Data Structures

There are four collection data types in the Python programming language

- Lists
- Tuples
- Sets
- Dictionaries

Python Lists

A list is a collection which is ordered and changeable. In Python lists are written with square brackets.

In [1]:
```
names = ['Kunal', 'Arjun', 'Priya','Kishor','Anu']
print(names)
print(type(names))
```

```
['Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu']
<class 'list'>
```

In [5]:
```
x = [1,5,8,'Kunal']
type(x)
```

Out[5]: list

In [2]:
```
type('Kunal')
```

Out[2]: str

In [6]:
```
# Access items
print(names[0])
print(names[2:])
print(names[:2])
print(names[1:3])
```

```
Kunal
['Priya', 'Kishor', 'Anu']
['Kunal', 'Arjun']
['Arjun', 'Priya']
```

In [11]:
```
type(names[0:3])
```

Out[11]: list

In [9]:
```
names
```

```
['Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu']
```

Out[9]:

In [12]:
```
# Negative index
print(names[-1])
print(names[-2:])
print(names[:-2])
print(names[-3:-1])
```

```
Anu
['Kishor', 'Anu']
['Kunal', 'Arjun', 'Priya']
['Priya', 'Kishor']
```

In [19]:
```
names[-3:3]
```

Out[19]:
```
['Priya']
```

In [20]:
```
# Change item in list
print(names)
names[3]="Vijay"
print(names)
```

```
['Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu']
['Kunal', 'Arjun', 'Priya', 'Vijay', 'Anu']
```

In [21]:
```
# Looping through lists
for j in names:
    print("Iterator now has value {}".format(j))
```

```
Iterator now has value Kunal
Iterator now has value Arjun
Iterator now has value Priya
Iterator now has value Vijay
Iterator now has value Anu
```

In [22]:
```
# Nested list
mylist=[1,2,3,['a','b',[5,6]]]
print(mylist)
```

```
[1, 2, 3, ['a', 'b', [5, 6]]]
```

In [30]:
```
type(mylist)
```

Out[30]:
```
int
```

In [28]:
```
mylist[3][2][1]
```

Out[28]:
```
6
```

Methods

- 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
- del : Delete list
- 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
- len(): Length of list
- list(): Copies list from given list
- pop(): Removes the element at the specified position
- remove(): Removes the item with the specified value
- reverse(): Reverses the order of the list
- sort(): Sorts the list

In [31]:
```
# append(): Adds an element at the end of the list
print(names)
names.append("Kumar")
print(names)
```

```
['Kunal', 'Arjun', 'Priya', 'Vijay', 'Anu']
['Kunal', 'Arjun', 'Priya', 'Vijay', 'Anu', 'Kumar']
```

In [32]:
```
# clear(): Removes all the elements from the list
names.clear()
print(names)
```

```
[]
```

In [33]:
```
# copy(): Returns a copy of the list
names = ['Kunal', 'Arjun', 'Priya','Kishor','Anu']
Names=names.copy()
print(Names)
```

```
['Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu']
```

In [37]:
```
# count(): Returns the number of elements with the specified value
names.append("Kunal")
x=names.count('Kunal')
print(x)
```

```
4
```

In [38]:
```
names
```

Out[38]:
```
['Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu', 'Kunal', 'Kunal', 'Kunal']
```

In [39]:
```
# del : Delete list
del Names
print(Names)
```

```
NameError                                 Traceback (most recent call last)
<ipython-input-39-e4f86090d768> in <module>
      1 # del : Delete list
      2 del Names
----> 3 print(Names)

NameError: name 'Names' is not defined
```

In [40]:
```
# extend(): Add the elements of a list (or any iterable),
# to the end of the current list
print(names)
cars = ['Ford', 'BMW', 'Volvo']
names.extend(cars)
print(names)
```

```
['Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu', 'Kunal', 'Kunal', 'Kunal']
['Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu', 'Kunal', 'Kunal', 'Kunal', 'Ford', 'BMW',
'Volvo']
```

In [42]:
```
# index(): Returns the index of the first element with the specified value
x=names.index('Kunal')
x
```

Out[42]:
```
0
```

In [43]:
```
# insert():     Adds an element at the specified position
names.insert(2,'Audi')
print(names)
```

```
['Kunal', 'Arjun', 'Audi', 'Priya', 'Kishor', 'Anu', 'Kunal', 'Kunal', 'Kunal', 'Ford',
'BMW', 'Volvo']
```

