Функции высшего порядка, работа с файлами.

В Python есть механизм, который позволяет превратить подобный вызов во что-то более красивое — lambda.

def sum(x, y):

return x + y # ⇔

(равносильно) sum = lambda x, y: x + y

Теперь, чтобы вызвать функцию суммы, достаточно просто вызвать sum. Также можно пропустить шаг создания переменной sum и сразу вызвать lambda:

calc(lambda x, y: x + y, 4, 5) # 9

Функция map() применяет указанную функцию к каждому элементу итерируемого объекта и возвращает итератор с новыми объектами.

list\_1 = [x for x in range (1,20)]

list\_1 = list(map(lambda x: x + 10, list\_1 ))

print(list\_1)

1. Маленькое отступление, функция строка.split() - убирает все пробелы и создаем список из значений строки, пример: data = '1 2 3 5 8 15 23 38'.split() print(data) # ['1', '2', '3', '5', '8', '15', '23', '38']
2. 2. Теперь вернемся к задаче. С помощью функции map():

data = list(map(int,input().split()))

Функция filter 💡

Функция filter() применяет указанную функцию к каждому элементу итерируемого объекта и возвращает итератор с теми объектами, для которых функция вернула True

data = [x for x in range(10)] res = list(filter(lambda x: x % 2 == 0, data))) print(res) # [0, 2, 4, 6, 8]

Функция zip() применяется к набору итерируемых объектов и возвращает итератор с кортежами из элементов входных данных

Функция enumerate 💡 Функция enumerate() применяется к итерируемому объекту и возвращает новый итератор с кортежами из индекса и элементов входных данных

Функция enumerate() позволяет пронумеровать набор данных.

Файлы Файлы в текстовом формате используются для: ● Хранения данных ● Передачи данных в клиент-серверных проектах ● Хранения конфигов ● Логирования действий Что нужно для работы с файлами: 1. Завести переменную, которая будет связана с этим текстовым файлом. 2. Указать путь к файлу. 3. Указать, в каком режиме мы будем работать с файлом. Варианты режима (мод): 1. a – открытие для добавления данных. ○ Позволяет дописывать что-то в имеющийся файл. ○ Если вы попробуете дописать что-то в несуществующий файл, то файл будет создан и в него начнётся запись. 2. r – открытие для чтения данных. ○ Позволяет читать данные из файла. ○ Если вы попробуете считать данные из файла, которого не существует, программа выдаст ошибку. 3. w – открытие для записи данных. ○ Позволяет записывать данные и создавать файл, если его не существует. Миксованные режимы: 4. w+ ○ Позволяет открывать файл для записи и читать из него. ○ Если файла не существует, он будет создан. 5. r+ ○ Позволяет открывать файл для чтения и дописывать в него. ○ Если файла не существует, программа выдаст ошибку.

1. Режим a

colors = ['red', 'green', 'blue']

data = open('file.txt', 'a') # здесь указываем режим, в котором будем работать

data.writelines(colors) # разделителей не будет

data.close()

● data.close() — используется для закрытия файла, чтобы разорвать подключение файловой переменной с файлом на диске. ● exit() — позволяет не выполнять код, прописанный после этой команды в скрипте. ● В итоге создаётся текстовый файл с текстом внутри: redbluedreen. ● При повторном выполнении скрипта redbluedreenredbluedreen — добавление в существующий файл, а не перезапись файлов.

Ещё один способ записи данных в файл:

with open('file.txt', 'w') as data:

data.write('line 1\n')

data.write('line 2\n')

1. Режим r ● Чтение данных из файла:

path = 'file.txt'

data = open(path, 'r')

for line in data:

print(line)

data.close()

1. Режим w

colors = ['red', 'green', 'blue']

data = open('file.txt', 'w')

data.writelines(colors) # разделителей не будет

data.close()

Модуль os Модуль os предоставляет множество функций для работы с операционной системой, причем их поведение, как правило, не зависит от ОС, поэтому программы остаются переносимыми. Для того, чтобы начать работать с данным модулем необходимо его импортировать в свою программу: import os Познакомимся с базовыми функциями данного модуля: ● os.chdir(path) - смена текущей директории. import os os.chdir('C:/Users/79190/PycharmProjects/GB') ● os.getcwd() - текущая рабочая директория import os print(os.getcwd()) # 'C:\Users\79190\PycharmProjects\webproject' 14 ● os.path - является вложенным модулем в модуль os и реализует некоторые полезные функции для работы с путями, такие как: ○ os.path.basename(path) - базовое имя пути import os print(os.path.basename('C:/Users/79190/PycharmProjects/webproject/main.py')) # 'main.py' ● os.path.abspath(path) - возвращает нормализованный абсолютный путь. import os print(os.path.abspath('main.py')) # 'C:/Users/79190/PycharmProjects/webproject/main.py' Это лишь малая часть возможностей модуля os.

Модуль shutil Модуль shutil содержит набор функций высокого уровня для обработки файлов, групп файлов, и папок. В частности, доступные здесь функции позволяют копировать, перемещать и удалять файлы и папки. Часто используется вместе с модулем os. Для того, чтобы начать работать с данным модулем необходимо его импортировать в свою программу: import shutil Познакомимся с базовыми функциями данного модуля: ● shutil.copyfile(src, dst) - копирует содержимое (но не метаданные) файла src в файл dst. ● shutil.copy(src, dst) - копирует содержимое файла src в файл или папку dst. ● shutil.rmtree(path) - Удаляет текущую директорию и все поддиректории; path должен указывать на директорию, а не на символическую ссылку