Learning functions increases your Python skills exponentially

This also means that the difficulties of problems you can solve also increases drastically

Lets get some practice with converting problem statements into Python code

We will go through a series of Function Practice Exercises

Warm Up Section

Lesser of Two Evens: Write a function that returns the lesser of two given numbers if both numbers are even, but returns the greater if one or both numbers are odd.

lesser_of_two_evens(2,4) --> 2
lesser_of_two_evens(2,5) --> 5

In [4]:
```
def lesser_of_two_evens(a,b):
    if a%2 == 0 and b%2 == 0:
        return min(a,b)
    else:
        return max (a,b)
```

In [5]:
```
#Check
lesser_of_two_evens(2,4)
```

Out[5]:
```
2
```

In [6]:
```
#Check
lesser_of_two_evens(2,5)
```

Out[6]:
```
5
```

In [ ]:
```

```

ANIMAL CRACKERS: Write a function that takes a two-word string and returns True if both words begin with the same letter

animal_crackers('Lazy Llama') --> True
animal_crackers('Crunk Kangaroo') --> False

In [7]:
```
def animal_crackers(text):
    wordlist = text.split()
    return wordlist[0][0] == wordlist[1][0]
```

In [8]:
```
#Check
animal_crackers('Lazy Llama')
```

Out[8]:
```
True
```

In [9]:
```
#Check
animal_crackers('Crunk Kangaroo')
```

Loading [MathJax]/extensions/Safe.js

```
Out[9]: False
```

```
In [ ]:
```

```
In [ ]:
```

MAKES TWENTY: Given two integers, return True if the sum of the integers is 20 or if one of the intergers is 20. If not, return False

makes_twenty(20,10) --> True
makes_twenty(12,8) --> False
makes_twenty(2,3) --> False

```
In [10]: def makes_twenty(n1,n2):
             return (n1+n2)==20 or n1==20 or n2==20
```

```
In [11]: #Check
         makes_twenty(20,10)

Out[11]: True
```

```
In [12]: makes_twenty(12,8)

Out[12]: True
```

```
In [13]: makes_twenty(2,3)

Out[13]: False
```

```
In [ ]:
```

```
In [ ]:
```

LEVEL 1 PROBLEMS

```
In [14]: OLD MACDONALD: Write a function that capitalizes the first and fourth letters of a name

             old_macdonald('macdonald') --> MacDonald

         Note: 'macdonald' .capitalize() returns 'Macdonald'

           File "C:\Users\Keegz\AppData\Local\Temp/ipykernel_13500/3425918243.py", line 1
             OLD MACDONALD: Write a function that capitalizes the first and fourth letters of a nam
         e
                 ^
         SyntaxError: invalid syntax
```

```
In [15]: def old_macdonald(name):
             if len(name) > 3:
                 return name[:3] .capitalize() + name[3:].capitalize()
```

Loading [MathJax]/extensions/Safe.js

```
    else:
        return 'Name is too short!'
```

In [16]:
```
#Check
old_macdonald('macdonald')
```

Out[16]:
```
'MacDonald'
```

In [ ]:
```

```

In [ ]:
```

```

MASTER YODA: Given a sentence, return a sentence with the words reversed

master_yoda('I am home') --> 'home am I'
master_yoda('We are ready') --> 'ready are We'

In [17]:
```
def master_yoda(text):
    return ' '.join(text.split() [::-1])
```

In [18]:
```
#Check
master_yoda('I am home')
```

Out[18]:
```
'home am I'
```

In [19]:
```
master_yoda('We are ready')
```

Out[19]:
```
'ready are We'
```

In [ ]:
```

```

In [ ]:
```

```

ALMOST THERE: Given an interger n, return True if n is within 10 of either 100 or 200

almost_there(90) --> True
almost_there(104) --> True
almost_there(150) --> False
almost_there(209) --> True

NOTE: abs(num) returns the absolute value of a number

In [20]:
```
def almost_there(n):
    return ((abs(100 - n) <= 10) or (abs(200 - n) <= 10))
```

In [21]:
```
#Check
almost_there(90)
```

Loading [MathJax]/extensions/Safe.js

```
Out[21]:
```

```
In [22]:
#Check
almost_there(104)
```

```
Out[22]: True
```

```
In [23]:
#Check
almost_there(150)
```

```
Out[23]: False
```

```
In [24]:
#Check
almost_there(209)
```

```
Out[24]: True
```

```
In [ ]:
```

```
In [ ]:
```

LEVEL 2 PROBLEMS

FIND 33

Given a list of ints, return True if the array contains a 3 next to a 3 somewhere

has_33([1,3,3]) -> True

has_33 ([1,3,1,3]) -> False

has_33([3,1,3]) -> False

```
In [25]:
def has_33(nums):
    for i in range(0, len(nums)-1):

        #nicer looking alternative in commected code
        #if nums[i] == 3 and nums[i+1] == 3:

        if nums[i:i+2] == [3,3]:
            return True

    return False
```

```
In [28]:
#Check
has_33([1,3,3])
```

```
Out[28]: True
```

```
In [26]:
#Check
has_33([1,3,1,3])
```

```
Out[26]: False
```