In [44]:
```
# len(): Length of list
print(len(names))
```

```
12
```

In [47]:
```
df = list(('Ford',)) # df = ['ford','audi']
df
```

Out[47]:
```
['Ford']
```

In [48]:
```
# list(): Creats list from given inputs
new_list=list(('Audi',"BMW",'Ford'))
print(new_list)
```

```
['Audi', 'BMW', 'Ford']
```

In [50]:
```
print(names)
```

```
['Kunal', 'Arjun', 'Audi', 'Priya', 'Kishor', 'Anu', 'Kunal', 'Kunal', 'Kunal', 'Ford',
'BMW', 'Volvo']
```

In [54]:
```
# pop(): Removes the element at the specified position
names.pop(5)
```

```
#print(names.pop(5))
print(names)
#names.pop(2)
#print(names)
```

```
BMW
['Kunal', 'Arjun', 'Kishor', 'Anu', 'Kunal', 'Volvo']
```

In [57]:
```
names.pop()
print(names)
```

```
['Kunal', 'Arjun', 'Kishor', 'Anu']
```

In [58]:
```
# remove(): Removes the item with the specified value
# names.remove('Raj','Ford'); remove function takes only one input
names.append('Delhi')
print(names)
```

```
['Kunal', 'Arjun', 'Kishor', 'Anu', 'Delhi']
```

In [59]:
```
names.remove('Delhi')
print(names)
```

```
['Kunal', 'Arjun', 'Kishor', 'Anu']
```

In [60]:
```
to_remove = ['Kunal', 'Anu']
for i in to_remove:
    names.remove(i)
names
```

Out[60]:
```
['Arjun', 'Kishor']
```

In [62]:
```
#names.extend(['Kunal']*5)
```

In [64]:
```
names
```

Out[64]:
```
['Arjun', 'Kishor', 'Kunal', 'Kunal', 'Kunal', 'Kunal', 'Kunal']
```

In [65]:
```
# reverse(): Reverses the order of the list
names.reverse()
print(names)
```

```
['Kunal', 'Kunal', 'Kunal', 'Kunal', 'Kunal', 'Kishor', 'Arjun']
```

In [70]:
```
names
```

Out[70]:
```
['Arjun', 'Kishor', 'Kunal', 'Kunal', 'Kunal', 'Kunal', 'Kunal']
```

In [69]:
```
# sort(): Sorts the list
names.sort()
print(names)
```

```
['Arjun', 'Kishor', 'Kunal', 'Kunal', 'Kunal', 'Kunal', 'Kunal']
```

In [71]:
```
names.sort(reverse = False)
names
```

Out[71]:
```
['Arjun', 'Kishor', 'Kunal', 'Kunal', 'Kunal', 'Kunal', 'Kunal']
```

Tuples

A tuple is a collection which is ordered and unchangeable. In Python tuples are written with round brackets.

In [72]:
```
t1=(1,5,8,3,10,3)
t2=('Kunal', 'Arjun', 'Priya','Kishor','Anu')
print(t1,"\n")
print(t2)
```

```
(1, 5, 8, 3, 10, 3)

('Kunal', 'Arjun', 'Priya', 'Kishor', 'Anu')
```

In [73]:
```
# Access elements
print(t1[0])
print(t1[2:])
print(t1[:3])
```

```
1
(8, 3, 10, 3)
(1, 5, 8)
```

In [75]:
```
type(t1[0:2])
```

Out[75]:
```
tuple
```

In [76]:
```
# Negative index
print(t1[-1])
print(t1[-4:])
print(t1[:-3])
```

```
3
(8, 3, 10, 3)
(1, 5, 8)
```

In [78]:
```
# Change tuple values
print(t1)
t1[0]=99
```

```
(1, 5, 8, 3, 10, 3)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-78-389c66f92d1c> in <module>
      1 # Change tuple values
      2 print(t1)
----> 3 t1[0]=99
```

```
TypeError: 'tuple' object does not support item assignment
```

Tuples are not interchangable. So no additional element can be added or removed. Try...

