```
In [1]: # boolean

In [2]: False

Out[2]: False

In [3]: print(True, False)

        True False

In [4]: True

Out[4]: True

In [5]: type(True)

Out[5]: bool

In [6]: type(False)

Out[6]: bool

In [7]: my_str = 'Apeksha'

In [8]: my_str

Out[8]: 'Apeksha'

In [9]: my_str.istitle()

Out[9]: True

In [10]: type(my_str)

Out[10]: str

In [11]: a=10

In [12]: a

Out[12]: 10
```

In [13]:
```
b='apeksha doctor'
```

In [14]:
```
type(b)
```

Out[14]:
```
str
```

mathematical operations

In [15]:
```
a=10
b=20
```

In [16]:
```
print(a+b)
print(a-b)
print(a/b)
print(a*b)
```

```
30
-10
0.5
200
```

In [17]:
```
# various ways of printing
print('Hello')
```

```
Hello
```

In [18]:
```
first_name='Apeksha'
last_name='doctor'
```

In [19]:
```
print('my first name is {} and my last name is {}'.format(first_name,last_name))
```

```
my first name is Apeksha and my last name is doctor
```

In [20]:
```
print('my first name is {a} and my last name is {b}'.format(a=first_name,b=last_name))
```

```
my first name is Apeksha and my last name is doctor
```

In [21]:
```
len('Apeksha')
# will give you the length of thr string
```

Out[21]:
```
7
```

```
In [22]: type([1,2,2,3,4])

Out[22]: list

In [23]: bool()

Out[23]: False

In [24]: my_str='Apeksha Doctor'

In [25]: my_str.isalnum()

Out[25]: False

In [26]: my_str.isalpha()

Out[26]: False

In [27]: my_str.isdigit()

Out[27]: False

In [28]: my_str.istitle()

Out[28]: True

In [29]: my_str.upper()

Out[29]: 'APEKSHA DOCTOR'

In [30]: my_str.lower()

Out[30]: 'apeksha doctor'

In [31]: my_str.isspace()

Out[31]: False

In [32]: my_str1='apeksha 1'

In [33]: my_str.isdigit()

Out[33]: False
```

```
In [34]: str_1='Hello World'
         my_str='Apeksha Doctor'

In [35]: my_str.isalpha() or str_1.isnumeric()

Out[35]: False

In [36]: # list
         type([])

Out[36]: list

In [37]: test=[]

In [38]: type(test)

Out[38]: list

In [39]: test=['maths','science','chemistry',100,200,300]

In [40]: test

Out[40]: ['maths', 'science', 'chemistry', 100, 200, 300]

In [41]: type(test)

Out[41]: list

In [42]: len(test)

Out[42]: 6

In [43]: test.append('Apeksha')

In [44]: test

Out[44]: ['maths', 'science', 'chemistry', 100, 200, 300, 'Apeksha']

In [45]: #indexing
         test[2]

Out[45]: 'chemistry'
```

```
In [46]: test[3]

Out[46]: 100

In [47]: #indexing range of values
         test[:]

Out[47]: ['maths', 'science', 'chemistry', 100, 200, 300, 'Apeksha']

In [48]: # if want to select all the values after chemistry.

         test[2:]

Out[48]: ['chemistry', 100, 200, 300, 'Apeksha']

In [49]: # if you want to filter only chemistry.
         test[2]

Out[49]: 'chemistry'

In [50]: # if want the values till 3rd values.
         test[:3]

Out[50]: ['maths', 'science', 'chemistry']

In [51]: # wan to select values between 2 and 5 th position
         test[2:5]

Out[51]: ['chemistry', 100, 200]

In [52]: # if want to add more values in the string

         test.append(['john','bala'])

In [53]: test

Out[53]: ['maths', 'science', 'chemistry', 100, 200, 300, 'Apeksha', ['john', 'bala']]

In [54]: test

Out[54]: ['maths', 'science', 'chemistry', 100, 200, 300, 'Apeksha', ['john', 'bala']]
```

