Python Basic Programming Exercises

Q1: What is the output of following expression 5 + 4 * 9 % (3 + 1) / 6 - 1

In [ ]:
```
4.0
```

Q2: Write a program to check if a Number is Odd or Even. Take number as a input from user at runtime.

In [18]:
```
def oddeven (a):
    print("Even") if a%2 == 0 else print("odd")
```

In [20]:
```
oddeven(3)
```

```
odd
```

Q3: Write a program to display the multiplication table by taking a number as input. [Hint : Use print statement inside of a loop]

In [49]:
```
def multiplicationtable(a):
    for i in range(1,11):
        print("{0:d}*{1:d} = {2:d}".format(a,i,a*i))
```

In [50]:
```
multiplicationtable(11)
```

```
11*1 = 11
11*2 = 22
11*3 = 33
11*4 = 44
11*5 = 55
11*6 = 66
11*7 = 77
11*8 = 88
11*9 = 99
11*10 = 110
```

Q4: Write a program which will find all numbers between 2000 and 3200 which are divisible by 7 but are not a multiple of 5. Note: The numbers obtained should be printed in a comma-separated sequence on a single line.

In [71]:
```
for i in range(2000,3200):
    if(i%7==0 and i%5!= 0):
        print(i, end=", ")
```

```
2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107, 2114,
2121, 2128, 2142, 2149, 2156, 2163, 2177, 2184, 2191, 2198, 2212, 2219, 2226, 2233,
2247, 2254, 2261, 2268, 2282, 2289, 2296, 2303, 2317, 2324, 2331, 2338, 2352, 2359,
2366, 2373, 2387, 2394, 2401, 2408, 2422, 2429, 2436, 2443, 2457, 2464, 2471, 2478,
2492, 2499, 2506, 2513, 2527, 2534, 2541, 2548, 2562, 2569, 2576, 2583, 2597, 2604,
2611, 2618, 2632, 2639, 2646, 2653, 2667, 2674, 2681, 2688, 2702, 2709, 2716, 2723,
2737, 2744, 2751, 2758, 2772, 2779, 2786, 2793, 2807, 2814, 2821, 2828, 2842, 2849,
2856, 2863, 2877, 2884, 2891, 2898, 2912, 2919, 2926, 2933, 2947, 2954, 2961, 2968,
2982, 2989, 2996, 3003, 3017, 3024, 3031, 3038, 3052, 3059, 3066, 3073, 3087, 3094,
3101, 3108, 3122, 3129, 3136, 3143, 3157, 3164, 3171, 3178, 3192, 3199,
```

Q5: Count the elements of each datatype inside the list and display in output [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]

In [2]:
```
list1=[2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]
no_of_int=0
no_of_str=0
no_of_bool=0
no_of_float=0
```

```
for i in range(len(list1)):
    if(type(list1[i])==int):
        no_of_int+=1
    if(type(list1[i])==float):
        no_of_float+=1
    if(type(list1[i])==str):
        no_of_str+=1
    if(type(list1[i])==bool):
        no_of_bool+=1

print("no_of_int = {0:d}".format(no_of_int))
print("no_of_float = {0:d}".format(no_of_float))
print("no_of_str = {0:d}".format(no_of_str))
print("no_of_bool = {0:d}".format(no_of_bool))
```

```
no_of_int = 5
no_of_float = 1
no_of_str = 4
no_of_bool = 1
```

Q6: Add all values from the list with numeric datatypes [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]

In [159…

```
#list1=[2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]
sum_of_nums=0

for i in range(len(list1)):
    if type(list1[i])==int or type(list1[i])==float:
        sum_of_nums+=list1[i]

print("sum of numbers = {0:.2f}".format(sum_of_nums))
```

```
sum of numbers = 21.50
```

Q7: Concat all str datatypes with hyphen as a delimiter [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]

In [4]:

```
str_concat=''

for i in range(len(list1)):
    if type(list1[i])==str:
        str_concat=str_concat+list1[i]+'-'

print("long string = {0:s}".format(str_concat))
```

```
long string = Py-10-SQL-John-
```

Q8: Write a UDF that takes list as input and returns sum of all numbers (exclude bool) and count of all str [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7] Hint: ----- def my_func: # your code my_func(l1) # output --> {'Sum': xxx, 'Count_of_Strs': xxx}

In [11]:

```
def UDF1 (l1):
    sum_of_nums=0
    count_strs=0
    for i in range(len(l1)):
        if type(l1[i])==int or type(l1[i])==float and type(l1[i])!=bool :
            sum_of_nums+=l1[i]
        if type(l1[i])==str:
            count_strs+=1
    return ("sum of numbers = {0:.2f}, count of string = {1:d}".format(sum_of_nums,c
```

In [12]:

```
UDF1([2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7])
```

Out[12]:

```
'sum of numbers = 21.50, count of string = 4'
```

Q9: Get only odd numbers from the following list and store the numbers in new list li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61] i. Use loops to get the answer ii. Use list comprehensions iii. Use lambda function with filter

In [88]:
```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
l2=[]
for i in range(len(li)):
    if(li[i]%2 != 0):
        l2.append(li[i])

print (l2)
```

```
[5, 7, 97, 77, 23, 73, 61]
```

Q10: Write a UDF to return the descriptives [sum, count, min, mean, max] for a list of n number of input numbers.

In [4]:
```
def UDF10 (n):
    output=[]
    sum1=0

    for i in range(len(n)):
        sum1=sum1+n[i]

    output.append(sum1)
    output.append(len(n))
    output.append(min(n))
    output.append(max(n))
    output.append((min(n)+max(n))/2)
    return(output)
```

In [5]:
```
UDF10([5,7,97,77,23,73,61])
```

Out[5]:
```
[343, 7, 5, 97, 51.0]
```

Q11: Write an udf to calculate the area of different shapes Take shape and dimensions as arguments to udf as follows : 1. square which has side 2. rectangle which has length and width 3. circle which has radius The shape should be a positional argument and it's dimensions are taken as kwargs Perform proper validation for the user inputs and then calculate area. E.g. if shape is square, ensure kwargs has "side" and if so, then you may return the area, else display appropriate error message like "Please enter 'side' for a square"

In [73]:
```
def shape_area (**kargs):
    Area_sq=0
    Area_rec=0
    Area_cir=0
    for key,value in kargs.items():
        if (key=='Square'):
            Area_sq=value*value
        if (key=='Rect'):
            Area_rec=value[0]*value[1]
        if (key=='Circle'):
            Area_cir=value*value*3.14
    print(Area_sq)
    print(Area_rec)
    print(Area_cir)
```

In [76]:
```
shape_area(Square=2,Rect=(3,4),Circle=6)
```

```
4
12
```

```
113.04
```

Q12: Write a UDF to reconcile the values within two lists. l1 = ['January', 'February', 'March', 'May', 'June', 'September', 'December'] l2 = ['January', 'February', 'April', 'June', 'October', 'December'] Hint: ----- def func(l1, l2): your code here... Output: {'Matched': ['January', 'February', 'June', 'December'], 'Only in l1': ['March', 'May', 'September'], 'Only in l2': ['April', 'October']}

In [51]:
```
#set1={'January', 'February', 'March', 'May', 'June', 'September', 'December'}
#set2={'January', 'February', 'April', 'June', 'October', 'December'}
def Set(l1,l2):
    set1=set(l1)
    set2=set(l2)

    print('Matched:',list(set1.intersection(set2)))
    print('Only in l1:',list(set1-set2))
    print('Only in l2:',list(set2-set1))
```

In [52]:
```
Set(['January', 'February', 'March', 'May', 'June', 'September', 'December'],['Janua
```

```
Matched: ['December', 'January', 'February', 'June']
Only in l1: ['May', 'September', 'March']
Only in l2: ['April', 'October']
```

Q13: write a UDF to check if a number is prime or not.

In [173…
```
import math
def calprime(num):
    sqnum=math.sqrt(num)
    r=0
    for i in range(2,int(sqnum)+1):
        return(print("Not a Prime number"))  if num%i == 0 else print()

    return(print('It is a prime number'))
```

In [168…
```
calprime(97)
```

```
It is a prime number
```

Q14. Write a program which can compute the factorial of a given numbers. # The results should be printed in a comma-separated sequence on a single line. # input() function can be used for getting user(console) input #Suppose the input is supplied to the program: 8 #Then, the output should be: 40320 #Hints: In case of input data being supplied to the question, it should be assumed to be a console input.

In [63]:
```
print('enter the number please')
n=int(input())
fact=1
for i in range(1,n+1):
    fact*=i

print(fact)
```

```
enter the number please
9
362880
```

Q15. With a given integral number n, write a program to generate a dictionary that contains (i, i*i) such that is an integral number between 1 and n (both included). and then the program should print the dictionary. #Suppose the following input is supplied to the program: 8 #Then, the output should be: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64} #Hints: In case of input data being supplied to the question, it should be assumed to be a console input. Consider using dict()

In [129…
```
def createdict(n):
    dict1={}
    for i in range(1,n+1):
        dict1[i]=i*i

    return dict1
```

In [130…
```
createdict(8)
```

Out[130… {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

Q16. Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number. #Suppose the following input is supplied to the program: 34,67,55,33,12,98 #Then, the output should be: ['34', '67', '55', '33', '12', '98'] ('34', '67', '55', '33', '12', '98') #Hints: In case of input data being supplied to the question, it should be assumed to be a console input. you may use tuple() method to convert list to tuple

In [113…
```
def list_tuple(l1):
    for i in range(len(l1)):
        l1[i]=str(l1[i])

    t1=tuple(l1)
    print(l1, t1, end=" ")
```

In [114…
```
list_tuple([34,67,55,33,12,98])
```

```
['34', '67', '55', '33', '12', '98'] ('34', '67', '55', '33', '12', '98')
```

Q17. Write a program that accepts a comma separated sequence of words as input and # prints the words in a comma-separated sequence after sorting them alphabetically. # Suppose the following input is supplied to the program: without,hello,bag,world # Then, the output should be: bag,hello,without,world #Hints: In case of input data being supplied to the question, it should be assumed to be a console input.

In [74]:
```
#list1=['without','hello','bag','world']
string1=(input())
list1=string1.split(',')
list1.sort()
print(list1)
```

```
without,hello,bag,world
['bag', 'hello', 'without', 'world']
```

Q18. Write a program that accepts a sequence of whitespace separated words # as input and prints the words after removing all duplicate words and sorting them alphanumerically. # Suppose the following input is supplied to the program: hello world and practice makes perfect and hello world again # Then, the output should be: again and hello makes perfect practice world #Hints: In case of input data being supplied to the question, it should be assumed to be a console input. #We use set container to remove duplicated data automatically and then use sorted() to sort the data.

In [3]:
```
string2=input()
set2=set(string2.split(' '))
list2=list(set2)
list2.sort()
print(list2)
```

```
hello world and practice makes perfect and hello world again
['again', 'and', 'hello', 'makes', 'perfect', 'practice', 'world']
```

Q19. Write a program that accepts a sentence and calculate the number of upper case # letters and lower case letters. #Suppose the following input is supplied to the program: Hello world! #Then, the output should be:

UPPER CASE 1 LOWER CASE 9 #Hints: In case of input data being supplied to the question, it should be assumed to be a console input.

In [ ]:
```
def UDF1 (str1):
    no_of_up=0
    no_of_lo=0
    for i in range(len(str1)):
    #no_of_lo += 1 if str1[i].islower() else no_of_up = no_of_up+1
    #no_of_lo += 1 if str1[i].isupper()

        if str1[i].islower():
            no_of_lo += 1
        elif str1[i].isupper():
            no_of_up += 1

    return print('UPPER CASE {0:d} LOWER CASE {1:d}'.format(no_of_up,no_of_lo))
```

In [112…
```
UDF1('Hello World!')
```

```
UPPER CASE 2 LOWER CASE 8
```

Q20. Write a program that takes a string and returns reversed string. i.e. if input is "abcd123" output should be "321dcba"

In [113…
```
def reversestr(str1):
    print(str1[::-1])
```

In [114…
```
reversestr("abcd123")
```

```
321dcba
```