In [ ]:
```
# Modules and Packages
```

In [1]:
```
# WAP to read excel file
```

In [2]:
```
! pip install excel
```

```
Collecting excel
  Downloading excel-1.0.0.tar.gz (2.3 kB)
Requirement already satisfied: xlrd in /opt/anaconda3/lib/python3.7/site-package
s (from excel) (1.2.0)
Building wheels for collected packages: excel
  Building wheel for excel (setup.py) ... done
  Created wheel for excel: filename=excel-1.0.0-py3-none-any.whl size=3007 sha25
6=5c4016cc219d7b012be5ca5bfa4e9628c2ae79bd54448743593b16224c3d795e
  Stored in directory: /Users/nvrsettle/Library/Caches/pip/wheels/40/20/eb/00974
261534573b83d9836784d45accfe983fa95804db30832
Successfully built excel
Installing collected packages: excel
Successfully installed excel-1.0.0
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is available.
You should consider upgrading via the '/opt/anaconda3/bin/python -m pip install
--upgrade pip' command.
```

In [7]:
```
from excel import OpenExcel

f = OpenExcel('exam.xls')

f.read() # read all

print(f.read('A')) # read 'A' row

print(f.read(1) )# f.read('1'), read '1' column

print(f.read(2)) # read 'A5' position
```

```
['No', 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.
0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0,
28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 4
1.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0]
['No', 'ID', 'Attempt', 'Candidate Name', 'Username', 'GroupName', 'Exam', 'Exam
ID', 'Marks or Points', 'Percentage', 'Result', 'examstatus', 'TimeTaken', 'Star
t_DateTime', 'Finish_DateTime', 'Mode', 'Email', 'Mobile', 'Certificateid', 'Can
didate registered on']
[1.0, 8542.0, 1.0, 'Letsupgradeb4fcs050 Letsupgradeb4fcs050', 'LetsUpgradeB4FCS0
50', 'LetsUpgrade FCS B4', 'ReExam LetsUpgrade FCS Python B4 Exam', 746.0, '22 O
ut of 91', 24.0, 'Fail', 'Completed', 0.028460648148148148, 44003.47224537037, 4
4003.500706018516, 'Exam Mode', 'praisey0234@gmail.com', '', 0.0, 43994.31856481
4814]
```

# Creating our Own Modules and Packages

In [8]:
```
%%writefile addition.py

def addition(*args):
    add = 0
    for i in args:
        add = add + i
    return add
```

```
Writing addition.py
```

In [4]:
```
from Calculator import addition

temp = addition.addition(4,5,6,7,8,5)
print(temp)
```

```
35
```

In [1]:
```
%%writefile subtraction.py

def sub(num1, num2):
    minus = num1 - num2
    return minus
```

```
Overwriting subtraction.py
```

In [3]:
```
from Calculator import subtraction

temp = subtraction.sub(4,2)
print(temp)
```

```
2
```

In [5]:
```
# __MAIN__ and __NAME___
```

1. YOUR .PY File can work as Module also and can work as Main Code also ?
2. If you want to know is my .py file is running as a Module or it is running as a Main Code ?
3. That time you use **main** == **name**

In [10]:
```
%%writefile file1.py

print("This is File1.py")

if __name__ == "__main__": # This is a Condition to check weather the
    #file is called directly or indirectly
    print("This File is running directly")
else:
    print("This file is running Indirectly as Module")
```

```
Overwriting file1.py
```

In [11]:
```
! python file1.py
```

```
This is File1.py
This File is running directly
```

In [8]:
```
%%writefile file2.py

import file1

print("This is File2.py")

if __name__ == "__main__":
    print("This is running directly")
else:
    print("This is running In-Directly as module")
```

```
Writing file2.py
```

In [9]:
```
! python file2.py
```

```
This is File1.py
This file is running Indirectly as Module
This is File2.py
This is running directly
```

# Decorators !!!

- You all have a some function1
- You are required to create one more function2
- function2 can be done with few modification in Function1
- But you are required to copy paste the whole stuff of function 1 and modify
- This will increase the numbers of lines of code in the program
- This will actually waste memory the computer
- This is a kind of static way of doing
- is there any better way ?