In [79]:
```
# Looping through tuple
for j in t1:
    print('The iterator now has value = {}'.format(j))
```

```
The iterator now has value = 1
The iterator now has value = 5
The iterator now has value = 8
The iterator now has value = 3
The iterator now has value = 10
The iterator now has value = 3
```

In [81]:
```
print('Kunal' in t2)
```

```
True
```

In [83]:
```
# Check if 'Kunal' in t2
if "Anu" in t2:
    print("Kunal is listed in t2")
```

```
Kunal is listed in t2
```

In [84]:
```
# tuple with one item
T1=("Hi",)
print(type(T1))
T2=("Hi")
print(type(T2))
```

```
<class 'tuple'>
<class 'str'>
```

Methods

- tuple(): Creates tuple with elements specified.
- len(): Gives length of tuple, no. of elements in tuple.
- t1 + t2 : Adds two tuples
- del(): Deletes tuple
- count(): Returns the number of times a specified value occurs in a tuple.
- index(): Searches the tuple for a specified value and returns the position of where it was found.

In [85]:
```
# tuple(): Creates tuple with elements specified.
t=tuple((0,2,4,6,0,2,2,5,2,0,2,3,4))
print(t)
print(type(t))
```

```
(0, 2, 4, 6, 0, 2, 2, 5, 2, 0, 2, 3, 4)
<class 'tuple'>
```

In [86]:
```
# len(): Gives length of tuple, no. of elements in tuple.
len(t)
```

Out[86]:
```
13
```

In [87]:
```
# t1 + t2 : Adds two tuples
t + t1
```

Out[87]:
```
(0, 2, 4, 6, 0, 2, 2, 5, 2, 0, 2, 3, 4, 1, 5, 8, 3, 10, 3)
```

In [88]:
```
# del(): Deletes tuple
print(T1)
del T1
print(T1)
```

```
('Hi',)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-88-6df59feedd1b> in <module>
      2 print(T1)
      3 del T1
----> 4 print(T1)

NameError: name 'T1' is not defined
```

In [93]:
```
# count(): Returns the number of times a specified value occurs in a tuple.
t.count(2)
```

Out[93]:
```
5
```

In [95]:
```
# index(): Searches the tuple for a specified value and returns the position of where i
t.index(4)
```

Out[95]:
```
2
```

In [96]:
```
t = (1,2,5,'a','b', (1,'c'))
print(t)
```

```
(1, 2, 5, 'a', 'b', (1, 'c'))
```

In [98]:
```
print(t[5])
print(type(t[5]))
```

```
(1, 'c')
<class 'tuple'>
```

Sets

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

In [103…
```
set1={1,3,8,9,4,3,5,2,1}
print(set1,"\n")
set2={'A','C','D','A','X'}
print(set2)
```

```
{1, 2, 3, 4, 5, 8, 9}

{'C', 'X', 'A', 'D'}
```

In [105…
```
set2={'A','D','C','A','X'}
print(set2)
```

```
{'C', 'X', 'A', 'D'}
```

In [107…
```
# Access items
# You cannot access items in a set by referring to an index,
# since sets are unordered the items has no index.
set1[3]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-107-296fa1aa445c> in <module>
      2 # You cannot access items in a set by referring to an index,
      3 # since sets are unordered the items has no index.
----> 4 set1[3]

TypeError: 'set' object is not subscriptable
```

In [108…
```
# Access items
# loop through the set items using a for loop
for j in set1:
    print(j)
```

```
1
2
3
4
5
8
9
```

In [111…
```
print('D' in set2)
```

```
True
```

Methods

- add(): Adds an element to the set
- clear(): Removes all the elements from the set
- copy(): Returns a copy of the set
- difference(): Returns a set containing the difference between two or more sets
- difference_update(): Removes the items in this set that are also included in another, specified set
- discard(): Remove the specified item
- intersection(): Returns a set, that is the intersection of two other sets
- intersection_update(): Removes the items in this set that are not present in other, specified set(s)
- isdisjoint(): Returns whether two sets have a intersection or not

- issubset(): Returns whether another set contains this set or not
- issuperset(): Returns whether this set contains another set or not
- len(): Finds length of set
- pop(): Removes an element from the set
- remove(): Removes the specified element
- symmetric_difference(): Returns a set with the symmetric differences of two sets
- symmetric_difference_update(): inserts the symmetric differences from this set and another
- union(): Return a set containing the union of sets
- update(): Update the set with the union of this set and others

