Module 1 - Python Basics

1. Your first program
2. Types
3. Expressions and Variables
4. String Operations

Module 2 - Python Data Structures

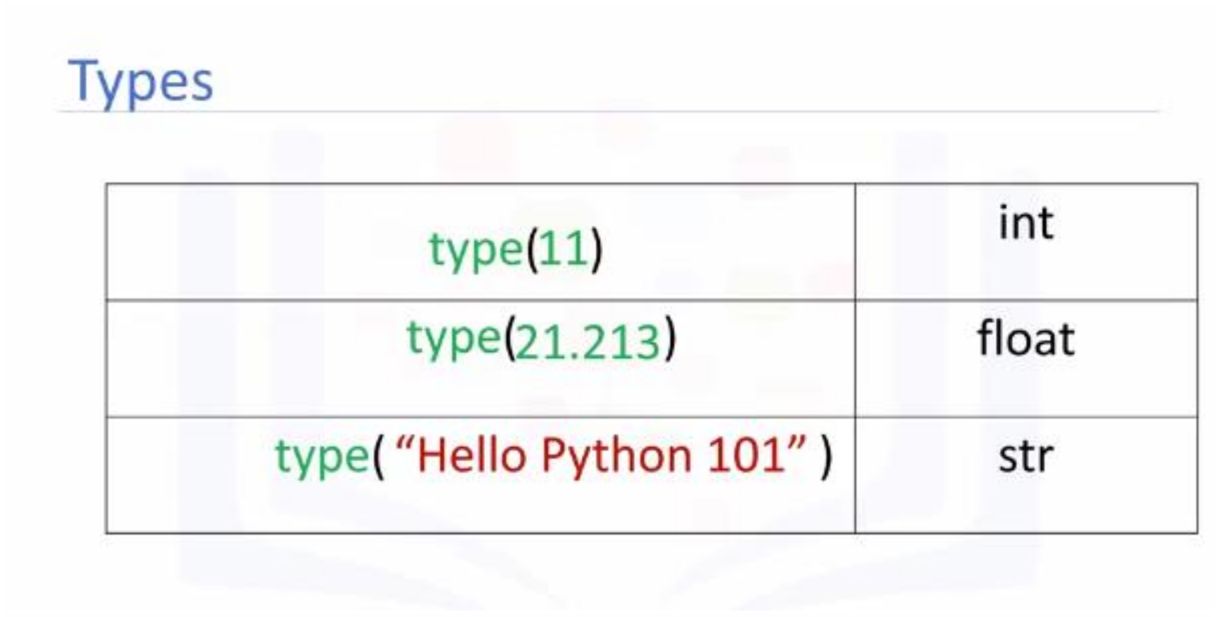
1. Lists and Tuples
2. Sets
3. Dictionaries

Module 3 - Python Programming Fundamentals

1. Conditions and Branching
2. Loops
3. Functions
4. Objects and Classes

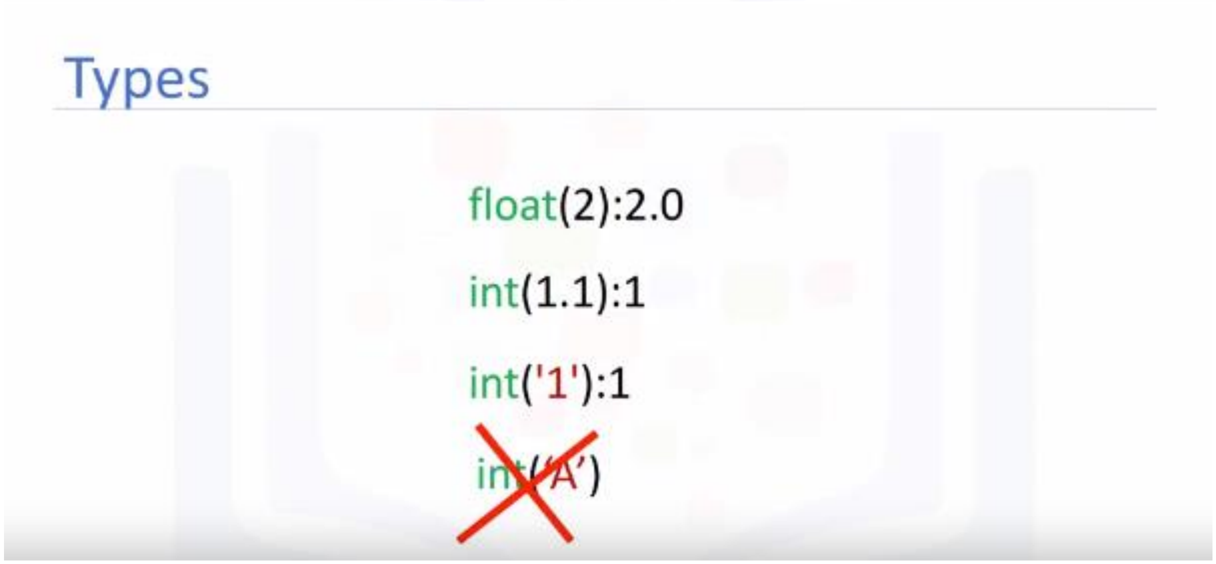
Module 4 - Working with Data in Python

1. Reading files with open
2. Writing files with open
3. Loading data with Pandas
4. Working with and Saving data with Pandas
5. Numpy



Types

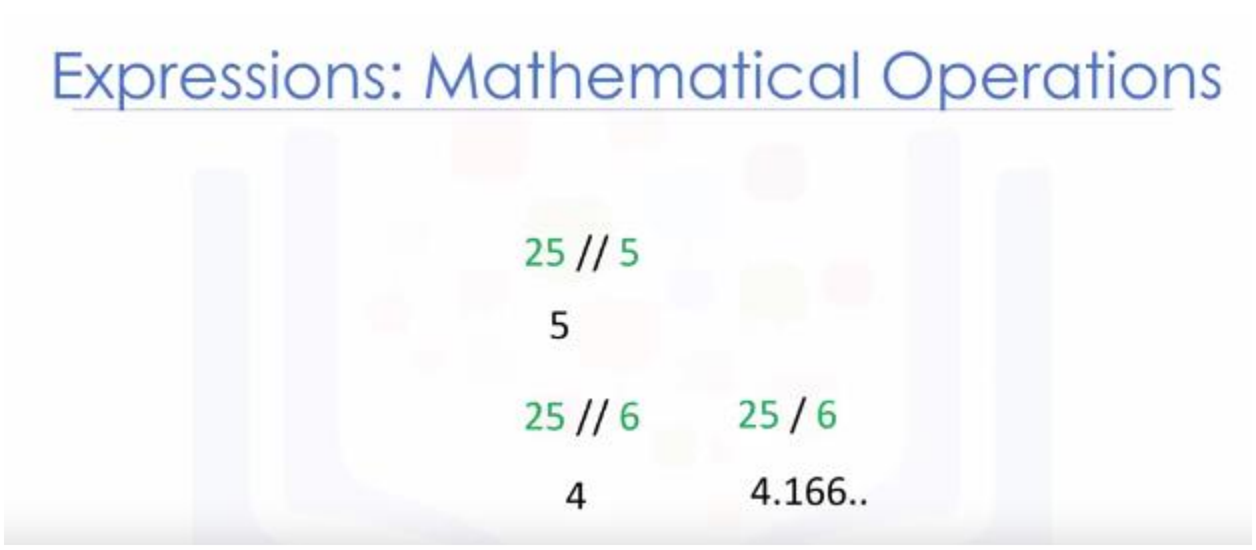
type(11)	int
type(21.213)	float
type(“Hello Python 101”)	str



Types

```
float(2):2.0
int(1.1):1
int('1'):1
```

~~int(‘A’)~~

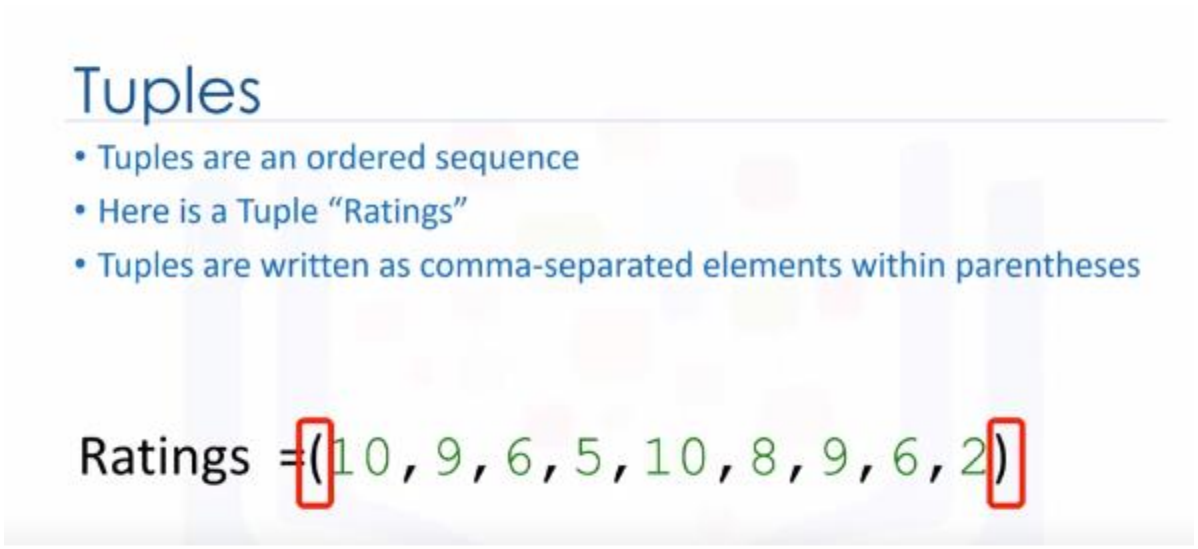


Expressions: Mathematical Operations

```
25 // 5
 5

25 // 6      25 / 6
  4           4.166..
```

Double // for integer division

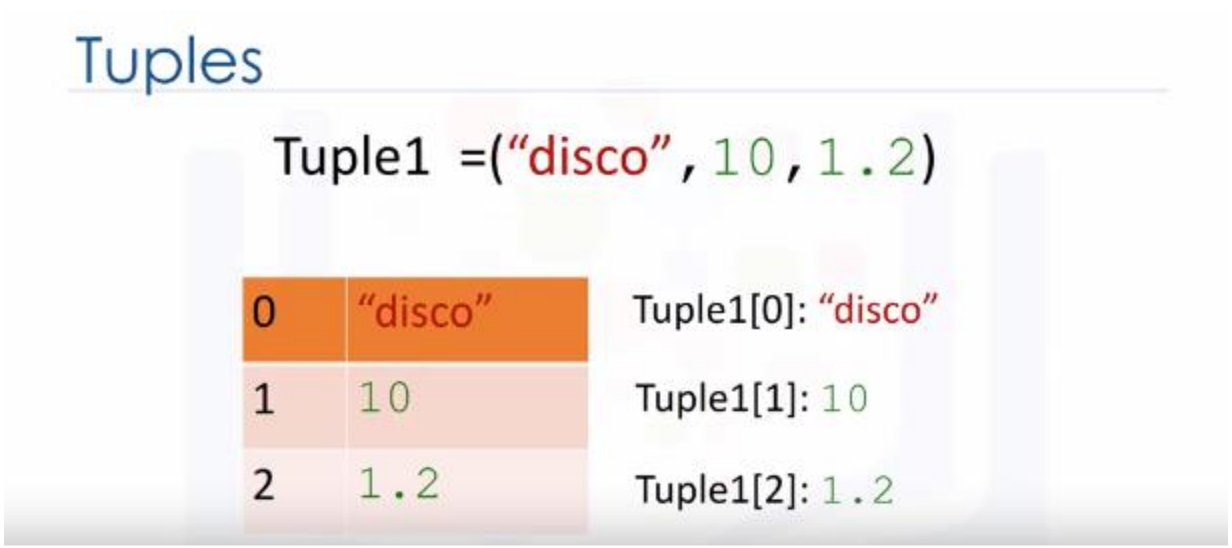
Week 2: LIST AND TUPLES



Tuples

- Tuples are an ordered sequence
- Here is a Tuple "Ratings"
- Tuples are written as comma-separated elements within parentheses

```
Ratings =(10,9,6,5,10,8,9,6,2)
```



Tuples

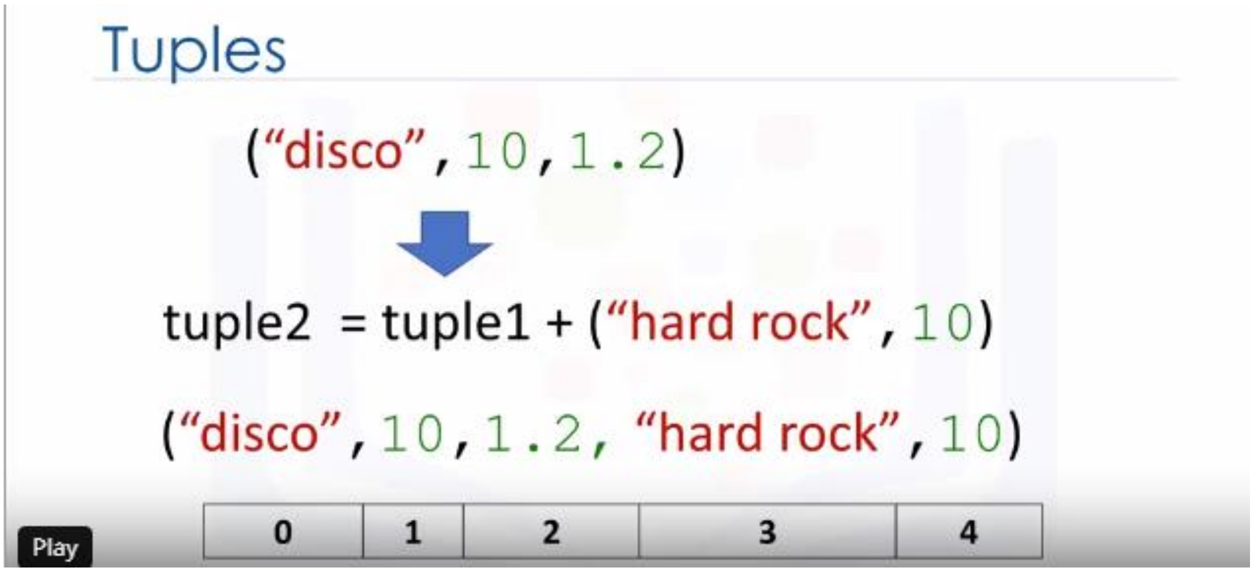
```
Tuple1 =("disco",10,1.2)
```

0	"disco"
1	10
2	1.2

Tuple1[0]: "disco"

