Московский государственный технический университет им. Н.Э.Баумана

Защищено:

Гапанюк Ю.Е.

"__"_____2016 г.

Демонстрация ЛР:

Гапанюк Ю.Е.

"__"_____2016 г.

Отчет по лабораторной работе №3
по курсу РИП

Вариант № <26>

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-52

_____

(подпись)

Чекулина М.Ю.

"__"_____2016 г.

Москва, МГТУ - 2016

1. Описание задания лабораторной работы.

В этой ЛР Вы знакомитесь с модулями и ООП в Python, а также осваиваете работу с сетью. В лабораторной работе необходимо создать набор классов для реализации работы с VK API.

Задание

Вход:

username или vk_id пользователя

Выход:

Гистограмма распределения возрастов друзей пользователя, поступившего на вход

Пример:

Вход:

```
reigning
```

Выход:

```
19 #
20 ##
21 ##
22 ####################
23 ################
24 ####
25 #
28 #
29 #
30 #
37 #
38 ##
45 #
```

Указания

За основу возьмите базовый класс:
https://gist.github.com/Abashinos/024c1dcaf92f1ff733c63a07e447ab51

Для реализации методов ВК наследуйтесь от этого базового класса. Создайте один класс для получения id пользователя из username и один для получения и обработки списка друзей. В классах-наследниках необходимо реализовать методы:

- get_params - если есть get параметры (необязательно) .
- get_json - если нужно передать post данные (необязательно) .
- get_headers - если нужно передать дополнительные заголовки (необязательно) .
- response_handler - обработчик ответа. В случае успешного ответа необходим, чтобы преобразовать результат запроса. В случае ошибочного ответа необходим, чтобы сформировать исключение.
- _get_data - внутренний метод для отправки http запросов к VK API.

Для решения задачи нужно обратиться к двум методам VK API

1) users.get - для получения vk id по username

2) friends.get - для получения друзей пользователя. В этом методе нужно передать в get параметрах fields=bdate для получения возраста. Нужно принять во внимание, что не у всех указана дата рождения.

Описание методов можно найти тут:

https://vk.com/dev/methods

Разнесите базовый класс, классы наследники и основную программу в разные модули.Про модули можно прочитать тут: https://docs.python.org/3/tutorial/modules.html
https://habrahabr.ru/post/166463/
Для выполнения запросов нужно использовать библиотеку *requests*
http://docs.python-requests.org/en/master/
Для обработки дат (дней рождения) используйте встроенную библиотеку *datetime*
https://docs.python.org/3/library/datetime.html
Чтобы установить библиотеку используйте пакетным менеджером *pip*
https://pip.pypa.io/en/stable/quickstart/
Подсказки:
1. Метод get библиотеки requests принимает вторым аргументом словарь get-параметров.
2. Не забывайте, что в классах-наследниках можно перегружать статические поля наследуемого класса.

Дополнительное задание
Постройте гистограмму с использованием *matplotlib*
http://matplotlib.org/examples/statistics/histogram_demo_features.html

2. Код программы

Файл base_client.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):
        pass

    # Получение данных POST запроса
    def get_json(self):
        pass

    # Получение HTTP заголовков
    def get_headers(self):
        pass

    # 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
        #передача параметров в URL
        #Requests позволяет передать эти аргументы в качестве словаря, используя
аргумент params.(1 аргумент-кому предаем, 2-что передаем)
        response = requests.get(self.generate_url(method), params=self.get_params())
        # todo выполнить запрос

