Reworked Python Problems

1. Count the letters in a string

```
Write a function that takes a string as input and counts the occ
lowercase letter in the string. Return the counts in a dictionar
letters are keys and their counts are values.

letter_count('launchschool') #=> { ’a’: 1, ‘c’: 2, ‘h’: 2, ‘l’:
```

2. Count of Pairs

```
Write a function that takes a list of integers as input and coun
pairs in the list. A pair is defined as two equal integers separ
other integer(s).

Examples:
pairs([1, 2, 5, 6, 5, 2]) --> 2
pairs([1, 2, 2, 20, 6, 20, 2, 6, 2]) --> 4
```

3. Count Substring Instances

```
Write a function that takes two strings as input, `full_text` an
and returns the number of times `search_text` appears in `full_t

Examples:
solution('abcdeb','b') # should return 2 since 'b' shows up twic
solution('aaabbbcccc', 'bbb') # should return 1
```

4. Alphabet Symmetry

```
Write a function that takes a list of words as input and returns
integers. Each integer represents the count of letters in the wo
their positions in the alphabet.

Examples:
solve(["abode","ABc","xyzD"]) # should return [4, 3, 1]
solve(["abide","ABc","xyz"]) # should return [4, 3, 0]
```

5. Longest Chain of Vowels

```
Write a function that takes a lowercase string as input and retu
length of the longest substring that consists entirely of vowels

Examples:
solve("roadwarriors") # should return 2
solve("suoidea") # should return 3
```

6. Odd Number Sub-strings

```
Write a function that takes a string of integers as input and re
number of substrings that result in an odd number when convertec

Examples:
solve("1341") # should return 7
solve("1357") # should return 10
```

7. The Nth Char

```
Write a function that takes a list of words and constructs a new
concatenating the nth letter from each word, where n is the posi
word in the list.

Example:
nth_char(['yoda', 'best', 'has']) # should return 'yes'
```

8. Smallest Substring Repeat

```
Write a function that takes a non-empty string `s` as input and
minimum substring `t` and the maximum number `k`, such that the
`s` is equal to `t` repeated `k` times.

Examples:
f("ababab") # should return ["ab", 3]
```

9. Typoglycemia Generator

```
Write a function that generates text following a pattern where:
1) the first and last characters of each word remain in their or
2) characters between the first and last characters are sorted a
3) punctuation should remain at the same place as it started

Examples:
scramble_words('professionals') # should return 'paefilnoorsss'
scramble_words("you've gotta dance like there's nobody watching,
```

10. Most Frequent Words

```
Write a function that, given a string of text, returns a list of
occurring words, in descending order of the number of occurrence
```

```
Assumptions:
- A word is a string of letters (A to Z) optionally containing
- Matches should be case-insensitive.
- Ties may be broken arbitrarily.
- If a text contains fewer than three unique words, then either

Examples:
top_3_words(" , e .. ") # ["e"]
top_3_words(" ... ") # []
top_3_words(" ' ") # []
top_3_words(" ''' ") # []
top_3_words("""In a village of La Mancha, the name of which I ha
mind, there lived not long since one of those gentlemen that ke
in the lance-rack, an old buckler, a lean hack, and a greyhound
coursing. An olla of rather more beef than mutton, a salad on m
nights, scraps on Saturdays, lentils on Fridays, and a pigeon o
on Sundays, made away with three-quarters of his income.""") #
```

11. Extract the domain name from a URL

```
Write a function that, given a URL as a string, parses out just
name and returns it.

Examples:
domain_name("http://github.com/carbonfive/raygun") # should ret
domain_name("https://www.cnet.com") # should return "cnet"
```

12. Detect the Pangram

```
A pangram is a sentence that contains every single letter of th
least once. Given a string, detect whether or not it is a pangr
```

```
Return True if it is, False if not. Ignore numbers and punctuati

Examples:
panagram?("The quick brown fox jumps over the lazy dog.") # shou
panagram?("This is not a pangram.") # should return False
```