Tuple1[1]: 10

Tuple1[2]: 1.2

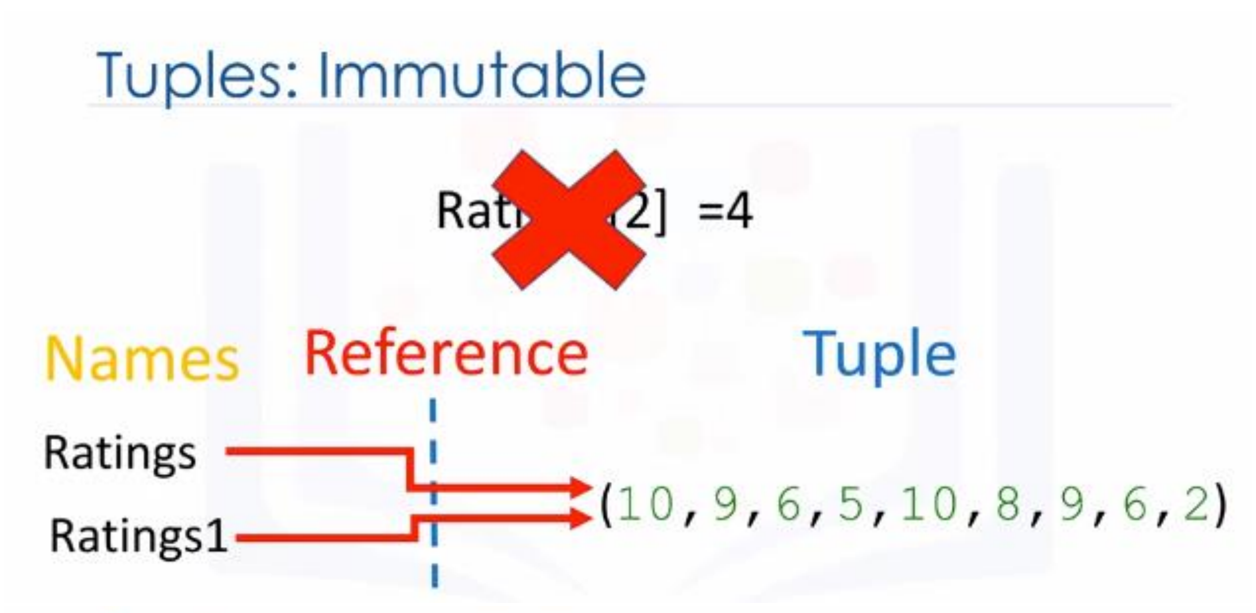


Tuples

("disco", 10, 1.2)

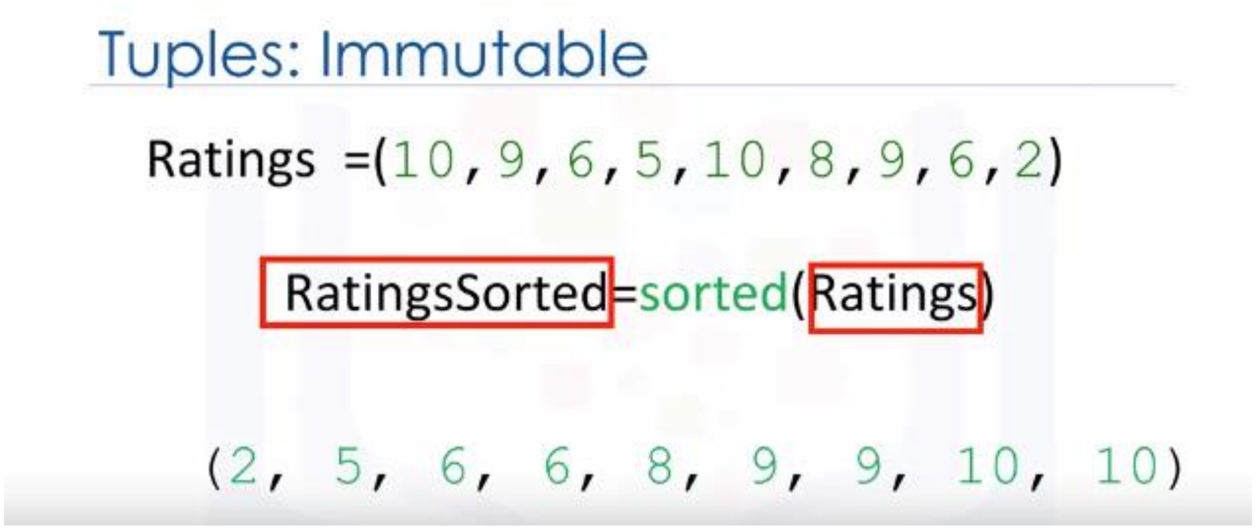
```
tuple2 = tuple1 + ("hard rock", 10)
```

("disco", 10, 1.2, "hard rock", 10)



Tuples: Immutable

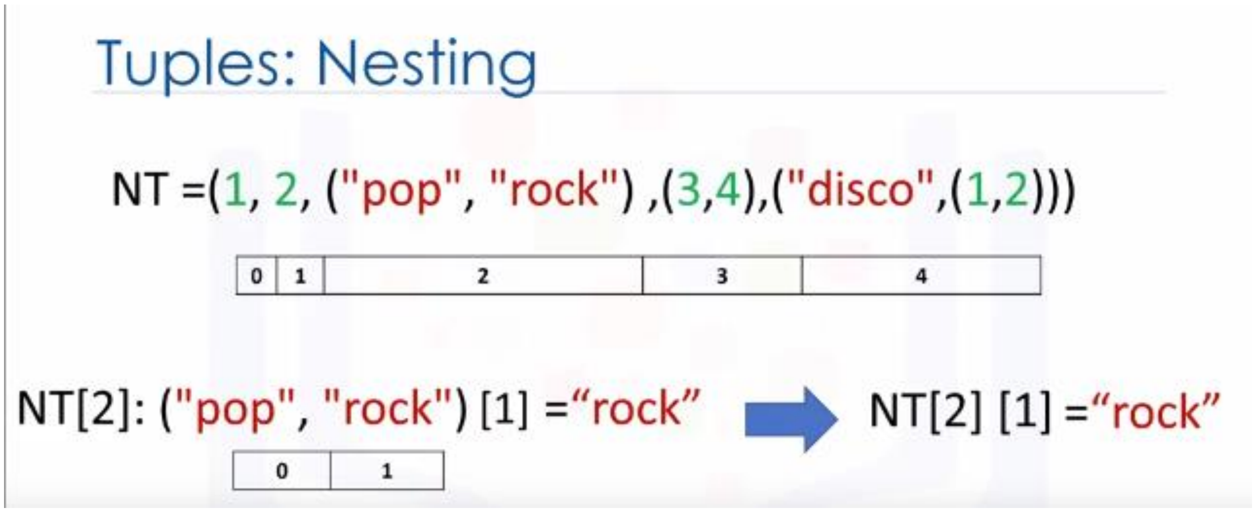
Rat[illegible]2] =4



Tuples: Immutable

```
Ratings =(10,9,6,5,10,8,9,6,2)

RatingsSorted=sorted(Ratings)
```

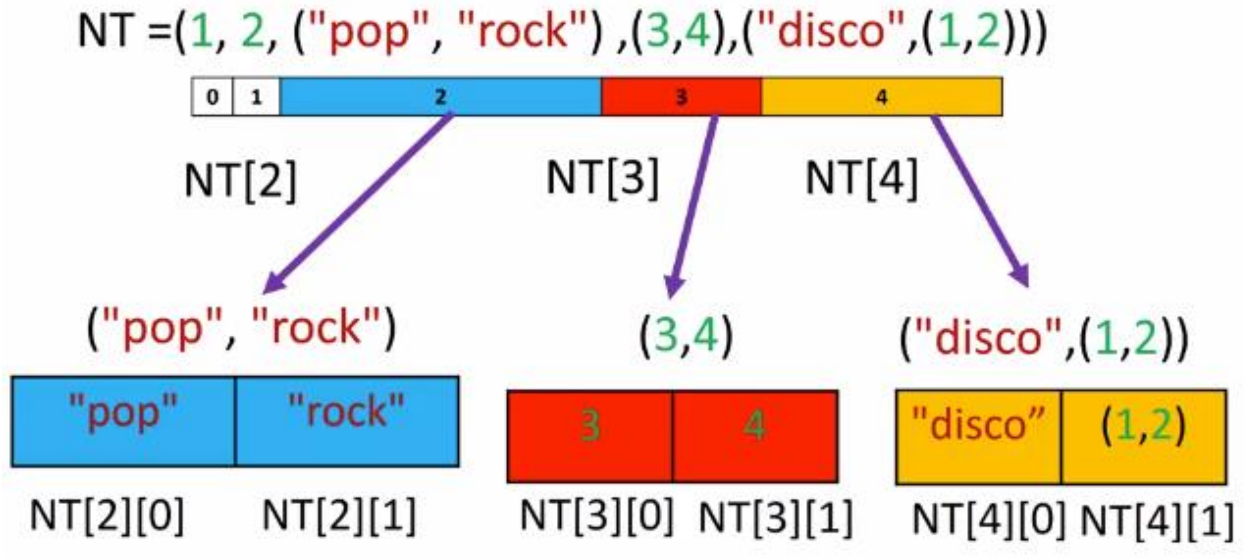
```
(2, 5, 6, 6, 8, 9, 9, 10, 10)
```

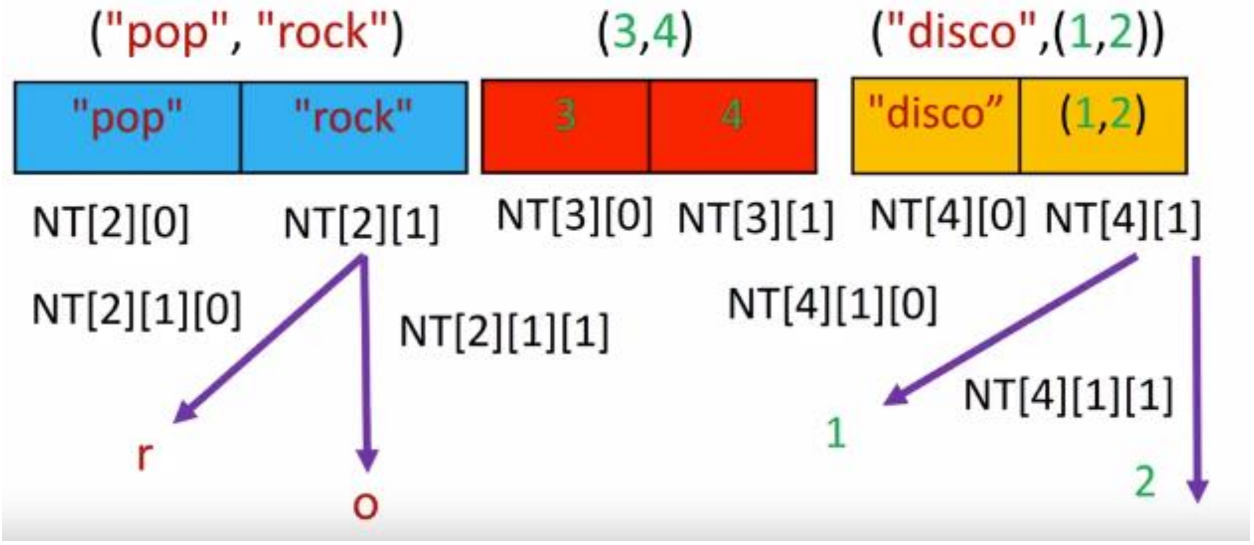


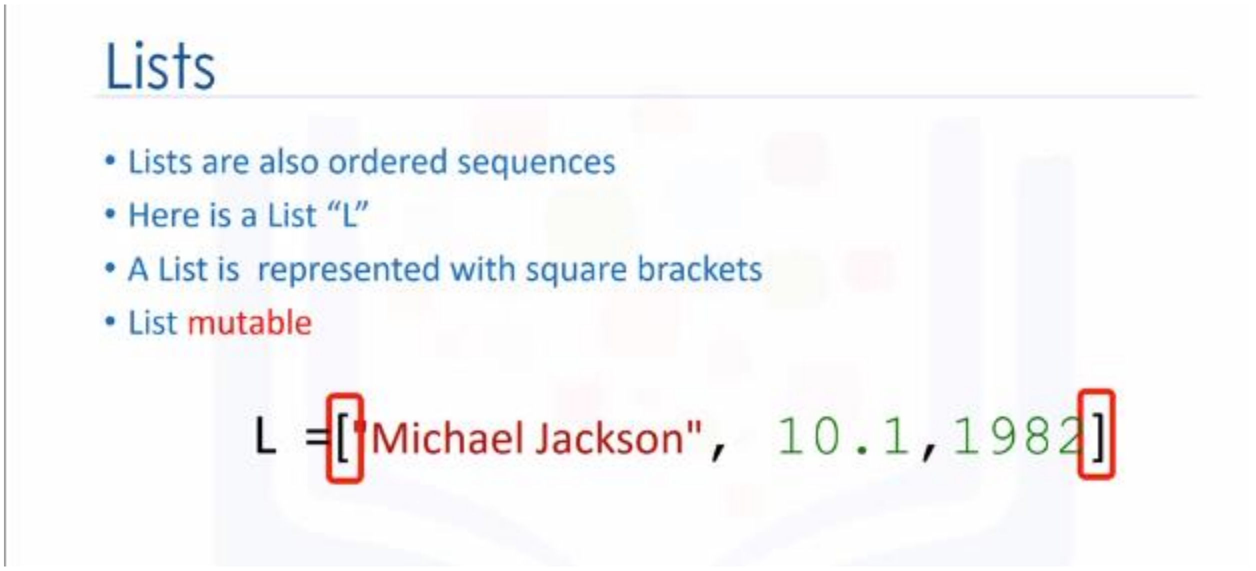
Tuples: Nesting

```
NT =(1, 2, ("pop", "rock") ,(3,4),("disco",(1,2)))
```

NT[2]: ("pop", "rock") [1] ="rock" → NT[2] [1] ="rock"



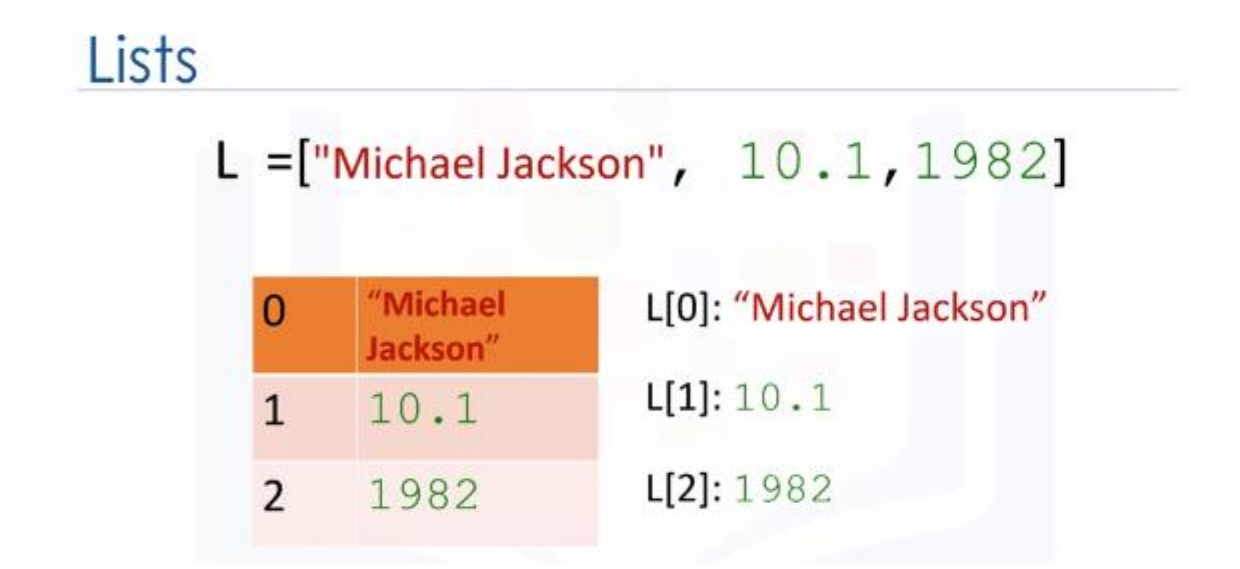




Lists

- Lists are also ordered sequences
- Here is a List "L"
- A List is represented with square brackets
- List mutable

```
L =["Michael Jackson", 10.1,1982]
```