        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
        )
```

Файл list_of_friends.py

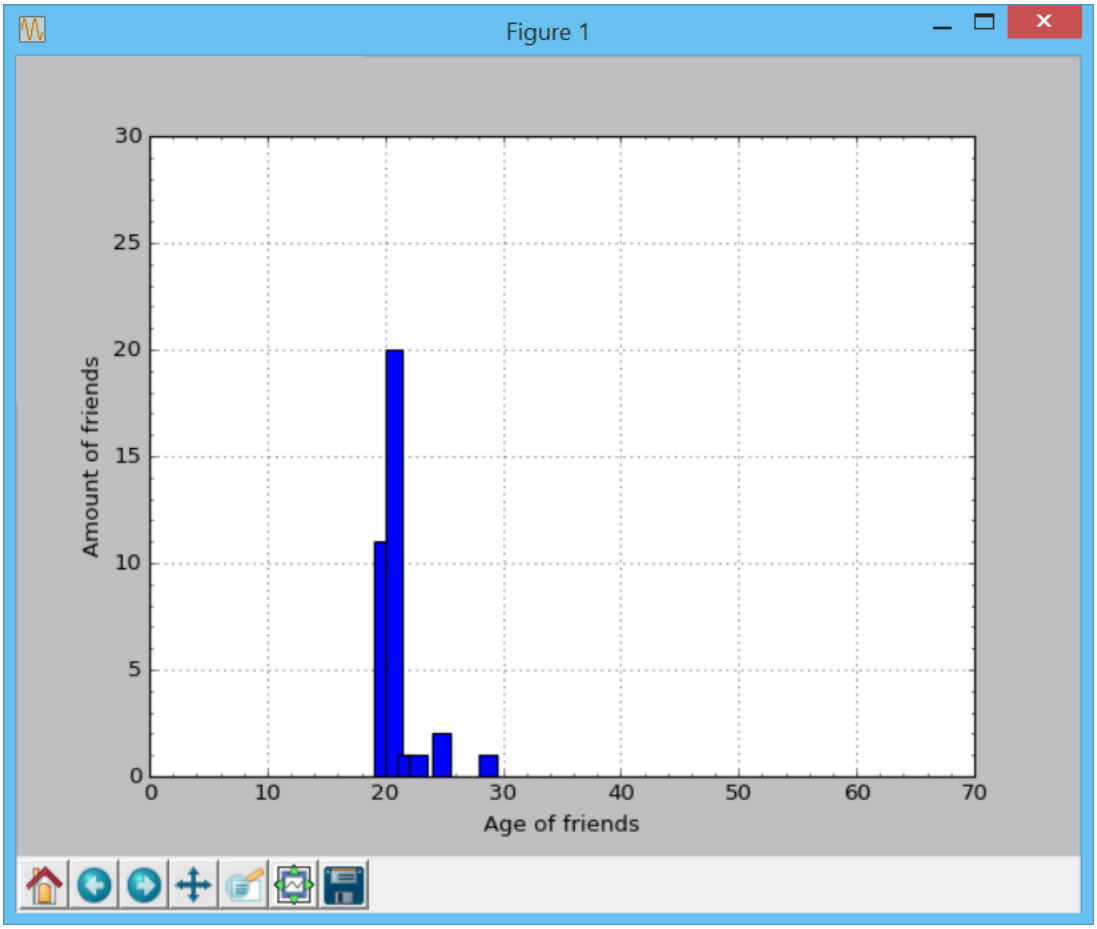
```
from base_client import BaseClient
from datetime import date

#получение возраста
def get_age(birthday):
    today = date.today()
    #формат даты рождения дд.мм.гггг
    #для года
    age = today.year - int(birthday[2], base=10)
    #проверка по месяцам
    if today.month < int(birthday[1], base=10):
        age -= 1
    #проверка по дню в месяце
    elif today.month == int(birthday[2], base=10) and today.day < int(birthday[0],
base=10):
        age -= 1
    return age

class GetUser(BaseClient):
# можно через браузер проверить http://api.vk.com/method/users.get?user ids=id33732528
    BASE_URL = 'https://api.vk.com/method/'
    method = 'users.get'

    def response_handler(self, response):
        # строка
        user_1 = response.json()
        # response-словарь см.main
        us_id = user_1['response'][0]['uid']
        return us_id


    def __init__(self, name):
        self.username = name

    def get_params(self):
        return {
            'user_ids': self.username
        }


class GetFriendsAges(BaseClient):
    BASE_URL = 'https://api.vk.com/method/'
    method = 'friends.get'

    def response_handler(self, response):
        result = []
        friends = response.json()  # массив словарей
        for i in range(len(friends['response'])):
            if ('bdate' in friends['response'][i] and
```

```
len(friends['response'][i]['bdate'].split('.')) == 3):
                result.append(get_age(friends['response'][i]['bdate'].split('.')))
        return result

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

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

Файл main.py

```
from collections import Counter
from list_of_friends import get_age, GetFriendsAges, GetUser
import matplotlib.pyplot as graph

a = GetUser('id33732528')
us_id = a.execute()

# для хранения друзей с возрастом

friends_list = GetFriendsAges(us_id)
result = friends_list.execute()

#гистограмма
#объект класса Counter, очень похожий на словарь (который dictionary), которому передан
список
result_1 = Counter(result)
#print(result_1)
data1 = result_1
#массив для возраста друзей пользователя
friends_ages = []

for i in list(result_1):
    friends_ages.append(result_1[i])
print(result_1)

#Сетка
graph.grid()
graph.minorticks_on()
# интервалы значений по x и y
graph.figure(num=1, figsize=(10, 4))
graph.axis([0, 70, 0, 30])
#Метка по оси x в формате TeX
graph.xlabel('Age of friends', size=12)
#Метка по оси y в формате TeX
graph.ylabel('Amount of friends', size=12)
graph.bar(data1, friends_ages, width=1.5)
#показать график
graph.show()
```

3.Результат выполнения программы




```
"C:\Program Files (x86)\Python 35\Python\Python35\python.exe" C:/Users/Марина/PycharmProjects/lab3/main.py
Counter({20: 20, 19: 11, 24: 2, 21: 1, 22: 1, 28: 1})
```