Python Workshop 1 Exercises

A.Write a function that returns sum of two numbers

```
def add(a,b):
  add=a+b
  return add
a=4
b=7
print(add(a,b))
```

```
11
```

B. Write a function that returns y where y=3x+2.

```
def f(x):
  y=3*x+2
  return y
x=int(input('Enter a number: '));
print(f(x))
```

```
Enter a number: 1
5
```

C.Write a function that calculate g(x) = x^2+2x+3.

```
def g(x):
  y= x^2+2*x+3
  return y
x=int(input('Enter a number: '));
print (g(x))
```

```
Enter a number: 1
6
```

D. Write a function f(x)=x^2 where x is an array [1,2,3,4,5] and the function returns y that is an array

```
def g(x):
  y=[] #defining y as an array
  for values in x:
    y.append(values**2)#adding in a list is called append
```

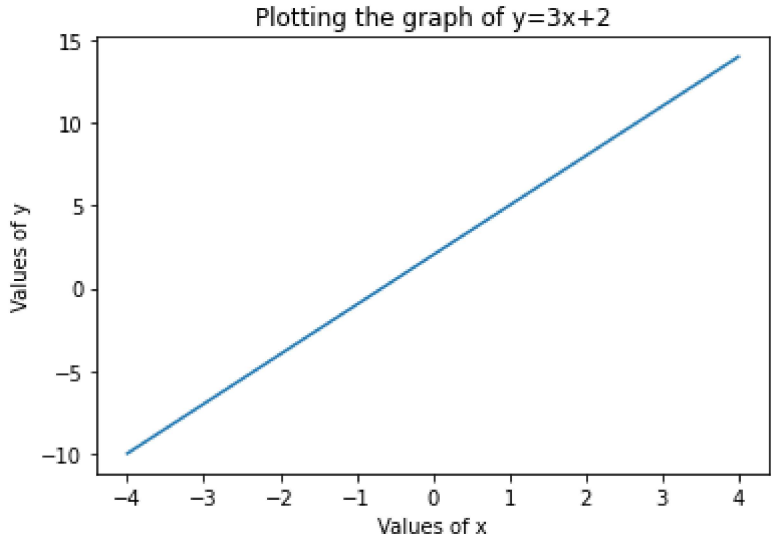
```
  return y
a=[1,2,3,4,5]
print(g(a))
```

```
[1, 4, 9, 16, 25]
```

E. Write a function i(x)=3x+2 that takes x as an array [-4,-3,-2,-1,0,1,2,3,4] and returns y. Also plot it in graph.

```
import matplotlib.pyplot as plt
def i(x):
  y=[]
  for values in x:
    y.append(3*values+2)
  return y
a=[-4,-3,-2,-1,0,1,2,3,4]
print(i(a))
plt.title("Plotting the graph of y=3x+2")#title of the whole graph
plt.plot(a,i(a))
plt.ylabel('Values of y') #label of y-axis
plt.xlabel('Values of x') #Label of x-axis
```

```
[-10, -7, -4, -1, 2, 5, 8, 11, 14]
Text(0.5, 0, 'Values of x')
```

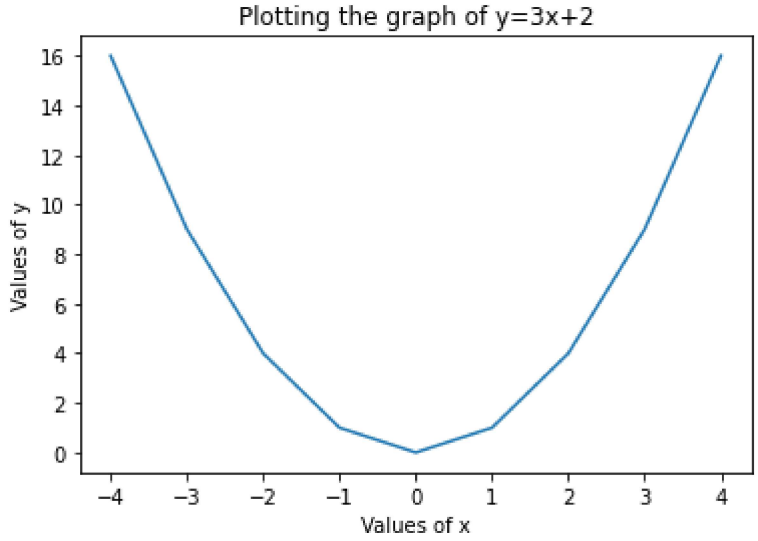


F. Write a function j(x)=x^2that takes x as an array [-4,-3,-2,-1,0,1,2,3,4] and returns y. Also plot it in graph.

```
import matplotlib.pyplot as plt
def j(x):
  y=[]
  for values in x:
    y.append(values**2)
```

```
  return y
a=[-4,-3,-2,-1,0,1,2,3,4]
print(i(a))
plt.title("Plotting the graph of y=3x+2")#title of the whole graph
plt.plot(a,j(a))
plt.ylabel('Values of y') #label of y-axis
plt.xlabel('Values of x') #Label of x-axis
```

```
[16, 9, 4, 1, 0, 1, 4, 9, 16]
Text(0.5, 0, 'Values of x')
```



G. Write a function f(X)=x-2 and g(x)=3x+2 and print gof(x)=g(f(x))

```
def f(x):
  a=x-2
  return a
def g(x):
  b=3*x+2
  return b
x=int(input("Enter a number"))
print (g(f(x)))
```

```
Enter a number 3
5
```