Lists

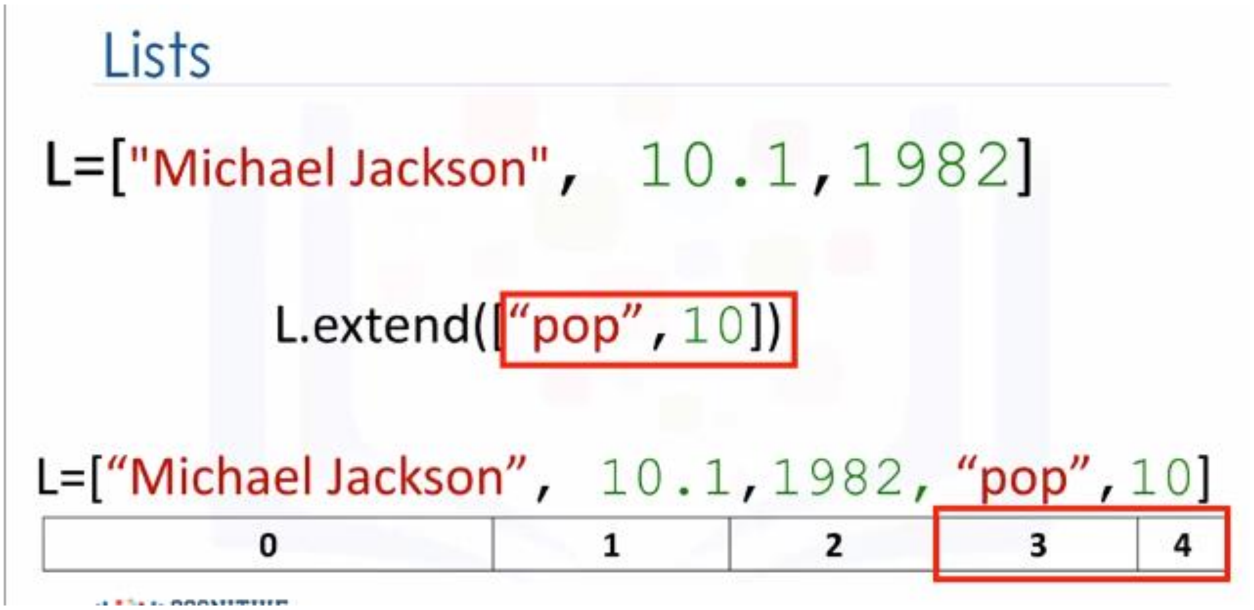
```
L =["Michael Jackson", 10.1,1982]
```

0	"Michael Jackson"
1	10.1
2	1982

L[0]: "Michael Jackson"

L[1]: 10.1

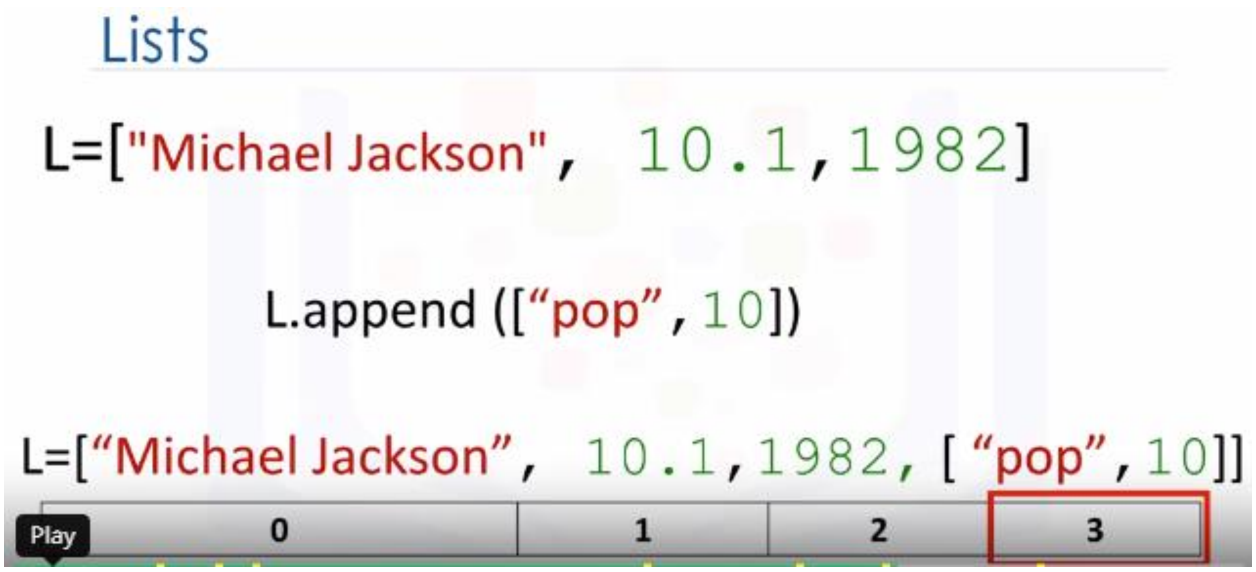
L[2]: 1982



Lists

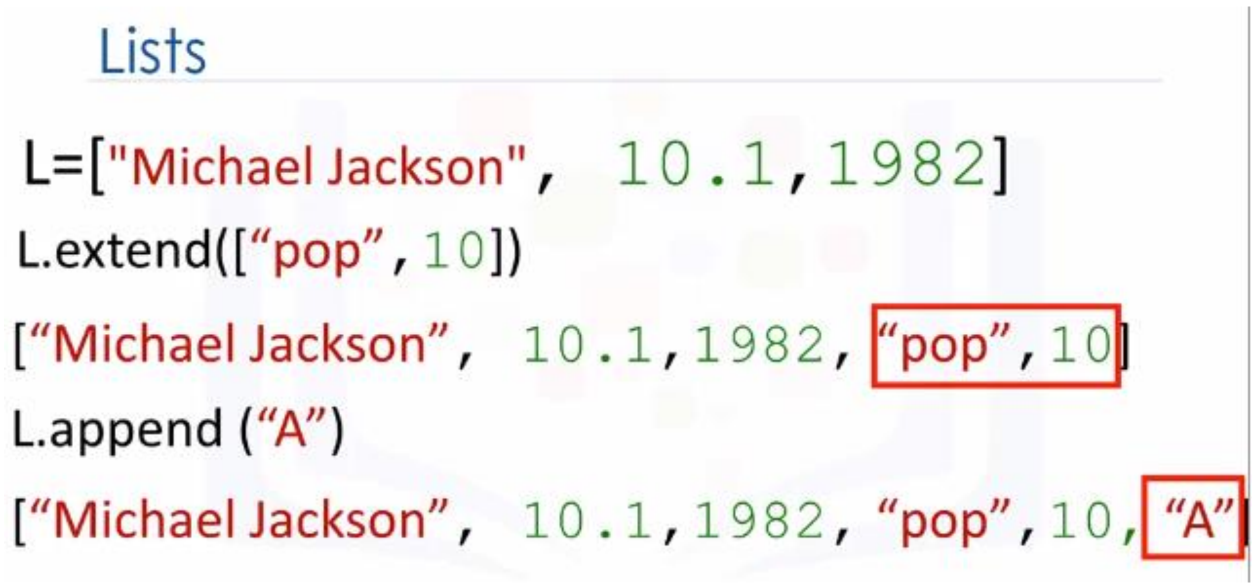
```
L=["Michael Jackson", 10.1,1982]

L.extend(["pop",10])

L=["Michael Jackson", 10.1,1982, "pop",10]
```

0	1	2	3	4



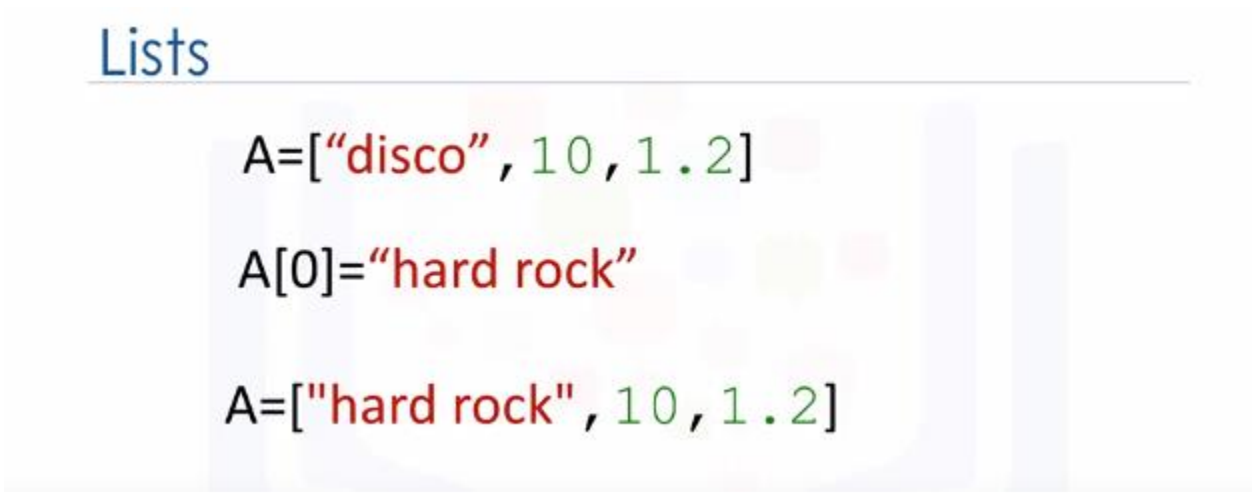
Lists

```
L=["Michael Jackson",  10.1,1982]

L.append (["pop",10])

L=["Michael Jackson",  10.1,1982, [ "pop",10]]
```



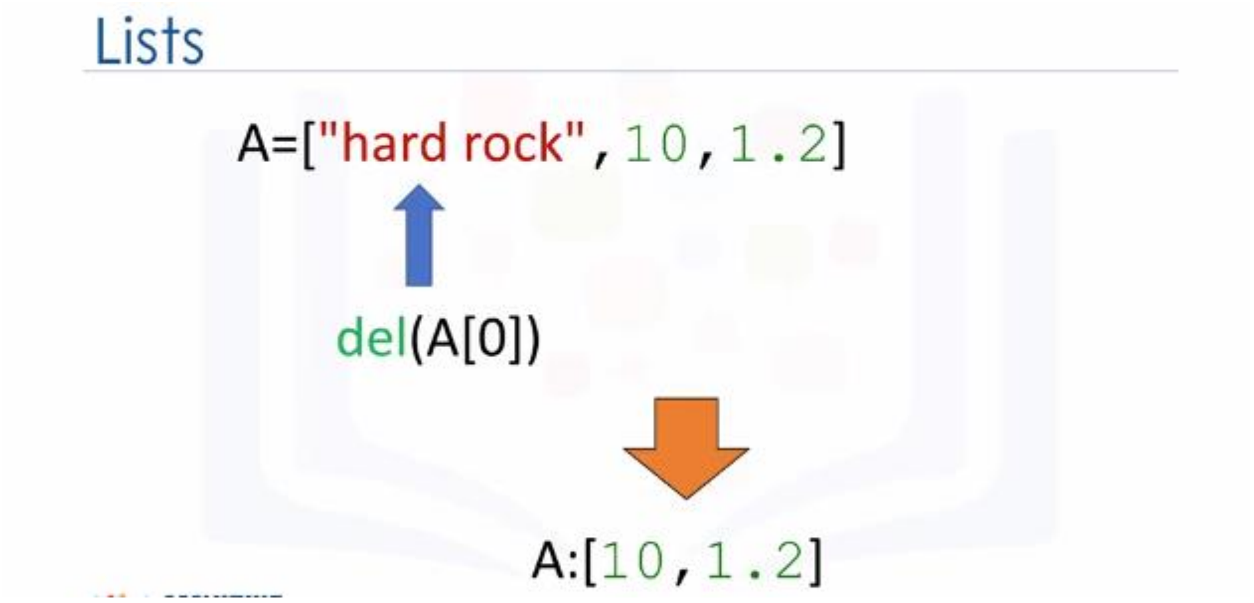
Lists

```
L=["Michael Jackson",  10.1,1982]
L.extend(["pop",10])
["Michael Jackson",  10.1,1982, "pop",10]
L.append ("A")
["Michael Jackson",  10.1,1982, "pop",10, "A"]
```



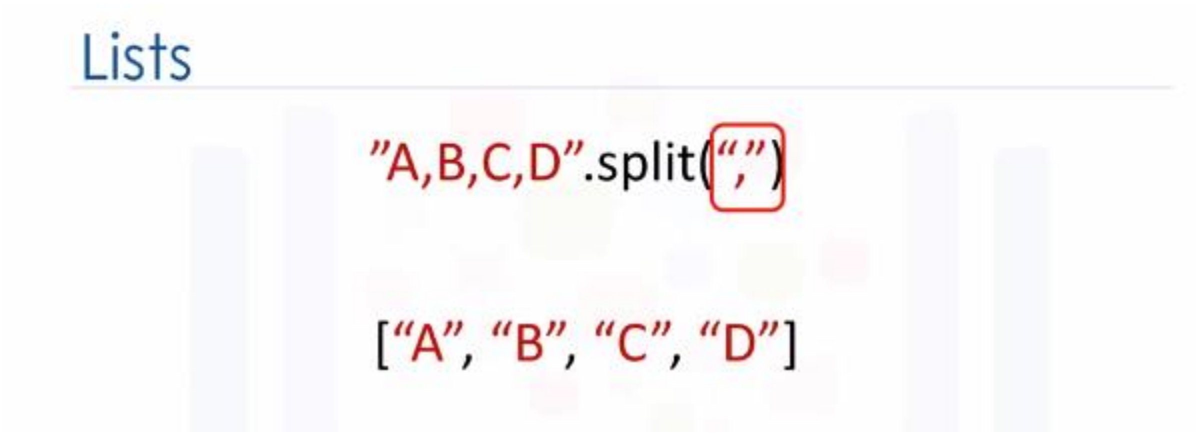
Lists

```
A=["disco",10,1.2]

A[0]="hard rock"

A=["hard rock",10,1.2]
```



