In [ ]:
```
f1.sh
====

cmd1
cmd2
cmd3--> Error
cmd4

output
=====
data1
data2
Command Error
data4


p1.py
====
lin1
line2
line3--> Erro
line4

output
=====

result1
result2
Erro
```

In [ ]:
```
try:
    lin1
    line2
    line3--> Erro
    line4
except Exception as obj:
    Handle error
```

In [1]:
```
try:
    name="Bob"
    print(NAME)
    age=12
    print(age)
except Exception as obj:
    print(obj)
```

```
name 'NAME' is not defined
```

In [2]:
```
name="Bob"
print(NAME)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-df010f8864a8> in <module>
      1 name="Bob"
----> 2 print(NAME)

NameError: name 'NAME' is not defined
```

In [4]:
```
try:
    name="Bob" #s1
    print(NAME) #s2
    print("After name display") #s3
except NameError as obj:
    print(obj) #s4
else:
    print("Else part message")#s5
```

```
name 'NAME' is not defined
```

In [ ]:
```
#Exception--> try->except
# s1-> s4
```

In [5]:
```
try:
    name="Bob" #s1
    print(name) #s2
    print("After name display") #s3
except NameError as obj:
    print(obj) #s4
else:
    print("Else part message")#s5
```

```
Bob
After name display
Else part message
```

In [ ]:
```
#Without Exception--> try->else
# s1->s2->s3->s5
```

In [6]:
```
#with Exception--> try->except->finally-> s1->s2->s4->s6
try:
    name="Bob" #s1
    print(NAME) #s2
    print("After name display") #s3
except NameError as obj:
    print(obj) #s4
else:
    print("Else part message")#s5
finally:
    print("Thankyou")   #s6
```

```
name 'NAME' is not defined
Thankyou
```

In [7]:
```
#No Exception  -> try->else->finally-> s1->s2->s3->s5->s6
try:
    name="Bob" #s1
    print(name) #s2
    print("After name display") #s3
except NameError as obj:
    print(obj) #s4
else:
    print("Else part message")#s5
finally:
    print("Thankyou")
```

```
Bob
After name display
```

```
Else part message
Thankyou
```

In [11]:
```python
name=input("Enter name")
try:
    if name!="root":
        raise NameError("you are not root user")
except Exception as obj:
    print(obj)
```

```
Enter namestudent
you are not root user
```

In [14]:
```python
#Task
#====

try:
    with open("D:\\emp1.csv") as fobj:
        L=fobj.readlines()
except Exception as eobj:
    print(eobj)
else:
    print(L)
finally:
    print("End of code")
print("In Main Script")
```

```
[Errno 2] No such file or directory: 'D:\\emp1.csv'
End of code
In Main Script
```

In [15]:
```python
#Without Exception Handling- control doesnt proceed after Exception
with open("D:\\emp1.csv") as fobj:
    L=fobj.readlines()
print(L)
print("End of code")
print("In Main Script")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-15-41beb1a20bf3> in <module>
      1 #Without Exception Handling- control doesnt proceed after Exception
----> 2 with open("D:\\emp1.csv") as fobj:
      3     L=fobj.readlines()
      4 print(L)
      5 print("End of code")

FileNotFoundError: [Errno 2] No such file or directory: 'D:\\emp1.csv'
```

In [16]:
```python
s="python"
print(s)
```

```
python
```

In [17]:
```python
iter(s)   # s|p|y|t|h|o|n
          #..............0x7ecd4f0
```

Out[17]:
```
<str_iterator at 0x7ecd4f0>
```

In [20]:
```
(hex(id('p')),hex(id('y')))
```

Out[20]:
```
('0x1ce4930', '0x1ce4cf0')
```

In [ ]:
```
reference- ox7ecd4f0
    |
    de reference
    1. manual way------> next(<iterator>).....StopIteration
    2. automatic --> for loop for v in <iterable>
```

In [27]:
```
obj=iter(s)
print(obj)
```

```
<str_iterator object at 0x0000000007ECD400>
```

In [28]:
```
print(next(obj))
print(next(obj))
print(next(obj))
print(next(obj))
print(next(obj))
print(next(obj))
print(next(obj))
```

```
p
y
t
h
o
n
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-28-269196dee9b1> in <module>
      5 print(next(obj))
      6 print(next(obj))
----> 7 print(next(obj))

StopIteration:
```

In [29]:
```
obj=iter(s)
for v in obj:
    print(v)
```

```
p
y
t
h
o
n
```

In [ ]:
```
fobj=open("D:\\emp.csv") # 2GB data
s=fobj.read() (or) L=fobj.readlines() #
fobj.close
```

In [30]:
```
fobj= open("D:\\emp.csv")
for v in fobj:
    print(v)
```

```
101,arun,sales,pune,1000

234,vijay,prod,bglore,2000
```

```
143,arun,sales,mumbai,3000

145,vinay,HR,hyderabad,4000

455,theeb,sales,chennai,3455

783,leo,prod,bglore,5678
```

In [37]:
```
#generator- function -if a function return an iterator (address), that is called genera
def f1():
    return 10
print(type(f1),type(f1()))
```

```
<class 'function'> <class 'int'>
```

In [36]:
```
def f2():
    yield 10

print(type(f2),type(f2()))
```

```
<class 'function'> <class 'generator'>
```

In [38]:
```
f2()
```

Out[38]:
```
<generator object f2 at 0x0000000008185F20>
```

In [39]:
```
f1()
```

Out[39]:
```
10
```

In [42]:
```
def f2():
    n=10
    yield n+100
    n=20
    yield n+200
    yield n+500
    yield "STDOUT"
    yield [1,2,3,4,5]
    yield "d",["11","12"],["13","14"]
```