13. Kebabize a String

```
Modify the kebabize function so that it converts a camel case st
kebab case. Kebab case separates words with dashes '-'; camel ca
separate words by upcasing the first character in each new word

Examples:
kebabize('camelsHaveThreeHumps') # should return 'camels-have-th
kebabize('myCamelHas3Humps') # should return 'my-camel-has-humps
```

14. Dubstep

```
Write a function to decode a dubstep string to its original form
may begin and end with one or more "WUB"s and there will be at l
possibly more) "WUB"s between each word.
The input consists of a single non-empty string, consisting only
English letters.

Examples:
song_decoder("WUBWEWUBAREWUBWUBTHEWUBCHAMPIONSWUBMYWUBFRIENDWUB"
```

15. Take a Walk

```
You live in the city of Cartesia where all roads are laid out
in a perfect grid. You arrived ten minutes too early to an ap
pointment, so you decided to take the opportunity to go for a
short walk. The city provides its citizens with a Walk Genera
ting App on their phones -- every time you press the button i
t sends you a list of one-letter strings representing directi
ons to walk (e.g., ['n', 's', 'w', 'e']). You always walk onl
y a single block in a direction, and you know it takes you on
e minute to traverse one city block. Create a function that w
ill return `True` if the walk the app gives you will take you
exactly ten minutes (you don't want to be early or late!) and
will, of course, return you to your starting point. Return `F
alse` otherwise.

Note: You will always receive a valid list containing a rando
m assortment of direction letters ('n', 's', 'e', or 'w' onl
y). It will never give you an empty list (that's not a walk,
that's standing still!).

Examples:
is_valid_walk(['n','s','n','s','n','s','n','s','n','s']) # sh
ould return True
is_valid_walk
(['w','e','w','e','w','e','w','e','w','e','w','e']) # should
return False
is_valid_walk(['w']) # should return False
is_valid_walk(['n','n','n','s','n','s','n','s','n','s']) # sh
ould return F
```

16. Spin Words

```
Write a function that takes in a string of one or more words
and returns the same string, but with all words of five or mo
```

```
re letters reversed. Strings passed in will consist of only l
etters and spaces. Spaces will be included only when more tha
n one word is present.

Examples:
spin_words("Hey fellow warriors") # should return "Hey wollef
sroirraw"
spin_words("This is a test") # should return "This is a test"
spin_words("This is another test") # should return "This is r
ehtona test"
```

17. Expanded Form of Number

```
You will be given a number, and you need to return it as a stri
expanded form. For example:

expanded_form(12) # should return '10 + 2'
expanded_form(42) # should return '40 + 2'
expanded_form(70304) # should return '70000 + 300 + 4'

Note: All numbers will be whole numbers greater than 0.
```

18. Multiplicative Persistence

```
Write a function, persistence, that takes in a positive paramete
`num` and returns its multiplicative persistence, which is the 
of times you must multiply the digits in `num` until you reach 

Examples:
persistence(39) # should return 3, because 3*9=27, 2*7=14, 1*4=
# and 4 has only one digit
persistence(999) # should return 4, because 9*9*9=729, 7*2*9=12
# 1*2*6=12, and finally 1*2=2
```

```
persistence(4) # should return 0, because 4 is already a one-dig
persistence(25) # should return 2, because 2*5=10, and 1*0=0
```

19. Title-ize

```
A string is considered to be in title case if each word in the s
a) Capitalized (that is, only the first letter of the word is in
b) Considered to be an exception and put entirely into lower cas

Write a function that will convert a string into title case, giv

Examples:
title_case('a clash of KINGS', 'a an the of') # should return 'A
title_case('THE WIND IN THE WILLOWS', 'The In') # should return
title_case('the quick brown fox') # should return 'The Quick Bro
```

