Dictionarys

- dictionarys are kind of key value pair
- it is similar to list but it has pair of values
- we define the data in the form of {}
- we can use any kind of data in the Dictionarys

In [1]:
```
people={'abhi':32,'anu':'bro'}
```

In [5]:
```
print(people['anu'])
```

```
bro
```

Dictionary functions

In [6]:
```
numbers={1:'anu',2:'abhi',3:'rameshbabu',4:'sridevi'}
```

In [8]:
```
print(5 in numbers)
```

```
False
```

get meathod in Dict it is actually a function to extract the value

In [9]:
```
print(numbers.get(5))
```

```
None
```

In [11]:
```
print(numbers.get(2,'key not found'))
```

```
abhi
```

Tuples are same as list but it is immutable

- we cannot change the data which is present in the list
- in this we use ()

In [12]:
```
fruits=('apple','mango','peach')
```

In [13]:
```
fruits
```

Out[13]:
```
('apple', 'mango', 'peach')
```

In [15]:
```
print(fruits[1])
```

```
mango
```

we cannot assign a value to the tuple

In [16]:
```
fruits[0]='banana'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-2a4b43d8c35b> in <module>
----> 1 fruits[0]='banana'

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

in tuples we does not need to give () to display the elememts

In [17]:
```
a='anu','abu','john'
```

In [18]:
```
a
```

Out[18]:
```
('anu', 'abu', 'john')
```

In [19]:
```
type(a)
```

Out[19]:
```
tuple
```

In [20]:
```
print(a[1])
```

```
abu
```

List slicing

- it is used in the case if we want only certain amount of data from the list then we use list slicing

- we can slice the index in many ways

In [21]:
```
num=[0,100,200,300,400,500,600]
```

In [22]:
```
print(num[1:4])
```

```
[100, 200, 300]
```

In [23]:
```
print(num[2:6])
```

```
[200, 300, 400, 500]
```

In [24]:
```
print(num[:3])
```

```
[0, 100, 200]
```

In [25]:
```
print(num[3:])
```

```
[300, 400, 500, 600]
```

we can jump the data as our wish like

In [26]:
```
print(num[1:6:2])
```

```
[100, 300, 500]
```

- in this case we jumped the data into 2 steps

List Comprehenstion

- it is nothing but creating a list by creating certain set of rules
- it saves our time and create list by itself
- we can use for and if conditions in the LC also

In [1]:
```
li=[x**2 for x in range(6)]
```

In [2]:
```
li
```

Out[2]:
```
[0, 1, 4, 9, 16, 25]
```

to print only even numbers we use if condition in the LC

In [3]:
```
l=[x**2 for x in range(10) if x**2 %2==0]
```

In [4]:
```
l
```

Out[4]:
```
[0, 4, 16, 36, 64]
```

- This is the way to create

String Formating

- it is nothing but to embed string with number
- to combine the string with non string
- the string formating will be mainly useful when we use the format of date

In [10]:
```
num=[14,4,94]
newstring="date:{0}/{1}/{2}".format(num[0],num[1],num[2])
newstring
```

Out[10]:
```
'date:14/4/94'
```

we can do string formating in onothr way that is

In [19]:
```
a="{x} / {y}".format(x=100, y=200)
```

In [20]:
```
a
```

Out[20]:
```
'100 / 200'
```

String Functions

- join function is used to join each and every function in a list

In [22]:
```
print(":".join(['apple','banana','mango']))
```

```
apple:banana:mango
```

- replace function is used to replace some content in the data

In [23]:
```
a='hello sonu'
a.replace('sonu','abhi')
```

Out[23]:
```
'hello abhi'
```

- Starts with

In [29]:
```
a='Hello abhi '
print(a.startswith('hello'))
```

```
False
```

In [30]:
```
print(a.swapcase())
```

```
hELLO ABHI
```

In [31]:
```
print(a.casefold())
```

```
hello abhi
```

In [35]:
```
print(a.count('hello'))
```

```
0
```

In [36]:
```
print(a.endswith('abhi'))
```

```
False
```

In [37]:
```
print(a.upper())
```

```
HELLO ABHI
```

In [38]:
```
print(a.lower())
```

```
hello abhi
```

Numeric Functions

- min function
- max function
- abs function

In [39]:
```
print(min(1,2,3,4,5))
```

```
1
```

In [40]:
```
print(max(12,2,3,45,5))
```

```
45
```

In [42]:
```
print(abs(-127))
```

```
127
```

In [43]:
```
print(bytes(12))
```

```
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

In [ ]:
```

```