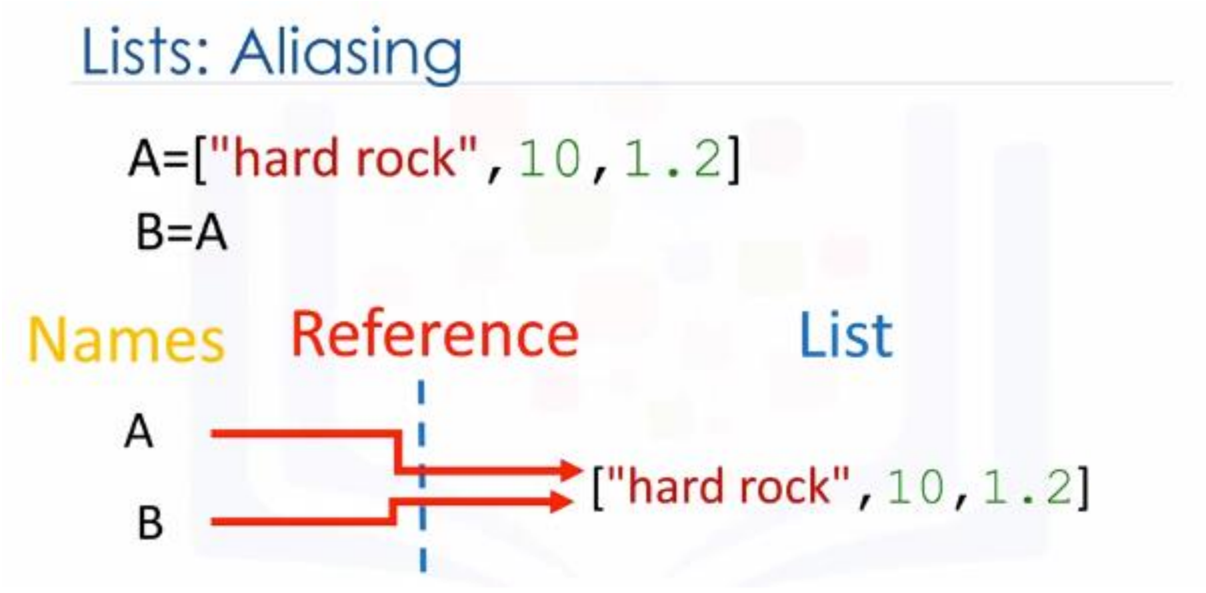
Lists

```
A=["hard rock",10,1.2]
del(A[0])
A:[10,1.2]
```



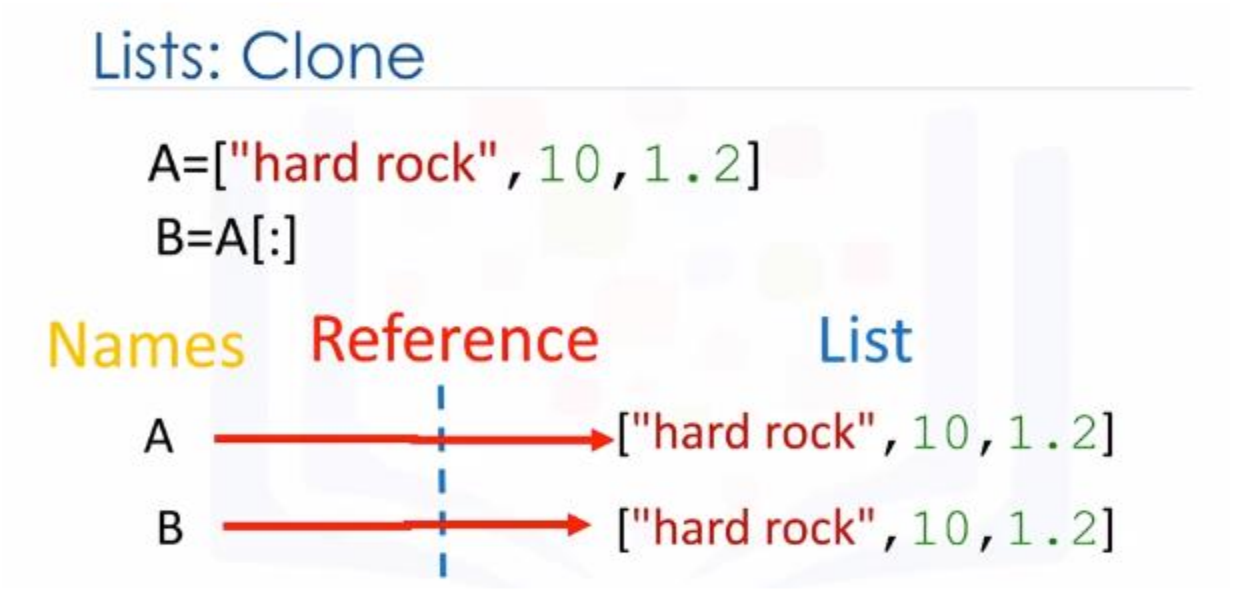
Lists

```
"A,B,C,D".split(",")

["A", "B", "C", "D"]
```

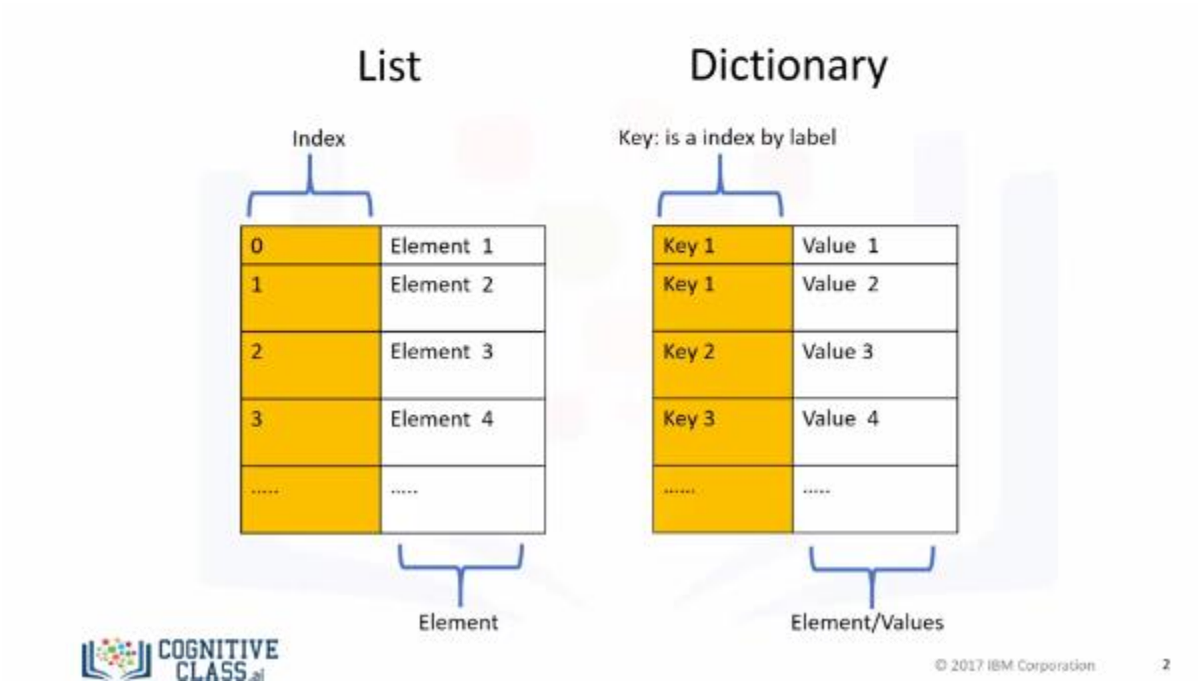


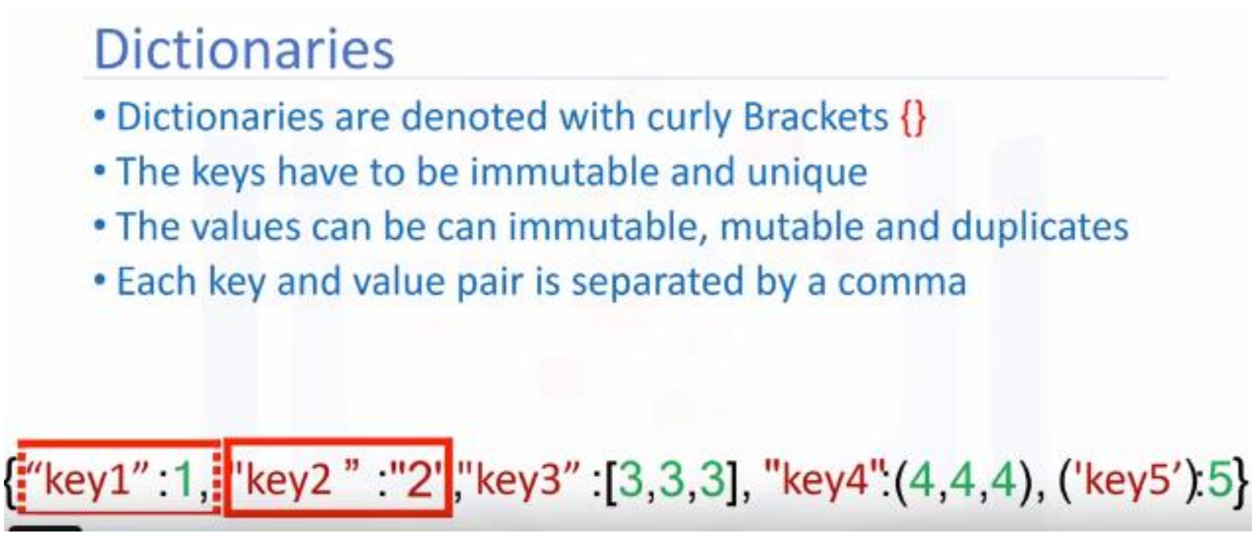
Lists: Aliasing

```
A=["hard rock",10,1.2]
B=A
```


Lists: Clone
A=["hard rock",10,1.2]
B=A[:]
Names
Reference
List
A
B
["hard rock",10,1.2]
["hard rock",10,1.2]

Dictionary

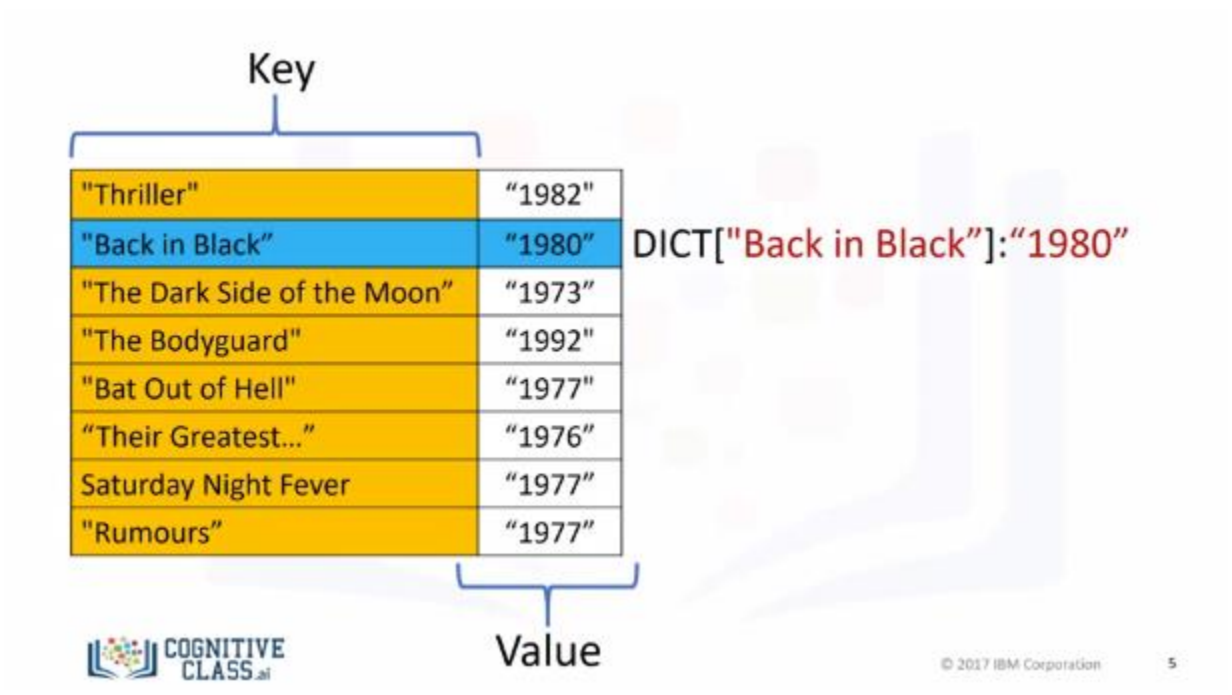

List
Dictionary
Index
Key: is a index by label
0
Element 1
1
Element 2
2
Element 3
3
Element 4
Key 1
Value 1
Key 1
Value 2
Key 2
Value 3
Key 3
Value 4
Element
Element/Values
COGNITIVE CLASS.ai
© 2017 IBM Corporation
2



Dictionaries

- Dictionaries are denoted with curly Brackets {}
- The keys have to be immutable and unique
- The values can be can immutable, mutable and duplicates
- Each key and value pair is separated by a comma

{"key1" :1, 'key2 " :"2","key3" :[3,3,3], "key4":(4,4,4), ('key5'):5}

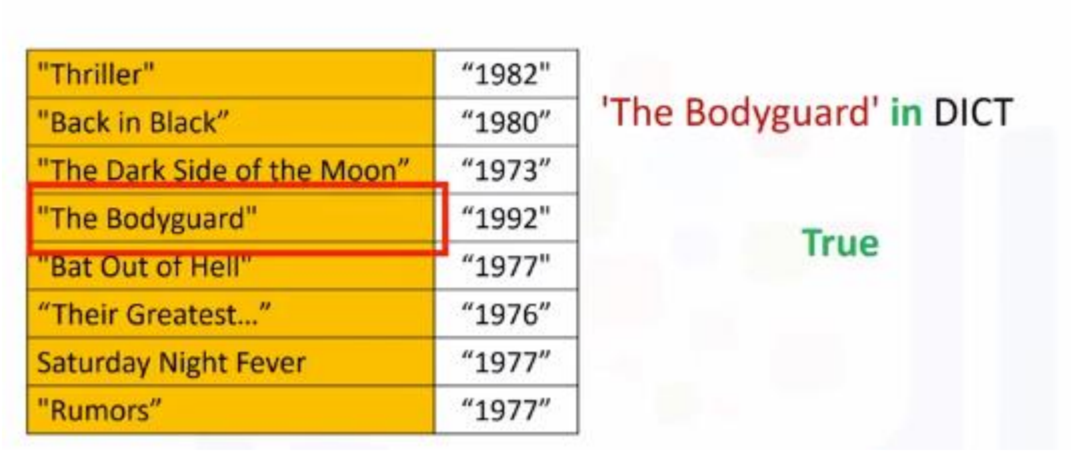


Key

"Thriller"	"1982"
"Back in Black"	"1980"
"The Dark Side of the Moon"	"1973"
"The Bodyguard"	"1992"
"Bat Out of Hell"	"1977"
"Their Greatest…"	"1976"
Saturday Night Fever	"1977"
"Rumours"	"1977"

DICT["Back in Black"]:"1980"

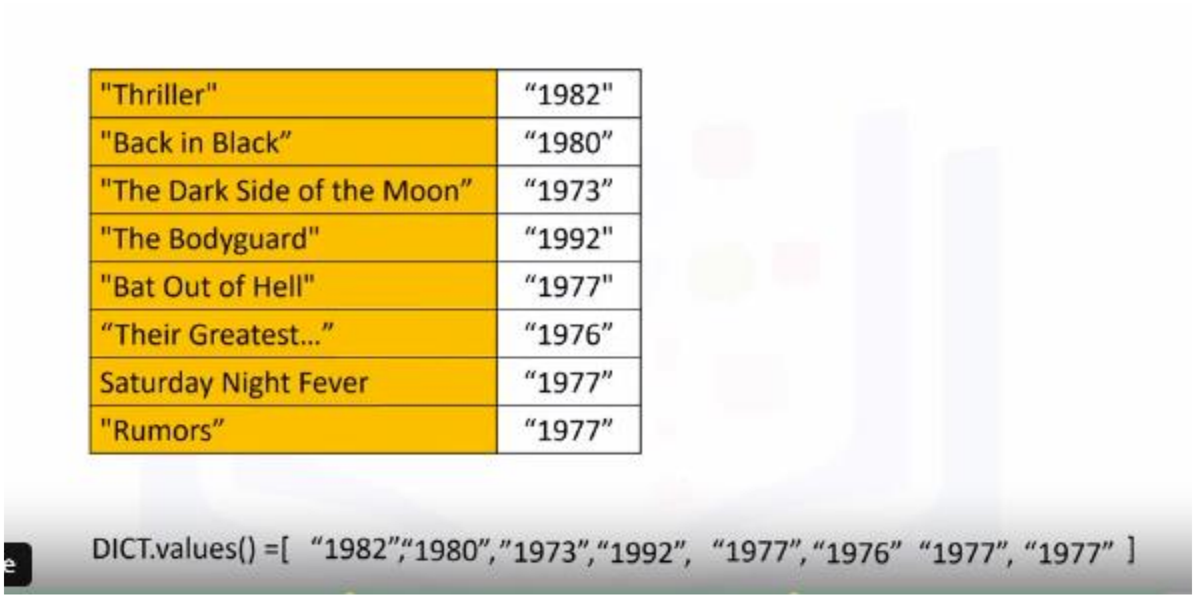
Value



"Thriller"	"1982"
"Back in Black"	"1980"
"The Dark Side of the Moon"	"1973"
"The Bodyguard"	"1992"
"Bat Out of Hell"	"1977"
"Their Greatest…"	"1976"
Saturday Night Fever	"1977"
"Rumors"	"1977"

'The Bodyguard' in DICT

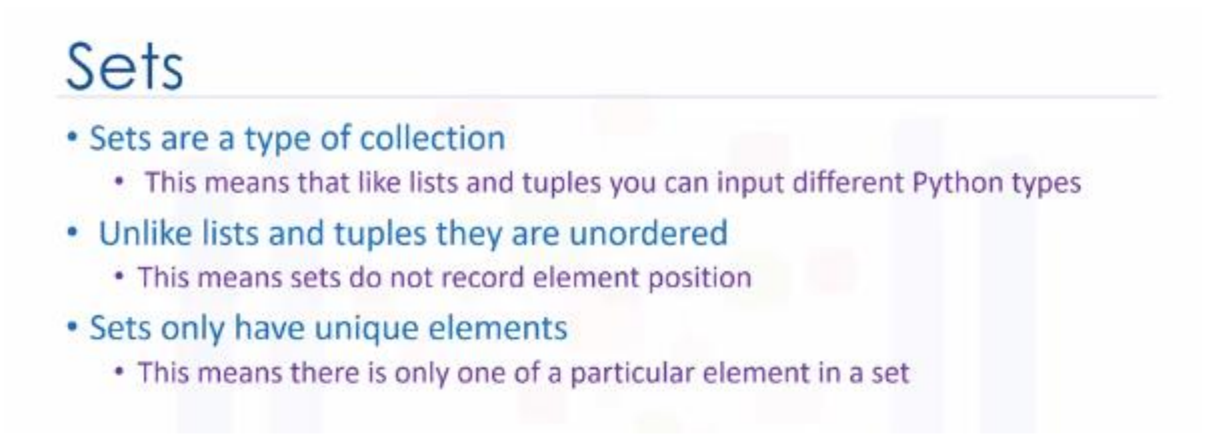
True



"Thriller"	"1982"
"Back in Black"	"1980"
"The Dark Side of the Moon"	"1973"
"The Bodyguard"	"1992"
"Bat Out of Hell"	"1977"
"Their Greatest..."	"1976"
Saturday Night Fever	"1977"
"Rumors"	"1977"

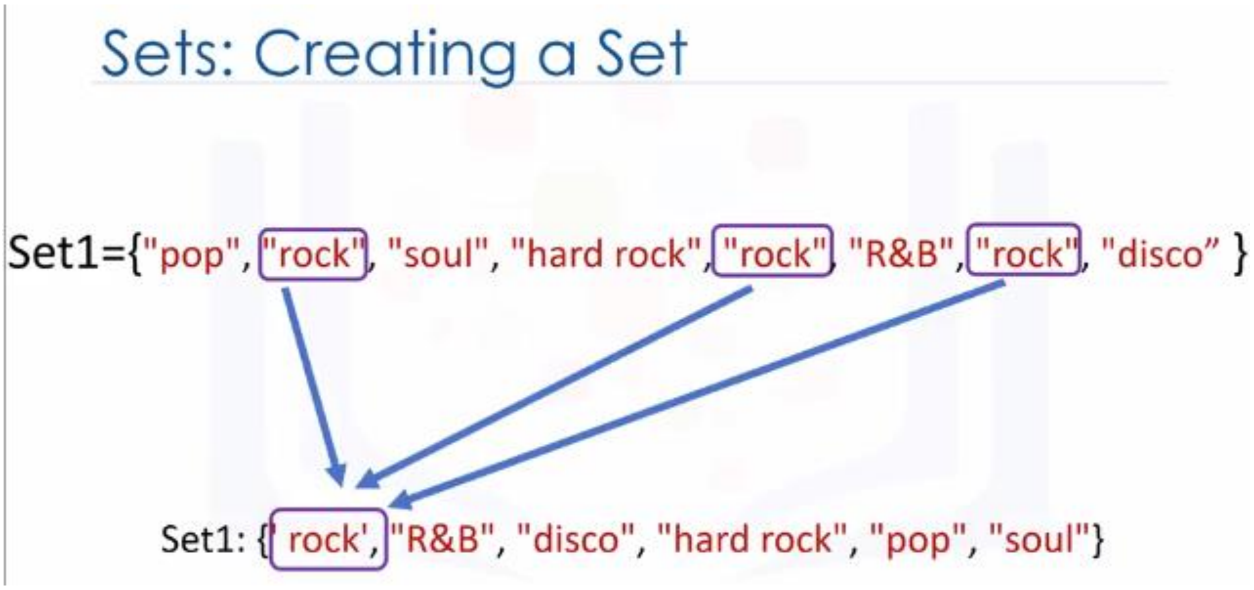
DICT.values() =["1982","1980","1973","1992", "1977","1976" "1977", "1977"]

SETS



Sets

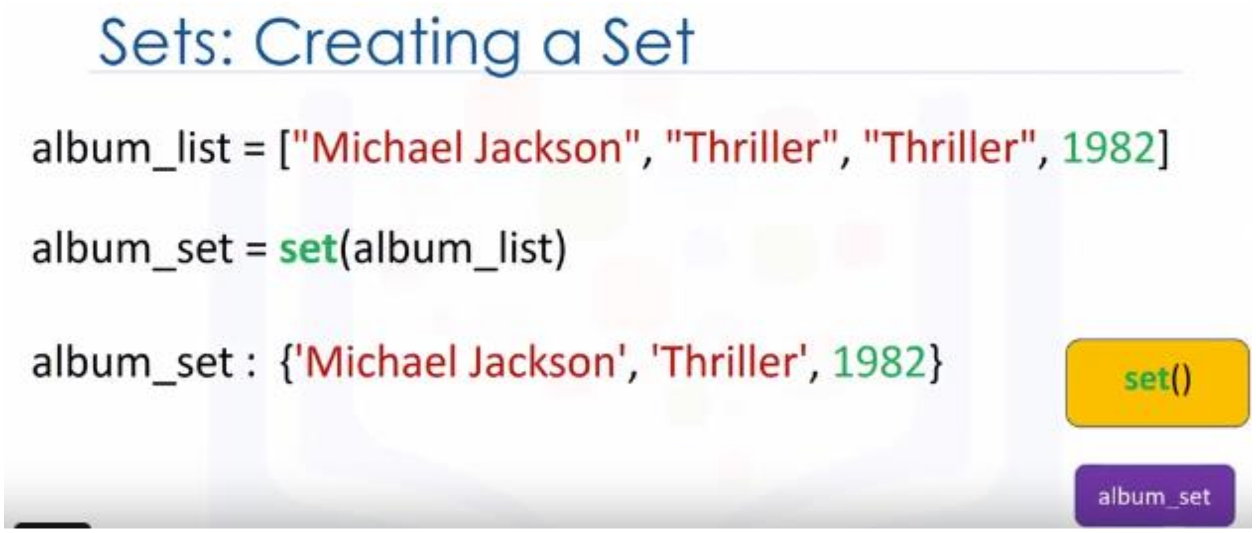
- Sets are a type of collection
  - This means that like lists and tuples you can input different Python types
- Unlike lists and tuples they are unordered
  - This means sets do not record element position
- Sets only have unique elements
  - This means there is only one of a particular element in a set



Sets: Creating a Set

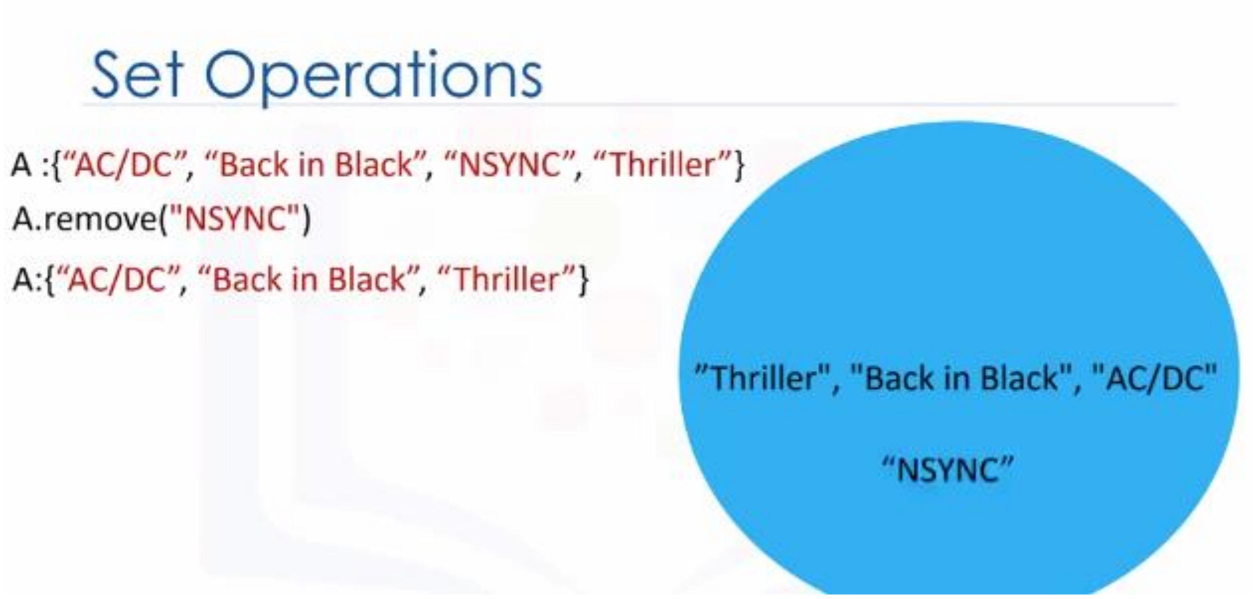
Set1={"pop","rock", "soul", "hard rock","rock", "R&B","rock", "disco" }

Set1: {'rock',"R&B", "disco", "hard rock", "pop", "soul"}



Sets: Creating a Set

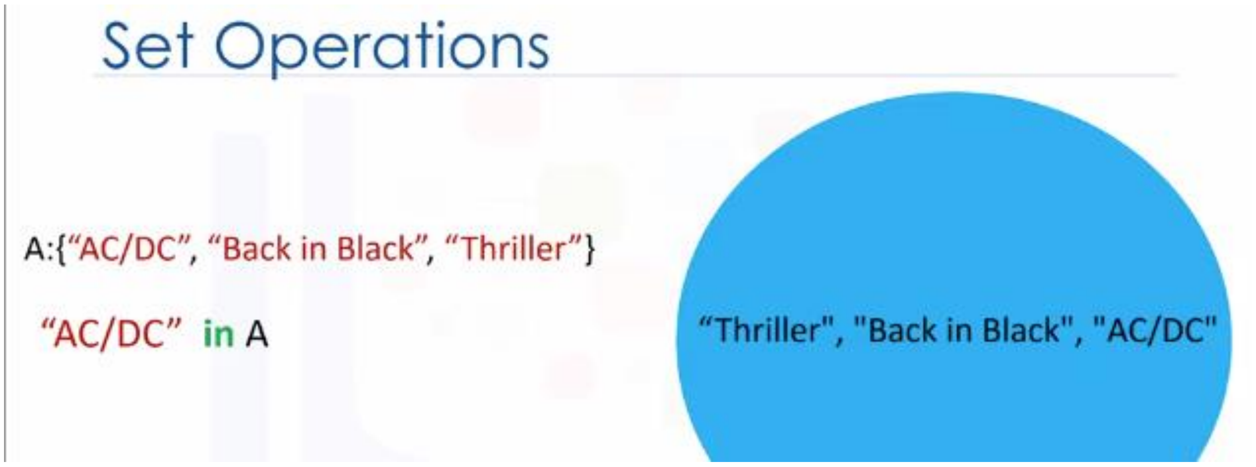
```
album_list = ["Michael Jackson", "Thriller", "Thriller", 1982]

album_set = set(album_list)

album_set :  {'Michael Jackson', 'Thriller', 1982}
```



Set Operations

```
A :{"AC/DC", "Back in Black", "NSYNC", "Thriller"}
A.remove("NSYNC")
A:{"AC/DC", "Back in Black", "Thriller"}
```

Add()

