**ЛР 1 square\_seq\_digit**

**Описание задачи**

Написать функцию squareSequenceDigit(), где решалась бы следующая задача. Найти n-ю цифру последовательности из квадратов целых чисел: 149162536496481100121144...

Например, 2-я цифра равна 4, 7-я 5, 12-я 6. Использовать операции со строками в этой задаче запрещается.

Протестировать выполнение программы со следующими значениями:

* при вызове squareSequenceDigit(1) должно быть 1;
* squareSequenceDigit(2) вернёт 4;
* squareSequenceDigit(7) вернёт 5;
* squareSequenceDigit(12) вернёт 6;
* squareSequenceDigit(17) вернёт 0;
* squareSequenceDigit(27) вернёт 9.

**ЛР 2 openweatherapi**

**Описание задачи**

Написать реализацию функции get\_weather\_data(place, api\_key=None), в которой необходимо получить данные о погоде с сайта <https://openweathermap.org/>.

Функция должна возвращать объект в формате JSON, включающий:

* информацию о названии города (в контексте openweathermap),
* код страны (2 символа),
* широту и долготу, на которой он находится,
* его временной зоне,
* а также о значении температуры (как она ощущается).

Значение временной зоны выводить в формате UTC±N, где N - цифра временного сдвига. Протестировать выполнение программы со следующими городами: Чикаго, СПб, Дакка.

Пример вызова функции и получаемого результата.

get\_weather\_data('Kiev', api\_key=key)

>>> {"name": "Kyiv", "coord": {"lon": 30.52, "lat": 50.43}, "country": "UA", "feels\_like": 21.96, "timezone": "UTC+3"}

При реализации программы, не публикуйте свой ключ для осуществления запросов. Сразу же после создания репозитория в классруме исключите из коммитов подключаемый файл, где разместите ключ, с помощью .gitignore. Для организации запросов используйте модуль requests. Для кодирования и декодирования json - одноименный модуль.

**Лабораторная работа 3. Реализация удаленного импорта**

Разместите представленный ниже код локально на компьютере и реализуйте механизм удаленного импорта. Продемонстрируйте в виде скринкаста или в текстовом отчете с несколькими скриншотами работу удаленного импорта.

По шагам:

1. Создайте файл myremotemodule.py, который будет импортироваться, разместите его в каталоге, который далее будет "корнем сервера" (допустим, создайте его в папке rootserver).
2. Разместите в нём следующий код:

def myfoo():

author = "" # Здесь обознаться своё имя (авторство модуля)

print(f"{author}'s module is imported")

1. Создайте файл Python с содержимым функций url\_hook и классов URLLoader, URLFinder из текста конспекта лекции со всеми необходимыми библиотеками (допустим, activation\_script.py).
2. Далее, чтобы продемонстрировать работу импорта из удаленного каталога, мы должны запустить сервер http так, чтобы наш желаемый для импорта модуль "лежал" на сервере (например, в корневой директории сервера). Откроем каталог rootserver с файлом myremotemodule.py и запустим там сервер:

python3 -m http.server

1. После этого мы запускаем файл, в котором содержится код, размещенный выше (обязательно добавление в sys.path\_hooks).

python3 -i activation\_script.py

1. Теперь, если мы попытаемся импортировать файл myremotemodule.py, в котором размещена наша функция myfoo будет выведен ModuleNotFoundError: No module named 'myremotemodule', потому что такого модуля пока у нас нет (транслятор про него ничего не знает).
2. Однако, как только мы выполним код:

sys.path.append("http://localhost:8000")

добавив путь, где располагается модуль, в sys.path, будет срабатывать наш "кастомный" URLLoader. В path\_hooks будет содержатся наша функция url\_hook.

1. Протестируйте работу удаленного импорта, используя в качестве источника модуля другие "хостинги" (например, gist, repl.it).
2. Переписать содержимое функции url\_hook с помощью модуля requests.

