Introduction to Python

Data types

Apart from numeric, string and boolean data types, Python supports storing of collection of values. This built-in data structures can hold any **collection** of objects. These are:

1. List – []
2. Tuple – ()
3. Set- {}
4. Dictionary – {key:value,…}

Set

Set is a well defined collection of distinct objects/elements/members. Set are unordered, unique mutable elements. It can be heterogeneous that means List can have elements of different data types (integer, string, etc). The elements can added to a list.

Introduction to Python

The **set** is initialized by:

a = set() -> it should be iterable such as list, tuple

a = {1,2,3,4}

a= {'idc101','idc','idc'}

```
varS=set()
print(type(varS))

varS={1,2,2,2,3}
print(varS)

varS={10,1,2,'idc','chm',2,11,1}
print(varS)
```

```
<class 'set'>
{1, 2, 3}
{1, 2, 10, 11, 'idc', 'chm'}
```

```
varS=set([10,(1,2,10,10),1,2])
print(varS)
```

```
{1, 10, 2, (1, 2, 10, 10)}
```

The elements of a set cannot be a mutable data type such as list/dict. Tuple can be element. But, a set cannot within another set.

Introduction to Python

The elements of a set can be string.

```
varS=set('idc101')
print(varS)

varP={'idc101'}
print(varP)

varQ={'idc101','idc201','idc101'}
print(varQ)
```

```
{'i', '1', 'd', 'c', '0'}
{'idc101'}
{'idc201', 'idc101'}
```

A single string is iterable

Introduction to Python

Set operations

- len() – size of set
- in - membership
- x1.union(x2) – union of x1 and x2 set
- X1 | X2 – union of X1 and X2 or it can be among many sets
- X1 | X2 | X3 – union of X1, X2 and X3

- x1.intersection(x2) – intersection of x1 and x2 set
- X1 & X2 – intersection of X1 and X2 or it can be among many sets
- X1 & X2 & X3 – intersection of X1, X2 and X3

```
x1={1,2,3}
x2={2,10,3}
x3={1,2,3,4,5}
print(x1|x2)
print(x1|x2|x3)
```

```
{1, 2, 3, 10}
{1, 2, 3, 4, 5, 10}
```

```
print(x1&x2)
print(x1&x2&x3)
```

```
{2, 3}
{2, 3}
```

Introduction to Python

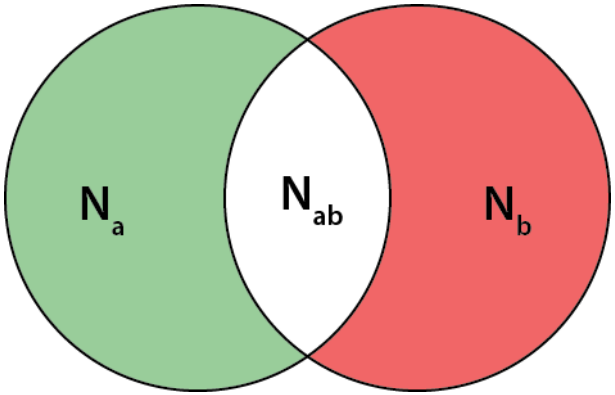
Set operations

- x1.difference(x2) − return the set of all elements that are in x1 but not in x2
- x1-x2 -return the set of all elements that are in x1 but not in x2

- x2.difference(x1) − return the set of all elements that are in x2 but not in x1
- x2-x1 -return the set of all elements that are in x2 but not in x1

- x1.symmetric_difference (x2) − return the set of all elements in either x1 or x2, but not both
- x1^x2 − same as above.

```
x1={1,2,3}
x2={2,10,3}
print(x1-x2)
print(x1^x2)

{1}
{1, 10}
```



N_a, N_b and N_{ab} are **set** of elements in the colored section.

x1 – N_a+N_{ab}
x2 – N_b+N_{ab}
(x1 – x2) –N_a
x1 | x2 - N_a+N_{ab} +N_b
x1 & x2 – N_{ab}
x1 ^x2 - N_a+N_b

Introduction to Python

Set operations

- x1.add(k) – adds only ONE element k to x1
- x1.update(x2) – adds another set x2 to x1
- x1.remove(k) –removes only ONE element k from x1

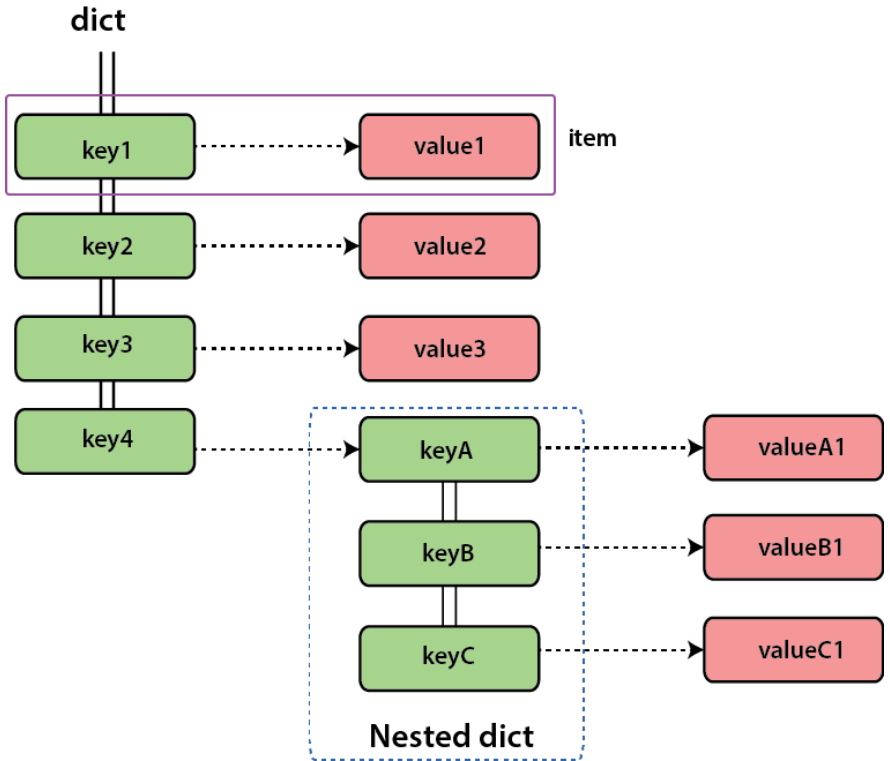
```
x1={1,2,3,4}
x2={10,2,3}
x1.add(121)
print(x1)

x1.update(x2)
print(x1)

x1.remove(2)
print(x1)
```

```
{1, 2, 3, 4, 121}
{1, 2, 3, 4, 10, 121}
{1, 3, 4, 10, 121}
```

