```
with open("MyNewFile.txt", 'w') as file:
  file.write("ABCDE")

with open("MyNewFile.txt", 'r') as file:
  a = (file.read(5)) # line1 Both  lines are in with Block(Under
Context of With Statement)
  print(file.closed) # line2
```

```
False
```

```
with open("MyNewFile.txt", 'w') as file:
  file.write("ABCDE")

with open("MyNewFile.txt", 'r') as file:
  a = (file.read(5))
print(file.closed)
```

```
True
```

```
Try Try but never cry


read(4)--> "Try "
read(5)--> "Try b"
```

```
with open("file1.txt", "w" ) as f:
    f.write("ABCD")

with open("file1.txt", "w" ) as f:
    f.write("A\nB\nC\nD\n")

with open("file2.txt", "w" ) as f:
    f.writelines(["A\n" ,"B\n","C\n", "D\n"])

with open("file2.txt", "w" ) as f:
    f.writelines(["A " ,"B ","C ", "D "])

output = "\n".join(["A" ,"B","C", "D"])
output
```

```
{"type":"string"}
```

```
with open("file1.txt", "r" ) as f:
    print(f.readlines())
```

```
['A\n', 'B\n', 'C\n', 'D\n']
```

```
with open("file1.txt", "r" ) as f:
    output_list = f.readlines()
print(output_list)
```

```
['A\n', 'B\n', 'C\n', 'D\n']
```

```
type(output_list)
```

```
list
```

```
with open("file1.txt", "r" ) as f:
    output_list = f.read()
print(output_list)
```

```
A
B
C
D
```

```
type(output_list)
```

```
str
```

```
with open("file1.txt", "r" ) as f:
    output_list = f.readline() # read only one row at a time into
string
print(output_list)
```

```
A
```

```
with open("file1.txt", "r" ) as f:
    print("reading: ", f.readline()) # read only one row at a time
into string
    print("reading again: ",f.readline()) # read only one row at a
time into string
```

```
reading:  A

reading again:  B
```

```
students = ["S ROY" , "B BOSE" ,"N KAR" , "C DUTTA" ,"G GHOSH", "B
KAR"]
scores = [1 , 3 ,2, 1 , 1, 2]
```

```
list(zip(students,scores))
```

```
[('S ROY', 1),
 ('B BOSE', 3),
 ('N KAR', 2),
 ('C DUTTA', 1),
```

```
 ('G GHOSH', 1),
 ('B KAR', 2)]

student_data = list(map(list,zip(students,scores))) # line 1
student_data
```

```
[['S ROY', 1],
 ['B BOSE', 3],
 ['N KAR', 2],
 ['C DUTTA', 1],
 ['G GHOSH', 1],
 ['B KAR', 2]]
```

```
# second step to get second lowest marks
second_low_score = sorted(list(set(scores)))[1] # line 2
second_low_score
```

```
2
```

```
# second step to get second Higest marks
# second_low_score = sorted(list(set(scores)))[-2] # replace -2  or
set reverse = True

list(filter(lambda x : x[1]==second_low_score, student_data ))
```

```
[['N KAR', 2], ['B KAR', 2]]
```

```
list(filter(lambda x : x[1]==second_low_score, student_data ))
```

```
[['N KAR', 2], ['B KAR', 2]]
```

```
list(map(lambda x: x[0], list(filter(lambda x :
x[1]==second_low_score, student_data )))) # line 3
```

```
['N KAR', 'B KAR']
```

```
output =list(filter(lambda x : x[1]==second_low_score, student_data ))
print(output)
```

```
[['N KAR', 2], ['B KAR', 2]]
```

```
for l in output:
    print(l[0])
```

```
N KAR
B KAR
```

```
[i[0] for i in output]
```

```
['N KAR', 'B KAR']
```

```
student_data[0]
```

```
['S ROY', 1]
```

```
student_data[1]
```

```
['B BOSE', 3]
```

```
(lambda x: x[1]==3) (student_data[1])
```

```
True
```

```
[['S ROY', 1.0], ['B BOSE', 3.0], ['N KAR', 2.0], ['C DUTTA', 1.0],
['G GHOSH', 1.0]]
2.0
["N KAR"] # ?
```

```
def second_lowest(students, scores):

    student_data, second_low_score, second_names = None, None, None

    # Your code starts here


    # Your code ends here
    return student_data, second_low_score, second_name
```

```
students = ["S ROY" , "B BOSE" ,"N KAR" , "C DUTTA" ,"G GHOSH", "B
KAR"]
scores = [1 , 3 ,2, 1 , 1, 2]
```

```
dict(zip(students,scores ))
```

```
{'S ROY': 1, 'B BOSE': 3, 'N KAR': 2, 'C DUTTA': 1, 'G GHOSH': 1, 'B
KAR': 2}
```

```
dict(zip(scores,students))
```

```
{1: 'G GHOSH', 3: 'B BOSE', 2: 'B KAR'}
```

```
# evaluate() printing the number of vowels if the Name is Smaller
otherwise it prints the number of consonants."

