ЛР 1. Численное интегрирование

Формулировка задания указана по ссылке: https://replit.com/@zhukov/prog7-t1-lr1?v=1

Сделайте форк данного борда, напишите код, позволяющий решить задачу и представьте ссылку в качестве ответа на данное задание.

```
# Написание программы для численного интегрирования площади под кривой.
def integrate(f, a, b, *, n_iter=1000):
    print(n_iter)



# или можно так:
def integrate2(f, a, b, n_iter=1000):
    print(n_iter)
```

Для вызова функций в первом и во втором случаях доступны следующие способы:

```
integrate(math.sin, 0, 1, n_iter=100)
# иначе аргумент n_iter не передать, неявно (как ниже) - не получится
integrate2(math.cos, 0, 1, 100) # или integrate2(math.cos, 0, 1, n_iter=100)
```

Оценить скорость выполнения интегрирования с помощью модуля timeit при количестве итерацией n_iter = 10**4, 10**5, 10**6 для какой-либо функции (в качестве функции предлагается sin, cos, или tg).

Для борда выше:

Существует два способа вызвать timeit для оценки времени для n_iter=1000 собственно внутри кода:

```
timeit.timeit("integrate(lambda x: x+1, 0, 1, n_iter=1000)",
 setup="from integrate import integrate")
```

или из командной строки:

```
>>>python -m timeit -s "from integrate import integrate"
                  "integrate(lambda x: x+1, 0, 1, n_iter=1000)"
```

Для импорта нескольких библиотек после ключа -s указываем импорты через ";" т.е. например, для импорта функции sin `-s "from integrate import integrate; from math import sin"`

Если будете использовать Jupyter Notebook:

```
%%timeit -n100


integrate(math.atan, 0, math.pi / 2, n_iter=10**5)


integrate(math.atan, 0, math.pi / 2, n_iter=10**6)
```

Если будете делать это в обычной среде, то см документацию.

Например, можно в режиме REPL или из командной строки (см. примеры):

```
>>> import timeit
>>> timeit.timeit('"-".join(str(n) for n in range(100))', number=10000)
```

ЛР 2. Численное интегрирование. Оптимизация

Дополните файл с кодом функции integrate следующим кодом, расположенном по ссылке (или тут) и также проведите замеры времени вычисления для аналогичных параметров модуля timeit для кратного числа потоков и процессов (2, 4, 6). Замеры вычислений — только для количества итераций n_iter=10**6:

```
integrate(math.atan, 0, math.pi / 2, n_iter=10**6)
```

Дополните этой информацией отчёт.

Количество повторений / repeat — 100, единицы измерения — msec. Ссылку на борд с кодом с комментариями на Python в repl.it приведите как ответ на это задание.

ЛР 3. Численное интегрирование. Cython, потоки, joblib

Перепишите функцию integrate с использованием Cython (см. документацию). Можно подумать о следующих вариантах оптимизации:

- объявите переменные с фиксированным типом данных;
- используйте конструкцию nogil (см. пример);
- используйте более быстрые Си-реализации известных мат. функций;
- используйте другой range.

Снова проведите замеры без потоков (аналогично предыдущим заданиям, для n_iter=10**5 и n_iter=10**6) и с потоками, процессами (аналогично для n_iter=10**6): 2, 4, 6. Зафиксируйте замеры.

Перепишите функцию integrate_async из ЛР 2 через Parallel из модуля joblib (см. документацию и пример) и снова сделайте замеры. Изменилось ли что-то?

Отчет в виде Jupyter Notebook или кода с комментариями на Python в GitHub приведите как ответ на это задание.

ЛР 4. Парсинг сайта herzen.spb.ru

Образец борда со стартовым кодом

Спарсить страницу https://www.herzen.spb.ru/main/structure/inst/ и создать json-файл со списком институтов, где структура файла будет такой:

```
[ {"institute_name":"", "url": "", "dep_list":
  [
   {"dep_name": "кафедра", "head_name":"имя", "email":"почта"},
   ...
  ]},
 ...
]
```

дополнить файл с данными, организовав парсинг страниц институтов на сайте https://atlas.herzen.spb.ru/faculty.php. Спарсить список кафедр этого института и дополнить файл информацией о руководителях кафедр этого института: имя и почта.

В качестве ответа приведите ссылку на Google Colab с кодом решения.

Базовый код:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html = urlopen('https://www.herzen.spb.ru/main/structure/inst/')
bs = BeautifulSoup(html, "html.parser")
nameList = bs.find('td',{'class':'block'}).children

# nameList = bs.findAll('td', {'class': 'block'})
# for i in range(10) # for (let i=0; i < 10; i++) {}
# for name in nameList:
# print(name.get_text())
for child in nameList:
  print(child)
```

Лабораторная работа 5

Дан Gist с текстом статьи сэра Тима Бернерса-Ли, оригинал тут: https://www.w3.org/DesignIssues/TimBook-old/History.html

Необходимо с использованием библиотеки nltk решить задачу частеречной разметки и найти 5 (пять) наиболее встречаемых частей речи в этом тексте.

В качестве ответа необходимо представить ссылку на *Google Colab* или на репозиторий GitHub (обозначьте ответы в **README.md**), в которой будет располагаться файл (**ipynb** или **py**) с выводом списка с обозначениями частей речи и количеством их в тексте.

Например:

1. Имя существительное - 123
2. Предлог - 456
3. Прилагательное - 789
4. Междометие - 89
5. Наречие - 42

Подсказка по алгоритму решешиня задачи:

1. Получить текст с помощью requests или urllib.request.
2. Преобразовать в utf-8.
3. Найти функцию для решения частеречной разметки.
4. Не забыть провести токенизацию текста перед разметкой.
5. Определить топ-5 частей речи и найти вывести их на экран в понятном виде.

6. В комментарии внутри борда отмечено какие части речи нам необходимо дополнительно вывести на экран, подсчитав количество встреченных и размеченных этими частями речи слов в тексте (обратите внимание на те части речи, которые нам нужно сложить).

Замечание 1

При работе в коллабе может потребоваться установить некоторые дополнения к nltk для получения ответа.

Например, автору задания потребовалось выполнить следующие команды:

```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

Стартовый борд для задания: https://replit.com/@zhukov/prog7-t2-lz6#main.py

ЛР 6. Обработка текста на Python / NLP with Python

Спарсить страницу https://www.herzen.spb.ru/main/news/ и с использованием регулярных выражений сохранить все ссылки на новости, расположенные на этой странице.

Открыть каждую сохраненную страницу и спарсить основное содержимое новости

Обработать текст и сохранить для каждой новости:
1. Ключевые термины.
2. Ключевых персонажей, упоминавшихся в новости.
3. Построить изображение с облаком тегов для извлеченных ключевых терминов и персонажей.
В качестве ответа приведите ссылку на Google Colab с кодом решения.

Для обработки текста использовать библиотеку natasha. (см. пример).

Для построения облака тегов использовать библиотеку wordcloud (см. пример)

ЛР 7. Счетчик (Flask) в Docker

На основе борда https://replit.com/@zhukov/prog7-t3-lr7#simpleapp.py реализуйте flask-приложение "Счетчик" со следующим функционалом:

- по адресу localhost:port/stat - происходит инкремент счетчика и возвращается текущее его значение в виде html-содержимого;
- по адресу localhost:port/about - вызывается функция hello, написанная в борде в текущей версии с добавлением имени и фамилии студента, выполнившего задание;

Счетчик инициализируется при запуске приложения и его значение хранится в переменной, не сохраняется, если приложение было остановлено.

"Упакуйте" приложение в докер-контейнейнер (в качестве основного образа используйте alpine-версию образа python) и опубликуйте в docker hub.

В ответе предоставьте ссылку на реплит с работающим приложением и на страницу приложения в докерхаб.

ЛР 8

Просмотрите ролик данной темы и выполните задание озвученное в конце ролика. Соответствующий ролику борд с кодом расположен по ссылке:

https://replit.com/@zhukov/prog7-t3-lr8#Dockerfile

Отчет по заданию представьте в виде:

- ссылки на борд в replit с кодом;
- ссылки на образ в Docker Hub, разместите в README.md в репозитории код Dockerfile для создания образа или образов, которые участвуют в создании приложения, инструкцию для развертывания приложения.

Итоговое групповое задание по дисциплине

Цель: реализовать полностью процесс развертывания какого-либо веб-приложения, созданного в рамках дисциплины "Программирование" с использованием методологии DevOps (CI/CD).

Для этого:

1. Выбрать какое-либо приложение (например, приложение для получения курсов валют; приложение, созданное в рамках ЛР 8 или какое-то другое приложение).
2. Выбрать какую-либо платформу для реализации DevOps (по умолчанию, GitHub или GitLab).
3. Внутри группы распределить отдельные этапы, реализуемые в рамках методологии DevOps, на всех участников.
   Например, участник 1 — берет на себя CI и в нем юнит-тестирование, а участник 2 — линтинг и валидацию кода / статический анализ кода, участник 3 — берет развертывание кода в продакшен и т.д.
4. Каждый участник формирует конкретный сценарий выполнения этого этапа с использованием выбранного (на этапе 2) инструмента сборки.
5. Объединить все этапы в общий сценарий и продемонстрировать любым образом процесс сборки.

В качестве ответа на задания представить ссылку на общий репозиторий с файлом README.md, где будет описан процесс сборки и каждый из этапов, реализованных участниками группы.