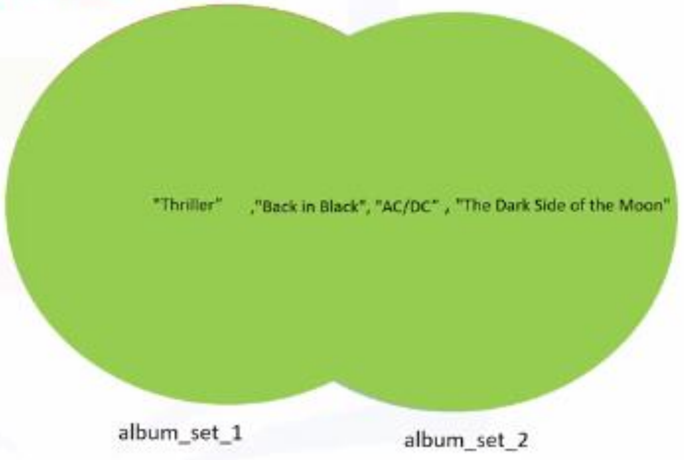


Set Operations

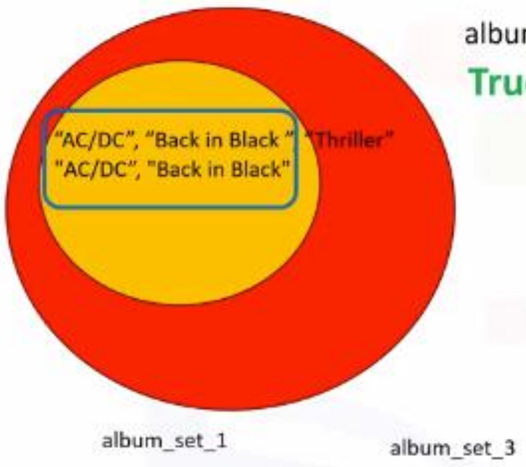
```
A:{"AC/DC", "Back in Black", "Thriller"}

"AC/DC"  in A
```

```
album_set_1 ={"AC/DC"," Back in Black ", "Thriller }

album_set_2={"AC/DC", "Back in Black", "The Dark Side of the Moon"}

album_set_3=album_set_1 & album_set_2

album_set_3: {"AC/DC"," Back in Black "}
```

```
album_set_1. union(album_set_2)
{AC/DC", "Back in Black", "The Dark Side of the Moon", "Thriller" }
```

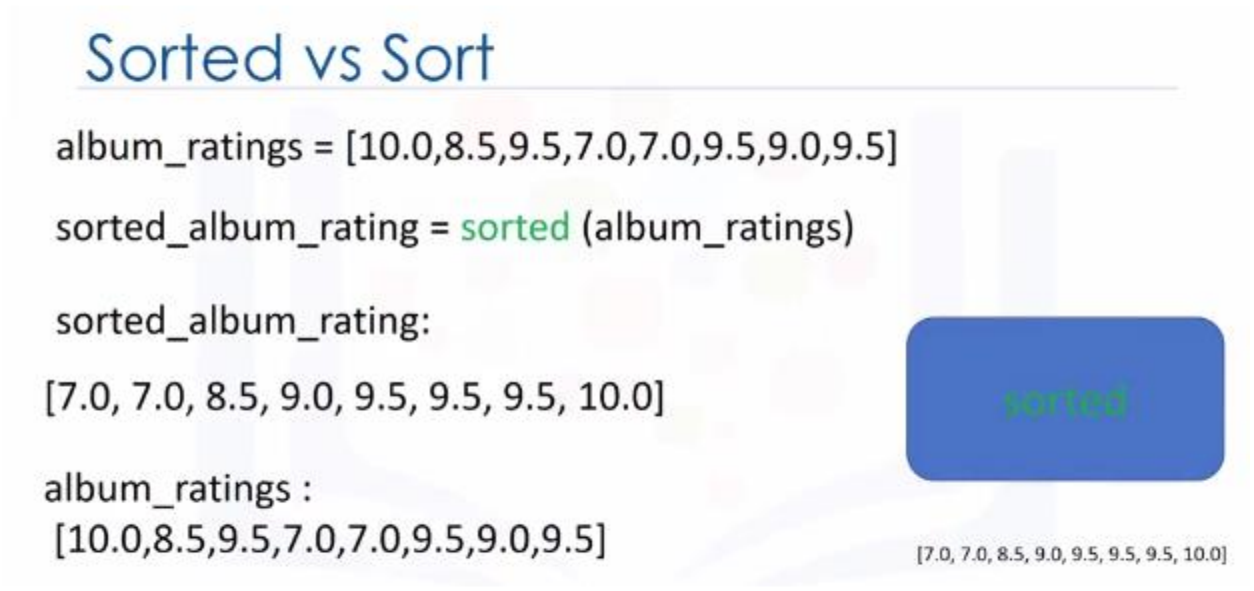


```
album_set_1 = {"AC/DC"," Back in Black ", "Thriller }
album_set_3 = {"AC/DC"," Back in Black "}
album_set_3.issubset(album_set1)
True
```



Week 3: Conditions and Branching

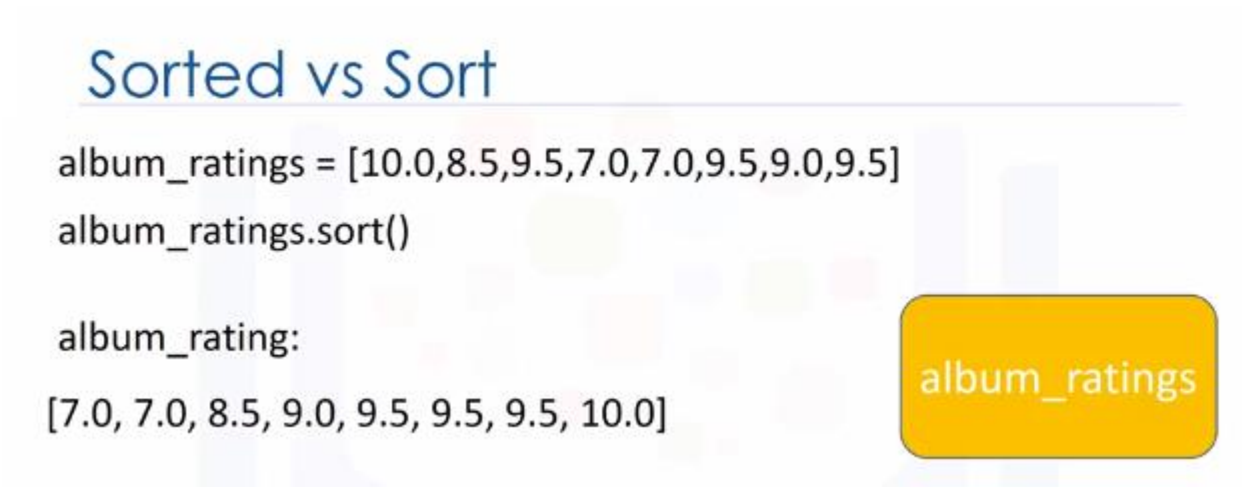
sorted is a function and returns a new list, it does not change the list **L**



Sorted vs Sort

```
album_ratings = [10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]

sorted_album_rating = sorted (album_ratings)

sorted_album_rating:

[7.0, 7.0, 8.5, 9.0, 9.5, 9.5, 9.5, 10.0]

album_ratings :
 [10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]
```

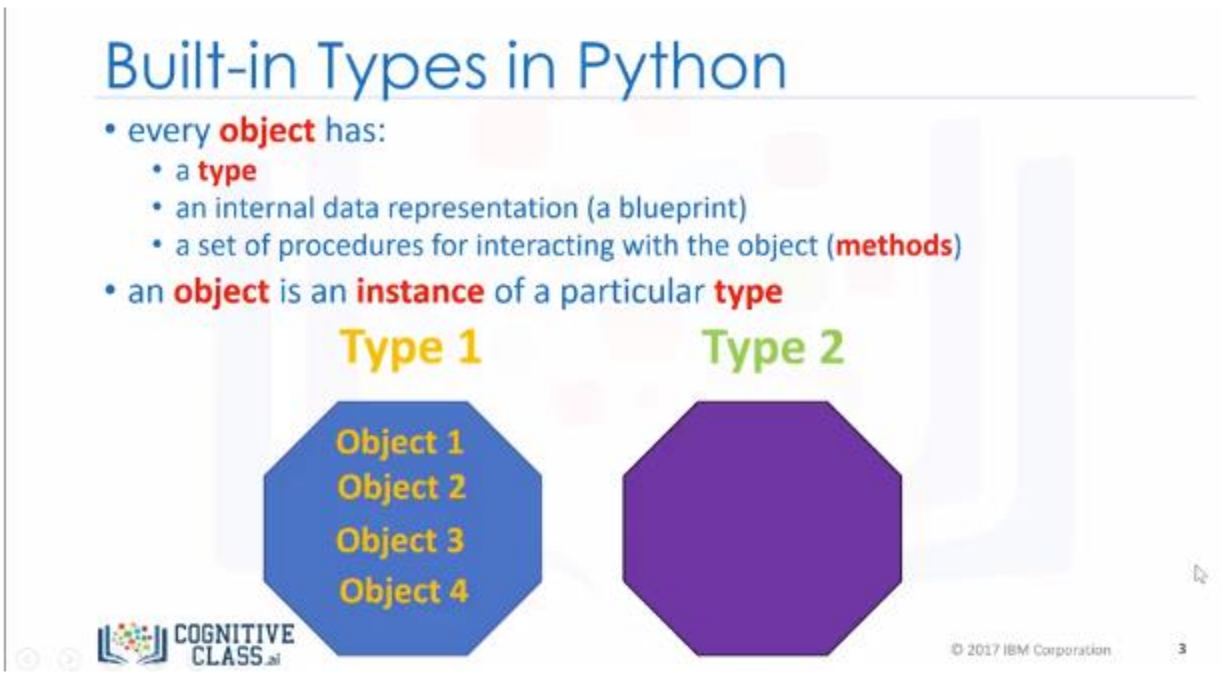
sorted
[7.0, 7.0, 8.5, 9.0, 9.5, 9.5, 9.5, 10.0]



Sorted vs Sort

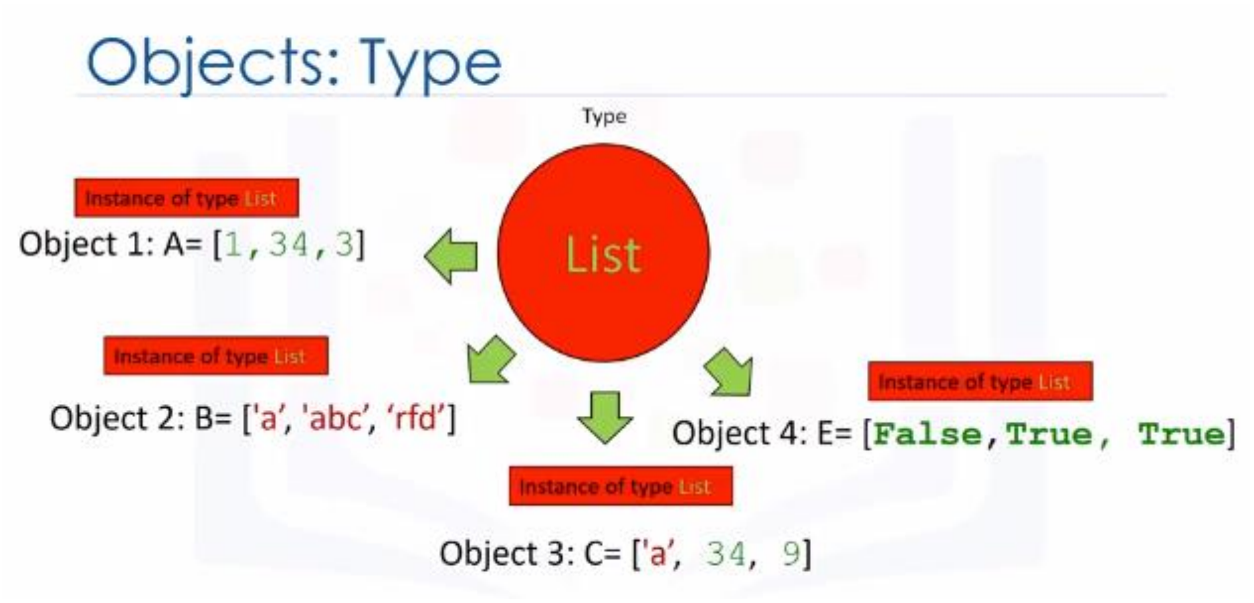
```
album_ratings = [10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]
album_ratings.sort()

album_rating:

[7.0, 7.0, 8.5, 9.0, 9.5, 9.5, 9.5, 10.0]
```

album_ratings



Built-in Types in Python

- every **object** has:
  - a **type**
  - an internal data representation (a blueprint)
  - a set of procedures for interacting with the object (**methods**)
- an **object** is an **instance** of a particular **type**



Objects: Type

Object 1: A= [1, 34, 3]

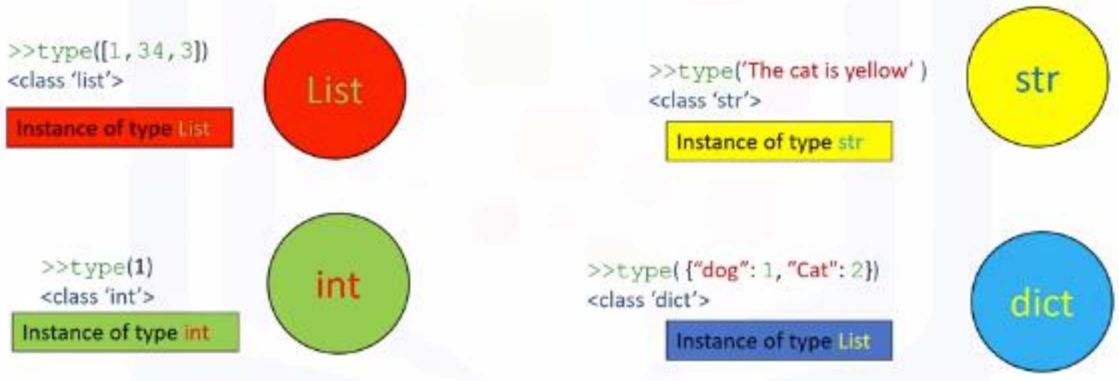
Object 2: B= ['a', 'abc', 'rfd']

Object 3: C= ['a', 34, 9]

Object 4: E= [False, True, True]

Objects: Type

- You can find the type of a object by using the command `type()`



```
>>type([1,34,3])
<class 'list'>
```

```
>>type('The cat is yellow' )
<class 'str'>
```