name = "Sachini"
name = set(name)
vowel_count = len(list(filter(lambda x :x in 'aeiouAEIOU',name )))
vowel_count
```

```
2
```

```
const_count =len(name)- vowel_count
const_count
```

```
4
```

```
### HOF: A fucntion that return a func

def gen_mul(x):
    def mul(y):
        return x*y
    return mul

a  = gen_mul(5)

### is this a HOF? Yes
### Is this a generator? No

print(a)


'''
def mul(y):
    return 5*y
'''
```

```
<function gen_mul.<locals>.mul at 0x7f86ed31aca0>
```

```
def mul(y):
    return 5*y
mul(2)
```

```
10
```

```
a(2)
```

```
10
```

```
b = gen_mul(101)
print(b)
```

```
<function gen_mul.<locals>.mul at 0x7f86ed3db4c0>
```

```
def mul(y):
    return 101*y

b(2)
```

```
202
```

```
b(3)
```

```
303
```

```
b(4)
```

```
404
```

```
# def funct():


#     return x
#     # yeild x
```

```
range(1,6)
```

```
range(1, 6)
```

```
a = iter(range(1,6))
```

```
a
```

```
<range_iterator at 0x7f86ed43a150>
```

```
next(a)
```

```
1
```

```
next(a)
```

```
2
```

```
next(a)
```

```
3
```

```
next(a)
```

```
4
```

```
next(a)
```

```
5
```

```
next(a)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-114-15841f3f11d4> in <cell line: 1>()
----> 1 next(a)
```

```
StopIteration:
```

```
list1 = [1, 2, 3]
print(list1.append(5))
```

```
None
```

```
str1 = "abc"
str1 = str1+"d"
```

```
str1
```

```
{"type":"string"}
```

```
[ 1,2, 3,4,5,6]
```

```
### decorator: decoratores accept function as input argument and
return func
def pretty(func):
    def func2():
        print("-"*50)
        func()
        print("-"*50)

    return func2


f1 = lambda : print(" "*25 + "Hello" + " "*25)

a = pretty(f1)

f1()
```

```
                         Hello
```

```
a()
```

```
--------------------------------------------------
                         Hello
--------------------------------------------------
```

```
-- helo --
```

```
f2 = lambda : print("World")
b = pretty(f2)
b()
```

```
--------------------------------------------------
World
--------------------------------------------------
```

```
@pretty
def print_w():
    print("World")
print_w()
```

```
--------------------------------------------------
World
--------------------------------------------------
```

```
def pretty2(func):
    def func2():
        print("#"*50)
        func()
        print("#"*50)

    return func2

@pretty
@pretty2
def print_w():
    string = "Hello"
    hyp = 50
    val = hyp//2 - len(string)//2
    print(" "*val + "Hello" + " "*val)
print_w()
```

```
--------------------------------------------------
##################################################
                      Hello
##################################################
--------------------------------------------------
```

```
##################################################
                      World
##################################################
```

```
names = ["Adam","Millie","Jacob","Michael","Maxine"]
loc = [(1,2),(5,4),(0,3),(2,3),(0,0)]

list1 = []
for i in range(len(names)):
    for j in range(i+1, len(names)):
        x1, y1 = loc[i]
        x2, y2 = loc[j]
        distance = abs(x1-x2) + abs(y1 - y2)
        print(names[i], names[j], distance)
        list1.append(distance)
print(list1)
```

```
Adam Millie 6
Adam Jacob 2
Adam Michael 2
```

```
Adam Maxine 3
Millie Jacob 6
Millie Michael 4
Millie Maxine 9
Jacob Michael 2
Jacob Maxine 3
Michael Maxine 5
[6, 2, 2, 3, 6, 4, 9, 2, 3, 5]

[6, 2, 2, 3, 6, 4, 9, 2, 3, 5][::-1]

[5, 3, 2, 9, 4, 6, 3, 2, 2, 6]

min(list1)

2

list1 = []
for i in range(len(names)):
    for j in range(i+1, len(names)):
        x1, y1 = loc[i]
        x2, y2 = loc[j]
        distance = abs(x1-x2) + abs(y1 - y2)
        list1.append(distance)

min_d = min(list1) # min element in all distances

list_output = []
for i in range(len(names)):
    for j in range(i+1, len(names)):
        x1, y1 = loc[i]
        x2, y2 = loc[j]
        distance = abs(x1-x2) + abs(y1 - y2)
        if distance == min_d:
            list_output.append([names[i], names[j]])

list_output

[['Adam', 'Jacob'], ['Adam', 'Michael'], ['Jacob', 'Michael']]
```

EXCEPTION HANDLING AND MODULES

- TRY EXCEPT AND FINALLY
- How to import modules