Python (BSU FAMCS Fall'18)

Семинар 1

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 0. Установите и настройте Python.

Задание 1. (0.4 балла). Напишите программу, которая будет выводить через пробел все нечетные числа от 1 до 101 (включительно), попутно заменяя числа, которые делятся на 3, на **Fizz**, делящиеся на 7 – на **Buzz**, а делящиеся и на 3, и на 7 – на **FizzBuzz**.
Замечание. Программу сохраните в файле *fizzbuzz.py*.

Задание 2. (0.8 балла). Напишите функцию, возвращающую число, битовое представление которого является k-тым членом последовательности Морса–Туэ (последовательность OEIS номер A010060, `https://en.wikipedia.org/wiki/Thue-Morse_sequence`). Значение k полагайте допустимым целым числом, большим либо равным нуля.
Подсказка. Возможно, полезной будет эта ссылка: `https://stackoverflow.com/a/12790495`.
Замечание. Программу сохраните в файле *thue_morse.py*, функцию назовите *get_sequence_item*. О быстрой скорости роста числа не беспокойтесь. Постарайтесь в решении обойтись без использования строк и списков и использовать операции с числами.

Пример

```
assert get_sequence_item(0) == 0b0
assert get_sequence_item(1) == 0b1
assert get_sequence_item(2) == 0b110
assert get_sequence_item(3) == 0b1101001
```

Задание 3. (0.8 балла). Билетик называется счастливым, если сумма первых трех цифр его номера равна сумме последних цифр. Напишите функцию, принимающую номер билета и возвращающую номер ближайшего счастливого билета (если их два – то любой из них). Номер переданного билета полагайте допустимым целым шестизначным числом (без нулей в начале).
Замечание. Программу сохраните в файле *ticket.py*, функцию назовите *get_nearest_lucky_ticket*.

Пример

```
assert get_nearest_lucky_ticket(111111) == 111111
assert get_nearest_lucky_ticket(123322) == 123321
assert get_nearest_lucky_ticket(123320) == 123321
assert get_nearest_lucky_ticket(333999) != 333900
assert get_nearest_lucky_ticket(333999) == 334019
```

Задание 4. (1 балл). Реализуйте функцию, которая объединяет две отсортированные последовательности в одну отстортированную (merge, асимптотическая сложность алгоритма – $O(n)$). Программа должна корректно работать для списков (возвращать новый отсортированный список) и кортежей (возвращать кортеж). Использовать встроенные функции сортировки и/или слияния запрещается.

Полный балл за задачу будет выставлен только в случае отсутствия итерирования по сортируемой коллекции с помощью индексов. Иначе говоря, попробуйте избегать произвольной индексации и использовать только методы объектов.

Функцию назовите *merge*. Программу сохраните в файле *merge.py*.

Пример

```
assert merge([1, 2, 7], [3]) == [1, 2, 3, 7]
assert merge((3, 15), (7, 8)) == (3, 7, 8, 15)
```