20. Character Count Sorting

```
Write a function that takes a string as an argument and groups t
number of times each character appears in the string as a dictio
sorted by the highest number of occurrences.

The characters should be sorted alphabetically, and you should i
spaces, special characters, and count uppercase letters as lower

Examples:
get_char_count("Mississippi") # should return {4: ['i', 's'], 2
get_char_count("Hello. Hello? HELLO!!") # should return {6: ['l
get_char_count("aaa...bb...c!") # should return {3: ['a'], 2: [
get_char_count("aaabbbccc") # should return {3: ['a', 'b', 'c']}
get_char_count("abc123") # should return {1: ['1', '2', '3', 'a
```

21. Mine Location

```
You've just discovered a square (NxN) field and you notice a war
The sign states that there's a single bomb in the 2D grid-like f
of you.

Write a function `mine_location` that accepts a 2D array, and re
location of the mine. The mine is represented as the integer 1 i
Areas in the 2D array that are not the mine will be represented 

The location returned should be an array where the first element

Examples:
mine_location([[1, 0, 0], [0, 0, 0], [0, 0, 0]]) # should return
mine_location([[0, 0, 0], [0, 1, 0], [0, 0, 0]]) # should return
mine_location([[0, 0, 0], [0, 0, 0], [0, 1, 0]]) # should return
mine_location([[1, 0], [0, 0]]) # should return [0, 0]
mine_location([[1, 0, 0], [0, 0, 0], [0, 0, 0]]) # should return
mine_location([[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 1, 0], [0, 0, 
```

22. Substring is Anagram?

```
Write a function `scramble(str1, str2)` that returns `True` if a
`str1` characters can be rearranged to match `str2`, otherwise r

Notes:
- Only lower case letters will be used (a-z). No punctuation or 
    be included.
- Performance needs to be considered.
- Input strings `str1` and `str2` are null terminated.

Examples:
```

```
scramble('rkqodlw', 'world') # should return True
scramble('cedewaraarossoqqyt', 'carrot') # should return True
scramble('katas', 'steak') # should return False
scramble('scriptjava', 'javascript') # should return True
scramble('scriptingjava', 'javascript') # should return True
```

23. Longest alphabetical substring

```
Write a function `longest(s)` that finds and returns the longest
`s` where the characters are in alphabetical order.

Example:
longest('asd')                  # should return 'as'
longest('nab')                  # should return 'ab'
longest('abcdeapbcdef')         # should return 'abcde'
longest('asdfaaaabbbbcttavvfffffdf') # should return 'aaaabbbbct
longest('asdfbyfgiklag')        # should return 'fgikl'
longest('z')                    # should return 'z'
longest('zyba')                 # should return 'z'
```

24. Generate Hashtags

```
Write a function `generate_hashtag(s)` that generates a hashtag

Rules:
- The hashtag must start with a '#' symbol.
- All words in the hashtag must start with a capital letter.
- If the resulting hashtag is longer than 140 characters, the fu
- If the input string or the resulting hashtag is an empty strin

Examples:
generate_hashtag("")                       # should return `Fals
generate_hashtag(" " * 200)                # should return `Fals
generate_hashtag("Do We have A Hashtag")   # should return "#DoV
generate_hashtag("Nice To Meet You")       # should return "#Nic
generate_hashtag("this is a test")         # should return "#Thi
generate_hashtag("this is a very long string" + " " * 140 + "enc
generate_hashtag("a" * 139)                # should return "#A"
generate_hashtag("a" * 140)                # should return `Fals
```