In [55]:
```
# insert when you want to add at specific order

test.insert(2,'shivom')
```

In [56]:
```
test
```

Out[56]:
```
['maths',
 'science',
 'shivom',
 'chemistry',
 100,
 200,
 300,
 'Apeksha',
 ['john', 'bala']]
```

In [57]:
```
# extend : if want to add more values.
lst= [1,2,3,4,5,6]
```

In [58]:
```
lst
```

Out[58]:
```
[1, 2, 3, 4, 5, 6]
```

In [59]:
```
lst.extend([7,8])
```

In [60]:
```
lst
```

Out[60]:
```
[1, 2, 3, 4, 5, 6, 7, 8]
```

In [61]:
```
sum(lst)
```

Out[61]:
```
36
```

In [62]:
```
# pop remove the last the value

lst.pop()
```

Out[62]:
```
8
```

In [63]:
```
lst.pop(3)
```

Out[63]:
```
4
```

```
In [64]: lst
Out[64]: [1, 2, 3, 5, 6, 7]
```

```
In [ ]: lst
```

```
In [ ]: lst*2
```

data Structures

```
In [ ]: #sets
        set_var=set()
        print(set_var)
```

```
In [ ]: print(type(set_var))
```

```
In [ ]: a1 ={1,2,2,3}
```

```
In [ ]: set_var(a1)
```

```
In [ ]: a1
```

```
In [ ]: a2 = {'a','b','b','c'}
```

```
In [ ]: # set will remove duplicates and set starts with {} brakets
        a2
```

```
In [ ]: test = {'avengers','ironman','ironman','hitman'}
```

```
In [ ]: test
```

```
In [ ]: # indexing in set, cant find indexing in sets
        test[1]
```

```
In [ ]: test['avengers']
In [ ]: # for addition in sets
        test.add('Hulk')
```

```
In [ ]: test
```

```
In [ ]: set1 = {'Hulk', 'avengers', 'hitman', 'ironman'}
        set2 = {'Hulk', 'avengers', 'hitman'}
```

```
In [ ]: set1.difference(set2)
```

```
In [ ]: set1
```

```
In [ ]: # differance update
        set1.difference_update(set2)
```

```
In [ ]: set2
```

```
In [ ]: set1
```

```
In [ ]: # to check the common elements
        set1.intersection(set2)
```

```
In [ ]: set.difference
```

```
In [ ]: # dictionary
        # it starts with {}, sets also starts with {} but in sets we use values like{1,2,3,4}
        # but in dictionary we use key values
```

```
In [ ]: apeksha={"car1":"Audi","car2":"BMW"}
```

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

```
In [ ]: a1={1,2,3,4}
```

```
In [ ]: type(a1)
```

```
In [ ]: # access the data from dictionary
        # indexing will be your key names in dictionary
        apeksha["car2"]
```

In [ ]:
```
# for loops
for x in apeksha:
    print(x)
```

In [ ]:
```
# dictionary values
for x in apeksha.values():
    print(x)
```

In [ ]:
```
# both key and values, we cn use items
for x in apeksha.items():
    print(x)
```

In [ ]:
```
# adding iteams in dictionary
apeksha["car3"]='mazda'
```

In [ ]:
```
apeksha
```

In [ ]:
```
# nested dictionary
```

In [ ]:
```
car1_model={'mazda':1990}
car2_model={'mazda5':1991}
car3_model={'jeep': 1984}

car_type={'car1':car1_model, 'car2':car2_model, 'car3':car3_model}
```

In [ ]:
```
print(car_type)
```

In [ ]:
```
# indexing
print(car_type['car2'])
```

In [ ]:
```
print(car_type['car2']['mazda5'])
```

In [ ]:
```
# Tuples not mutable, means tuple cant be change once it created
# for tupel use()
```