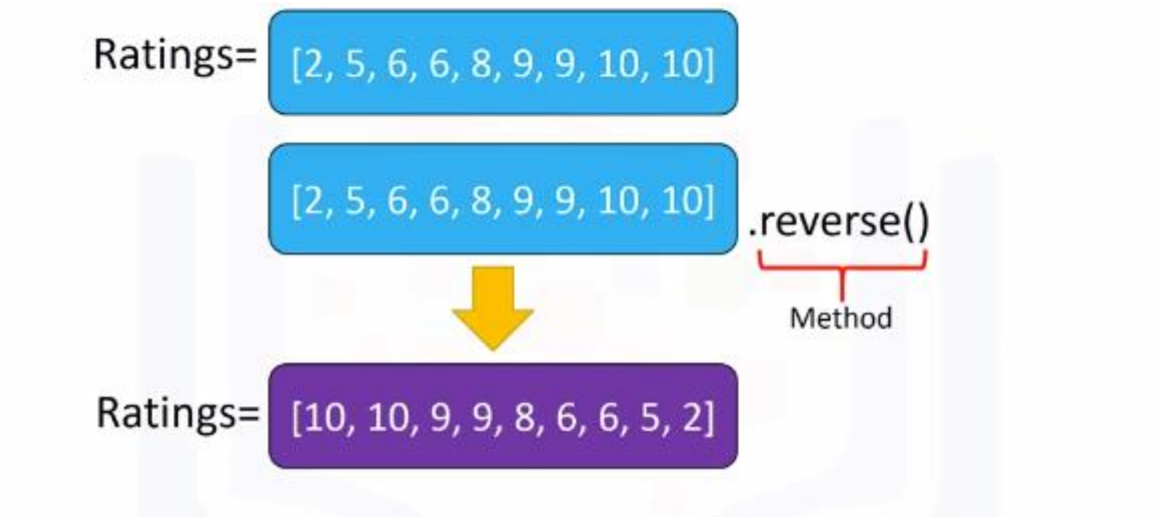
```
>>type(1)
<class 'int'>
```

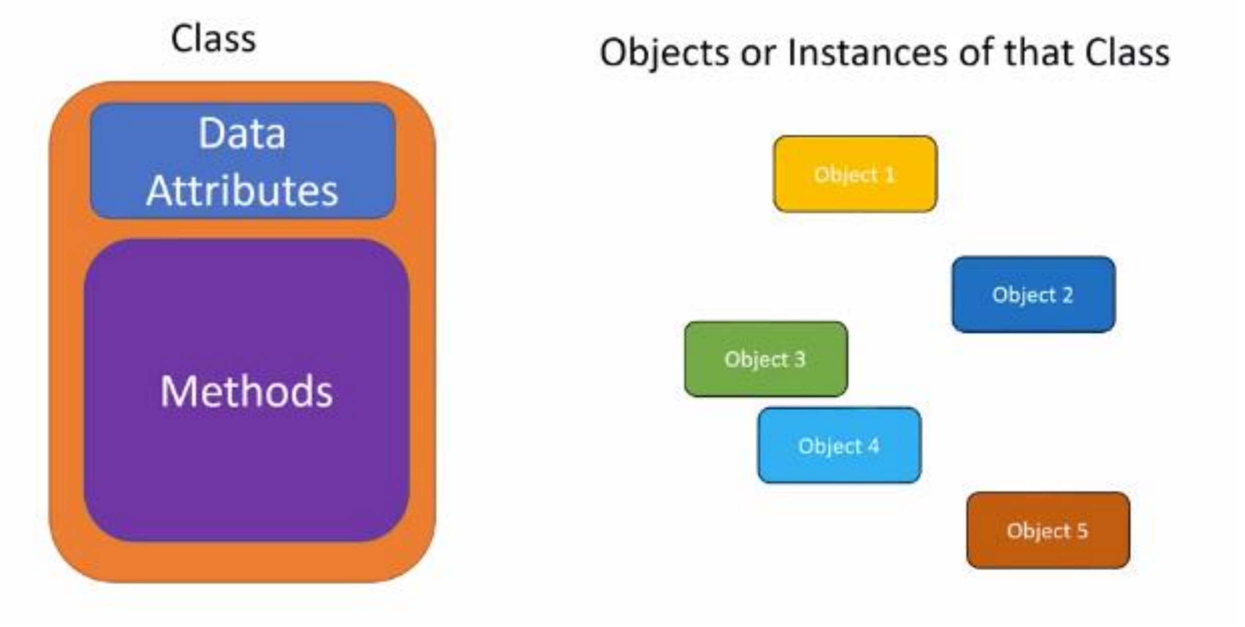
```
>>type( {"dog": 1, "Cat": 2})
<class 'dict'>
```

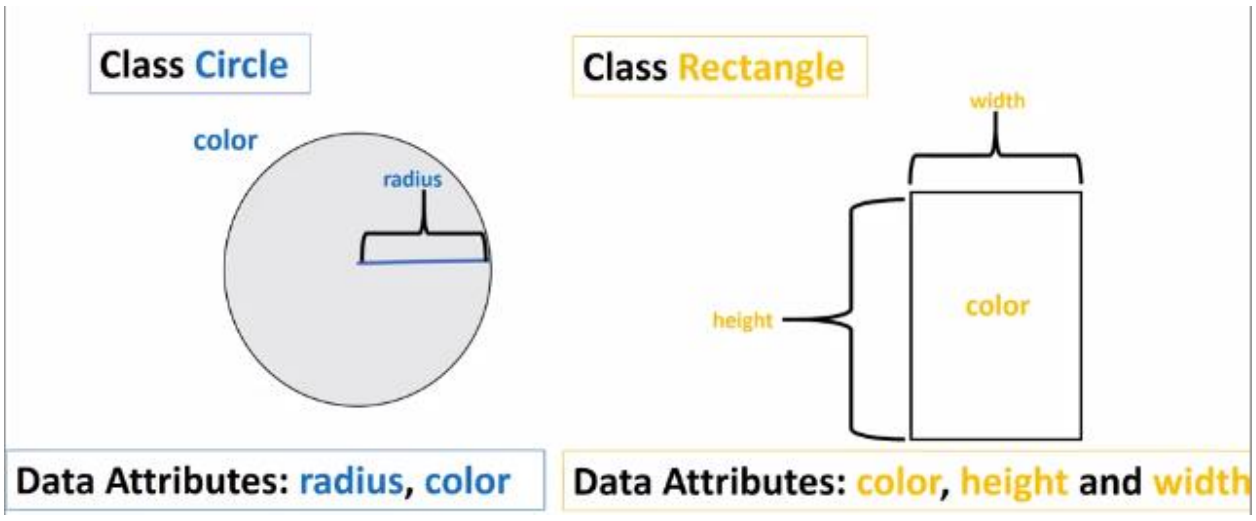
Methods

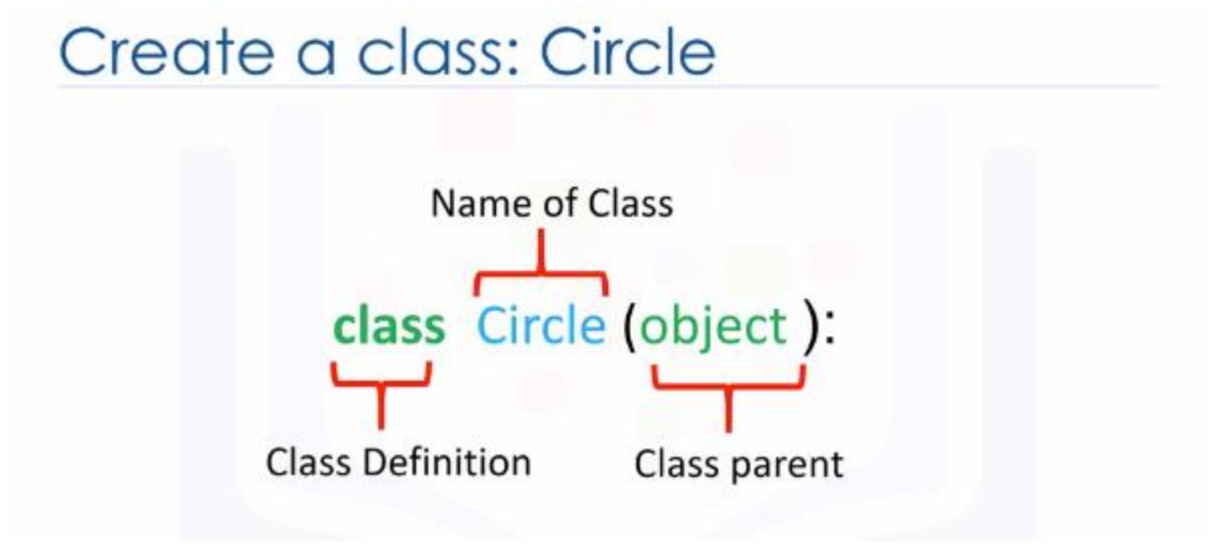
- A class or type's methods are functions that every instance of that class or type provides
- It's how you interact with the data in a object
- Sorting is an example of a method that interacts with the data in the object

```
Ratings=[10,9,6,5,10,8,9,6,2]

Ratings.sort()
```


Ratings=
[2, 5, 6, 6, 8, 9, 9, 10, 10]
[2, 5, 6, 6, 8, 9, 9, 10, 10]
.reverse()
Method
Ratings=
[10, 10, 9, 9, 8, 6, 6, 5, 2]

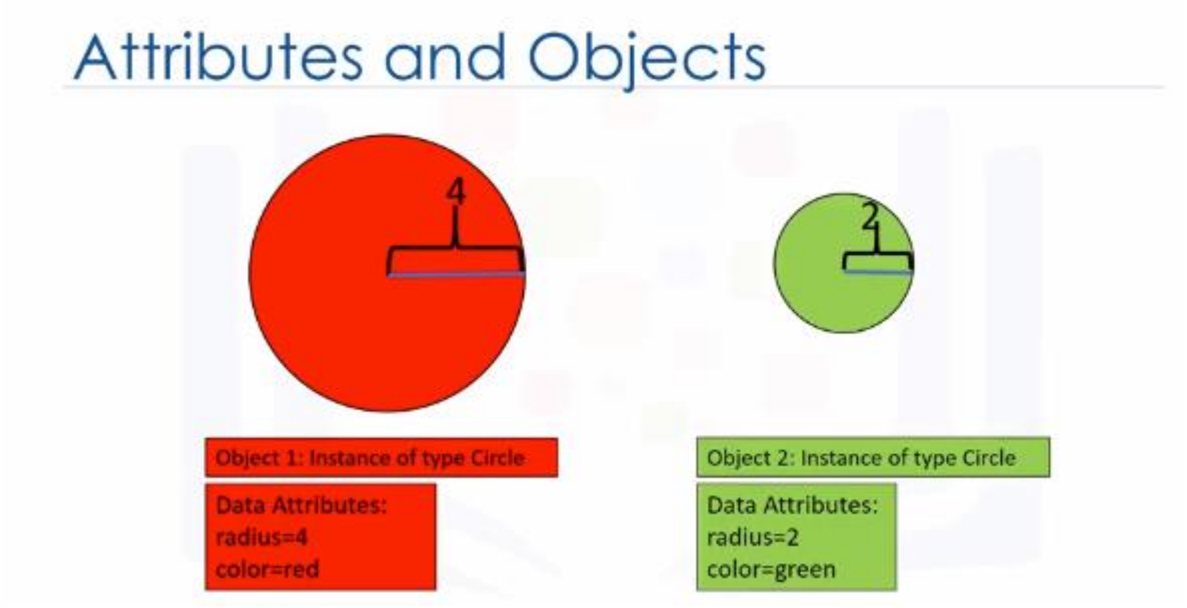

Class
Data Attributes
Methods
Objects or Instances of that Class
Object 1
Object 2
Object 3
Object 4
Object 5


Class Circle
color
radius
Class Rectangle
width
height
color
Data Attributes: radius, color
Data Attributes: color, height and width

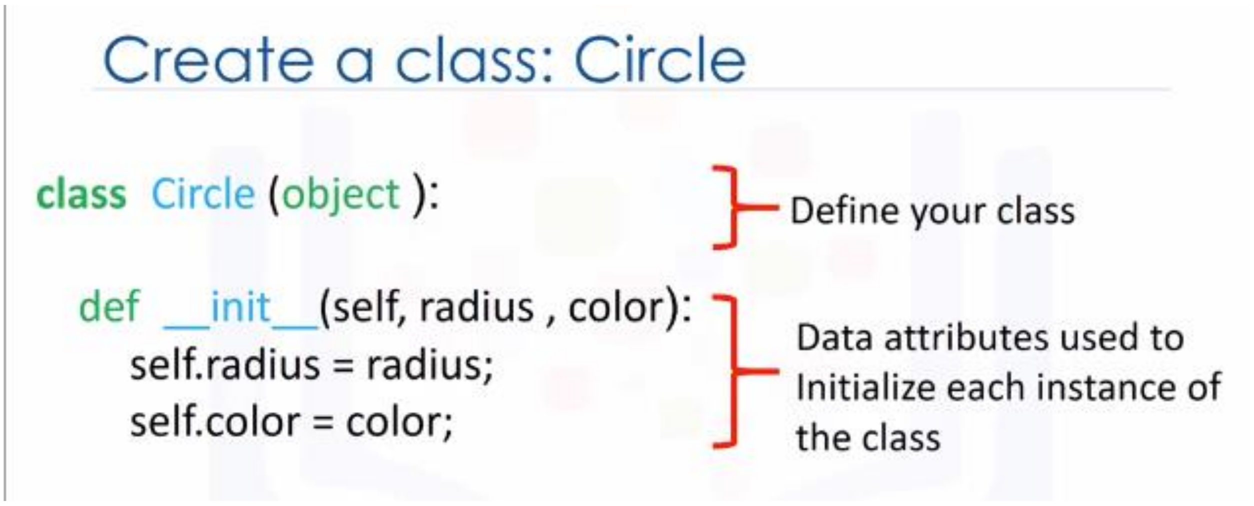


Create a class: Circle

```
class  Circle (object ):
```



Attributes and Objects



Create a class: Circle

```
class  Circle (object ):

  def  __init__(self, radius , color):
      self.radius = radius;
      self.color = color;
```

Define your class
Data attributes used to Initialize each instance of the class

Create a class: Circle

special method or constructor used to initialize data attributes

parameters

```
def __init__(self, radius , color):
```

The self parameter

```
    self.radius = radius;
    self.color = color;
```

Create an Instance of a Class: Circle

How to create an object of class circle:

Name of Class

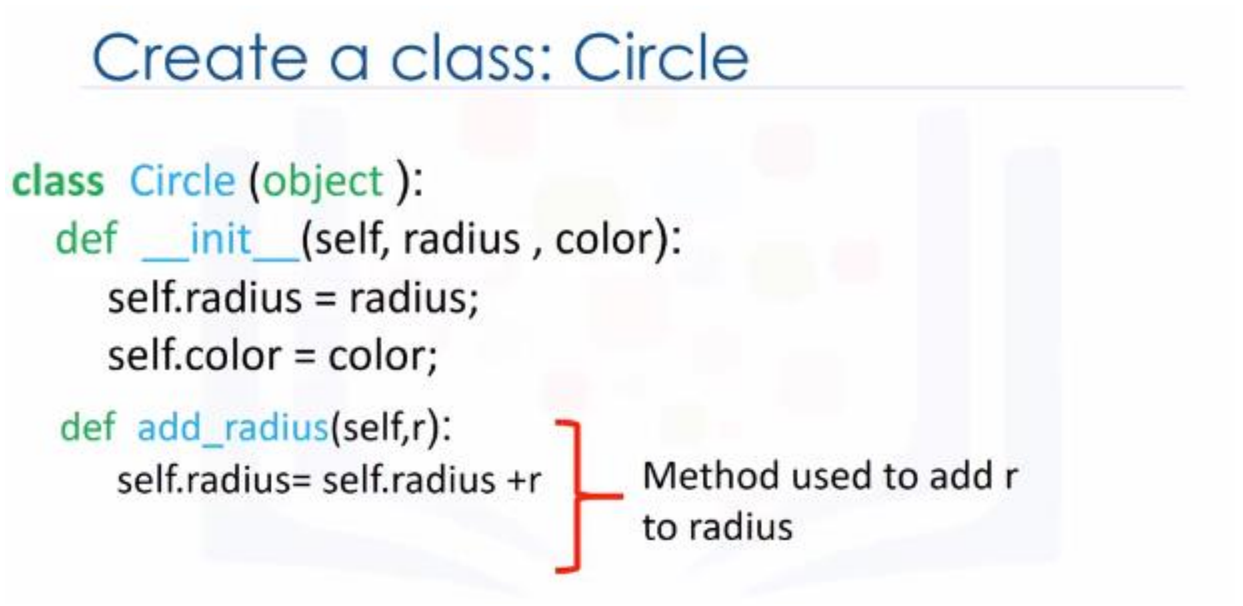
```
RedCircle =Circle (10, “red” )
```

Attributes

Create an Instance of a Class: Circle

```
C1=Circle (10, “red”)

C1.color=“blue”

C1.color
```

