Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский государственный технический университет им. Н. Э. Баумана»

Лабораторная работа №2

Отчёт о выполненной работы

Выполнил: студент группы СГН3-71

Волынов М.М.

Проверил: кандидат технических наук

Гапанюк Ю.Е.

Москва 2017

Задание на выполнение работы

Основная задача данной работы - знакомство с базовым синтаксисом Python. Дополнительная задача - знакомство с git и github. Git вам понадобится для выполнения и сдачи домашнего задания.

1. Создайте новый проект в PyCharm
    a. в поле Location **untitled** заменить, например, на **lab2**
    b. в поле Interpreter должен быть Python 3.5.x
2. Массивы
    a. Добавьте в проект новый **Python File** с именем **arr_algs.py**
    b. Реализуйте в нём следующие функции:
        i. Нахождение минимума в массиве
        ii. Нахождение среднего арифметического в массиве
    c. Проверьте правильность работы ваших функций, вызвав их несколько раз в том же модуле в конце файла
3. Строки
    a. Добавьте в проект новый **Python File** с именем **str_algs.py**
    b. Реализуйте в нём следующие функции:
        i. Переворот строки (“hello, world” -> “dlrow ,olleh”)
    c. Проверьте правильность работы ваших функций, вызвав их несколько раз в том же модуле в конце файла
4. Словари
    a. Добавьте в проект новый **Python File** с именем **dict_algs.py**
    b. Реализуйте в нём следующий алгоритм:
        i. есть несколько сотрудников, описанных в виде массива словарей emps (данные приведены ниже в конце этого раздела)
        ii. выведите имена тех сотрудников, у которых есть дети старше 18 лет
    c. Проверьте правильность работы вашего алгоритма, вызвав его в том же модуле в конце файла
5. Github
    a. Зайдите на https://github.com
    b. Создайте репозиторий с названием **lab2_repo**
    c. Склонируйте его себе на диск
    d. Добавьте туда ваши .py файлы
    e. commit, push
    f. подробнее - см. в разделе Теория.Git

Исходный код

Содержимое файла **main.py**

```
from utils.clients import VKClientGetID, VKClientGetFriends
from utils.stats import get_age_by_list, plot_ages_histogram

if __name__ == '__main__':
    ids = input('Введите id пользователей (через запятую): ').replace(' ', '').split(',')
    vk_id_getter = VKClientGetID(ids)
    result = vk_id_getter.execute()
    print('Найденные id: {}'.format(', '.join([str(i) for i in result])))

    for user_id in result:
        vk_friends_getter = VKClientGetFriends(user_id)
        result = vk_friends_getter.execute()
        plot_ages_histogram(get_age_by_list(result))
```

Содержимое файла **base.py**

```
import requests


class BaseClient:
    # URL vk api
    BASE_URL = None
    # метод vk api
    method = None
    # GET, POST, ...
    http_method = None

    # Получение GET параметров запроса
    def get_params(self):
        return None

    # Получение данных POST запроса
    def get_json(self):
        return None

    # Получение HTTP заголовков
    def get_headers(self):
        return None

    # Склейка url
    def generate_url(self, method):
        return '{0}{1}'.format(self.BASE_URL, method)

    # Отправка запроса к VK API
    def _get_data(self, method, http_method):
        response = None

        return self.response_handler(response)
```

```
    # Обработка ответа от VK API
    def response_handler(self, response):
        return response

    # Запуск клиента
    def execute(self):
        return self._get_data(
            self.method,
            http_method=self.http_method
        )


class BaseVKClient(BaseClient):
    BASE_URL = 'https://api.vk.com/method/'

    _request_mapper = {
        'GET': requests.get,
        'POST': requests.post,
    }
```

Содержимое файла **clients.py**

```
import json

from utils.base import BaseVKClient


class VKClientGetID(BaseVKClient):
    method = 'users.get'
    http_method = 'GET'

    def __init__(self, user_ids):
        self.user_ids = user_ids

    def response_handler(self, response):
        loaded = json.loads(response.content)
        users = loaded['response']
        return [user['uid'] for user in users]

    def _get_data(self, method, http_method):
        request_provider = self._request_mapper.get(http_method)

        if request_provider is None or self.user_ids is None:
            return None

        url = self.generate_url(method)
        payload = {'user_ids': ','.join(self.user_ids)}
        response = request_provider(url, params=payload)

        return self.response_handler(response)
```

```
class VKClientGetFriends(BaseVKClient):
    method = 'friends.get'
    http_method = 'GET'

    def __init__(self, user_id):
        self.user_id = user_id

    def response_handler(self, response):
        loaded = json.loads(response.content)
        users = loaded['response']
        return users

    def _get_data(self, method, http_method):
        request_provider = self._request_mapper.get(http_method)

        if request_provider is None or self.user_id is None:
            return None

        url = self.generate_url(method)
        payload = {'user_id': self.user_id, 'fields': 'bdate'}
        response = request_provider(url, params=payload)

        return self.response_handler(response)
```

Содержимое файла **stats.py**

```
import datetime
from matplotlib import pyplot as plt


def parse_date(date_str):
    if date_str is None or not isinstance(date_str, str):
        return None

    temp = [int(i) for i in date_str.split('.')]
    l = len(temp)
    result = dict.fromkeys(['day', 'month', 'year'])
    if l == 2:
        result['day'] = temp[0]
        result['month'] = temp[1]
    elif l == 3:
        result['day'] = temp[0]
        result['month'] = temp[1]
        result['year'] = temp[2]

    return result


def get_age(date_dict):
    if not (date_dict and date_dict.get('day') and date_dict.get('month') and date_dict.get('year')):
        return None

    today = datetime.date.today()
    return today.year - date_dict['year'] - ((today.month, today.day) < (date_dict['month'], date_dict['day']))
```

```python
def get_age_by_list(users):
    ages = []
    for user in users:
        bdate_str = user.get('bdate')
        bdate = parse_date(bdate_str)
        age = get_age(bdate)
        if age:
            ages.append(age)
    return ages


def plot_ages_histogram(ages):
    bins = 25
    x_ticks = 5

    plt.hist(ages, bins=bins, rwidth=0.95)
    plt.title('Возраст пользователей')
    plt.xlabel('Возраст')
    plt.xticks(range(0, bins * x_ticks, x_ticks))
    plt.ylabel('Количество')
    plt.grid()
    plt.show()


def analyze_users(users):
    ages = get_age_by_list(users)
    plot_ages_histogram(ages)
```

Скриншоты работы

```
Введите id пользователей (через запятую): stealth_tech
Найденные id: 62071911
```

Рис 1. Ввод изначальных данных

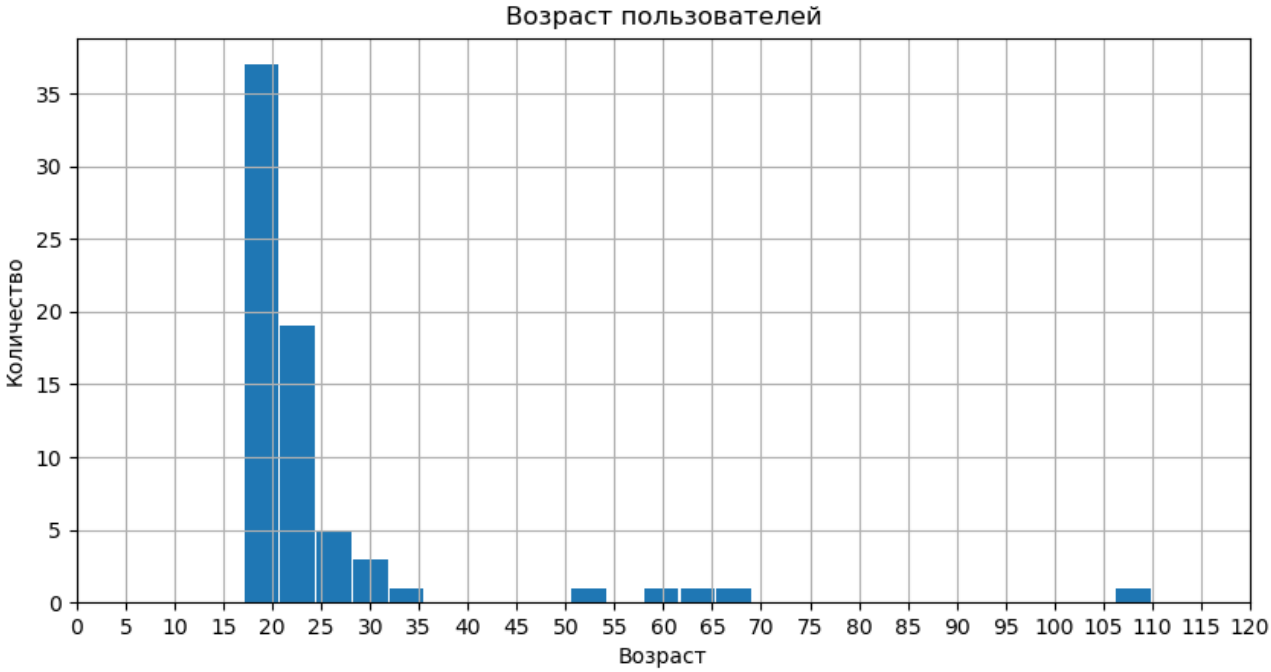

Рис. 2. Визуализация результатов работы

Ссылка на репозиторий с исходным кодом:

https://github.com/StealthTech/WebLanguagesHomework/tree/master/Task2