Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [2]:

```
"""5 Years+ Experience in Software Development,Python Programmer,Genuinely Working on Machine Lear
ning /Deep Learning Projects
Call: 8237180203
Email Id: abhaychougule01@gmail.com"""
```

Out[2]:

```
'5 Years+ Experience in Software Development,Python Programmer,Genuinely Working on Machine
Learning /Deep Learning Projects \nCall: 8237180203\nEmail Id: abhaychougule01@gmail.com'
```

In [ ]:

```
print('python')
```

In [ ]:

```
print("python")
```

In [ ]:

```
print("""python""")
```

In [ ]:

```
a=10
b=20.10
c="abhay"


#c=a+b
print(c)
print(a+b)
```

In [ ]:

```
a=20
b=30
```

In [ ]:

```
c=a+b
print(c)
```

In [ ]:

```
r=20
f=30
```

In [ ]:

```
z=r+f
print(z)
```

In [ ]:

```
counter=10
miles=20

print("counter",counter)
```

In [ ]:

```
print('miles',miles)
```

In [ ]:

```
city="Pune"
print("city")
print(city)
```

In [ ]:

```
a=b=c=1
print(c)
```

In [ ]:

```
str="IlovemyIndia"

#print(str)

#print(str[0])

#print(str[1:7])1==========7

#print(str[6:7])

#print(str[1:-3])

#print(str[4:-5])


#print(str + "   Rack")


0,1,2,3,4,5
```

In [ ]:

```
a="msdhoni"

print(a[-6:-3])

#sdh
```

In [ ]:

```
list=[2,3,4,5,6,"Rack","Sachin"]
print(list)
```

In [ ]:

```
print(list[1:3])
```

In [ ]:

```
print(list[3:6])
```

In [ ]:

```
print(list[:])
```

Basic Operator #Python Programming by Abhay Chougule (5+ Yrs of

Exp.)

In [ ]:

```
### Arithmetic Operatoers
```

In [ ]:

```
a=20
b=30
c=a+b
```

In [ ]:

```
print(a+b)
print(c)
```

In [ ]:

```
print(a-b)
print(c)
```

In [ ]:

```
### Comparison Operatoers
```

In [ ]:

```
a=20
b=30
c=a+b

#a==b
#a!=b
#a>b
#a<b
a>=b
```

In [ ]:

```
### Assignment Operatoers
c=a+b
c
print(c)
```

String #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [ ]:

```
print('Hello')
print("Hello")
print("""Abhay""")
```

In [ ]:

```
a = "Hello"
print(a)
```

In [ ]:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

In [ ]:

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

In [ ]:

```
a = "HelloWorld!"
print(a[8:6])
```

In [ ]:

```
print(a[5])
```

In [ ]:

```
b = "HelloWorld!"
print(b[2:7])
```

In [ ]:

```
b = "HelloWorld!;'/.,'"
print(b[-6])
```

In [ ]:

```
a = "Hello World!"
print(len(a))
```

In [ ]:

```
a = " Hello World!             "
print(a)
#print(a.strip())
```

In [ ]:

```
a ="Hello WORLD!"
print(a.lower())
```

In [ ]:

```
a = "Hello, World!"
print(a.upper())
```

In [ ]:

```
a = "1,2,3"
print(a.upper())
```

In [ ]:

```
a = "Hello World!"
print(a.replace("H", "J"))
print(a.replace("Hello", "My"))
```

In [ ]:

```
txt = "The rain in Spain stays mainly in the plain"
x = "in Spain" in txt
print(x)
```

Python Decision Making #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [ ]:

```
var=150
if(var<=100):
    print("The value is 100")
else:
    print("Hey Everyone")
```

In [ ]:

```
a=20
b=10
if(a>b):
    print("Its True")
else:
    print("Its False")

print("Now i am not happy")
```

Python Loops Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [ ]:

```
fruits = ["apple", "banana", "cherry"] #list
for x in fruits:
  print(x)
```

In [ ]:

```
for x in "banana": #With String
  print(x)
```

In [ ]:

```
fruits = ["apple", "banana", "cherry"]  #with Break Statement
for x in fruits:
  print(x)
  if x == "cherry":
    break
```

In [ ]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

In [ ]:

```
fruits = ["apple", "banana", "cherry"]  #Continue Statement
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

In [ ]:

```
for x in range(10):  # Range Function
  print(x)
```

In [ ]:

```
for x in range(2, 6):
  print(x)
```

In [ ]:

```
for x in range(3, 10, 3):
  print(x)
```

Python Collections (Arrays) Python Programming by Abhay Chougule (5+ Yrs of Exp.)

List : is a collection which is ordered and changeable. Allows duplicate members.

Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

In [ ]:

```
#list
#A list is a collection which is ordered and changeable. In Python lists are written with square b
rackets.
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
print(mylist)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(mylist[4:6])
```

In [ ]:

```
mylist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(mylist[2:])
```

In [ ]:

```
mylist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(mylist[0:-1])
```

In [ ]:

```
mylist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(mylist[:4])
```

In [ ]:

```
mylist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(mylist[2:4])
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
mylist[1] = "blackcurrant"
print(mylist)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
for x in "banana":
  print(x)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
mylist.append("orange")
print(mylist)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
mylist.append("apple")
print(mylist)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
mylist.insert(1, "orange")
print(mylist)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
mylist.remove("banana")
print(mylist)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
mylist.pop()
print(mylist)
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
del mylist[0]
print(mylist)
#mylist
```

In [ ]:

```
mylist = ["apple", "banana", "cherry"]
del mylist
```

In [ ]:

```
print(mylist)
```

Tuples #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

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

In [ ]:

```
#tuples

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

mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru")
print(mylist)
print(type(mylist))
```

In [ ]:

```
mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru")
print(mylist[1])
```

In [ ]:

```
mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru")
print(mylist[2:-4])
```

In [ ]:

```
mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru")
print(mylist[4:5])
```

In [ ]:

```
mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi")
print(mylist[5:-2])
```

In [ ]:

```
x = ("apple", "banana", "cherry")   # Convert the tuple into a list to be able to change it:
y = list(x)

y[2] = "kiwi" #update with banana

x = tuple(y)

print(y)
print(x)
```

In [ ]:

```
a=(1005,-14)
b=(61654655,0)

if(a>b):
    print("a is bigger")
else:
    print("b is bigger")
```

In [ ]:

```
mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru")
for x in mylist:
  print(x)
```

In [ ]:

```
mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru")
print(len(mylist))
```

In [ ]:

```
mylist = ("abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru")
mylist.pop()
#print(x)
print(mylist)                                 #this will raise an error because the tuple no longer
xists
```

In [ ]:

```
#Joining 2 tuples

tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
```

```
tuple3 = tuple1 + tuple2
print(tuple3)
```

In [ ]:

```
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
print(thistuple)
```

sets #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

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

In [ ]:

```
myset = {"abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru"}
print(myset)
```

In [ ]:

```
myset = {"abhay", "amar", "ankita","sagar","rahul","mahi","pooja","viru"}
for x in "mahi":
    print(x)



#for(int i=1;i<n;i++)
```

In [ ]:

```
myset = {"abhay","sagar","rahul","pooja","Mahi","viru"}

print("pooja" in myset)
```

In [ ]:

```
myset = {"abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru"} #Add an item to a set

myset.add("orange")
myset.add("Mangos")
myset.add("orange")
myset.add("Bluee")
print(myset)
```

In [ ]:

```
myset = {"abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru"} #Add multiple items to a
set,

myset.update(["orange", "mango", "grapes"])

print(myset)
```

In [ ]:

```
myset = {"abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru"}

print(len(myset))
```

In [ ]:

```
myset = {"abhay","ankita","sagar","rahul","pooja","Mahi","viru"}

myset.remove("a")
```

```
print(myset)
```

In [ ]:

```
myset = {"abhay","ankita","sagar","rahul","pooja","Mahi","viru"}

myset.discard("a")

print(myset)
```

In [ ]:

```
myset = {"pooja","Mahi","viru","AMAR","adi","Ankita","rahul","Krishna"}

x = myset.pop()

print(x)

print(myset)
```

In [ ]:

```
myset = {"abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru"}

myset.clear()

print(myset)
```

In [ ]:

```
set1 = {"abhay", "amar", "ankita","sagar","rahul","pooja","Mahi","viru"}

set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

In [ ]:

```
set1 = {"abhay", "Amar", "Ankita"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

In [ ]:

```
a=set()
b={}
c=[]

print(type(a))
print(type(b))
print(type(c))
```

In [ ]:

```
"""Method Description
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
issubset() Returns whether another set contains this set or not
issuperset() Returns whether this set contains another set or not
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
"""
```

dictionary Python Programming by Abhay Chougule (5+ Yrs of Exp.)

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

In [1]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopper",
  "rack":"Python",
  "year": 2020
}
print(mydict)
```

```
{'Abhay': 'Developer', 'Amar': 'Shopper', 'rack': 'Python', 'year': 2020}
```

In [2]:

```
x = mydict["rack"]  # Find the value
x
```

Out[2]:

```
'Python'
```

In [3]:

```
x = mydict.get("Amar")
x
```

Out[3]:

```
'Shopper'
```

In [4]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}
mydict["year"] = 1921
```

In [5]:

```
print(mydict)
```

```
{'Abhay': 'Developer', 'Amar': 'Shopkeeper', 'year': 1921}
```

In [6]:

```
for x in mydict:
  print(x)
```

```
Abhay
Amar
year
```

In [7]:

```
for x in mydict:
  print(mydict[x])
```

```
Developer
Shopkeeper
1921
```

In [8]:

```
for x in mydict.values():
  print(x)
```

```
Developer
Shopkeeper
1921
```

In [9]:

```
for x, y in mydict.items():
  print(x, y)
```

```
Abhay Developer
Amar Shopkeeper
year 1921
```

In [10]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}

if "Abhay" in mydict:
  print("Yes,  in dictionary")
else:
  print("Yes not  in dictionary")
```

```
Yes,  in dictionary
```

In [11]:

```
print(len(mydict))
```

```
3
```

In [12]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}
mydict["color"] = "red"
print(mydict)
```

```
{'Abhay': 'Developer', 'Amar': 'Shopkeeper', 'year': 2020, 'color': 'red'}
```

In [13]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}
mydict.pop("Abhay")
print(mydict)
```

```
{'Amar': 'Shopkeeper', 'year': 2020}
```

In [14]:

```
mydict = {
  "111": "333",
  "": "",
  "year": 2020,
   "rack":5555
}
x=mydict.popitem()
print(x)
print(mydict)
```

```
('rack', 5555)
{'111': '333', '': '', 'year': 2020}
```

In [15]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}
del mydict["Amar"]
print(mydict)
```

```
{'Abhay': 'Developer', 'year': 2020}
```

In [16]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}
mydict.clear()
print(mydict)
```

```
{}
```

In [17]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}
mydict1 = mydict.copy()
print(mydict1)
```

```
{'Abhay': 'Developer', 'Amar': 'Shopkeeper', 'year': 2020}
```

In [18]:

```
mydict = {
  "Abhay": "Developer",
  "Amar": "Shopkeeper",
  "year": 2020
}
mydict1 = dict(mydict)
print(mydict1)
```

```
{'Abhay': 'Developer', 'Amar': 'Shopkeeper', 'year': 2020}
```

Method Description

clear() Removes all the elements from the dictionary

copy() Returns a copy of the dictionary

fromkeys() Returns a dictionary with the specified keys and value

get() Returns the value of the specified key

items() Returns a list containing a tuple for each key value pair

keys() Returns a list containing the dictionary's keys

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

values() Returns a list of all the values in the dictionary

Python If ... Else # Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [ ]:

```
""" Equals: a == b
Not Equals: a != b
Less than: a < b
Less than or equal to: a <= b
Greater than: a > b
Greater than or equal to: a >= b """
```

In [ ]:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

In [ ]:

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

In [ ]:

```
a = 33
b = 300
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

In [ ]:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a < b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

In [ ]:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

In [ ]:

```
if a > b: print("a is greater than b")
```

In [ ]:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

In [ ]:

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

In [ ]:

```
a = 200
b = 3030
c = 500
if c > b and b > a:
  print("Both conditions are True")
else:
  print("Its totally wrong")
```

In [ ]:

```
a = 200
b = 33
c = 500
if a > b and a > c:
  print("At least one of the conditions is True")
else:
  print("Its totally wrong")
```

In [19]:

```
x = 100

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
else:
  print("but not above 20.")
```

```
Above ten,
and also above 20!
```

Python While Loops #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [ ]:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

In [ ]:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

In [ ]:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

In [ ]:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

In [ ]:

Python For Loops Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [20]:

```
fruits = ["apple", "banana", "cherry"]
for x in "apple":
  print(x)
```

```
a
p
p
l
e
```

In [21]:

```
for x in fruits:
  print(x)
```

```
apple
```

```
banana
cherry
```

In [22]:

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
banana
```

In [ ]:

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

In [ ]:

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

In [ ]:

```python
for x in range(6):
  print(x)
```

In [ ]:

```python
for x in range(2, 6):
  print(x)
```

In [ ]:

```python
for x in range(2, 30, 3):
  print(x)
```

In [ ]:

```python
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

In [ ]:

```python
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in "apple":
    print(x, y)
```

Python Arrays #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

Arrays are used to store multiple values in one single variable:

In [25]:

```
"""car1 = "Ford"
car2 = "Volvo"
car3 = "BMW" """
```

Out[25]:

```
'car1 = "Ford"\ncar2 = "Volvo"\ncar3 = "BMW" '
```

In [26]:

```
cars = ["Ford", "Volvo", "BMW"]
```

In [29]:

```
x = cars[2]


x
```

Out[29]:

```
'BMW'
```

In [30]:

```
cars[0] = "Toyota"
```

In [31]:

```
cars
```

Out[31]:

```
['Toyota', 'Volvo', 'BMW']
```

In [33]:

```
x = len(cars)

x
```

Out[33]:

```
3
```

In [41]:

```
cars.append("Honda")


cars
```

Out[41]:

```
['Toyota',
 'Volvo',
 'BMW',
 'Honda',
 'Honda',
 'Kiya',
 'Kia',
 'Honda',
 'Honda',
 'Kia',
 'Honda']
```

In [54]:

```
cars.append("Kia")


cars
```

Out[54]:

```
['Toyota', 'Kia']
```

In [55]:

```
p=cars.pop(1)

print(p)

print(cars)
```

```
Kia
['Toyota']
```

In [57]:

```
cars.remove("Toyota")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-57-b640feaf1199> in <module>
----> 1 cars.remove("Toyota")

ValueError: list.remove(x): x not in list
```

In [ ]:

```
"""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
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
pop() Removes the element at the specified position
remove() Removes the first item with the specified value
reverse() Reverses the order of the list
sort() Sorts the list"""
```

Python Lambda #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

In [59]:

```
x = lambda a : a / 200
print(x(5))
```

```
0.025
```

In [60]:

```
x = lambda a, b : a * b
print(x(5, 6))
```

```
30
```

In [61]:

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

```
13
```

In [62]:

```
def myfunc(n):
  return lambda a : a * n
```

In [64]:

```
def myfunc(n):
  return lambda a : a * n

"""x = lambda a : a / 200
print(x(5))"""

x = myfunc(2)

print(x(11))
```

```
22
```

In [65]:

```
def myfunc(n):
  return lambda a : a * n

x = myfunc(2)
y = myfunc(3)

print(x(11))
print(y(11))
```

```
22
33
```

In [ ]:

```

```

Python Functions #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

In [66]:

```
def rack():
  print("Hello from a function")

rack()
```

```
Hello from a function
```

In [69]:

```
def rack():
  print("a function")

rack()

def racksonsit():
  print("a Racksonsit developers")

#rack()
racksonsit()
```

```
a function
a Racksonsit developers
```

In [70]:

```
def rack(fname):
  print(fname + " Developers")

rack("Software")
rack("Machine")
rack("Network")
```

```
Software Developers
Machine Developers
Network Developers
```

In [74]:

```
def rack(fname,lname):
    print(fname + "                                         " + lname)

rack("                                       Abhay","                           Chougule")
```

```
                                       Abhay
Chougule
```

In [76]:

```
def rack(c1,c2,c3):
    print("Who is the Younger:"+ c2)

rack(c1="Abhay", c2="Amar",c3="Ankita")
```

```
Who is the Younger:Amar
```

Class #Python Programming by Abhay Chougule (5+ Yrs of Exp.)

In [77]:

```
class SoftClass:
  x = 100


p1 = SoftClass()
print(p1.x)


#class A()    obj=new Class A();
#obj.rack(a)
```

```
100
```

To understand the meaning of classes we have to understand the built-in init() function.

All classes have a function called init(), which is always executed when the class is being initiated.

Use the init() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

In [82]:

```
class Maxclass:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Maxclass("Abhay Chougule", 28)

print(p1.name)
print(p1.age)
```

```
Abhay Chougule
28
```

In [81]:

```
class Maxclass:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)
  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Maxclass("John", 36)

#del p1.age
#print(p1.age)
#print(p1.name)


#print(p1.age)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-81-aa8bb4fa8107> in <module>
     10
     11 del p1.age
---> 12 print(p1.age)
     13 #print(p1.name)
     14

AttributeError: 'Maxclass' object has no attribute 'age'
```

In [ ]:

```
class emp:
    empc=0
    def __init__(self,name,salary):
        self.name=name
        self.salary=salary
        emp.empc +=1

    def displayC(self):
        print("Total Emp % d" %employee.empc)
    def displayEmp(self):
        print("Name:",self.name,"Salary:",self.salary)

emp1=emp("Amar",5000)
emp2=emp("Ankita",5500)

emp1.displayEmp()
emp2.displayEmp()
print("Total Emp %d"%emp.empc)
```

In [1]:

```
"""5 Years+ Experience in Software Development,Python Programmer,Genuinely Working on Machine Lear
ning /Deep Learning Projects
Call: 8237180203
Email Id: abhaychougule01@gmail.com"""
```

Out[1]:

```
'5 Years+ Experience in Software Development,Python Programmer,Genuinely Working on Machine
Learning /Deep Learning Projects \nCall: 8237180203\nEmail Id: abhaychougule01@gmail.com'
```

In [ ]:

```

```