Date: 20-01-2020

# Practice Problem Statements Set 1

1. Create a function that takes a number as an argument and returns True or False depending on whether the number is symmetrical or not. A number is symmetrical when it is the same as its reverse.

   **Input:** 454 **Output:** True

   **Input:** 9939 **Output:** False

   **Input:** 1112111 **Output:** True

2. Create a function that takes a single string as argument and returns an ordered list containing the indices of all capital letters in the string.

   **Input:** "eDaBiT" **Output:** [1,3,5]

   **Input:** "determine" **Output:** []

   **Input:** "STRIKE" **Output:** [0,1,2,3,4,5]

3. Write a function grab_city to return the city from each of these vacation spots. The city will always be in the last bracket pair.

   **Input:** "[Last Day!] Beer Festival [Munich]" **Output:** "Munich"

   **Input:** "Cheese Factory Tour [Portland]" **Output:** "Portland"

   **Input:** "[50% Off!][Group Tours Included] 5-Day Trip to Onsen [Kyoto]" **Output:** "Kyoto"

4. Write a function that takes two lists and adds the first element in the first list with the first element in the second list, the second element in the first list with the second element in the second list, etc, etc. Return True if all element combinations add up to the same number. Otherwise, return False.

   **Input:** [1, 2, 3, 4], [4, 3, 2, 1] **Output:** True

   **Input:** [1, 2], [-1, -1] **Output:** False

   **Input:** [9, 8, 7], [7, 8, 9, 10] **Output:** False

5. Given a list with an odd number of elements, return whether the scale will tip 'left' or 'right' based on the sum of the numbers. The scale will tip on the direction of the largest total. If both sides are equal, return 'balanced'.

   **Input:** [0, 0, 'I', 1, 1 ]  **Output:** Right

   **Input:** [1, 2, 5, 'I', 4, 1,0 ] **Output:** Left

   **Input:** [5, 5, 5, 0, 'I', 10, 2, 2, 1 ] **Output:** Balanced

6. Create a function that returns all combinations of size n from a list. Sort the list in ascending lexicographical order.

   **Input:** [1, 2, 3, 4], 1  **Output:** [[1], [2], [3], [4]]

   **Input:** [2, 1, 4, 3], 2  **Output:** [[1, 2], [1, 3], [1, 4], [2, 3], [2, 4], [3, 4]]

   **Input:** [1, 2, 3, 4], 5  **Output:** []

   **Input:** [1, 2, 3, 4], 0  **Output:** [[]]

7. You work for a manufacturer, and have been asked to calculate the total profit made on the sales of a product. You are given a dictionary containing the cost price per unit (in dollars), sell price per unit (in dollars), and the starting inventory. Return the total profit made, rounded to the nearest dollar. Assume all of the inventory has been sold.

   **Input:** {"cost_price": 32.67, "sell_price": 45.00, "inventory": 1200}

   **Output :** 14796

8. Create a function that returns the smallest number of letter removals so that two strings are anagrams of each other.

   An anagram is any string that can be formed by shuffling the characters of the original string.

   For example: baedc is an anagram of abcde.

   An empty string can be considered an anagram of itself.

   Characters won't be used more than once per string.

   **Input:** "abcde", "cab"  **Output:** 2

   **Input:** "deafk", "kfeap"  **Output:** 2

   **Input:** "acb", "ghi"  **Output:** 6

9. Create a function that takes: A list of keys, A list of values (same size), True, if key and value should be swapped, else False. The function returns the constructed dict. Empty lists return an empty dict.

   **Input:** [1, 2, 3], ["one", "two", "three"], False

   **Output:** { 1: "one", 2: "two", 3: "three" }

   **Input:** [1, 2, 3], ["one", "two", "three"], True

   **Output:** { "one": 1, "two": 2, "three": 3 }

10. A "magic square" is a square divided into smaller squares each containing a number, such that the numbers in each vertical, horizontal, and diagonal row add up to the same value. Write a function that takes a 2D list, checks if it's a magic square and returns either True or False.