25. How many cakes can the baker make?

```
# Pete is baking cakes and needs help calculating how many he ca
# Write a function cakes() that takes two dictionaries: the reci

# Rules:
# - Ingredients not present in the objects can be considered as

# must return 2
cakes({"flour"=>500, "sugar"=>200, "eggs"=>1},{"flour"=>1200, "s

# must return 11
cakes({"cream"=>200, "flour"=>300, "sugar"=>150, "milk"=>100, "c
"milk"=>20000, "oil"=>30000, "cream"=>5000}) == 11

# must return 0
cakes({"apples"=>3, "flour"=>300, "sugar"=>150, "milk"=>100, "oi
"milk"=>2000}) == 0

# must return 0
cakes({"apples"=>3, "flour"=>300, "sugar"=>150, "milk"=>100, "oi
"milk"=>2000, "apples"=>15, "oil"=>20}) == 0

# must return 0
cakes({"eggs"=>4, "flour"=>400},{}) == 0

# must return 1
```

```
cakes({"cream"=>1, "flour"=>3, "sugar"=>1, "milk"=>1, "oil"=>1,
"cream"=>1, "oil"=>1, "milk"=>1}) == 1
```

26. Mean Square

```
# Create a function that takes two integer arrays of equal lengt
# squares the absolute value difference between those two values

# Examples
# [1, 2, 3], [4, 5, 6] --> 9 because (9 + 9 + 9) / 3
# [10, 20, 10, 2], [10, 25, 5, -2] --> 16.5 because (0 + 25 + 25
# [-1, 0], [0, -1] --> 1 because (1 + 1) / 2

p solution([1, 2, 3], [4, 5, 6]) == 9
p solution([10, 20, 10, 2], [10, 25, 5, -2]) == 16.5
p solution([-1, 0], [0, -1]) == 1
```

27. List Anagrams

```
# Write a function that finds all the anagrams of a word from a
# Two words are anagrams of each other if they both contain the

# Examples
# 'abba' & 'baab' == true
# 'abba' & 'bbaa' == true
# 'abba' & 'abbba' == false
# 'abba' & 'abca' == false

p anagrams('abba', ['aabb', 'abcd', 'bbaa', 'dada']) == ['aabb',
p anagrams('racer', ['crazer', 'carer', 'racar', 'caers', 'racer
p anagrams('laser', ['lazing', 'lazy', 'lacer']) == []
```

28. Group by 2 chars

```
# Write a function that splits the string into pairs of two char
# If the string contains an odd number of characters, replace th

p solution('abc') == ['ab', 'c_']
p solution('abcdef') == ['ab', 'cd', 'ef']
p solution("abcdef") == ["ab", "cd", "ef"]
p solution("abcdefg") == ["ab", "cd", "ef", "g_"]
p solution("") == []
```

29. Anagram Difference Count

```
# Given two words, determine the number of letters you need to r

p anagram_difference('', '') == 0
p anagram_difference('a', '') == 1
p anagram_difference('', 'a') == 1
p anagram_difference('ab', 'a') == 1
p anagram_difference('ab', 'ba') == 0
p anagram_difference('ab', 'cd') == 4
p anagram_difference('aab', 'a') == 2
p anagram_difference('a', 'aab') == 2
```

30. Is anagram?

```
# Write a function to determine if two words are anagrams of eac

p is_anagram('Creative', 'Reactive') == true
p is_anagram("foefet", "toffee") == true
p is_anagram("Buckethead", "DeathCubeK") == true
```

```
p is_anagram("Twoo", "WooT") == true
p is_anagram("dumble", "bumble") == false
```

31. Highest Scoring Word

```
# Find the highest scoring word in a string.
# Each letter scores points based on its position in the alphabe
# Return the highest scoring word. If two words score the same,

p high('man i need a taxi up to ubud') == 'taxi'
p high('what time are we climbing up the volcano') == 'volcano'
p high('take me to semynak') == 'semynak'
p high('aaa b') == 'aaa'
```

32. Replace Char with Score

```
# Given a string, replace every letter with its position in the
# If anything in the text isn't a letter, ignore it and don't re

p alphabet_position("The sunset sets at twelve o' clock.") == "2
p alphabet_position("-.-'") == ""
```