In [112…
```
# add(): Adds an element to the set
print(set1)
set1.add(12)
print(set1)
```

```
{1, 2, 3, 4, 5, 8, 9}
{1, 2, 3, 4, 5, 8, 9, 12}
```

In [113…
```
# clear(): Removes all the elements from the set
set1.clear()
print(set1)
```

```
set()
```

In [114…
```
# copy(): Returns a copy of the set
set1=set2.copy()
print(set1)
```

```
{'C', 'X', 'A', 'D'}
```

In [116…
```
# difference(): Returns a set containing the difference between two or more sets
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

z = x.difference(y)
print(z)
```

```
{'cherry', 'banana'}
```

In [118…
```
# difference_update(): Removes the items in this set that are also included in another,
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

x.difference_update(y)
print(x)
```

```
{'cherry', 'banana'}
```

In [119…
```
# discard(): Remove the specified item
print(set1)
set1.discard('X')
print(set1)
```

```
{'C', 'X', 'A', 'D'}
```

```
{'C', 'A', 'D'}
```

In [121…
```
# intersection(): Returns a set, that is the intersection of two other sets
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

z = x.intersection(y)
print(z)
```

```
{'apple'}
```

In [123…
```
# intersection_update(): Removes the items in this set that are not present in other, s
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

x.intersection_update(y)
print(x)
print(y)
```

```
{'apple'}
{'samsung', 'mi', 'apple'}
```

In [126…
```
# isdisjoint(): Returns whether two sets have a intersection or not
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

z = x.isdisjoint(y)
print(z)
```

```
False
```

In [131…
```
# issubset(): Returns whether another set contains this set or not
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}#,'banana','cherry'}

z = x.issubset(y)
print(z)
```

```
False
```

In [134…
```
# issuperset(): Returns whether this set contains another set or not
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple",'banana','cherry'}

z = y.issuperset(x)
print(z)
```

```
True
```

In [135…
```
# len(): Finds length of set
len(y)
```

Out[135… 5

In [144…
```
# pop(): Removes an element from the set and returns the removed value.
x = {"apple", "banana", "cherry"}
```

```
print(x)
print(x.pop())
print(x)
```

```
{'cherry', 'apple', 'banana'}
cherry
{'apple', 'banana'}
```

In [141…

```
#x = {"apple", "banana", "cherry"}
#print(x)
```

```
{'cherry', 'apple', 'banana'}
```

In [142…

```
# remove(): Removes the specified element
y={'samsung', 'apple', 'mi'}
print(y)
y.remove('mi')
#y.remove('MI')
#y.discard('MI')
print(y)
```

```
{'samsung', 'mi', 'apple'}
{'samsung', 'apple'}
```

In [145…

```
# symmetric_difference(): Returns a set with the symmetric differences of two sets
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

z = x.symmetric_difference(y)
print(z)
```

```
{'samsung', 'cherry', 'mi', 'banana'}
```

In [146…

```
# symmetric_difference_update(): inserts the symmetric differences from this set and an
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

y.symmetric_difference_update(x)
print(y)
```

```
{'samsung', 'cherry', 'mi', 'banana'}
```

In [147…

```
# union(): Return a set containing the union of sets
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

z=x.union(y)
print(z)
```

```
{'cherry', 'samsung', 'mi', 'apple', 'banana'}
```

In [148…

```
# update(): Update the set with the union of this set and others
x = {"apple", "banana", "cherry"}
y = {"samsung", "mi", "apple"}

x.update(y)
```

```
print(x)
print(y)
```

```
{'cherry', 'samsung', 'mi', 'apple', 'banana'}
{'samsung', 'mi', 'apple'}
```

Dictionary

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

In [ ]:
```
d1={"Name": "Ankit","Class": '12th','Roll No.': 25, 'Result': "Pass"}
print(d1)
```

In [ ]:
```
# Accessing Items
# You can access the items of a dictionary by referring to its key name, inside square
d1['Name']
```

