Lambda functions

- Lambda functions represnts function concept
- But we can write in a single line
- Like list comprehension, lambda functions similar only
- It will decrease the time complexity
- always remember if we use many for loops or many conditions using multiple line , the time complexity will increase

In [ ]:
```
**It will use a keyword lambda**

# lambda <arguments>: <expression>
```

In [ ]:
```
What is the interview process for a data science position with 3 years of experi

1) coding+sql
2) based on ML DL
3) Techn: 2,3

bond : 1.5 bonus

ds2 === 2 2ppl
ds3 ====

do you know
```

In [ ]:
```
strings
list
tuple set
dictionary
landad
file handling

oops : kwargs

Sir do we have right to break bond with company
```

In [1]:
```
def summ(n):
    return(n+10)
summ(10)
```

Out[1]:
```
20
```

$Pattern - 1$

Function with only one argument

- function name
- argument name

- return output

```
In [ ]: # syntax: <function name>= lambda <argument name>: <return output>
        def summ(n):
            return(n+10)
        summ(10)

        # function name= summ
        # argument name = n
        # return output= n+10
```

```
In [4]: summ= lambda n:n+10
        summ(100)
```

```
Out[4]: 110
```

```
In [5]: def cube(n):
            return(n**3)
        cube(10)
```

```
Out[5]: 1000
```

```
In [7]: cube= lambda n:n**3
        cube(10)
```

```
Out[7]: 1000
```

pattern-2

Two arguments

```
In [8]: # syntax :  <function name>= lambda <arg1>,<arg2>: <return output>
        def add(a,b):
            return(a+b)
        add(50,50)

        # Function name: add
        # arg1=a
        # arg2=b
        # return= a+b
```

```
Out[8]: 100
```

```
In [10]: add=lambda n1,n2: n1+n2
         add(60,50)
```

```
Out[10]: 110
```

```
In [13]: average=lambda a,b,c:round((a+b+c)/3,2)
         average(10,202,30)
```

```
Out[13]: 80.67
```

Pattern-3

Default arguments

```
In [14]: average=lambda a,b,c=500:round((a+b+c)/3,2)
         average(10,202)

Out[14]: 237.33
```

Pattern-4

if-else

```
In [15]: def max(a,b):
             if a>b:
                 return(a)
             else:
                 return(b)
         max(10,20)

Out[15]: 20
```

```
In [17]: # syntax : function name = lambda <arg1>,<arg2>: <list comprehension>
         # syntax : function name = lambda <arg1>,<arg2>: <if_out><if_con><else><else_out
         maxx=lambda a,b: a if a>b else b
         maxx(30,20)

Out[17]: 30
```

Pattern-5

using List

```
In [18]: l=['hyd','chennai','mumbai']
         # op=['Hyd','Chennai','Mumbai']
         op=[]
         for i in l:
             op.append(i.capitalize())
         op

Out[18]: ['Hyd', 'Chennai', 'Mumbai']
```

```
In [ ]: lambda <variable>:<op>
        # variable:i
        # op: i.capitalize()
        lambda <variable>:<op>,<iterator>
        # Qn: from where you are getting 'i'
        # <iterator>: list
```

map

- the function and iterator are available now
- we need to map both

```
In [21]: l=['hyd','chennai','mumbai']
         lambda i: i.capitalize(),l

Out[21]: (<function __main__.<lambda>(i)>, ['hyd', 'chennai', 'mumbai'])
```

```
In [22]: l=['hyd','chennai','mumbai']
         map(lambda i: i.capitalize(),l)

Out[22]: <map at 0x227209e9cf0>
```

```
In [23]: # apply the list to see the values
         l=['hyd','chennai','mumbai']
         list(map(lambda i: i.capitalize(),l))

Out[23]: ['Hyd', 'Chennai', 'Mumbai']
```

```
In [24]: l=['hyd','chennai','mumbai']
         tuple(map(lambda i: i.capitalize(),l))

Out[24]: ('Hyd', 'Chennai', 'Mumbai')
```

```
In [ ]: # step1: Write your normal expression
        #        ex: lambda <var>: <op>===>lambda i: i.capitalize()
        # step2: add the iterator
        #        ex: lambda <var>: <op>,<list>===>lambda i: i.capitalize(),list1
        # Step-3: Map the both
        #        ex: map(lambda <var>: <op>,<list>)===>map(lambda i: i.capitalize(),list
        # Step-4: save the values in a list,
        #       ex: list(map(lambda <var>: <op>,<list>))===>list(map(lambda i: i.capital

        #Note: Those who are getting list object not callable use tuple
```