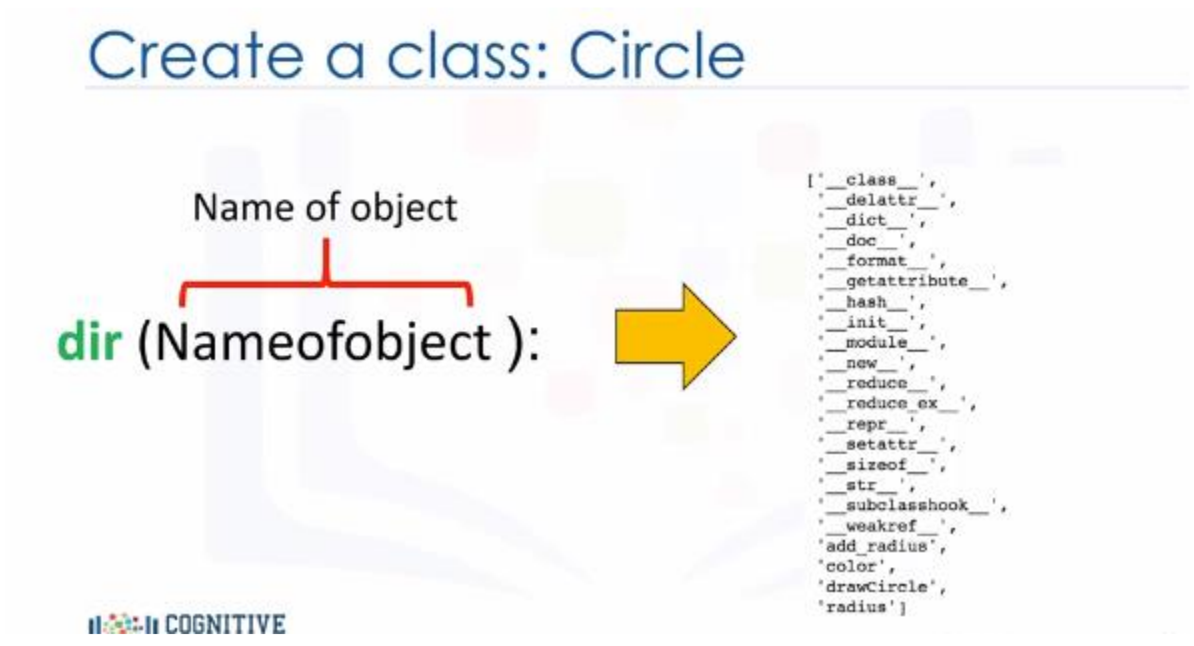
```
“blue”
```



Create a class: Circle

```
class  Circle (object ):
   def  __init__(self, radius , color):
      self.radius = radius;
      self.color = color;
   def  add_radius(self,r):
      self.radius= self.radius +r
```

Method used to add r to radius



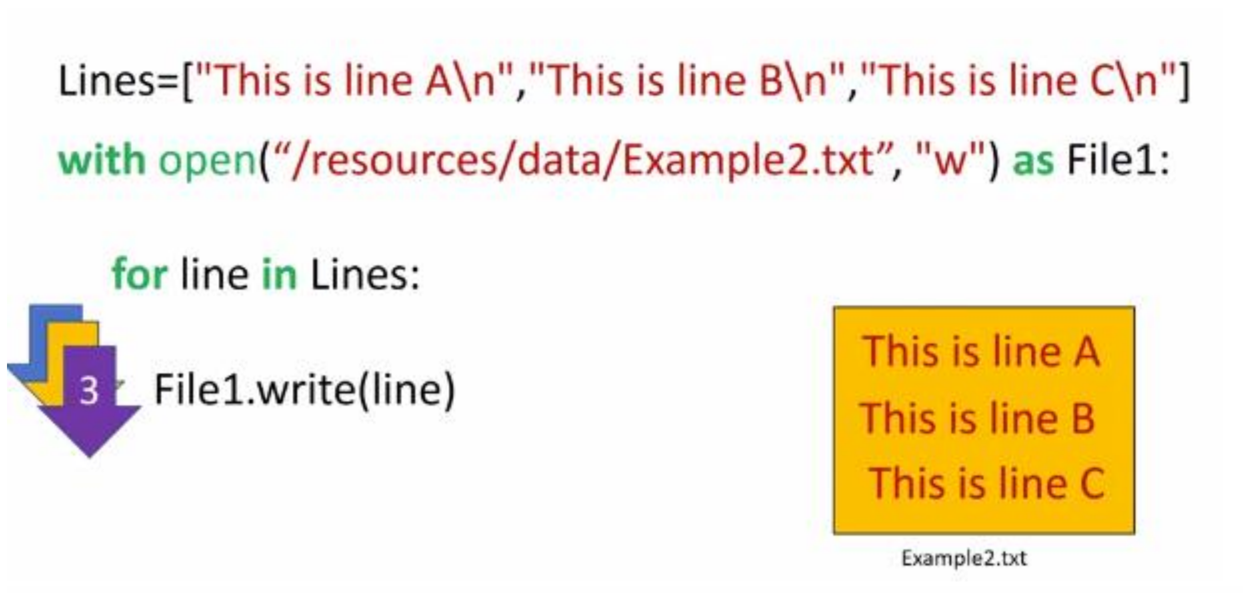
Create a class: Circle

Name of object

```
dir (Nameofobject ):
```

```
['__class__',
 '__delattr__',
 '__dict__',
 '__doc__',
 '__format__',
 '__getattribute__',
 '__hash__',
 '__init__',
 '__module__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'add_radius',
 'color',
 'drawCircle',
 'radius']
```

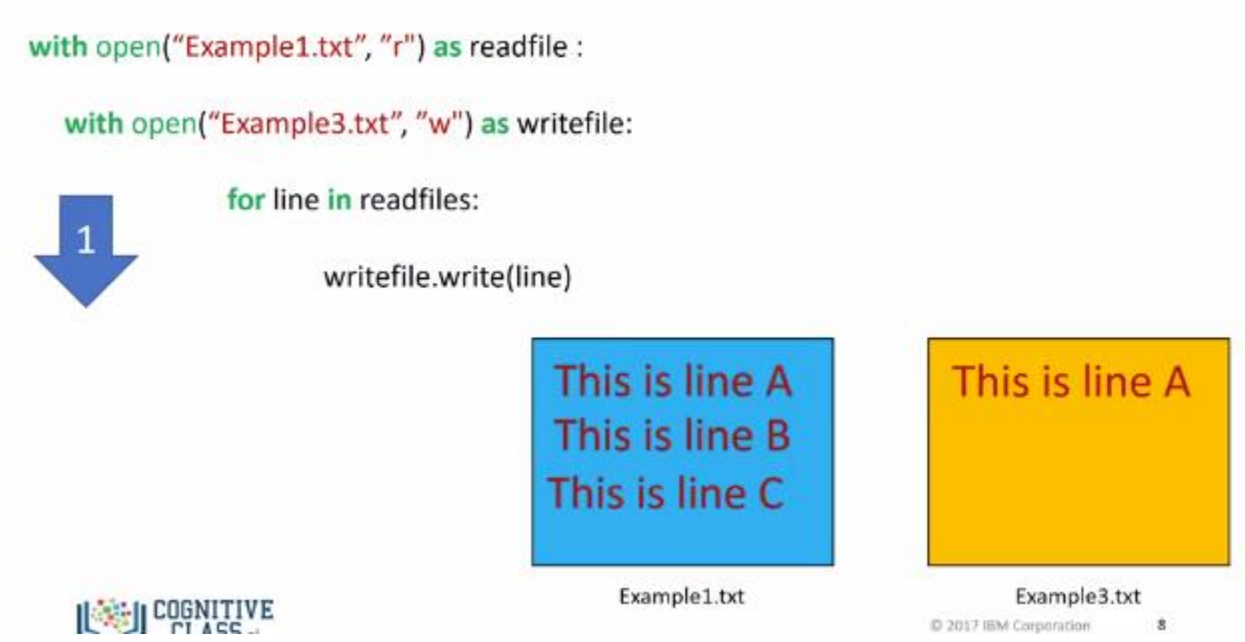
WEEK 4:



```
Lines=["This is line A\n","This is line B\n","This is line C\n"]
with open("/resources/data/Example2.txt", "w") as File1:

    for line in Lines:
        File1.write(line)
```

```
This is line A
This is line B
This is line C
```

Example2.txt



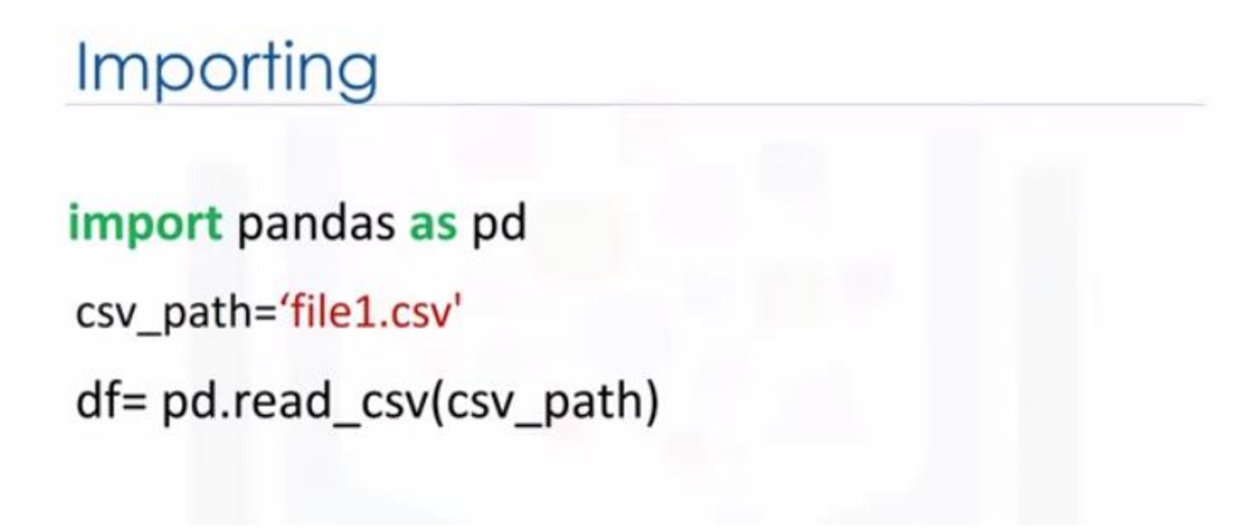
```
with open("Example1.txt", "r") as readfile :

   with open("Example3.txt", "w") as writefile:

          for line in readfiles:

                 writefile.write(line)
```

```
This is line A
This is line B
This is line C
```

Example1.txt

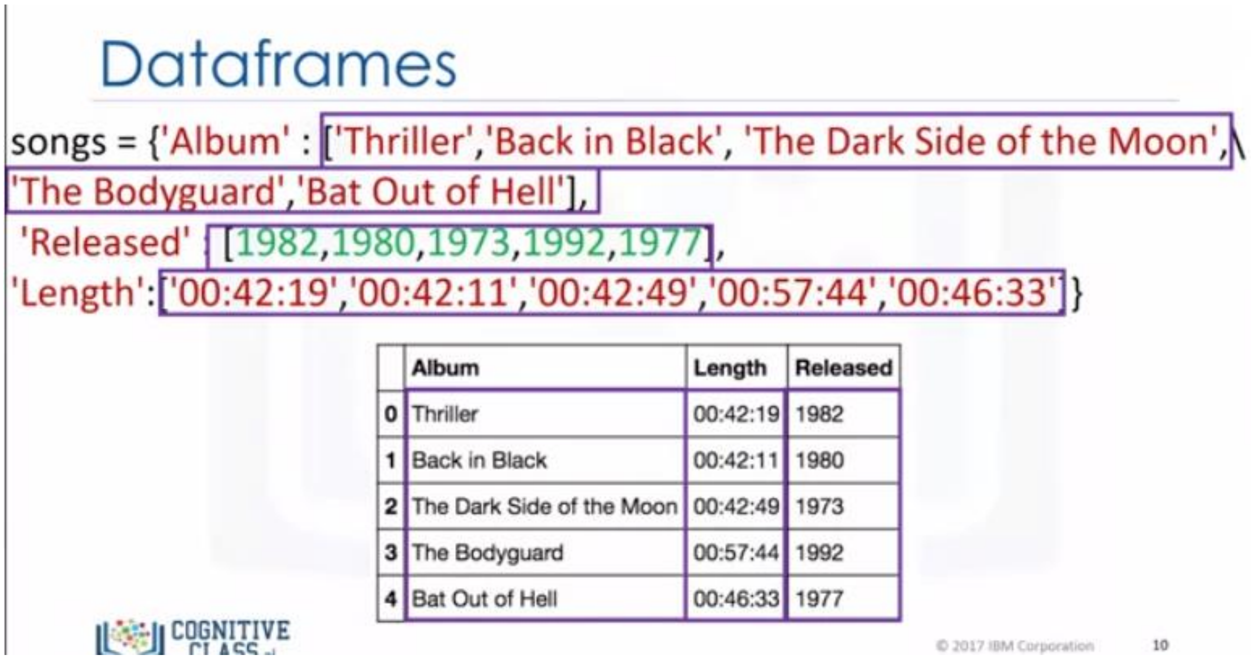
```
This is line A
```

Example3.txt



Importing

```
import pandas as pd
csv_path='file1.csv'
df= pd.read_csv(csv_path)
```



Dataframes

```
songs = {'Album' : ['Thriller','Back in Black', 'The Dark Side of the Moon',\
'The Bodyguard','Bat Out of Hell'],
 'Released' : [1982,1980,1973,1992,1977],
'Length':['00:42:19','00:42:11','00:42:49','00:57:44','00:46:33']}
```

	Album	Length	Released
0	Thriller	00:42:19	1982
1	Back in Black	00:42:11	1980
2	The Dark Side of the Moon	00:42:49	1973
3	The Bodyguard	00:57:44	1992
4	Bat Out of Hell	00:46:33	1977