Московский государственный университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Лабораторные работы по курсу:

«Разработка Интернет Приложений»

ЛР3. Python-классы

Исполнитель:

Студент группы РТ5-51

Крутов Т.Ю.

Преподаватель:

Гапанюк Ю. Е.

«___»_____

Москва 2017 г.

Цель работы:

В этой лабораторной работе необходимо познакомиться с модулями и ООП в Python, а также освоить работу с сетью. Кроме того, необходимо создать набор классов для реализации работы с VK API.

Листинг:

base_client.py:

```
import requests


class BaseClient:
    # URL vk api
    BASE_URL = "https://api.vk.com/method/"
    # 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):

        # Выполнение запроса
        response = requests.get(self.BASE_URL + method + "." + http_method,
                                params=self.get_params())

        if response.status_code != 200:
            print('Error')
            input()

        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
        )
```

ClientGetFriendsAges.py:

```
from datetime import datetime
from BaseClient import BaseClient

class ClientGetFriendsAges(BaseClient):
    # метод vk api
    method = "friends"
    # id пользователя
    user_id = None
    # GET, POST, ...
    http_method = "get"

    json_data = None

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

    def get_params(self):
        return {
            "user_id": self.user_id,
            "fields": "bdate"
        }

    def calculate_age(self, born, today):
        if today is None:
            today = datetime.utcnow()
        return today.year - born.year - ((today.month, today.day) < (born.month,
born.day))

    def response_handler(self, response):
        self.json_data = response.json()
        ages = list()
        today = datetime.utcnow()
        date_tmp = None
        for friend in self.json_data["response"]:
            date_tmp = friend.get("bdate")
            if date_tmp is None or len(date_tmp) < 6:
                continue
            ages.append(self.calculate_age(datetime.strptime(date_tmp, "%d.%m.%Y"),
                                           today))
        return ages

    def get_json(self):
        return self.json_data
```

ClientGetID.py:

```
from BaseClient import BaseClient


class ClientGetID(BaseClient):
    # метод vk api
    method = "users"
    # GET, POST, ...
    http_method = "get"

    json_data = None

    #Создание экземпляра
    def __init__(self, username):
        self.username = username

    # Получение GET параметров запроса
    def get_params(self):
        return {
```

```
            "user_ids": self.username
        }

    # Обработка ответа от VK API
    def response_handler(self, response):
        self.json_data = response.json()
        ids = list()
        try:
            ids.append(self.json_data["response"][0]["uid"])
        except Exception:
            print('Id не найден')
            return None

        return ids

    # Получение данных POST запроса
    def get_json(self):
        return self.json_data
```

gist.py:

```
import matplotlib.pyplot as plt


class Gist:
    # данные гистаграммы
    _data = None
    _data_sorting = dict()

    def _init_(self, date):
        self._data = date
        for number in date:
            self._data_sorting.update({number: self._data_sorting.get(number, 0) + 1})

    def get_data(self):
        return self._data

    def print_hist(self):
        str_out = ""
        for age, stat in self._data_sorting.items():
            str_out += str(age).ljust(4) + str().ljust(stat, '#') + '\n'
        print(str_out)

        plt.hist(self._data)
        plt.xlabel('Возраст')
        plt.ylabel('Кол-во людей')
        plt.title('Гистограмма распределения возрастов друзей')

        plt.show()
```

main.py:

```
from ClientGetID import ClientGetID
from ClientGetFriendsAges import ClientGetFriendsAges
from Gist import Gist

friends_id_arr = None
while friends_id_arr is None:
    input_username = input()

    friends_id_arr = ClientGetID(input_username).execute()

friends_ages = ClientGetFriendsAges(friends_id_arr).execute()

if len(friends_ages) == 0:
    print('Нет друзей :(')
    input()
```

```
else:
    print("ID: ", friends_id_arr)
    print("Ages: ", friends_ages)

    Gist(friends_ages).print_hist()
```

main()

Вывод:

```
52  ##
64  #
31  #
53  #
34  ##
84  #
36  ###
10  #
55  ##
42  #
95  #
20  ##
69  #
48  #
70  ##
43  #
6   #
35  #
81  #
25  #
60  #
38  #
```