**Лабораторная работа 4. Ряд Фибоначчи с помощью итераторов**

Лабораторная работа состоит из трех заданий:

Разработать функцию, возвращающую элементы ряда Фибоначчи по данному максимальному значению.

Создание программы, возвращающей список чисел Фибоначчи с помощью итератора.

Разработать функцию, возвращающую список чисел ряда Фибоначчи с использованием бесконечных итераторов (модуль itertools).

Создание программы с классическим генератором (использовать yield).

Рассмотрим особенности каждого из них.

Задание 1

Стартовый борд: https://replit.com/@zhukov/sem5-lr4-fib#main.py

Требуется реализовать код для функции fib такой что, для данного n функция возвращала бы максимальное число элементов ряда Фибоначчи не превосходящих данное n.

Например: для n = 1, функция должна вернуть список [0, 1, 1]. Для n = 2, соответственно [0, 1, 1, 2]. Для n = 5, соответственно [0, 1, 1, 2, 3, 5].

Предлагается использовать не рекурсивный способ решения, а использовать цикл while или for .. in .. и по заданному n вычислять значение очередного элемента ряда Фибоначчи. Разрешается хранить внутри функции первые 2 элемента, поскольку их невозможно получить с помощью арифметических действий.

Требуется написать необходимые тесты в файле test\_fib.py.

Задание 2

Дополните код классом FibonacchiLst, который бы позволял перебирать элементы из ряда Фибоначчи по данному ей списку. Итератор должен вернуть очередное значение, которое принадлежит ряду Фибоначчи, из данного ей списка. Например: для lst = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], FibonacchiLst должен вернуть [0, 1, 1, 3, 5, 8]

Решение может быть выполнено с помощью реализации содержимого методов \_\_init\_\_,\_\_iter\_\_, \_\_next\_\_ или с помощью реализации метода \_\_getitem\_\_.

class FibonacchiLst:

def \_\_init\_\_(self):

pass

def \_\_iter\_\_(self):

pass

def \_\_next\_\_(self):

pass

Пример реализации итератора, возвращающего четные элементы, из iterable-объекта представлен в файле even\_numbers\_iterator.py.

Задание 3

Для выполнения задания требуется написать такую функцию fib\_iter, которая принимала бы iterable-объект с числами и возвращала бы числовые значения (принадлежащие ряду Фибоначчи) с помощью модуля itertools (например, с помощью метода islice()):

from itertools import islice

l = list(range(14))

print(l) # [0, 1, 1, 3, 5, 8, 13]

list(islice(l,0,2)) # [0, 1]

Функция fib\_iter например, может принять range(14) и должна вернуть [0, 1, 1, 3, 5, 8, 13]

Задание 4

Пример функции-генератора представлен в файле gen\_fib.py. Генератор my\_genn возвращает нечетные элементы.

## Лабораторная работа 5. Графики matplotlib

Выполните [лабораторную работу](https://colab.research.google.com/drive/1tT93ly-SQBMcP09bppDGMFwdId110jQf?usp=sharing)  и опубликуйте ссылку на репозиторий в GitHub или на Google Colab, предварительно удостоверившись, что ссылка открывается в режиме инкогнито.

Лабораторная работа 5. Визуализация данных о погоде с помощью matplotlib.

Цель работы: научиться обрабатывать и визуализировать данные, полученные с помощью API (на примере сервиса openweathermap).

Описание работы: получить данные о погоде за 5 последних дней и визуализировать эти данные, используя диаграмму рассеяния (scatterplot). Затем, посчитать среднюю температуру за каждый день и построить рядом (на этом же изображении) линейную диаграмму изменения температур.

Замечание: можно использовать другие сервисы для получения прогноза погоды на 7 дней (gismeteo, pogoda.yandex.ru), но сигнатура функций должна быть такая же как в примере ниже.

Лабораторная работа состоит из 2-х основных частей:

1. Получение данных посредством API.
2. Визуализация данных.