Advance Python Questions

1) Write a function that returns True if two lists, when combined, form a consecutive sequence.

- consecutive_combo([7, 4, 5, 1], [2, 3, 6]) → True
- consecutive_combo([1, 4, 6, 5], [2, 7, 8, 9]) → False
- consecutive_combo([1, 4, 5, 6], [2, 3, 7, 8, 10]) → False
- consecutive_combo([44, 46], [45]) → True

2) Create a function that takes a list of numbers or strings and returns a list with the items from the original list stored into sublists. Items of the same value should be in the same sublist.

- advanced_sort([2, 1, 2, 1]) → [[2, 2], [1, 1]]
- advanced_sort([5, 4, 5, 5, 4, 3]) → [[5, 5, 5], [4, 4], [3]]
- advanced_sort(["b", "a", "b", "a", "c"]) → [["b", "b"], ["a", "a"], ["c"]]

3) This is a companion to my previous challenge Numbers to English.
Given an English description of an integer in the range 0 to 999, devise a function that returns the integer in numeric form.
Examples:-

- eng2nums('four') → 4
- eng2nums('forty') → 40
- eng2nums('six hundred') → 600
- eng2nums('one hundred fifteen') → 115
- eng2nums('seven hundred sixty seven') → 767

4) Imagine you took all the numbers between 0 and n and concatenated them together into a long string.
How many digits are there between 0 and n? Write a function that can calculate this.
There are 0 digits between 0 and 1, there are 9 digits between 0 and 10 and there are 189 digits between 0 and 100.

EXAMPLES ->

- digits(1) → 0
- digits(10) → 9
- digits(100) → 189
- digits(2020) → 6969

5) Create a function that returns the characters from a list or string r on odd or even positions, depending on the specifier s.

The specifier will be "odd" for items on odd positions (1, 3, 5, ...) and "even" for items on even positions (2, 4, 6, ...).

EXAMPLES-

--> char_at_pos([2, 4, 6, 8, 10], "even") ➞ [4, 8]

--> 4 & 8 occupy the 2nd & 4th positions

--> char_at_pos("EDABIT", "odd") ➞ "EAI"

--> "E", "A" and "I" occupy the 1st, 3rd and 5th positions

--> char_at_pos(["A", "R", "B", "I", "T", "R", "A", "R", "I", "L", "Y"], "odd") ➞ ["A", "B", "T", "A", "I", "Y"]

6) you are given three inputs: a string, one letter, and a second letter.

Write a function that returns True if every instance of the first letter occurs before every instance of the second letter.

Examples:-

--> first_before_second("a rabbit jumps joyfully", "a", "j") ➞ True
--> Every instance of "a" occurs before every instance of "j".

--> first_before_second("knaves knew about waterfalls", "k", "w") ➞ True
--> first_before_second("happy birthday", "a", "y") ➞ False
--> The "a" in "birthday" occurs after the "y" in "happy".
--> first_before_second("precarious kangaroos", "k", "a") ➞ False

7) Given a list of coins, father needs to distribute them amongst his three children.

Write a function to determine if the coins can be distributed equally or not.

Return True if each son receives the same amount of money, otherwise return False.

EXAMPLES

[1, 2, 3, 2, 2, 2, 3] ➞ True
Amount to be distributed to each child = (1+2+3+2+4+3)/3 => 15/3 => 5

Possible set of coin to be distributed to children = [(1,2,2),(2,3),(2,3)]
[5, 3, 10, 1, 2] → False

Amount to be distributed to each child = (5+3+10+1+2)/3 => 21/3 => 7
But there are no combination such that each child get equal value which is 7.

coins_div([2, 4, 3, 2, 4, 9**, 7, 8, 6, 9]) → True**

8) In this challenge, you have to establish if the digits of a given number form a sequence (ascending or descending).

Given an integer n, implement a function that returns a string:

- "Metadrome" if the digits of n form an ascending sequence without repeating digits.
- "Plaindrome" if the digits of n form an ascending sequence with repeating digits.
- "Katadrome" if the digits of n form a descending sequence without repeating digits.

- "Nialpdrome" if the digits of n form a descending sequence with repeating digits.
- "Repdrome" if n contains a single repeating digit.
- "Nondrome" if none of the above conditions is true.

9) Create a function that takes an integer argument and returns a list of prime numbers found in the decimal representation of that number.

For example, extract_primes(1717) returns [7, 7, 17, 17, 71].

The list should be in acending order. If a prime number appears more than once, every occurance should be listed.

If no prime numbers are found, return an empty list.

Examples

- extract_primes(1) ➞ []
- extract_primes(7) ➞ [7]
- extract_primes(73) ➞ [3, 7, 73]
- extract_primes(103) ➞ [3]
- extract_primes(1313) ➞ [3, 3, 13, 13, 31, 131, 313]

10) Scoring plays in American football count as either 2, 3, 6, 7, or 8 points.

Write a function that has as it's argument a football score and returns the number of possible ways that score can be achieved.

Order is not important.

Examples

- football(4) → 1
- # 2+2

- football(6) → 3
- # 2+2+2 or 3+3 or 6

- football(7) → 2
- # 2+2+3 or 7

- football(9) → 4
- # 2+2+2+3 or 3+3+3 or 3+6 or 7+2