33. Find the Suspect

```
# Sherlock has to find suspects on his latest case. He will use
# Suspect in this case is a person which has something not allo
# pockets.
# Allowed items are defined by an array of numbers.
# Pockets contents are defined by map entries where key is a per
# value is one or few things represented by an
# array of numbers (can be nil or empty array if empty).

pockets = {
    'bob': [1],
    'tom': [2, 5],
    'jane': [7]
}

def find_suspects(pockets, allowed_items):
    if not pockets:
        return None

    suspects = [person for person, items in pockets.items() if a

    return suspects if suspects else None

p find_suspects(pockets, [1, 2]) == ['tom', 'jane']
p find_suspects(pockets, [1, 7, 5, 2]) == None
p find_suspects(pockets, []) == ['bob', 'tom', 'jane']
p find_suspects(pockets, [7]) == ['bob', 'tom']
```

34. Do the Wave

```
# Create a function that turns a string into a Wave. You will be
# and you must return that string in an array where an uppercase

p wave("hello") == ["Hello", "hEllo", "heLlo", "helLo", "hellO"]
p wave("") == []
p wave("two words") == ["Two words", "tWo words", "twO words", '
p wave(" gap ") == [" Gap ", " gAp ", " gaP "]
```

35. Delete a Digit

```
# Given an integer n, find the maximal number you can obtain by
# exactly one digit of the given number.

p delete_digit(152) == 52
p delete_digit(1001) == 101
p delete_digit(10) == 1
```

36. Largest Product in a series

```
# Complete the greatestProduct method so that it'll find the gre
# in the given string of digits.

def greatest_product(n):
    pass

p greatest_product("123834539327238239583") == 3240
p greatest_product("395831238345393272382") == 3240
p greatest_product("92494737828244222221111111532909999") == 529
p greatest_product("92494737828244222221111111532909999") == 529
p greatest_product("02494037820244202221011110532909999") == 0
```

37. Encode Duplicates

```
# The goal of this exercise is to convert a string to a new stri
# is "(" if that character appears only once in the original str
# more than once in the original string. Ignore capitalization w


p duplicate_encode("din") == "((("
p duplicate_encode("recede") == "()()()"
p duplicate_encode("Success") == ")())())"
p duplicate_encode("(( @") == "))(("
```

38. Update string

```
# Assume "#" is like a backspace in string. This means that stri
# Your task is to process a string with "#" symbols and return t

p clean_string('abc#d##c') == "ac"
p clean_string('abc####d##c#') == ""
```

39. Sort Arrays (Case-Insensitive)

```
# Sort the given strings in alphabetical order, case insensitive

p sortme(["Hello", "there", "I'm", "fine"]) == ["fine", "Hello",
p sortme(["C", "d", "a", "Ba", "be"]) == ["a", "Ba", "be", "C",
```

40. Difference of Sum from Next Prime Number

```
# Given a List [] of n integers, find the minimum number to be i
# in the list, so that the sum of all elements of the list shoul
# equal the closest prime number.



p minimum_number([3,1,2]) == 1
p minimum_number([5,2]) == 0
p minimum_number([1,1,1]) == 0
```

```
p minimum_number([2,12,8,4,6]) == 5
p minimum_number([50,39,49,6,17,28]) == 2
```

41. Counting Duplicates

```
# Count the number of Duplicates
# Write a function that will return the count of distinct case-i
# alphabetic characters and numeric digits that occur more than

p duplicate_count("") == 0
p duplicate_count("abcde") == 0
p duplicate_count("abcdeaa") == 1
p duplicate_count("abcdeaB") == 2
p duplicate_count("Indivisibilities") == 2
```

42. Find the Parents

```
# Mothers arranged a dance party for the children in school. At
# there are only mothers and their children. All are having grea
# dance floor when suddenly all the lights went out. It's a dark
# one can see each other. But you were flying nearby and you can
# dark and have ability to teleport people anywhere you want.
# Legend:
# - Uppercase letters stands for mothers, lowercase stand for th
# i.e. "A" mother's children are "aaaa".
# - Function input: String contains only letters, uppercase lett

p find_children("abBA") == "AaBb"
p find_children("AaaaaZazzz") == "AaaaaaZzzz"
p find_children("CbcBcbaA") == "AaBbbCcc"
p find_children("xXfuUuuF") == "FfUuuuXx"
p find_children("") == ""
```