In [43]:
```
f2()
```

Out[43]:
```
<generator object f2 at 0x00000000081BD660>
```

In [44]:
```
obj=f2()
for v in obj:
    print(v)
```

```
110
220
520
STDOUT
[1, 2, 3, 4, 5]
('d', ['11', '12'], ['13', '14'])
```

In [45]:
```
#iterator-Address
#function returning addresss--->generator

def fn():
```

```
    return 10  #

def f1():
    yield 10
 #fn()->value
#f1()--->generator ref/address- which needs deferencing to use the value
```

In [47]:
```
fn()
f1()
```

Out[47]:
```
<generator object f1 at 0x00000000081BDCF0>
```

In [ ]:
```
lambda expression/ function-  unnamed function

syntax:-
    def functionname(parameters):
        code
functionname(parametr)  #function call--->calls fn defn
syntax:-

var=    lambda parameter:code

var(parameter)#
```

In [48]:
```
rv=lambda a1,a2:a1+a2   # lambda inputparameter:action
print(rv(1,2))# call
```

Out[48]:
```
3
```

In [49]:
```
rv("Python","programming")
```

Out[49]:
```
'Pythonprogramming'
```

In [50]:
```
rv=lambda a:a.upper()
rv("hai")
```

Out[50]:
```
'HAI'
```

In [54]:
```
check=lambda a:a>100
check(11)
#map(lambda),filter(lambda)
print(check)
```

```
<function <lambda> at 0x00000000080B2430>
```

In [55]:
```
#list comprehension
L=[]
for v in [10,20,30]:
    r=v+100
    L.append(r)
print(L)
```

```
[110, 120, 130]
```

In [56]:
```
[v+100 for v in [10,20,30]] # List comprehension
```

Out[56]:
```
[110, 120, 130]
```

In [ ]:
```
#creating new list from existing list
#syntax:-

[var+100 for var in collection]
```

In [59]:
```
d={}

d["emp"]=[v.upper() for v in open("D:\\emp.csv")]
print(d)
```

```
{'emp': ['101,ARUN,SALES,PUNE,1000\n', '234,VIJAY,PROD,BGLORE,2000\n', '143,ARUN,SALES,M
UMBAI,3000\n', '145,VINAY,HR,HYDERABAD,4000\n', '455,THEEB,SALES,CHENNAI,3455\n', '783,L
EO,PROD,BGLORE,5678']}
```

In [65]:
```
[v+10 if v>3 else v+50 for v in [1,2,3,4,5]]
#[true_stmt if condition else false_stmt for var in collection]
```

Out[65]:
```
[51, 52, 53, 14, 15]
```

In [66]:
```
L=[]
for v in [1,2,3,4,5]:
    if(v>3):
        L.append(v+10)
    else:
        L.append(v+50)
print(L)
```

```
[51, 52, 53, 14, 15]
```

In [67]:
```
f1=lambda a:a+".log"
Files=[]
for v in ["p1","p2","p3,","p4"]:
    r=f1(v)
    Files.append(r)
print(Files)
```

```
['p1.log', 'p2.log', 'p3,.log', 'p4.log']
```

In [79]:
```
empSal=[1000,2000,3000,4000]

r=list(map(lambda a:a+500,empSal))#map(function,collection)
print(r)

for v in r:
    print(v)
```

```
[1500, 2500, 3500, 4500]
1500
2500
3500
4500
```

In [73]:
```
r=map(lambda a:a>500,empSal)
for v in r:
    print(v)
```

```
True
True
True
True
```

In [78]:
```
r=list(filter(lambda a:a<3500,empSal))# python 3.x - explicit typecast to list
for v in r:
    print(v)
print(r)
```

```
1000
2000
3000
[1000, 2000, 3000]
```

In [80]:
```
import functools
functools.reduce(lambda a,b:a+b,[10,20,30,40,50])# reduce(function,collection)
```

Out[80]:
```
150
```

In [82]:
```
if functools.reduce(lambda a,b:a+b,[10,20,30,40,50])>100:
    print("yes")
else:
    print("No")
```

```
yes
```

In [83]:
```
list(map(lambda a:a+".txt",["p1","p2","p3","p4"]))
```

Out[83]:
```
['p1.txt', 'p2.txt', 'p3.txt', 'p4.txt']
```

In [84]:
```
#comprehensive using filter
list(filter(lambda a:a>5,range(2,10)))#[6,7,8,9]
```

Out[84]:
```
[6, 7, 8, 9]
```

In [85]:
```
#Block style
L=[]
for i in range(2,10):
    if(i>5):
        L.append(i)
print(L)
```

```
[6, 7, 8, 9]
```

In [89]:
```
dept=["admin","sales","crm","QA","HR","prod"]
#filter sales,QA,prod from list

#blockstyle
result=[]
for i in dept:
    if i=="sales" or i=="QA" or i=="prod":
        result.append(i)

#comprehensive style
result1=list(filter(lambda a:a in ["sales","QA","prod"],dept))

result2=list(filter(lambda i:i=="sales" or i=="QA" or i=="prod",dept))

print(result)

print(result1)

print(result2)
```

```
['sales', 'QA', 'prod']
['sales', 'QA', 'prod']
['sales', 'QA', 'prod']
```

In [90]:
```python
from functools import reduce
reduce(lambda a,b:a+b, ["p","y","t","h","o","n"])
```

Out[90]: 'python'

In [91]:
```python
LB=[0.25,1.55,2.05,4.50,2.40]
#block style way
r=0
for v in LB:
    r=r+v

if r >10.5:
    print("High CPU Utilization")
```

```
High CPU Utilization
```

In [92]:
```python
#comprehensive style
from functools import reduce
if reduce(lambda a,b:a+b,LB)>10.5:
    print("High CPU Utilization")
```

```
High CPU Utilization
```