Python Practice

Ndugba

Problem # 1

Biggie Size - Given a list, write a function that changes all positive numbers in the list to "big". Example: make_it_big([-1, 3, 5, -5]) returns that same list, #changed to [-1, "big", "big", -5].

In [1]:
```
##how can i remove the none at the end?
def positive_numbers(list):
    for number in list:
        if number >0:
            print("big")
        else:
            print(number)
    return
```

In [2]:
```
example1= [-1,3,5,-5]
print(positive_numbers(example1))
```

```
-1
big
big
-5
None
```

In [3]:
```
def positive_numbers(list):
    for index, value in enumerate(list):
        if value>0:
            list[index]="big"

    return list
```

In [4]:
```
example1= [-1,3,5,-5]
print(positive_numbers(example1))
```

```
[-1, 'big', 'big', -5]
```

In [5]:
```
def positive_numbers(list):
    for number in range(len(list)):
        if list[number] >0:
            list[number]= "big"
    return list
```

In [6]:
```
example1= [-1,3,5,-5]
print(positive_numbers(example1))
```

```
[-1, 'big', 'big', -5]
```

In [ ]:
```

```

Problem # 2

Count Positives - Given a list of numbers, create a function to replace last value with number of positive values. Example, count_positives([-1,1,1,1]) changes list #to [-1,1,1,3] and returns it. (Note that zero is not considered to be a positive number).

unsure what question is asking- shouldnt the above response be (-1,1,1,2) ? or (-1, 0, 1, 2)?

In [ ]:

In [ ]:

Problem # 3

SumTotal - Create a function that takes a list as an argument and returns the sum of all the values in the list. For example sum_total([1,2,3,4]) should return 10

In [7]:
```
def sum_total(list):
    return sum(list)
```

In [8]:
```
example1= [-1,3,5,-5]
print(sum_total(example1))
```

```
2
```

In [9]:
```
example1= [1,3,5,7]
print(sum_total(example1))
```

```
16
```

In [10]:
```
def sum(list):
    counter = 0
    for number in range(len(list)):
        counter = counter + list[number]
    return counter
```

In [11]:
```
example1= [-1,3,5,-5]
print(sum(example1))
```

```
2
```

In [12]:
```
example1= [1,2,3,4]
print(sum(example1))
```

```
10
```

In [13]:
```
def sum_1(list):
    counter = 0
    for number in list:
        counter = counter + number
    return counter
```

In [14]:
```
example1= [1,21,3,4]
print(sum_1(example1))
```

```
29
```

Problem # 4

Average - Create a function that takes a list as an argument and returns the average of all the values in the list. For example multiples([1,2,3,4]) should return #2.5

In [15]:
```
def average(list):
    return sum(list)/len(list)
```

In [16]:
```
example1 = [2,4,6,8]
print(average(example1))
```

```
5.0
```

In [17]:
```
def average_1(list):
    sum=0
    for number in list:
        sum =sum + number
    return sum/len(list)
```

In [18]:
```
example1 = [2,4,6,8]
print(average(example1))
```

```
5.0
```

Problem # 5

Length - Create a function that takes a list as an argument and returns the length of the list. For example length([1,2,3,4]) should return 4

In [19]:
```
def length(list):
    return len(list)
```

In [20]:
```
example1 = [1,2,3,4]
print(length(example1))
```

```
4
```

In [21]:
```
example1 = [1,2,3,4,6,7,8,11,21]
print(length(example1))
```

```
9
```

In [ ]:

Problem # 6

Minimum - Create a function that takes a list as an argument and returns the minimum value in the list. If the passed list is empty, have the function return false. #For example minimum([1,2,3,4]) should return 1; minimum([-1,-2,-3]) should return -3.

In [22]:
```
def minimum(list):
    return min(list)
```

In [23]:
```
example1 = [1,2,3,4]
example2 = [-1,-2,-3]
#example3 = [ , ]
print(min(example1))
print(min(example2))
#print(min(example3))
```

```
1
-3
```

In [24]:
```
def minimum_1(list):
    if len(list) > 0:
        return min(list)
    else:
        return False
```

In [25]:
```
example1 = [1,2,3,4]
example2 = [-1,-2,-3]
example3 = [ ]
print(minimum_1(example1))
print(minimum_1(example2))
print(minimum_1(example3))
```

```
1
-3
False
```

In [26]:
```
def minimum_2(list):
    if list!=[]:
        return min(list)
    else:
        return False
```

In [27]:
```
example1 = [1,2,3,4]
example2 = [-1,-2,-3]
example3 = [ ]
print(minimum_2(example1))
print(minimum_2(example2))
print(minimum_2(example3))
```

```
1
-3
False
```

Problem # 7

Maximum - Create a function that takes a list as an argument and returns the maximum value in the list. If the passed list is empty, have the function return false. #For example maximum([1,2,3,4]) should return 4; maximum([-1,-2,-3]) should return -1.

In [28]:
```
def maximum(list):
    return max(list)
```

In [29]:
```
example1 = [1,2,3,4]
example2 = [-1,-2,-3]
#example3 = [ , ]
print(maximum(example1))
print(maximum(example2))
#print(maximum(example3))
```

```
4
-1
```

In [30]:
```
def maximum_1(list):
    if len(list) > 0:
        return max(list)
    else:
        return False
```

In [31]:
```
example1 = [1,2,3,4]
example2 = [-1,-2,-3]
example3 = [ ]
print(maximum_1(example1))
print(maximum_1(example2))
print(maximum_1(example3))
```

```
4
-1
False
```

In [32]:
```
def maximum_2(list):
    if list!=[]:
        return max(list)
    else:
        return False
```

In [33]:
```
example1 = [1,2,3,4]
example2 = [-1,-2,-3]
example3 = [ ]
print(maximum_2(example1))
print(maximum_2(example2))
print(maximum_2(example3))
```

```
4
-1
False
```

Problem # 8

Ultimateaalyze - Create a function that takes a list as an argument and returns a dictionary that has the sumTotal, average, minimum, maximum ad length of the list.

In [34]:
```
def stats(list):
    return max(list), min(list), (sum(list)/len(list)), sum(list)
```

In [35]:
```
example1 = [1,2,3,4]
```

In [36]:
```
example1 = [10,20,30,40]
example2 = [-15,25,35,-10]
example3 = [1,2,3,4,5]
print(stats(example1))
print(stats(example2))
print(stats(example3))
```

```
(40, 10, 25.0, 100)
(35, -15, 8.75, 35)
(5, 1, 3.0, 15)
```

In [ ]:
```

```

Problem # 9

ReverseList - Create a function that takes a list as a argument and return a list in a reversed order. Do this without creating a empty temporary list. For example #reverse([1,2,3,4]) should return [4,3,2,1]. This challenge is known to appear during basic technical interviews.

In [37]:
```
def reverse(list):
    return(list[::-1])
```

In [38]:
```
example1 = [1,2,3,4]
print(reverse(example1))
```

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

In [39]:
```
def reverse_1(list):
    #item = len(list) -1
    for item in range(len(list)):
        if len(list):
            print (list[-(item+1)])
```

In [40]:
```
example1 = [1,2,3,4]
print(reverse_1(example1))
```

```
4
3
2
1
None
```

In [41]:
```
def reverse_2(list):
    #item = len(list) -1
    for item in range(len(list)-1,-1,-1):
        print (list[item])
```

In [42]:
```
example1 = [1,2,3,4]
print(reverse_2(example1))
```

```
4
3
2
1
None
```

In [43]:
```
def reverse_3(list):
    for item in reversed(list):
        print(item)
```

In [44]:
```
example1 = [1,2,3,4]
print(reverse_3(example1))
```

```
4
3
2
1
None
```

In [45]:
```
def reverse_4(list):
    new_list=[]
    for item in range(len(list)-1,-1,-1):
          new_list.append(item)
    return new_list
```

In [46]:
```
example4 = [1,2,3,4]
print(reverse_4(example4))
```

```
[3, 2, 1, 0]
```

Problem # 10

Ispalindrome- Given a string, write a python function to check if it is palindrome or not. A string is said to be palindrome if the reverse of the string is the same as string. For example, “radar” is a palindrome, but “radix” is not a palindrome.

