

Задачи на функциональное программирование 1

Задача 1.

Пусть имеется многомерный вложенный список, например:

```
sp = [[[5, 7, 2], [4, 9, 5]], [[2, 5, 4]], [[3, 2, 1], [[5], [9, 5]]],  
[4, 3, 1, 2], [[4, 7, 2], [6, 4]], [[[4, 1, 6], [3, 8]], [4, 5]],  
[9, 1], [3, 1], [[5, 6], [[4, 2, 1], [2, 5], [[6, 8, 2, 3, 4]]],  
[5, 3, 2], [2, [1], 4], [2, 5, [4, 3, 1], 6, 7, [9, 0, 5, 2, 4]],  
[7, 3, [4]], [4, 2, [[[5, 6, 7], 5, 7]], 1], [3, 4, 6, [6, 4, 5]],  
]
```

Напишите рекурсивную и итеративную функции для преобразования списка в одномерный.

Ожидаемый результат:

```
[5, 7, 2, 4, 9, 5, 2, 5, 4, 3, 2, 1, 5, 9, 5, 4, 3, 1, 2, 4, 7, 2, 6, 4, 4, 1, 6, 3,  
8, 4, 5, 9, 1, 3, 1, 5, 6, 4, 2, 1, 2, 5, 6, 8, 2, 3, 4, 5, 3, 2, 2, 1, 4, 2, 5, 4, 3,  
1, 6, 7, 9, 0, 5, 2, 4, 7, 3, 4, 4, 2, 5, 6, 7, 5, 7, 1, 3, 4, 6, 6, 4, 5]
```

Задача 2.

Напишите программу, которая получает от пользователя список, состоящий из целых чисел, и находит произведение его элементов. Решите задачу тремя способами:

1. с помощью reduce и лямбда-функции;
2. с math.prod();
3. с использованием пользовательской функции.

Задача 3.

Перепишите следующий код, используя map, reduce и filter.

```
height_total = 0  
height_count = 0  
for person in people:  
    if 'рост' in person:  
        height_total += person['рост']  
        height_count += 1  
if height_count > 0:  
    average_height = height_total / height_count  
    print(average_height)
```

Задача 4.

Напишите программу, которая сортирует полученную от пользователя строку с числами следующим образом:

4. отрицательные числа идут после положительных;
5. и положительные, и отрицательные числа упорядочены по возрастанию.

Пример ввода:

```
6 -1 3 -5 -15 4 1 9 8 6 -3 -4 12 -10 7
```

Вывод:

```
1 3 4 6 7 8 9 12 -15 -10 -6 -5 -4 -3 -1
```

Задача 5.

Пусть есть две операции: $x+1$ и $x*3$. Напишите рекурсивную функцию, подсчитывающую число вариантов достижения числа 203 от заданного числа n путем применения указанных операций.

Задача 6.

Напишите рекурсивную функцию, которая определяет, является ли введенная пользователем строка палиндромом.

Пример ввода:

Лёша на полке клопа нашёл

Вывод:

True

Задача 7.

Сформируйте колоду карт (52 штуки) CardDeck. Каждая карта представлена в виде кортежа типа (2, Пик). Реализуйте перемешивание карт колоды. Далее, реализуйте итератор колоды карт. При вызове функции next() должна быть представлена следующая карта. По окончании перебора всех элементов - возникать ошибка StopIteration.

Задача 8.

На рисунке приведена лесенка, представляющая собой набор кубиков. В этом наборе каждый последующий ряд состоит из меньшего числа кубиков, чем предыдущий. Надо написать программу для вычисления количества лесенок, которое можно построить из n кубиков. В решении используйте рекурсивную функцию.

Для кэширования промежуточных результатов примените мемоизацию с использованием декоратора @lru_cache() модуля functools. Сравните скорость выполнения решений с мемоизацией и без.



Задача 9.

Напишите программу для возведения числа n в степень m . Решите задачу двумя способами – итеративным и рекурсивным. Сравните скорость выполнения обоих решений.

Примечание для рекурсивного решения: предположим, что нужно возвести число 5 в степень 6. Свойства степени позволяют разбить процесс на более мелкие операции и представить выражение $5 ** 6$ в виде $(5 ** 3) ** 2$. Этот подход работает в том случае, если степень представляет собой четное число. Если степень нечетная, следует воспользоваться другим свойством: $(n ** m) \times n = n ** (m + 1)$. Поскольку может ввести как четное, так и нечетное значение m , в функции должны быть два рекурсивных случая. В качестве граничного случая используется еще одно свойство степени: $n ** 1 = n$.

Задача 10.

Функция zero принимает строку s . Если первый символ есть 0, то возвращает остаток строки. Если нет – тогда None.

Функция one делает то же самое, если первый символ есть 1.

```
def zero(s):  
    if s[0] == "0":  
        return s[1:]
```

```
def one(s):  
    if s[0] == "1":  
        return s[1:]
```

Напишите функцию `rule_sequence()`, которая принимает на вход строку из 0 и 1, и список из функций-правил, состоящий из функций `zero` и `one`.

Она вызывает первое правило, передавая ему строку.

Если правило не возвратило `None`, то берёт возвращённое значение и вызывает следующее правило. И так далее.

Если возвращается `None`, `rule_sequence()` останавливается и возвращает `None`.

Иначе – значение последнего правила.

Примеры вывода:

```
print (rule_sequence('0101', [zero, one, zero]))
```

```
1
```

```
print (rule_sequence('0101', [zero, zero]))
```

```
None
```

Если ваш код использует цикл, перепишите код без использования циклов с использованием рекурсии.