Лабораторная работа №4

Тема: Прикладные пакеты Python.

Цель: Научиться проектировать и создавать одно- и многопоточные скрипты с графическим интерфейсом, позволяющие работать с базами данных, сетью и слабоструктурированной информацией.

Темы для предварительной проработки [УСТНО]:

- Работа с бинарными данными в Python.
- Пакеты Python для работы с файлами форматов json и xml, базами данных и объектно-реляционными отображениями.
- Пакеты requests, BeautifulSoup.
- Пакеты для научно-исследовательской деятельности: sciPy, numPy, symPy, pandas.

Индивидуальное задание [КОД]:

1. Напишите скрипт, читающий во всех mp3-файлах указанной директории ID3v1-теги и выводящий информацию о каждом файле в виде: [имя исполнителя] - [название трека] - [название альбома]. Если пользователь при вызове скрипта задает ключ -d, то выведите для каждого файла также 16-ричный дамп тега. Скрипт должен также автоматически проставить номера треков и жанр (номер жанра задается в параметре командной строки), если они не проставлены. Используйте модуль struct.
   ID3v1-заголовки располагаются в последних 128 байтах mp3-файла. Структура заголовка отражена в табл. 2.

Таблица 2 – Структура ID3v1-заголовка mp3-файла

Поле	Длина (байт)	Описание
header	3	3 символа: "TAG"
title	30	30 символов названия трека
artist	30	30 символов имени исполнителя
album	30	30 символов названия альбома
year	4	4 символа года издания
comment	28 или 30	Комментарий
zero-byte	1	Если в теге хранится номер трека, то этот байт зарезервирован под 0.
track	1	Номер трека альбома или 0. Имеет смысл, если предыдущий байт равен 0
genre	1	Индекс в списке жанров или 255.

2. Напишите скрипт для информационной системы библиотеки. База данных библиотеки включает таблицы «Авторы» с полями «id», «имя», «страна», «годы жизни», и «Книги» с полями «id автора», «название», «количество страниц», «издательство», «год издания»). Необходимо производить авторизацию пользователей, логины и пароли которых хранятся в отдельной таблице. Пароли должны храниться в зашифрованном виде (например, хэш SHA-1 или MD5). В программе должны быть окна для отображения информации о всех книгах и авторах, окно добавления книги/автора. Реализуйте также возможность сохранения информации о выделенном авторе в файле в формате json или XML (по выбору пользователя). При добавлении нового автора в базу допускается не заполнять поля в соответствующем окне, а распарсить файл, указанный пользователем (файл необходимо заранее создать и заполнить информацией вручную, в текстовом редакторе). Для *преобразования в формат XML и json* напишите собственный код; *парсинг* можно делать с помощью сторонних библиотек. Форматы файлов:

```
JSON:                          XML:
{
  "name": "L.N.Tolstoi",       <author>
  "country": "Russia",           <name>L.N.Tolstoi</Name>
  "years": [1828, 1910]          <country>Russia</Country>
}                                <years born="1828" died="1910"/>
                               </author>
```

3. Выполните задание № 2 средствами SQLAlchemy, включая создание и редактирование таблиц, а также выполнение таких запросов, как:
   - вывод фамилий всех авторов, родившихся в диапазоне между X и Y годами (задайте программно числа X и Y);
   - вывод всех книг, написанных авторами из России;
   - вывод всех книг с количеством страниц более N;
   - вывод всех авторов с числом книг более N.
4. Выполните задание № 3, используя в качестве базы данных NoSql-технологию MongoDB.
5. Напишите приложение для загрузки файлов из интернета. В главном окне должно быть три текстовых поля, в которые можно вводить URL файла на закачку; под каждым из текстовых полей должны быть индикаторы загрузки и рядом поля с процентом загрузки каждого файла. Необходимо организовать возможность качать от одного до трех файлов параллельно (использовать потоки обязательно, файлы загружать фрагментами по 4 Кб). Загрузка должна инициироваться нажатием кнопки «Start downloading!». По окончанию загрузки последнего файла должно появиться окно со столбчатой диаграммой со значениями времени загрузки каждого

файла в формате «2s 322ms» и размерами файлов (используйте библиотеку matplotlib).