43. Digit Power Play

```
# Some numbers have funny properties. For example:
# 89 --> 8¹ + 9² = 89 * 1
# 695 --> 6² + 9³ + 5⁴= 1390 = 695 * 2
# 46288 --> 4³ + 6⁴+ 2⁵ + 8⁶ + 8⁷ = 2360688 = 46288 * 51
# Given a positive integer n written as abcd... (a, b, c, d...
# and a positive integer p we want to find a positive integer k,
# such as the sum of the digits of n taken to the successive pow
# equal to k * n.
# In other words:
# Is there an integer k such as : (a ^ p + b ^ (p+1) + c ^(p+2)
# = n * k
# If it is the case we will return k, if not return -1.
# Note: n and p will always be given as strictly positive integ

p dig_pow(89, 1) == 1
p dig_pow(92, 1) == -1
p dig_pow(46288, 3) == 51
p dig_pow(695, 2) == 2
```

44. Squared Array Check

```
# Given two arrays a and b write a function comp(a, b) that chec
# the two arrays have the "same" elements, with the same multipl
# "Same" means, here, that the elements in `b` are the elements
# regardless of the order.

p comp([121, 144, 19, 161, 19, 144, 19, 11], [121, 14641, 20736,
p comp([121, 144, 19, 161, 19, 144, 19, 11], [132, 14641, 20736,
p comp(None, [1, 2, 3]) == False
p comp([1, 2], []) == False
p comp([1, 2], [1, 4, 4]) == False
```

45. Count Digit Occurences

```
# Your goal is to write the group_and_count method, which shoul
# as a unique parameter and return a hash. Empty or nil input mu
# instead of a hash. This hash returned must contain as keys the
# of the array, and as values the counting of each value.



p group_and_count([1,1,2,2,2,3]) == {1: 2, 2: 3, 3: 1}
p group_and_count([]) == None
p group_and_count(None) == None
p group_and_count([1, 7, 5, -1]) == {1: 1, 7: 1, 5: 1, -1: 1}
```

46. Triple double

```
# Write a function triple_double(num1, num2) which takes number
# and returns 1 if there is a straight triple of a number at an
# and also a straight double of the same number in num2. If thi
# return 0



p triple_double(12345, 12345) == 0
p triple_double(666789, 12345667) == 1 # 3 straight 6's in num1
```

47. Find the missing letter

```
# Write a method that takes an array of consecutive (increasing
# and that returns the missing letter in the array.
# You will always get an valid array. And it will be always exa
# The length of the array will always be at least 2.
# The array will always contain letters in only one case.
# Example:
```

```
# ['a','b','c','d','f'] -> 'e'
# ['O','Q','R','S'] -> 'P'

p find_missing_letter(['a','b','c','d','f']) == 'e'
p find_missing_letter(['O','Q','R','S']) == 'P'
```

48. Reverse and combine text

```
# Your task is to Reverse and Combine Words.
# Input: String containing different "words" separated by spaces
# 1. More than one word? Reverse each word and combine first wit
# (odd number of words => last one stays alone, but has to be re
# 2. Start it again until there's only one word without spaces
# 3. Return your result…

p reverse_and_combine_text("abc def") == "cbafed"
p reverse_and_combine_text("abc def ghi jkl") == "defabcjklghi"
p reverse_and_combine_text("dfghrtcbafed") == "dfghrtcbafed"
p reverse_and_combine_text("234hh54 53455 sdfqwzrt rtteetrt hjhj
"trzwqfdstrteettr45hh4325543544hjhjh21lllll"
p reverse_and_combine_text("sdfsdf wee sdffg 342234 ftt") == "g
```