'm', 'p'}
```

Accessing/printing set elements

```
s={10,20,30}
```

```
print(s)
```

```
for  i in s:
        print(i)
```

Error

```
for  i in range(0,len(s))
          print(sub[i])
```

Error

```
i=0
while(i<len(s):
       print(s[i])
        i=i+1
```

Error

```
print(s[0:len(s):1])
```

No concept of indexing/slicing

TypeError: 'set' object does not support indexing

Functions on Sets

```
s={10,20,30}

print(len(s))

print(max(s))

print(min(s))

print(sum(s))

print(sorted(s))

print(type(s))
```

Built in functions, used directly without object

Set methods

```
s={10,20,30}
s.add(40) #adds element to the set
s.discard(10) #removes the element
s.remove(10) #error
s.discard(10) # no error nothing to discard
s.remove(20) #{40,30}
s.pop() # pops an arbitrary item
s.clear() #clears all elements, set becomes empty

s={1,2,3}
s.update({4,5},{6,7,8})
print(s) #{1, 2, 3, 4, 5, 6, 7, 8}
```

Set methods

```
s1 = {1, 2, 3, 4, 5}
s2 = {1, 3, 5, 7, 9}

print(s1.union(s2)) # {1, 2, 3, 4, 5, 7, 9}

print(s1.intersection(s2)) # {1, 3, 5}

print(s1.difference(s2)) # {2, 4}

print(s1.symmetric_difference(s2)) # {2, 4, 7, 9}

print(s1.issubset(s2)) #False

print(s1.issuperset(s2)) #False
```

Operations on sets in Python

Membership

```
a = { 10, 30, 10, 40, 20, 50, 30 }
# check for membership
print(100 in a) # False
print(20 in a) # True
```

No Concatenation

```
Print(s1+s2)
```

Relational operators (<,>…)

Identity

```
a={10,20,30}
b=a
print(a is b) # true
```

set operations (using operators)

```
s1 = {1, 2, 3, 4, 5}
s2 = {1, 3, 5, 7, 9}
# union
print(s1 | s2) # {1, 2, 3, 4, 5, 7, 9}
# intersection
print(s1 & s2) # {1, 3, 5}
# set difference
print(s1 - s2) # {2, 4}
# symmetric difference
print(s1 ^ s2) # {2, 4, 7, 9}
```

set operations(using set methods)

```
s1 = {1, 2, 3, 4, 5}
s2 = {1, 3, 5, 7, 9}

print(s1.union(s2))

print(s1.intersection(s2))

print(s1.difference(s2))

print(s1.symmetric_difference(s2))

print(s1.issubset(s2))

print(s1.issuperset(s2))
```

```
{1, 2, 3, 4, 5, 7, 9}
{1, 3, 5}
{2, 4}
{2, 4, 7, 9}
False
False
```

Removing duplicate elements in a list

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

```
a = [11, 33, 11, 33, 11, 44, 22, 55, 55, 11]
print(list(set(a)))
```

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

final_list=[]

for item in a:

    if item not in final_list:

        final_list.append(item)

print(final_list)
```

To check given sentence or string is pangram or not

```
import string
pgm=True
str="the quick brown fox jumps over the lazy dog"
chars=[]
for i in range(97,122):
     chars.append(chr(i))
chars = list(string.ascii_lowercase)
print(chars)
for i in chars:
   if(i not in str):
      pgm=False
      break
if(pgm):
   print ("yes")
else:
   print("no")
```

```
import string
pgm=True
str="the quick brown fox jumps over the lazy
dog"
chars = list(string.ascii_lowercase)
print(chars)
for i in chars:
   if(i not in str):
      pgm=False
      break
if(pgm):
   print ("yes")
else:
   print("no“)
```

To check whether a string is pangram or not

```
import string
a="the quick brown fox jumps over the lazy dog“
b = list(string.ascii_lowercase)
if not set(b)-set(a):
    print("pangram")
else:
    print("not pangram")
```

To find Cartesian product

```
a=(1,2)
a=a+((3,4),(5,6))
print(a)

a = {1,2}
b = {3,4}
res = ()
for i in a:
    for j in b:
        res = res+((i, j),(i,j))
    print(res)
print(set(res))
```

Program to generate a set with n elements and remove multiples of 2
OR
Program remove multiples of 2 from a set

```
Enter an integer : 20

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
     13, 14, 15, 16, 17, 18, 19, 20  }

{1, 3, 5, 7, 9, 11, 13, 15, 17, 19}
```

Program to remove multiples of 2 from a set

```
n = int(input("Enter an integer : "))
s = set(range(2, n + 1))
print(s)
s = s - set(range(2, n + 1,2))
print (s)
```

To remove multiples of 3,5 ,7,

To print all prime numbers upto n

1) **Division method**
2) **Sieve of Eratosthenes**

```
#Division method
n=397
prime=True
for i in range(2,n//2):
        if(n%i==0):
                prime=False
                break;
print(i)
if(prime==True):
        print("prime")
else:
        print("not prime")
```

Prime numbers up to 10

```
Enter an integer : 10
{2, 3, 4, 5, 6, 7, 8, 9, 10}
2 {9, 3, 5, 7}
3 {5, 7}
5 {7}
7
```

```
#generate prime numbers (no division; efficient algorithm)
# use sieve of Eratosthenes
1) get a number(say n)
2) make a set of numbers from 2 to n - say sieve
3) while sieve is not empty
        find the smallest (small)
        print it (that is a prime)
        remove small and its multiples from the sieve
```

Prime numbers upto n

```
n = int(input("Enter an integer : "))
# make a set of numbers from 2 to n - say sieve
sieve = set(range(2, n + 1))
while sieve :
    small = min(sieve)
    #print(sieve)
    print(small, end = " ")
    sieve = sieve - set(range(small, n + 1,small))
```

Python Frozen set: immutable

```
vowels = {'a', 'e', 'i', 'o', 'u'} #normal set
vowels.add('z') #mutable
print(vowels)

fset = frozenset(vowels)
print('The frozen set is:', fset) #fset is frozen set
#immutable
fset.add('z')  #AttributeError: 'frozenset' object has no attribute 'add'
print(vowels)
```