    Input: [[8, 1, 6], [3, 5, 7], [4, 9, 2]] **Output:** True

11. Create a function that returns True if the given circles are intersecting, otherwise return False. The circles are given as two lists containing the values in the following order: Radius of the circle, Center position on the x-axis, Center position on the y-axis.

    **Input:** [10, 0, 0], [10, 10, 10] **Output:** True

    **Input:** [1, 0, 0], [1, 10, 10] **Output:** False

12. Create a function that calculates what percentage of the box is filled in. Give your answer as a string percentage rounded to the nearest integer.

    **Input:** ["####", "#  #", "#o #", "####"] **Output:** 25%

    **Input:** [ "#######", "#o oo #", "#######"] **Output:** 60%

    **Input:** ["######", "#ooo #," "#oo  #," "#    #," "#    #," "######"] **Output:** 31%

13. Create a function that accepts a string, checks if it's a valid email address and returns either True or False.

    1. The string must contain an @ character.

    2. The string must contain a . character.

    3. The @ must have at least one character in front of it.

    e.g. "e@edabit.com" is valid while "@edabit.com" is invalid.

    4. The . and the @ must be in the appropriate places.

    e.g. "hello.email@com" is invalid while "john.smith@email.com" is valid.

14. Your job is to create a function, that takes 3 numbers: a, b, c and returns True if the last digit of (the last digit of a * the last digit of b) = the last digit of c.

    **Input:** 25, 21, 125 **Output:** True

    **Input:** 55, 226, 5190 **Output:** True

    **Input:** 12, 215, 2142 **Output:** False

15. Given an integer limit being the upper limit of the range of interest, implement a function that returns the last 15 palindromes numbers lower or equal to limit as a list sorted ascending.

    **Input:** 151 **Output:** [11, 22, 33, 44, 55, 66, 77, 88, 99, 101, 111, 121, 131, 141, 151]

    **Input:** 600 **Output:** [454, 464, 474, 484, 494, 505, 515, 525, 535, 545, 555, 565, 575, 585, 595]

    **Input:** 999999 **Output:** [985589, 986689, 987789, 988889, 989989, 990099, 991199, 992299, 993399, 994499, 995599, 996699, 997799, 998899, 999999]

16. To train for an upcoming marathon, Johnny goes on one long-distance run each Saturday. He wants to track how often the number of miles he runs exceeds the previous Saturday. This is called a progress day.

    Create a function that takes in a list of miles run every Saturday and returns Johnny's total number of progress days.

    **Input:** [3, 4, 1, 2] **Output:** 2

    **Input:** [10, 11, 12, 9, 10] **Output:** 3

    **Input:** [6, 5, 4, 3, 2, 9] **Output:** 1

    **Input:** [9,9] **Output:** 0

17. Create a function that tests whether or not an integer is a perfect number. A perfect number is a number that can be written as the sum of its factors, excluding the number itself.

    e.g. - 6 is a perfect number, since 1 + 2 + 3 = 6

    **Input:** 6 **Output:** True

    **Input:** 28 **Output:** True

    **Input:** 24 **Output:** False

18. A pandigital number contains all digits (0-9) at least once. Write a function that takes an integer, returning True if the integer is pandigital, and False otherwise.

    **Input:** 98140723568910 **Output:** True

    **Input:** 90864523148909 **Output:** False

    **Input:** 112233445566778899 **Output:** False

19. The Atbash cipher is an encryption method in which each letter of a word is replaced with its "mirror" letter in the alphabet: A <=> Z; B <=> Y; C <=> X; etc.

    Create a function that takes a string and applies the Atbash cipher to it.

    Capitalisation should be retained.

    Non-alphabetic characters should not be altered.

20. Return the coordinates ([row, col]) of the element that differs from the rest.

    **Input:** [["A", "A", "A"],["A", "A", "A"],["A", "B", "A"]]

    **Output:** [3, 2]

    **Input:** [["c", "c", "c", "c"], ["c", "c", "c", "d"]]

    **Output:** [2, 4]