Опис лабораторної роботи №8

Всі функції, які будуть розроблені при виконанні завдань цієї лабораторної роботи повинні відповідати вимогам по стилю програмування PEP8 та мати належну документацію, яка дозволить здійснювати їх тестування за допомогою модуля doctest.

1. Розробити функцію sort_songs(song_titles, length_songs, key), що сортує пісні за вказаним параметром (key). Функція при виклику приймає наступні аргументи:
параметр song_titles - це список назв пісень (наприклад гурту Океан Ельзи);
параметр length_songs - це відповідний список тривалостей пісень;
параметр key - ключ за яким буде здійснюватися сортування;
та повертає один відсортований список кортежів першим елементом якого є назва пісні, а другим тривалість цієї пісні.

Серед ключів обов'язково передбачити сотрування за тривалістю пісень (song_length), за довжиною назви (title_length) та за першою літерою останнього слова назви (last_word). Якщо списки мають різну довжину або їх елементи не є відповідного типу, то функція повинна повернути None.
Підказка: Метод sort дозволяє здійснювати сортування елементів списків, за будь якими ознаками, наприклад сортувати список кортежів (назва пісні, тривалість пісні) за одним з елементів кортежу (тривалість пісні). Для такого сортування потрібно у виклику метода використовувати аргумент key (list.sort([ ], key = ...)) . Значенням цього аргументу може бути назва функції, яка повертає значення за яким буде відбуватися сортування. Наприклад, сортування списку рядків за останнім елементом рядка можна реалізувати наступним чином (документація: https://docs.python.org/3.5/howto/sorting.html.):

```
def sort_by_last(input_str):
    return input_str[-1]

print(list.sort([], key=sort_by_last))
```

Приклади значень аргументів функції:

```
song_titles = ['Янанебібув', 'Той день', 'Мало мені', 'Сосни', Кавачай',
               'Відпусти', 'Африка', 'Поясни', 'Фіалки', 'Коли тебе нема', 'Етюд']
length_songs = ['3.19', '3.58', '5.06', '4.31', '4.39', '3.52', '4.24', '3.39, '3.43', '3.17', 2.21']
key -  song_length, title_length, last_word
```

Якщо недоступна автоматична перевірка то збережіть та завантажте модуль з функцією у форматі Name_Surname_ sort_songs.py, вказавши своє прізвище та ім'я.

2. Напишіть тіло функції equal_parts(numbers), яка ділить вхідний список цілих чисел на два списки з однаковими сумами елементів. Приклади виклику цієї функції можна знайти в документації до функції , яка знаходиться у розділі для автоматичної перевірки. У вкладеному файлі problem3_flow.jpg зображено блок-схему найпростішого алгоритму для вирішення цієї задачі. В документації до функції вкажіть застереження у яких випадках функція не буде коретно працювати.

Якщо недоступна автоматична перевірка то збережіть та завантажте модуль з функцією у форматі Name_Surname_equal_parts.py, вказавши своє прізвище та ім'я.

3. Розробити функцію sieve_flavius(n) для побудови списку вдалих чисел (lucky number) .

В теорії чисел вдале число — натуральне число з множини, яку генерує решето Йосипа Флавія.
Побудова решета Йосипа Флавія подібна до побудови решета Ератосфена, яке генерує прості числа.
Процес побудови починається з запису повного списку натуральних чисел:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, …

Далі здійснюються наступні дії:
Перше число 1 - вважається вдалим.
Кожне друге число (всі парні числа) викреслюються і залишаються тільки непарні числа:
1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25,
Друге число в послідовності — число 3.
Число 3 - вважається вдалим.
Визначаються числа, які є третіми у поточному списку. Число 1 - перше, 3 - друге, 5 - третє...
Кожне третє число, викреслюється з послідовності:
1, 3, 7, 9, 13, 15, 19, 21, 25,
Наступне число 7 - вважається вдалим.
Всі сьомі числа викреслюються з послідовності:
1, 3, 7, 9, 13, 15, 21, 25,
Процедура повторюється поки не буде досягнуто число n.
Всі числа які залишилися це вдалі числа:
1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, 43, 49, 51, 63, 67, 69, 73, 75, 79, 87, 93, 99, …

Приклади виклику функції:

```
>>> sieve_flavius(100)
[1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, 43, 49, 51, 63, 67, 69, 73, 75, 79, 87, 93, 99]
>>> sieve_flavius(10)
[1, 3, 7, 9]
>>> sieve_flavius(0)
[]
```

Якщо недоступна автоматична перевірка то збережіть та завантажте модуль з функцією у форматі Name_Surname_ sieve_flavius.py, вказавши своє прізвище та ім'я.

4. Розробити модуль happy для пошуку щасливих трамвайних квитків. Трамвайний квиток вважається щасливим, якщо сума першої половини цифр його номера дорівнює сумі цифр його другої половини. Цифри сумуються до тих пір поки сумою не буде цифра. Номери квитків це восьмирозрядні числа. Якщо номер квитка містить непарну кількість цифр то номер доповнюється нулями згідно вказаної кількості розрядів. Наприклад, 12345 при кількості розрядів 8 - 00012345.
Модуль повинен містити наступні функції:
- happy_number(num) для визначення чи квиток щасливий; num - додатнє натуральне число. Функція повинна повертати будеве значення True або False.
- count_happy_numbers(n) для визначення кількості щасливих квитків в серії з n квитків; n - загальна кількість квитків. Функція повинна повертати число типу int.
- happy_numbers(m, n) для побудови списку щасливих квитків в заданому проміжку номерів квитків; m - нижня границя номерів квитків, n - верхня границя номерів квитків. Функція повинна повертати список чисел типу int.
Приклади виклику функцій:

```
>>> happy_number(12345)
False
>>> happy_number(43211234)
True
>>> happy_number(191234)
True
```

Якщо недоступна автоматична перевірка то збережіть та завантажте модуль у форматі Name_Surname_ happy.py, вказавши своє прізвище та ім'я.