Dictionaries (dict)

It is unordered set of key/value pairs. This has has **unique key** and it has **value** associated with it. The **value** could be any object. The dictionaries are mutable and is dynamic (grow or shrink on demand). This is also called as hash/associative arrays/ map. A **key** cannot be a mutable object.

It is map between set of keys to set of values. The key-value pair is called as an **item**.



Introduction to Python

The **dict** is initialized by:

a = dict() -> it should be list with tuple of two elements or dict(a='b',c='d');

dict([('key1','value'), ('key2','value')])

a = {} -> empty dict()

a= {key:value,key1:value1}

Key should be unique non-mutable. Value can be anything or any other object

```
## various ways to initialize dict
varD={} # empty dictionary
print (type(varD))

## define data as {<key1>:<value>,<key2>:<value>}
varD={'name':'michal','section':'va','subject':'maths'}
print(varD)

## Data has following key and value
## dict(key1='value',key2='value')
## dict(name='michal',section='va',subject='maths')
varD=dict(name='michal',section='vb')
print(varD)

## dict([ ('key1','value'), ('key2','value') ])
varD=dict([('name','michal'),('section','vc')])
print(varD)

## dict({'key1:'value','key2':'value'})
varD=dict({'name':'michal','section':'vd','subject':'maths'})
print(varD)
```

```
<class 'dict'>
{'name': 'michal', 'section': 'va', 'subject': 'maths'}
{'name': 'michal', 'section': 'vb'}
{'name': 'michal', 'section': 'vc'}
{'name': 'michal', 'section': 'vd', 'subject': 'maths'}
```

Introduction to Python

Value can be list, tuple or dict()

```
varD={'name':'Tom','childern':('Michal','Jeff','Mu'),'friends':['Adam','Pat']}
print(varD)
```

```
{'name': 'Tom', 'childern': ('Michal', 'Jeff', 'Mu'), 'friends': ['Adam', 'Pat']}
```

```
varD={'name':'Tom','subjects':{ 'biology':50,'chemistry':90 } }
varD
```

```
{'name': 'Tom', 'subjects': {'biology': 50, 'chemistry': 90}}
```

Introduction to Python

The dict value is accessed by:

dictName['key']

On accessing value without a 'key' returns error 'KeyError'

```
## accessing values
varData={'name':'Tom','childern':('Michal','Jeff','Mu'),'friends':['Adam','Pat']}
print(varData['childern'])
print(varData['childern'][1])
```

```
('Michal', 'Jeff', 'Mu')
Jeff
```

```
varD={'name':'Tom','subjects':{ 'biology':50,'chemistry':90 } }
print(varD['subjects']['biology'])
```

```
50
```

dict operations

x.items() – returns tuple of (key,value) in the dictionary
x.keys() – returns keys of dictionary
x.values() – returns values of dictionary
x.update(d) – returns updated dict (**d** could be another dictionary or key,value pairs as used for initializing dict object.
del x['key'] – deletes **'key'** in the dict object **x**

```
## x.items() returns key,value as tuple
print(varData.items())

## x.keys() returns key as list
print(varData.keys())

## x.values() returns value as list
print(varData.values())
```

```
dict_items([('name', 'Tom'), ('childern', ('Michal', 'Jeff', 'Mu')), ('friends', ['Adam', 'Pat'])])
dict_keys(['name', 'childern', 'friends'])
dict_values(['Tom', ('Michal', 'Jeff', 'Mu'), ['Adam', 'Pat']])
```

Accessing key/value in dict

```
data=dict([(1,'a'),(2,'b'),(3,'c'),(4,'d')])
data
for k,v in data.items():
  print(k,v)

for k in data.keys():
  print('The key is: ',k)

for v in data.values():
  print('The value is: ',v)
```

```
1 a
2 b
3 c
4 d
The key is:  1
The key is:  2
The key is:  3
The key is:  4
The value is:  a
The value is:  b
The value is:  c
The value is:  d
```

Name	Batch	Course	Maths	Biology	Chemistry	Physics
Adrian	2020	Sem1	59	93	78	85
Jasuz	2020	Sem1	78	61	65	80
Jeffrey	2020	Sem1	77	58	56	80
Michal	2020	Sem1	51	93	67	56
Piotr	2020	Sem1	91	76	64	70