Suppose we don't know the formula for the sum and want to use a "brute force" approach. We can calculate the sum *1 + 2 + ... + n* using a loop:

```
sum1n = 0 i = 1 while i <= n: sum1n += i # same as sum1n = sum1n + i i += 1
```

This loop can also be nested within another loop to calculate a series of numbers:

```
## Sums1toN program nMax = 10 n = 1 while n <= nMax: sum1n = 0 i = 1 while i <= n: sum1n
+= i i += 1 print('{0:3d} {1:6d}'.format(n, sum1n)) n += 1
```

Type in the little program above into the cell below. To run it select the cell and then press the **run-button** from the button bar above. Test the program and ensure it runs correctly.

```
nMax = 10
n = 1
while n <= nMax:
    sum1n = 0
    i = 1
    while i <= n:
        sum1n += i
        i += 1
    print('{0:3d} {1:6d}'.format(n, sum1n))
    n += 1
```

```
  1      1
  2      3
  3      6
  4     10
  5     15
  6     21
  7     28
  8     36
  9     45
 10     55
```

Now try to write the Python code to answer the following questions in the cells below

1. Modify the Sums1toN.py program to accept only an odd positive integer nMax from the user and print the sums of positive odd integers for *n = 1, 3, ..., nMax*.

```
num = input ("enter a number")
num = int(num)
n=1
sum=0
if(num%2==1 and n>0):
  while(n<=num):
    sum=n+sum
    print("{0:3d}{1:6d}".format(n,sum))
    n=n+2

else:
  print("please enter odd number")
```

```
enter a number15
  1    1
  3    4
  5    9
  7   16
  9   25
 11   36
 13   49
 15   64
```

2. Write a program that prompts the user for a positive integer n and prints all positive multiples of 6 *ie. (6, 12, 18, etc.)* that do not exceed n.

```
num = input ("enter a number")
num = int(num)
n=0
while n<=num:
  print('{0:3d}'.format(n))
  n =n+6
```

```
enter a number49
  0
  6
 12
 18
 24
 30
 36
 42
 48
```

3. Using Sums1toN.py as a prototype, write a program that prompts the user to enter a positive integer nMax and displays n and n! (n-factorial) for n from 1 to nMax. .

```
nmax = input("enter a number")
nmax = int(nmax)
n=0
while n<=nmax:
  sum1n = 1
  i = 1
  while i<=n:
    sum1n *=i
    i = i+1
  print ('{0:3d}!{1:6d}'.format(n,sum1n))
  n = n+1
```

```
enter a number7
  0!     1
  1!     1
  2!     2
  3!     6
  4!    24
  5!   120
  6!   720
  7!  5040
```

Note: most of the programs for this module do not require you to use the *input* statement to load data into your code. To avoid loosing marks (for over use of the input statement) You should only use input statements where the question includes the phrases like *"prompts the user.."*, *"from the user.."* or *"ask the user.."*. **ie. for questions 4,5 & 6 hard code the data in the program**

4. Write a program that prints n, $s_1(n) = \sum_{k=1}^{n} k$, $s_2(n) = \sum_{k=1}^{n} k^2$, and $\frac{3s_2(n)}{s_1(n)}$ for n = 1, 2, 3, ..., 20. Try to guess the general formula for $\sum_{k=1}^{n} k^2$ from the resulting table.

```
num = input("enter a number")
num =  int(num)
n=1
while n<=num:
  s1 = 0
  k = 1
  while k <=n:
    s1 += k
    k += 1
  s2 = 0
  k = 1
  while k <= n:
      s2 += k**2
      k += 1
  s3 = 3*s2//s1
  print('{0:3d} {1:6d} {2:6d} {3:6d}'.format(n,s1,s2,s3))
  n += 1
```

```
enter a number5
  1      1      1      3
  2      3      5      5
  3      6     14      7
  4     10     30      9
  5     15     55     11
```

5. Write a function printSquare(n) that displays a “square” whose side has n stars. For example, for n = 5, the output should be:

```
*****
* *
* *
* *
*****
```

Use only one while loop. (Actually, in Python, you can write this function without any loops, on one line. Can you figure out how?)

```
side = int(input("Please Enter any Side of a Square : "))

print("Hollow Square Star Pattern")
for i in range(side):
  for j in range(side):
    if(i == 0 or i == side - 1 or j == 0 or j == side - 1):
      print('*', end = ' ')
    else:
      print(' ', end = ' ')
  print()
```

```
 Please Enter any Side of a Square : 5
 Hollow Square Star Pattern
 * * * * *
 *       *
 *       *
 *       *
 * * * * *
```

6. Write a function *myPow(x, n)* that returns x^n, where and n is a non-negative integer. Do not use the ** operator or the math.pow function — use one *while* loop. Hint: $x^0 = 1$.

```
def myPow(x, n):
    res = 1

    for count in range(n):
        res *= x
    return res

myPow(4, 2)
```

```
 16
```

Downloading your workbook

Please note even if you are Jupyter on your computer (ie. not on a network) the Jupyter notebook runs as a server, so you are not running and editing a file on your filing system. So to obtain your workbook (to use on a different computer or upload as an assignment, etc.) it must be downloaded to your file system. To do this...

Goto the "**File**" menu and select the "**Download as**" item. You can then select "**notebook (ipynb)**" to download.

✓ 4s completed at 9:34 AM