In [ ]:
```
# get() function
d1.get('Name')
```

In [ ]:
```
# Change values
d1['Name']='Aniket'
print(d1)
```

In [ ]:
```
# Looping through dictionary
for j in d1:
    print(j)
```

In [ ]:
```
for j in d1:
    print(d1[j])
```

In [ ]:
```
# Extract keys
d1.keys()
print(d1.keys())
```

In [ ]:
```
# Extract values
d1.values()
```

In [ ]:
```
# Extract keys and values both
d1.items()
```

In [ ]:
```
for k,v in d1.items():
    print(k,v)
```

In [ ]:
```
# Check existance of key
print('Result' in d1,'\n')

if 'Name' in d1:
    print('Name exists in disctionary')
```

In [ ]:
```
# Nested dictionary
myfamily = {
  "child1" : {
    "name" : "kruti",
    "year" : 2004
  },
  "child2" : {
    "name" : "Akruti",
    "year" : 2007
  },
  "child3" : {
    "name" : "Sanskruti",
    "year" : 2011
  }
}
print(myfamily)
```

In [ ]:
```
child1 = { "name" : "kruti","year" : 2004}
child2 = {"name" : "Akruti","year" : 2007}
child3 = {"name" : "Sanskruti","year" : 2011}

myfamily = { "child1" : child1, "child2" : child2, "child3" : child3}

print(myfamily)
```

In [ ]:
```
y = myfamily['child1'].keys()
print(y)
```

Methods

- clear(): Removes all the elements from the dictionary
- copy(): Returns a copy of the dictionary
- fromkeys(): Returns a dictionary with the specified keys and values
- pop(): Removes the element with the specified key
- popitem(): Removes the last inserted key-value pair
- setdefault(): Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
- update(): Updates the dictionary with the specified key-value pairs

In [ ]:
```
# clear(): Removes all the elements from the dictionary
print(d1)
d1.clear()
print(d1)
```

In [ ]:
```
# copy(): Returns a copy of the dictionary d2 = d1
d1={'Name': 'Ankit', 'Class': '12th', 'Roll No.': 25, 'Result': 'Pass'}
d3 = d1
print(d3)
D =d1.copy()
print(D)
```

In [ ]:
```
# fromkeys(): Returns a dictionary with the specified keys and values
x = ('key1', 'key2', 'key3')
y = 'A'
z=dict.fromkeys(x,y)
print(z)
```

In [ ]:
```
# pop(): Removes the element with the specified key
print(d1)
d1.pop('Name')
print(d1)
```

In [ ]:
```
# popitem(): Removes the last inserted key-value pair
print(d1)
d1.popitem()
print(d1)
```

In [ ]:
```
# setdefault(): Returns the value of the specified key.
# If the key does not exist: insert the key, with the specified value
d1={'Name': 'Ankit', 'Class': '12th', 'Roll No.': 25, 'Result': 'Pass'}
print(d1)
d2=d1.setdefault("Class")
print(d2)
d3=d1.setdefault("Location",'Pune')
print(d3)
```

In [ ]:
```
d1
```

In [ ]:
```
# update(): Updates the dictionary with the specified key-value pairs
print(d1)
d1.update({"Division":"A"})
print(d1)
```

In [ ]:
```
d1.update({"Gender":'M',"Age":19})
print(d1)
```

In [ ]:
```
d3 = d1.copy()
print(d3)
```

In [ ]:
```
d3.update({'XYZ':446})
print(d3)
print(d1)
```

In [ ]:
```
d4 = d1
print(d4)
```

In [ ]:
```
d4.update({'xyz':454})
print(d4)
print(d1)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Basics of Python Programming

1. Below is a part of a DNA sequence. Copy that to python in a variable named as 'DNA'.

   a. What is the length of the sequence?

   b. Write a code that replaces specified name like A or C from sequence by some number.

   c. Split the DNA sequence so we can have each character as new entry.

   'CCAGCAGCTCCTTGCCGAGATGGGATTCGGTTATCTTGCCTTTGAAAAAATCCAGGTAACCTTCCGCATCATC

In [ ]:
```
y = 'CCAGCAGCTCCTTGCCGAGATGGGATTCGGTTATCTTGCCTTTGAAAAAATCCAGGTAACCTTCCGCATCATCGCTCGCCCG
```

In [ ]:
```
print(len(y))
```

In [ ]:
```
print(y.replace('A','1'))
```

In [ ]:
```
y.split()
```

In [ ]:
```
Y = list(y)
len(Y)
```

In [ ]:
```
type(Y)
```

In [ ]:
```

```