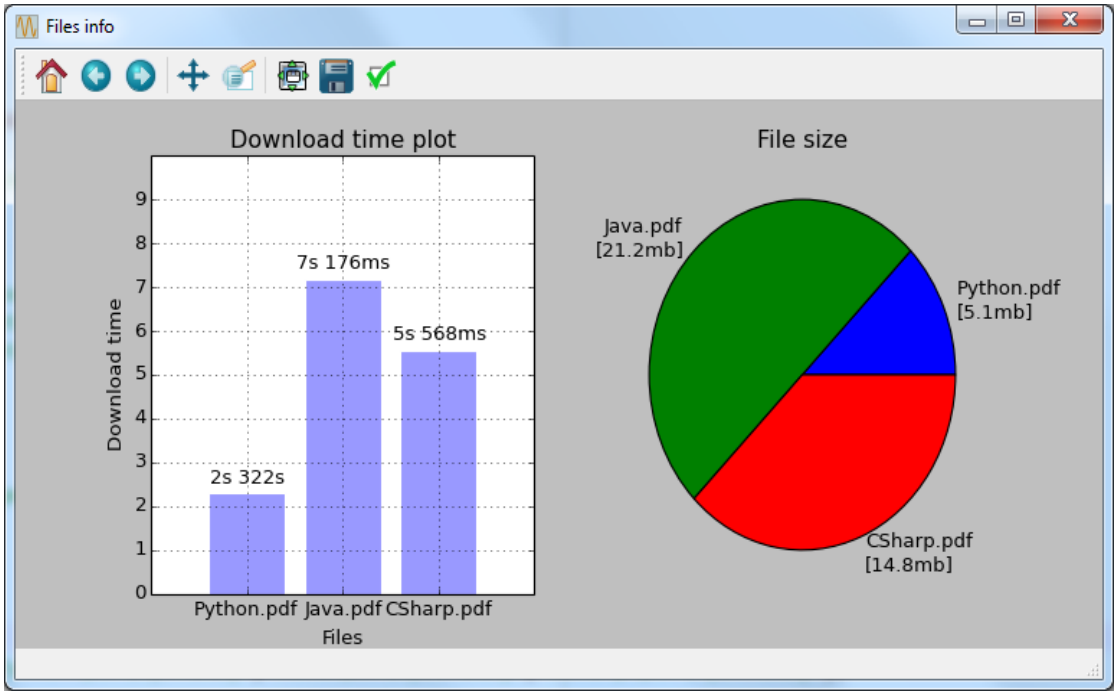

Рисунок 5 – Вид окна загрузки файлов

6. С помощью модуля numPy реализуйте следующие операции: 1) умножение произвольных матриц A (размерности 3x5) и B (5x2); 2) умножение матрицы (5x3) на трехмерный вектор; 3) решение произвольной системы линейных уравнений; 4) расчет определителя матрицы; 5) получение обратной и транспонированной матриц. Также продемонстрируйте на примере матрицы 5x5 тот факт, что определитель равен произведению собственных значений матрицы.
7. Выберите произвольную дифференцируемую и интегрируемую функцию одной переменной. С помощью модуля symPy найдите и отобразите ее производную и интеграл в аналитическом и графическом виде. Напишите код для решения произвольного нелинейного урванения и системы нелинейных уравнений.
8. Скачайте файл с информацией о всех государствах мира по адресу: https://github.com/mledoze/countries/blob/master/dist/countries.csv. С помощью модуля pandas отобразите: 1) 10 самых маленьких и самых больших стран мира по территории; 2) 10 самых маленьких и самых больших стран мира по населению; 3) все франкоязычные страны мира; 4) только островные государства; 5) все страны, находящиеся в южном полушарии. Сгруппируйте страны по первой букве; по населению; по территории. Программно сохраните в таблицу Excel все страны с выборочной информацией: название, столица, население, территория, валюта, широта, долгота.

Контрольные вопросы [ОТЧЕТ]:

1. Какие возможности предоставляет Python для работы с произвольными бинарными данными?
2. Какова типовая схема действий при работе с базами данных в Python?
3. Что такое объектно-реляционные отображения? Какие есть средства в Python по работе с ними?
4. Какие возможности предоставляет Python для работы с сетью?
5. Каковы особенности работы с потоками в Python?
6. Какой функционал предоставляют пакеты sciPy, numPy, symPy, pandas?

Краткая теоретическая справка.

Работа с базами данных в Python производится как через SQL-запросы, так и через объектно-реляционные отображения (Object-Relational Mapping, ORM) SQLAlchemy, Django ORM и другие. Типовая схема выполнения запросов не зависит от СУБД и включает такие блоки, как: 1) установка соединения с БД; 2) получение курсора; 3) функция execute(), непосредственно выполняющая SQL-запрос. Пример кода работы с БД, содержащей две таблицы «Альбом» и «Исполнитель»:

```
import sqlite3
# в этом примере используется SQLite;
# альтернативные коннекторы для других СУБД:
# import psycopg2 для PostgreSQL,
# import pymysql для MySql

DBname = r"db\sqlite3\catalog.db"

# 1) установка соединения
db = sqlite3.connect(DBname)

with db:
     # 2) получение курсора
    cursor = db.cursor()

     # 3) непосредственное выполнение SQL-запроса…
    cursor.execute("SELECT id, name from performers")
     # … и заполнение данных результатами выполнения запроса
    performers = cursor.fetchall()

    for performer in performers:
        print(performer)
         cursor.execute("""
                  SELECT performers.name, albums.name,
                  albums.release_year FROM albums JOIN
                  performers ON albums.perfID=performers.id
                         """)
        albums = cursor.fetchall()
        for album in albums:
            print(album)
```

