Python (BSU FAMCS Fall'18)

Семинар 2

Преподаватели: Дмитрий Косицин, Светлана Боярович

Общие замечания ко всем заданиям

Свой тестирующий код можно размещать под условием `if __name__ == '__main__'` внизу файла. Такой код выполнится только если запустить этот файл, но не импортировать его.

Просьба использовать те имена для функций и файлов, которые указаны в замечаниях к заданиям.

Для отправки заданий выберите в anytask нужную задачу и там к сообщению прикрепите свое решение (один или несколько .py файлов).

Для тестирования интерфейса ваших заданий выложен специальный скрипт. Просьба также не оставлять debug print'ы в ваших программах, которые нужны исключительно для вывода отладочной информации.

Задание 1. (0.5 балла). Напишите функцию, которая принимает один аргумент n – натуральное число – и считает (возвращает) сумму $1*2+2*3+...+(n-1)*n$. Используйте comprehensions для решения задачи, желательно, только одно.
Замечание. Функцию назовите *calculate_special_sum*. Программу сохраните в файле *special_sum.py*.

Пример

```
assert calculate_special_sum(3) == 8
```

Задание 2. (1 балл). Напишите функцию, подсчитывающую количество различных способов выдать со счета сумму от 0 до 50 копеек монетами по 1, 2, 5 и 10 копеек.
Замечание. Программу сохраните в файле *exchange.py*, функцию назовите *exchange_money*.

Пример

```
assert exchange_money(0) == 0
assert exchange_money(1) == 1
assert exchange_money(2) == 2
assert exchange_money(3) == 2
```

Задание 3. (0.5 балла). Напишите функцию, которая принимает на вход последовательность (кортеж или список) и возвращает список пар из уникальных элементов и количества раз, сколько они встретились в переданной последовательности. Парой называется кортеж из двух элементов. Порядок пар в списке не важен.
Замечание. Функцию назовите *compress*. Программу сохраните в файле *unique.py*.

Пример

```
expected_sorted = [(1, 2), (2, 1)]
actual_sorted = sorted(compress([1, 2, 1]))
assert expected_sorted == actual_sorted
```

Задание 4. (1 балл). Напишите функцию, которая принимает один аргумент n – натуральное число – и возвращает все простые числа, не превосходящие n. Используйте comprehensions для решения задачи.

Полный балл за задачу будет выставлен только в случае написания функции из одного comprehension выражения.

Функцию назовите *get_primes*. Программу сохраните в файле *primes.py*.

Пример

```
assert [2, 3, 5, 7, 11] == sorted(get_primes(11))
```

Задание 5. (1 балл). Напишите функцию, которая для выборки, заданной списком, и заданного натурального числа k посчитает и выведет гистограмму распределения шириной k. Другими словами, найдет максимум и минимум, поделит интервал на k частей ($1 \leqslant k \leqslant d$, d – количество различных элементов в списке) и посчитает, сколько элементов выборки попало в каждый интервал.

Верните список длины k, содержащий количество элементов в каждой ячейке. Левую границу интервалов считайте включая, правую – исключая (кроме последнего интервала). Желательно реализовать алгоритм со сложностью $O(n)$, где n – длина входного списка.

Замечание. Функцию назовите *distribute*. Программу сохраните в файле *hist.py*.

Пример

```
assert distribute([1.25, 1, 2, 1.75], 2) == [2, 2]
```

Задание 6. (1 балл). Напишите функцию, которая первым аргументом на вход принимает строку с цифрами некоторого целого числа. Второй аргумент функции – числа, вхождения которых нужно найти в переданной строке цифр – является либо целым числом (int), либо кортежем целых чисел (если их несколько).

Функция должна возвращать количество всех позиций, на которых обнаружены вхождения, а также отсортированный список обнаруженных позиций. Если передан кортеж, то возвращать нужно суммарное количество вхождений, а также объединенный отсортированный список позиций найденных чисел. Если длина списка больше чем k, то возвращайте только первые k элементов, и логируйте факт того, что вхождений на самом деле больше. Параметр k передается третим аргументом – натуральное число, по умолчанию 5.

Замечание. Индексацию цифр ведите с единицы, т.е. первая цифра некоторого числа имеет индекс *один*, а не *ноль*.

Функцию назовите *index*. Программу сохраните в файле *big_ number.py*.

Пример

```
assert (1, [1]) == index('123', 1)
assert (13, [1, 1, 2]) == index('1212122222', (1, 2, 12), 3)
```