ЛР 1 square_seq_digit

Описание задачи

Написать функцию squareSequenceDigit(), где решалась бы следующая задача. Найти n-ю цифру последовательности из квадратов целых чисел: 14916253649648110012114 4...

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")
```

3. Создайте файл Python с содержимым функций url_hook и классов URLLoader, URLFinder из текста конспекта лекции со всеми необходимыми библиотеками (допустим, activation_script.py).
4. 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
```

5. После этого мы запускаем файл, в котором содержится код, размещенный выше (обязательно добавление в sys.path_hooks).

```
python3 -i activation_script.py
```

6. 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).
7. 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.

8. Протестируйте работу удаленного импорта, используя в качестве источника модуля другие "хостинги" (например, gist, repl.it).
9. 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 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.

Лабораторная работа 6. «Одиночка» и получение курсов валют

Примените паттерн одиночка к функции получения валют и протестируйте получившийся код (при применении шаблона у вас не может существовать более одного инстанса объекта, к которому вы применили паттерн).

Стартовый борд: https://replit.com/@zhukov/DistinctRareBundledsoftware-API#main.py

Вставьте в поле ответа ссылку на replit с получившимся решением.

Лабораторная работа 7. Использование шаблона «Декоратор»

Цель работы

Примените паттерн декоратор и реализуйте объектно-ориентированную версию программы получения курсов валют с сайта Центробанка таким образом, чтобы:

- было возможно использовать базовую версию для получения информации о валютах (возвращает словарь со структурой, описанной в одной из предыдущих лабораторных работ) (`class CurrenciesList`);

- было возможно применить декоратор к базовой версии и получить данные в формате JSON (`class ConcreteDecoratorJSON)`;
- было возможно использовать декоратор к базовой версии (`CurrenciesList`) или к другому декоратору (`ConcreteDecoratorJSON`) и получить данные в формате csv (`class ConcreteDecoratorCSV`).

Комментарии по выполнению

Изучите пример реализации схемы шаблона «Декоратор»: https://replit.com/@zhukov/decorator-example и стартовый борд для реализации задания: https://replit.com/@zhukov/DistinctRareBundledsoftware-API-decorator#main.py. В них сопоставляются классы, представляющие схему устройства шаблона "Декоратор". Для некоторых компонент код уже написан. Для корректного выполнения клиентского кода требуется реализовать магический метод `__str__` и/или `__repr__` в классах-декораторах.

Вставьте в поле ответа ссылку на replit с получившимся решением.

ЛР 8 mvc-simple-task

1. Запустить приложение.
2. Реализовать сохранение данных, получаемых из метода POST в файл (json, csv) или базу данных sqlite.
3. Корректно подключить и использовать шаблонизатор Jinja2 (реализовать приложение как пакет и подключить Jinja2 корректно).
4. Реализовать содержимое класса Record (или Item), в котором содержатся и проверяются данные, связанные с регистрацией на конференцию.
5. Использовать шаблонизатор Jinja2 и реализовать три шаблона: один - базовый с head, title, body. Второй — содержимое формы, которая отображается на индексной странице; Третий шаблон - отображение всех записей, которые были добавлены в базу данных / файл.
6. Создавать / генерировать qrcode на страницу пользователя (http://localhost/user/id), где id — это идентификатор пользователя.

По пункту 2:

from jinja2 import Environment, PackageLoader, select_autoescape

env = Environment(loader=PackageLoader('app', 'templates'),

autoescape=select_autoescape(['html', 'xml']))

Соответствующее задание ИСР 1.2. Создание пользовательского пакета для приложения "Гостевая книга"

Лабораторная работа 9. Визуализация курсов валют

На основе https://colab.research.google.com/drive/1qXLB5qT0mgPvLAjU7-Q0D5U67MRN8PDv?usp=sharing

создайте собственный борд с реализованными пунктами, отмеченными в TODO colaboratory-блокнота.

В качестве ответа предоставьте собственный colab-блокнот. Проверьте, что он открывается в режиме инкогнито и доступ к нему есть у всех, у кого есть ссылка.