# Функции как объект. Лямбда-функции

# Набор юного арифметика

Kлассная работа

макс. 1 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Напишите функцию arithmetic\_operation(operation), которая принимает символ одной из четырех арифметических операций, а возвращает функцию двух аргументов для соответствующей операции.

## Пример

| Ввод | Вывод |
| --- | --- |
| operation = arithmetic\_operation('+')  print(operation(1, 4)) | 5 |

# Просто map

Kлассная работа

макс. 1 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Напишите свою функцию simple\_map(transformation, values), которая из одного списка делает другой. Каждый элемент результирующего списка должен получаться применением функции преобразования transformation к соответствующему элементу списка values.

## Пример

| Ввод | Вывод |
| --- | --- |
| values = [1, 3, 1, 5, 7]  operation = lambda x: x + 5  print(\*simple\_map(operation, values)) | 6 8 6 10 12 |

## Примечания

Использовать стандартную функцию map не получится.

# Мимикрия

Kлассная работа

макс. 1 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

У вас есть код, который вы не можете менять (так часто бывает, когда код в глубине программы используется множество раз и вы не хотите ничего сломать):

transformation = <???>   
values = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29] *#*илилюбойдругойсписок   
transformed\_values = **list**(**map**(transformation, values))

Единственный способ вашего взаимодействия с этим кодом – посредством задания функции transformation.

Однако вы поняли, что для вашей текущей задачи вам не нужно никак преобразовывать список значений, а нужно получить его как есть.

Напишите такое лямбда-выражение transformation, чтобы transformed\_values получился копией values. Переменная transformation должна быть глобальной, чтобы проверяющая система смогла его найти. Кроме transformation вам ничего писать не нужно. Печатать на экран – тоже.

## Пример

| Ввод | Вывод |
| --- | --- |
| values = [1, 23, 42, "asdfg"]  transformed\_values = list(map(transformation, values))  if values == transformed\_values:  print('ok')  else:  print('fail') | ok |

# Коллбэки

Kлассная работа

макс. 3 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Иногда бывают ситуации, когда вы хотите поручить управление программой функции, а не ориентироваться на результат её выполнения. Этому может быть много причин.  
  
Во-первых, это упрощает код, который пользуется вашей функцией.

Во-вторых, бывают программы, в которых несколько наборов команд выполняется параллельно и независимо друг от друга. Например, когда ваша программа подключается к интернету, это обычно делается параллельно с работой остальной программы, ведь запрос может занять как долю секунды, так и минуты. Так же обычно поступают, когда вычисление занимает очень много времени. Взаимодействие с пользователем тоже может занимать очень много времени по меркам компьютерной программы - секунды на то, чтобы пользователь ввёл текст. Было бы очень неудобно, если бы, пока выполняется запрос, зависла бы вся программа.  
  
Проблема с частями программы, которые выполняются независимо друг от друга – в том, что основная программа не может отследить, когда интернет-запрос завершится, если только не будет постоянно это проверять.

Зато сам запрос точно знает, когда он завершился и может что-то сделать по завершении. Например, показать вам загруженную картинку или обновить ленту сообщений. А иногда – сообщить о том, что произошла ошибка.

Но на результат завершения одного и того же запроса можно реагировать по-разному: иногда не нужно делать ничего, в других случаях – показать результат, в третьих – сохранить результат в файл и так далее.

Значит, программе нужно знать, как реагировать на окончание запроса. Это делается при помощи так называемых коллбэков (callback), что переводится как обратный звонок. Коллбэком называется специальная функция, которая вызывается, когда ваше вычисление завершилось. Для программы, запустившей запрос или долгое вычисление, коллбэк – это способ сообщить, что надлежит сделать, когда вычисление завершится. Коллбэк обычно передают как аргумент функции запуска вычисления.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Ваша задача – написать функцию ask\_password(login, password, success, failure), которая получает логин и пароль пользователя и проверяет их правильность. Пароль считается правильным, если в нём содержится ровно три английские гласные буквы (гласными считать буквы a,e,i,o,u,y) и ровно такой же набор согласных (все буквы, кроме перечисленных шести), как в логине. Порядок и количество согласных также должно совпадать. Логин и пароль приводятся к нижнему регистру и передаются.

Пример: для логина “login” подойдут пароли “aaalgn” и “luagon“.

Если пароль правильный, функция должна вызвать коллбэк **success**, передав ему в качестве аргумента логин. А если пароль был неверный - **failure**, передав ему в качестве аргументов логин и сообщение об ошибке (в таком порядке). Сообщение об ошибке должно быть одним из трёх вариантов:

* “Wrong number of vowels”, если в пароле неверное число гласных;
* “Wrong consonants”, если в пароле набор согласных отличается от набора согласных логина;
* “Everything is wrong”, если оба условия нарушены.

Также напишите функцию main(login, password), которая вызывает написанную функцию ask\_password так, чтобы в случае успеха она печатала “Привет, {логин}!”, а в случае ошибки – “Кто-то пытался притвориться пользователем {логин}, но в пароле допустил ошибку: {текст ошибки, большими буквами}.”.

Внимание! При вводе логина и пароля *НЕ*выводите подсказку для пользователя вида “Введите логин:”, так как это собьет с толку проверяющую систему.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| main("anastasia", "nsyatos") | Привет, anastasia! |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| main("eugene", "aanig") | Кто-то пытался притвориться пользователем eugene, но в пароле допустил ошибку: WRONG CONSONANTS. |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| ask\_password("anastasia", "nsyatos", lambda login: print('super'), lambda login, err: print('bad')) | super |

# Самая далёкая планета

Kлассная работа

макс. 3 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Планеты вращаются вокруг звезд по эллиптическим орбитам. Назовём самой далёкой планетой ту, орбита которой имеет самую большую площадь.

Напишите функцию find\_farthest\_orbit(list\_of\_orbits), которая среди списка орбит планет найдет ту, по которой вращается самая далёкая планета. Круговые орбиты не учитывайте: вы знаете, что у вашей звезды таких планет нет, зато искусственные спутники были запущены на круговые орбиты.

Результатом функции должен быть кортеж, содержащий длины полуосей эллипса орбиты самой далёкой планеты.

Каждая орбита представляет из себя кортеж из пары чисел – полуосей её эллипса. Площадь эллипса вычисляется по формуле S=πabS=πab, где aa и bb – длины полуосей эллипса.

При решении задачи используйте списочные выражения.

Подсказка: проще всего будет найти эллипс в два шага: сначала вычислить самую большую площадь эллипса, а затем найти и сам эллипс, имеющий такую площадь.

Гарантируется, что самая далёкая планета ровно одна.

## Пример

| Ввод | Вывод |
| --- | --- |
| orbits = [(1, 3), (2.5, 10), (7, 2), (6, 6), (4, 3)]  print(\*find\_farthest\_orbit(orbits)) | 2.5 10 |

# Все равны, как на подбор

Kлассная работа

макс. 3 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Напишите функцию same\_by(characteristic, objects), которая проверяет, все ли объекты имеют одинаковое значение некоторой характеристики, и возвращает **True**, если это так. Если значение характеристики для разных объектов отличается – то **False**. Для пустого набора объектов, функция должна возвращать True. Аргумент characteristic – это функция, которая принимает объект и вычисляет его характеристику.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| values = [0, 2, 10, 6]  if same\_by(lambda x: x % 2, values):  print('same')  else:  print('different') | same |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| values = [1, 2, 3, 4]  if same\_by(lambda x: x % 2, values):  print('same')  else:  print('different') | different |