С помощью модулей requests и BeautifulSoup можно осуществлять простой разбор HTML-страниц. Например, скрипт, находящий на странице все ссылки на mp3-файлы и загружающий их на клиентский компьютер, выглядит так:

```
import requests
from bs4 import BeautifulSoup

WEBSITE = 'http://www.somesite.foo/audio/'
content = requests.get(WEBSITE)

soup = BeautifulSoup(content.text, 'html.parser')

links = soup.find_all('a')
links = filter(lambda a: a.get('href'), links)
links = filter(lambda a: a.get('href').endswith('mp3')
                 and not a.get('href').startswith('?'), links)

for link in links:
    print('Downloading: {}'.format(link.get('href')))
    download_file(link.get('href'))
```

В функции загрузки файла download_file() также используется модуль requests и, в частности, функция iter_content():

```
import requests

DOWNLOAD_FOLDER = r'C:\User\Downloads'

def download_file(url):
    local_path = os.path.join(DOWNLOAD_FOLDER,
                              url.split('/')[-1])

    r = requests.get(url, stream=True)

    with open(local_path, 'wb') as f:
        for chunk in r.iter_content(chunk_size=4096):
            if chunk:
                f.write(chunk)

    return local_path
```

Язык Python очень популярен в научных кругах по всему миру, благодаря таким библиотекам, как numPy (линейная алгебра), sciPy (численные методы, обработка сигналов и др.), symPy (символьные вычисления), pandas (статистика и анализ данных) и matplotlib (визуализация данных). Ниже приведен список только некоторых возможностей, предоставляемых этими библиотеками.

```
np.array([2, 3, 4])              # инициализация вектора напрямую
np.empty(20, dtype=np.float32)   # вектор 20 вещественных чисел
np.zeros(200)                    # инициализация 200 нулями
np.ones((3,3), dtype=np.int32)   # создание матрицы 3x3 из единиц
np.eye(200)                      # создание единичной матрицы
np.zeros_like(a)                 # матрица нулей формы матрицы a
np.linspace(0., 10., 100)        # вектор 100 точек от 0 до 10
np.arange(0, 100, 2)             # точки от 0 до 99 с шагом 2
np.logspace(-5, 2, 100)          # 100 точек на логарифмической
                                 # шкале от 1e-5 до 1e2
np.copy(a)              # копирование матрицы
a.shape                 # кортеж с длинами всех измерений матрицы
a.ndim                  # количество измерений
a.sort(axis=1)          # сортировка вектора
a.flatten()             # формирование одномерного вектора
a.conj()                # комплексно-сопряженная матрица
a.astype(np.int16)      # преобразование к типу int
a.tolist()              # преобразование вектора в список
np.argmax(a, axis=1)    # индекс максимального элемента
np.cumsum(a)            # кумулятивная сумма
np.any(a)               # True, если хоть один элемент равен True
np.all(a)               # True, если все элементы равны True
np.where(cond)          # индексы элементов, для которых
                        # выполняется условие cond
np.where(cond, x, y)    # элементы от x до y, для которых
                        # выполняется условие cond
np.dot(a, b)                  # матричное произведение
np.sum(a, axis=1)             # сумма по измерению 1
a[None, :] * b[:, None]       # внешнее произведение матриц
np.outer(a, b)                # внешнее произведение матриц
np.sum(a * a.T)               # норма матрицы

plot(x,y, '-o', c='red', lw=2, label='bla') # вывод графика
scatter(x,y, s=20, c=color)                 # диаграмма разброса

# 3D-график на плоскости (быстрая версия)
pcolormesh(xx, yy, zz, shading='gouraud')

# 3D-график на плоскости (более медленная версия)
colormesh(xx, yy, zz, norm=norm)

contour(xx, yy, zz, cmap='jet')             # контурный график
n, bins, patch = hist(x, 50)                # гистограмма
imshow(matrix, origin='lower',              # вывод изображения
          extent=(x1, x2, y1, y2))
text(x, y, string, fontsize=12, color='m')  # текстовая надпись
```