In [16]:
```
# basic Function
def hi():
    print("HI Sonali")
```

In [17]:
```
hi()
```

```
HI Sonali
```

In [21]:
```
# basic Functuion with arg
def hi(arg = "LetsUpgrader"):
    print("HI - ", arg)
```

In [22]:
```
hi("Sonali Agwane")
```

```
HI -  Sonali Agwane
```

In [23]:
```
hi()
```

```
HI -  LetsUpgrader
```

In [24]:
```
# renaming the functionm
hello = hi
```

In [25]:
```
hello("Music Lover")
```

```
HI -  Music Lover
```

In [27]:
```
# Can we pass a function as Argument ?
```

In [29]:
```
def Amigo(arg):
    print("we are inside amigo")

    arg()

    print("We are done executing a function inside a function")
```

In [30]:
```
Amigo(hi)
```

```
we are inside amigo
HI -  LetsUpgrader
We are done executing a function inside a function
```

In [31]:
```
# Can you return a function from the Function ?

def hola():
    print( " we will be returning a function from the function ")

    def namaskar(arg):
        print("Hello - ", arg)

    return namaskar
```

In [32]:
```
abc = hola()
```

```
 we will be returning a function from the function
```

In [33]:
```
abc("Sai")
```

```
Hello -  Sai
```

In [35]:
```
# Can we decorate a function inside a function with other functionality
```

In [37]:
```
def needRapper():
    print("Hey I am the Patty i need bread around me")
```

In [38]:
```
def wrapperFun(arg):

    def wrappedCode():
        print("This is Top bread code")

        arg()

        print("This is Bottom bread code")

    return wrappedCode
```

In [40]:
```
burger = wrapperFun(needRapper)
```

In [41]:
```
burger()
```

```
This is Top bread code
Hey I am the Patty i need bread around me
This is Bottom bread code
```

In [56]:
```
def addition(num1,num2):
    print("This is a addition function")
    print("Result - ",num1 + num2)
```

In [53]:
```
def subtraction(num1,num2):
    print("This is a Subtraction function")
    print("Result - ", num1 - num2)
```

In [60]:
```
def inputCode(arg):
    def modifiedFunction():
        num1 = int(input("Enter the Number 1 - "))
        num2 = int(input("Enter the Number 2 - "))
        arg(num1,num2)
    return modifiedFunction
```

In [61]:
```
fullAdditionFunWithIP = inputCode(addition)
```

In [62]:
```
fullAdditionFunWithIP()
```

```
Enter the Number 1 - 56
Enter the Number 2 - 12
This is a addition function
Result -  68
```

In [54]:
```
fullSubtractionFunWithIP = inputCode(subtraction)
```

In [55]:
```
fullSubtractionFunWithIP()
```

```
Enter the Number 156
Enter the Number 134
This is a Subtraction function
Result -  22
```

In [65]:
```
@inputCode # This is known as Decorator
def multipy(num1,num2):
    print(num1* num2)
```

In [66]:
```
multipy()
```

```
Enter the Number 1 - 56
Enter the Number 2 - 12
672
```

# Generators

In [67]:
```
# How many things can you return from a function ??
```

In [68]:
```
def isOdd(num):
    if num%2 != 0:
        return num
```

In [70]:
```
isOdd(45)
```

Out[70]:
```
45
```

In [81]:
```
# this code will return the first ODD and exit
def isOdd(range1 = 1, range2 = 10):
    for num in range(range1, range2):
        if num%2 != 0:
            return num
```

In [82]:
```
oddNumbers = isOdd(1,100)
```

In [83]:
```
oddNumbers
```

Out[83]:
```
1
```

In [88]:
```
# We want our code to have all the odd numbers in the range and o/p
def isOdd(range1 = 1, range2 = 10):
    for num in range(range1, range2):
        if num%2 != 0:
            yield num
```

In [89]:
```
oddNumbers = isOdd(1,100)
```

In [90]:
```
print(list(oddNumbers))
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41,
43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81,
83, 85, 87, 89, 91, 93, 95, 97, 99]
```

In [ ]:
```

```