Loading [MathJax]/extensions/Safe.js

In [27]:
```
#Check
has_33([3,1,3])
```

Out[27]:
```
False
```

In [ ]:
```

```

In [ ]:
```

```

PAPER DOLL: Given a string, return a string where for every character in the original there are three characters

paper_doll('Hello) --> 'HHHeeellllllooo'
paper_doll('Mississippi') --> 'MMMiiissssssiiissssssiiippppppiii'

In [32]:
```
def paper_doll(text):
    result = ''
    for char in text:
        result += char * 3
    return result
```

In [33]:
```
#Check
paper_doll('Hello')
```

Out[33]:
```
'HHHeeellllllooo'
```

In [34]:
```
#Check
paper_doll('Mississippi')
```

Out[34]:
```
'MMMiiissssssiiissssssiiippppppiii'
```

In [ ]:
```

```

In [ ]:
```

```

BLACKJACK: Given three integers between 1 and 11, if their sum is less than or equal to 21, return their sum.

If their sum exceeds 21 and theres an eleven, reduce the total sum by 10.

Finally, if the sum (even after adjustment) exceeds 21, return 'BUST'

blackjack(5,6,7) --> 18
blackjack(9,9,9) --> 'BUST'
blackjack(9,9,11) --> 19

In [35]:
```
def blackjack(a,b,c):
    if sum((a,b,c)) <= 21:
        return sum((a,b,c))
    elif sum((a,b,c)) <=31 and 11 in (a,b,c):
                 um ((a,b,c)) - 10
    else:
        return 'BUST'
```

In [36]:
```
#Check
blackjack(5,6,7)
```

Out[36]:
```
18
```

In [37]:
```
#Check
blackjack(9,9,9)
```

Out[37]:
```
'BUST'
```

In [38]:
```
#Check
blackjack(9,9,11)
```

Out[38]:
```
19
```

In [ ]:
```

```

SUMMER OF '69: Return the sum of the numbers in the array, except ignore sections of numbers starting with a 6 and extending to the next 9 (every 6 will be followed by at least one 9).

Return 0 for no numbers

summer_69([1,3,5}) ---> 9
summer_69([4,5,6,7,8,9]) --> 9
summer_69([2,1,6,9,11]) --> 14

In [47]:
```
def summer_69(arr):
    total = 0
    add = True
    for num in arr:
        while add:
            if num != 6:
                total += num
                break
            else:
                add = False
        while not add:
            if num != 9:
                    break
            else:
                    add = True
                    break
        return total
```

In [48]:
```
#Check

summer_69([1,3,5])
```

Out[48]:
```
1
```

Loading [MathJax]/extensions/Safe.js

```
summer_69([4,5,6,7,8,9])
```

Out[44]:
```
4
```

NEED DEBUGGING ^^^

In [ ]:

In [ ]:

In [ ]:

CHALLENGING PROBLEMS

SPY GAME: Write a function that takes in a list of integers and returns True if it contains 007 in order:

spy_game([1,2,4,0,0,7,5]) --> True
spy_game([1,0,2,4,0,5,7]) --> True
spy_game([1,7,2,0,4,5,0]) --> False

In [50]:
```
def spy_game(nums):

    code = [0,0,7,'x']

    for num in nums:
        if num == code[0]:
            code.pop(0)      #code.remove(num) also works

    return len(code) == 1
```

In [51]:
```
#Check

spy_game([1,2,4,0,0,7,5])
```

Out[51]:
```
True
```

In [52]:
```
spy_game([1,0,2,4,0,5,7])
```

Out[52]:
```
True
```

In [53]:
```
spy_game([1,7,2,0,4,5,0])
```

Out[53]:
```
False
```

In [ ]:

In [ ]:

Loading [MathJax]/extensions/Safe.js

COUNT PRIMIES: Write a fucntion that retusn the number of prime numbers that exist up to and including a given number

count_primes(100) ---> 25

In [57]:
```
def count_primes(num):
    primes = [2]
    x = 3
    if num < 2: #for the case of num =0 or 1
        return 0
    while x <= num:
        for y in range (3,x,2): #test all odd factors up to x-1
            if x%y == 0:
                x += 2
                break
        else:
            primes.append(x)
            x+= 2
    print(primes)
    return len(primes)
```

In [58]:
```
count_primes(100)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 8
9, 97]
```

Out[58]:
```
25
```

BONUS: Here is a faster version that makes use of the prime numbers we are collecting as we go!

In [59]:
```
def count_primes2(num):
    primes = [2]
    x = 3
    if num < 2:
        return 0
    while x <= num:
        for y in primes:       #use the primes list! :- )
            if x%y == 0:
                x+= 2
                break
        else:
            primes.append(x)
            x += 2
    print(primes)
    return len(primes)
```

In [60]:
```
count_primes2(100)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 8
9, 97]
```

Out[60]:
```
25
```

In [ ]:

Loading [MathJax]/extensions/Safe.js