In [ ]:
```
test1 = ('apeksha','nilesh','shivom')
```

In [ ]:
```
test1
```

In [ ]:
```
test1[2]
```

In [ ]:
```
test1[0]
```

In [ ]:
```
# tuple can chenge completely but not partially
test1 =('a','b')
```

In [ ]:
```
test1
```

In [ ]:
```
test1=('a','nilesh','shivom')
```

In [ ]:
```
test1
```

In [ ]:
```
# Numpy
import numpy as np
```

In [ ]:
```
my_lst=[1,2,3,4,5]
```

In [ ]:
```
arr=np.array(my_lst)
```

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

In [ ]:
```
# shape for how many rows and columns if you have 2D array here we have one dimetion array
arr.shape
```

In [ ]:
```
# multinested array
my_lst1=[1,2,3,4,5]
my_lst2=[2,3,4,5,6]
my_lst3=[3,4,5,6,7]
```

In [ ]:
```
a=np.array([my_lst1,my_lst2,my_lst3])
```

In [ ]:
```
a
```

In [ ]:
```
a.shape
```

In [ ]:
```
a.reshape(5,3)
```

indexing

In [ ]:
```
b =np.array([1,2,3,4,5,6,7,8,9])
```

In [ ]:
```
b[3]
```

In [ ]:
```
b[0]
```

In [ ]:
```
a
```

In [ ]:
```
a[0:2,]
```

In [ ]:
```
a[1:3,3:]
```

In [ ]:
```
a[1,1:3]
```

In [ ]:
```
# inbuilt function
```

In [ ]:
```
a1=np.arange(0,10)
```

In [ ]:
```
a1
```

In [ ]:
```
a1=np.arange(0,10,step=4)
```

In [ ]:
```
a1
```

In [ ]:
```
a1=np.arange(0,10,step=2)
```

In [ ]:
```
a1
```

In [ ]:
```
np.linspace(1,10,50)
```

In [ ]:
```
# copy function and broadcasting
a1
```

In [ ]:
```
a2=a1
```

In [ ]:
```
a2[3:]=200
```

```
In [ ]: a2
```

```
In [ ]: a2[3:]=500
```

```
In [ ]: a2
```

```
In [ ]: a2
```

```
In [ ]: # to prevent this we have copy function
```

```
In [ ]: a2=a1.copy()
```

```
In [ ]: print(a1)
        a2[3:]=300
        print(a2)
```

```
In [ ]: # some conditions very useful for data exploratory analysis
```

```
In [ ]: val=2
        a1>2
```

```
In [ ]: a2[a1<2]
```

```
In [ ]: # random distribution
        np.random.rand(3,3)
```

```
In [ ]: t=np.random.rand(4,4)
```

```
In [ ]: t
```

```
In [ ]: s=np.random.randint(1,100,10)
```

```
In [ ]: s
```

```
In [ ]: s=np.random.randint(1,100,10).reshape(5,2)
```

```
In [ ]: s
```

In [ ]:
```
# Pandas
```

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
a=pd.DataFrame(np.arange(0,20).reshape(5,4),index=['row1','row2','row3','row4','row5'],columns=['col1','col2','col3','c
ol4'])
```

In [ ]:
```
a
```

In [ ]:
```
a.to_csv('a.csv')
```

In [ ]:
```
# accessing th elements
a.loc['row1']
```

In [ ]:
```
type(a.loc['row1'])
```

In [ ]:
```
a.iloc[:,:]
```

In [ ]:
```
a.iloc[1:3,1:3]
```

In [ ]:
```
a.iloc[2:4,2:]
```

In [ ]:
```
type(a.iloc[2:4,2:])
```

In [ ]:
```
# DataFrame can convert in to array
a.iloc[:,1:].values
```

In [ ]:
```
# how to check the null conditions
a.isnull()
```

In [ ]:
```
a.isnull().sum()
```

In [ ]:
```
# to check the value or duplicate in the columns
a['col1'].value_counts()
```

In [ ]:
```
# if want to filter multiple columns
a[['col1','col2','col3']]
```

```
In [ ]: import pandas as pd
```

```
In [ ]: import numpy as np
```

```
In [ ]: df=pd.read_csv('mercedesbenz.csv')
```

''' Function in pandas '''

```
In [ ]: num=24

        if num%2==0:
            print('The num is even')
        else:
            print('The num is odd')
```

```
In [ ]: def even_oddfun(num):
            if num%2==0:
                print('The num is even')
            else:
                print('The num is odd')
```

```
In [ ]: even_oddfun(45)
```

```
In [ ]: even_oddfun(12)
```

```
In [ ]: # print vs return
        def hello_world():
            print('Hello world')
```

```
In [ ]: hello_world()
```

```
In [ ]: val=hello_world()
```

```
In [67]: print('val')
```

```
val
```

```
In [ ]: def hello_world1():
            return('Hello world')
```

```
In [ ]: hello_world1()
```

```
In [ ]: a=hello_world1()
```

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

Lambda function

```
In [ ]: def addition(a,b):
            return a+b
```

```
In [ ]: addition(4,5)
```

```
In [ ]: addition(10,100)
```

```
In [ ]: lambda a,b:a+b
```

```
In [ ]: # store in some variable name
        addition= lambda a,b:a+b
```

```
In [ ]: addition(12,13)
```

```
In [ ]: def even(num):
            if num%2==0:
                return True
```

```
In [ ]: even(12)
```

```
In [ ]: even(15)
```

```
In [ ]: even1=lambda a:a%2==0
```

```
In [ ]: even1(13)
In [ ]: even(14)
```

```
In [ ]: # 3 parameters
        def addition(x,y,z):
            return x+y+z
```

```
In [ ]: addition(2,4,5)
```

```
In [ ]: # convert to lambda function
        w=lambda x,y,z:x+y+z
```

```
In [ ]: w(10,20,30)
```

Map function

```
In [ ]: def even_or_odd(num):
            if num%2==0:
                return 'even'
            else:
                return 'odd'
```

```
In [ ]: even_or_odd(25)
```

```
In [ ]: def even_odd(num):
            if num%2==0:
                return'The number {} is even'.format(num)
            else:
                return'The number {} is odd'.format(num)
```

```
In [ ]: even_odd(12)
```

```
In [ ]: even_odd(15)
```

```
In [ ]: # if you have to apply for multiple numbers
```

```
In [ ]: A = [1,2,3,4,5,6,7,45,34,23,67]
In [ ]: map(even_odd,A)
```

```
In [ ]: # now convert to the list for the result
        list(map(even_odd,A))
```

Filter function

```
In [ ]: def even(num):
            if num%2==0:
                return True
```

```
In [ ]: lst=[1,2,4,5,7,35,65,76,87]
```

```
In [ ]: list(filter(even,lst))
```

```
In [ ]: list(filter(lambda num: num%2==0,lst))
```

```
In [ ]: list(map(lambda num:num%2==0,lst))
```

List Comprehension

```
In [ ]: lst1=[]
        def lst_square(lst):
            for i in lst:
                lst1.append(i*i)
            return lst1
```

```
In [ ]: lst_square([1,2,3,4,5,6])
```

with list comprehension we can wright the upper code in one line

```
In [ ]: lst=[1,2,3,4,5,6]
```

```
In [ ]: [i*i for i in lst]
```

In [ ]:
```
lst1=[i*i for i in lst if i%2==0]
```

In [ ]:
```
lst1
```

string formatting

In [ ]:
```
print('Hello world')
```

In [ ]:
```
def greeting(name):
    return "Hello {}.Welcome to thr community".format(name)
```

In [ ]:
```
greeting('Shiv')
```

In [ ]:
```
def Welcome_email(firstname,lastname):
    return 'welcome {}. please welcome {}'.format(firstname,lastname)
```

In [ ]:
```
Welcome_email('shiv','nilesh')
```

list iterabels vs iterators

In [ ]:
```
lst=[1,2,3,4,5,6,7]
for i in lst:
    print(i)
```

In [ ]:
```
# what is iterators
iter(lst)
```