In [47]:
```
def palindrome(string):
    string = string
    new_string = string[::-1]
    if new_string == string:
        return True
    else:
        return False
```

In [48]:
```
string = 'radar'
print(palindrome(string))
string2 = 'Borrow or rob'
print(palindrome(string2))
```

```
True
False
```

In [49]:
```
def palindrome_1(string):
    string = string
    new_string = reversed(string)
    if list(new_string) == list(string):
        return True
    else:
        return False
```

In [50]:
```
string3 = 'madam'
print(palindrome_1(string3))
string2 = 'peaceful'
print(palindrome_1(string2))
```

```
True
False
```

In [51]:
```
def palindrome_2(string):
    string_lower = string.lower()
    string_lower_no_punctuation = ''.join(char for char in string_lower if char.isalnum(
    new_string = string_lower_no_punctuation[::-1]
    if new_string == string_lower_no_punctuation:
        return True
    else:
        return False
```

In [52]:
```
string = 'madam'
print(palindrome_2(string))
string2 = 'peaceful'
print(palindrome_2(string2))
string3 = 'Radar'
print(palindrome_2(string3))
string4 = 'Do geese see God?'
print(palindrome_2(string4))
string5 = 'Mr. Owl ate my metal worm!'
print(palindrome_2(string5))
string6 = '12 34 5432'
print(palindrome_2(string6))
string7 = '02-02/2020'
print(palindrome_2(string7))
```

```
True
False
True
True
True
False
True
```

In [53]:
```python
def palindrome_3(string):
    string_lower = string.lower()
    string_lower_no_punctuation = ''.join(char for char in string_lower if char.isalnum(
    new_string = reversed(string_lower_no_punctuation)
    if list(new_string) == list(string_lower_no_punctuation):
        return True
    else:
        return False
```

In [54]:
```python
string = 'madam'
print(palindrome_3(string))
string2 = 'peaceful'
print(palindrome_3(string2))
string3 = 'Radar'
print(palindrome_3(string3))
string4 = 'Do geese see God?'
print(palindrome_3(string4))
string5 = 'Mr. Owl ate my metal worm!'
print(palindrome_3(string5))
string6 = '12 34 5432'
print(palindrome_3(string6))
string7 = '02-02/2020'
print(palindrome_3(string7))
```

```
True
False
True
True
True
False
True
```

Problem # 11

Fizzbuzz- Create a function that will print numbers from 1 to 100, with certain exceptions:

If the number is a multiple of 3, print “Fizz” instead of the number.

If the number is a multiple of 5, print “Buzz” instead of the number.

If the number is a multiple of 3 and 5, print “FizzBuzz” instead of the number.

In [55]:
```
def fizzbuzz(number):
    while number >=1 and number <=100:
        if number %3 ==0 and number %5 ==0:
            number = "FizzBuzz"
        elif number %3 ==0:
            number= "Fizz"
        elif number %5 ==0:
            number= "Buzz"
        else:
            number = number
        return number
```

In [56]:
```
number1 = (33)
number2 = (15)
number3 = (13)
number4 = (25)
print(fizzbuzz(number1))
print(fizzbuzz(number2))
print(fizzbuzz(number3))
print(fizzbuzz(number4))
```

```
Fizz
FizzBuzz
13
Buzz
```

In [57]:
```
# if I wanted to return
for number in range(1,101):
    if number %3 ==0 and number %5 ==0:
        number = "FizzBuzz"
    elif number %3 ==0:
        number= "Fizz"
    elif number %5 ==0:
        number= "Buzz"
    else:
        number = number
    print(number)
```

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
31
32
Fizz
34
Buzz
Fizz
37
38
Fizz
Buzz
41
Fizz
43
44
FizzBuzz
46
47
Fizz
49
Buzz
Fizz
52
53
Fizz
Buzz
56
Fizz
58
59
FizzBuzz
61
62
```

```
Fizz
64
Buzz
Fizz
67
68
Fizz
Buzz
71
Fizz
73
74
FizzBuzz
76
77
Fizz
79
Buzz
Fizz
82
83
Fizz
Buzz
86
Fizz
88
89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

In [ ]:

Problem # 12

Fibonacci- The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, #starting from 0 and 1. That is,

F(0) = 0, F(1) = 1

F(n) = F(n - 1) + F(n - 2), for n > 1.

Create a function that accepts any number and will create a sequence based on the fibonacci sequence.

In [58]:
```
#assuming postive numbers
def Fibonacci(number):
    if number < 0:
        return False
    elif number ==0:
        number =0
    elif number ==1:
        number= 1
    else:
        number = (number - 1) + (number - 2)
    return number
```

In [59]:
```
number1 = (-5)
number2 = (0)
number3 = (1)
number4 = (5)
number5 = (30)
print(Fibonacci(-5))
print(Fibonacci(0))
print(Fibonacci(1))
print(Fibonacci(5))
print(Fibonacci(30))
```

```
False
0
1
7
57
```

In [ ]:

In [ ]: