```
In [ ]:
```

argument

```
def add(x):
 return(x+10)
```

```
In [1]:
# Create basic
```

```
add(20)
```

function **one**

```
Out[1]: 30
```

```
return(summ)
```

```
In [2]:
def add(x):
 summ=x+10
```

```
add(20)
```

```
Out[2]: 30
```

```
add(20)
```

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

```
# how many arguemts are
present? : x # what you
are returning?: x+10

# format: lambda
<argument_name>:<output>


add=lambda x:x+10
add(20)
```

```
def add(x):
 return(x+10)
```

```
Out[3]: 30
```

```
def
square(x):
 return(x*x)
```

```
In [ ]: In
```

```
[4]:
```

```
square=lambda
a:a*a
square(5)
```

```
Out[4]: 25
```

```
return(x*x*x)
```

```
In [ ]: In
```

```
cube=lambda
```

```
[5]:
```

```
x:x*x*x

cube(30)
```

```
def cube(x):
Out[5]: 27000
```

```
 return(a+b)
```

In [ ]: In [6]:

```
add(20,30)
```

```
# how many arguemts are
present? : a,b # what
you are returning?: a+b

# format: lambda
<arg1>,<arg2>:<output>
```

```
add=lambda a,b:a+b
add(20,30)
```

two arguments

```
def add(a,b):
```

```
Out[6]: 50
```

In [7]:

```
# implement average of 3
numbers using lambda
```

```
avrg=lambda
a,b,c:(a+b+c)/3
avrg(5,5,5)
```

```
Out[7]: 5.0
```

In [10]:

```
# implemet avergae make c
as default parameter
```

```
avg=lambda a,b,c=30:
round((a+b+c)/3,2)
avg(20,30)
```

```
Out[10]: 26.67
```

In [11]:

```
round((a+b+c)
/3,2)
```

```
Out[11]: 26.67
```

if-else

In [12]:

```
# Create a function for finding
greater number between two numbers
def greater(n1,n2):
 if n1>n2:
 return(n1)
 else:
 return(n2)

greater(100,200)
```

```
Out[12]: 200
```

In [13]: In [14]:

```
def greater(n1,n2):
 if n1>n2:
 l1.append(n1)
 else:
 l1.append(n2)

greater(100,200)
```

```
l1=[<if_output> <if_con>
else <else_op> <loop>]
```

```
l1=[]
```

```
Out[14]: [200]
```

```
In [ ]: In [15]:
```

```
 return(n1)
 else:
 return(n2)

greater(100,200)

# format :
# lambda <arg1>,<arg2>:
<if_output> <if_con> else
<else_op>


greater=lambda a,b: (a if a>b else
b)
greater(8,3)
```

```
def greater(n1,n2):
 if n1>n2:
```

```
Out[15]: 8
```

lambda function is nothing but create a function

one argument

multiple arguments

if else conditions

if else conditions same like list compehenshion

```
In [2]:
list1=['hyd','mumbai'

,'chennai'] #output:

['Hyd','Mumbai','Chen

nai'] # M-1: use

append method

list2=[]
```

```
for i in list1:

list2.append(i.capita
lize())
print(list2)

# M-2: use list
comprhenshion
[i.capitalize() for i
in list1]

# M-3: make a lambda
function
```

```
['Hyd', 'Mumbai', 'Chennai']
```

```
Out[2]: ['Hyd', 'Mumbai', 'Chennai']
```

```
In [ ]: In [ ]:
```

```
iterbale/ you can print using for loop

# list ,string, tuple, dictionary


lambda <arguments>:
<output>,<iterator>

[i.capitalize() for i in list1]


list1=['hyd','chennai','mumbai']
lambda i:i.capitalize(),list1
```

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

```
lambda <arguments>: <output>


# whenever you use iterations

# iterator: some thing can be
```

```
Out[3]: (<function __main__.<lambda>(i)>, ['hyd', 'chennai', 'mumbai'])
                         input and output

In [ ]: In [ ]: In
                         list1=['hyd','chennai'
[4]:                     ,'mumbai'] map(lambda
                         i:i.capitalize(),list1
                         )
lambda <arg>:<output>


- next thing is map

Out[4]: <map at 0x1979c06cfd0>


In [ ]: In [5]:            list(map(lambda
                           i:i.capitalize(),list1))

- store the output

Out[5]: ['Hyd', 'Chennai', 'Mumbai']
```

first make a lambda function
second add your iterator
map both function and iterator
save the result in a list

```
In [ ]:
list1=['hyd','chennai','
mumbai'] lambda
i:i.capitalize(),list1
map(lambda
i:i.capitalize(),list1)
list(map(lambda
i:i.capitalize(),list1))
In [8]:


                        In [10]: In [15]:











                        list1=[1,2,3,4,5]
                        # [1,4,9,16,25]


                        list(map(lambda
                        i:i*i,list1))

                        for i in map(lambda
 i:i*i,list1):  print(i)
                         for i,j in
 1                       zip(list1,list2):
 4                       print(i+j)
 9
 16                      12
 25                      24
                         36

 list1=[1,2,3]
 list2=[11,22,33]        str(map(lambda i,j:i+j,
                         list1,list2))
 # [12,24,36]

Out[15]: '<map object at 0x000001979C06CC40>'
                     map(lambda i,j:i+j,
                     list1,list2)
In [16]:

Out[16]: <map at 0x1979c06dcc0>
         str([])

In [18]:

Out[18]: '[]'
            e() i*i
 In [ ]:    i+j
 i.capitaliz
In [20]:                 print(list2)
list1=['h#d','mum#bai
','chennai']
#['h#d','mum#bai']    [i for i in list1 if

list1=['h#d','mum#bai '#' in i] ['h#d',
','chennai'] list2=[]
for i in list1:       'mum#bai']
 if '#' in i:
 list2.append(i)
Out[20]: ['h#d', 'mum#bai']
                           <argument>:<condition>,<i
                           terator>
In [25]:
#lambda

 Out[25]: (<function __main__.<lambda>(i)>, ['h#d', 'mum#bai', 'chennai'])
                      in i,list1))

In [28]:
list1=['h#d','mum#bai' # condition mapping to
,'chennai']            list of items
list(map(lambda i: '#'

Out[28]: [True, True, False]
                          'chennai']
                          list(filter(lambda i:
In [31]:                  '#' in i,list1))
list1=['h#d','mum#bai',

Out[31]: ['h#d', 'mum#bai']
                        ','chennai'] '#' in
                        'h#d'
In [30]:                '#' in 'mum#bai'
list1=['h#d','mum#bai

Out[30]: True
```

In [ ]: