

**Отчет по лабораторной работе № 3
по курсу
"Разработка интернет приложений"**

" Python-классы "

Бодунов А.Г., ИУ5-53

Москва, МГТУ - 2016 год

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/024c1dc92f1ff733c63a07e447ab51>

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 необходимо, чтобы преобразовать результат запроса. В случае ошибочного ответа необходимо, чтобы сформировать исключение.
- `_get_data` - внутренний метод для отправки http запросов к VK API.

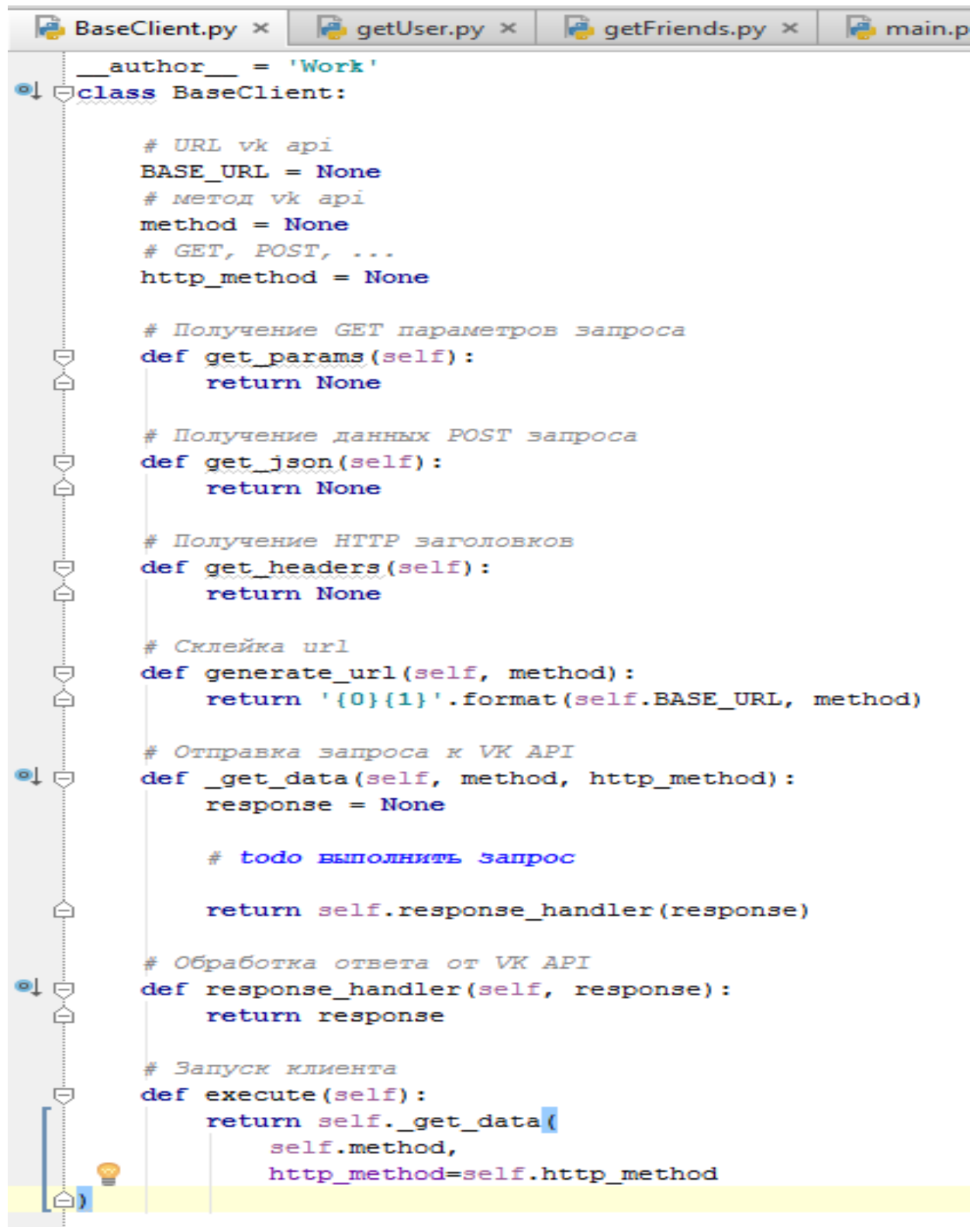
Для решения задачи нужно обратиться к двум методам VK API

- 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>

BaseClient.py:



```
BaseClient.py x  getUser.py x  getFriends.py x  main.p
__author__ = 'Work'
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

        # 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
        )
```

getUser.py:

```
BaseClient.py x  getUser.py x  getFriends.py x  main.py x

__author__ = 'Work'
import BaseClient
import requests
import json

class GetUser(BaseClient.BaseClient):
    def __init__(self, method):
        self.BASE_URL = "http://api.vk.com/method/"
        self.method = method
        self.http_method = 'get'

    def get_dict_data(self): #получаем словарь в результате запросов к серверу
        dict_data = json.loads(self.execute())
        return dict_data

    def _get_data(self, method, http_method):
        response = requests.get(self.generate_url(method))
        return self.response_handler(response)

    def response_handler(self, response):
        dict_data = json.loads(response.text)
        if "error" in dict_data.keys():
            raise Exception
        return response.text
```

getFriends.py:

```
BaseClient.py x  getUser.py x  getFriends.py x  main.py x

__author__ = 'Work'
import BaseClient
import requests
import json

class GetFriends(BaseClient.BaseClient):
    def __init__(self, method):
        self.BASE_URL = "http://api.vk.com/method/"
        self.method = method #в переменной method хранится запрос к серверу (у нас то что нужно получить - пользователя или список друзей пользователя)
        self.http_method = 'get'

    def get_dict_data(self): #получаем словарь в результате запросов к серверу
        dict_data = json.loads(self.execute()) #метод execute() объявлен в родительском классе BaseClient. Он вызывает метод _get_data().
        #метод _get_data() возвращает файл в формате json, а нам нужен словарь
        return dict_data

    def _get_data(self, method, http_method):
        response = requests.get(self.generate_url(method)) #посылаем запрос на сервер
        return self.response_handler(response) #возвращаем только текстовую часть ответа

    def response_handler(self, response): #обработка ответа
        text_part = response.text
        dict_data = json.loads(text_part) #в переменной dict_data содержится словарь, она нужна для проверки наличия ошибок в пределах данной функции
        if "error" in dict_data.keys():
            raise Exception
        return text_part
```

Main.py:

```
BaseClient.py x getUser.py x getFriends.py x main.py x
__author__ = 'Work'
import BaseClient
import getUser
import getFriends
import datetime

def create_args(args):
    return "&".join("%s=%s" % (key, value) for key, value in args.items())

def check_bdatt(obj): #получаем на вход словарь
    return ("bdate" in obj.keys()) and (len(obj["bdate"].split('.')) == 3)

if __name__ == "__main__":
    name=input("Введите ID пользователя ")
    try:
        vk_name=getUser.GetUser('users.get?user_ids=' + name) #создаем объект класса getUser через конструктор с параметром method = "получить пользователей(id=id пользователя)"
        name=vk_name.get_dict_data() #получаем список пользователей в виде словаря
        names=name['response'][0] #так как пользователь с одним ID один единственный, а поле response представляет собой массив, выбираем первый элемент массива
        if (name is None) or ('uid' not in names):
            raise Exception
    except Exception:
        print('Пользователь с данным ID не найден')
        exit()

    #Выведем имя и фамилию пользователя

    print('Имя пользователя VK: ', names['first_name'],
          ' ', names['last_name'])

    now=datetime.date.today()

    old_arr = {} #словарь для количества #

    args = {
        "user_id":names['uid'],
        "fields":"bdate",
        "v":"5.57"
    }
    try:
        vk=getFriends.GetFriends('friends.get?' + create_args(args)) #создаем объект класса getUser через конструктор с параметром method = "получить список друзей пользователя(
        # (преобразованный к параметру get запроса словарь, содержащий id пользователя, поля, которые необходимо вернуть, и версия vk)"
        friends = vk.get_dict_data()["response"]["items"] #получаем массив словарей, каждый из которых содержит возраст друга
    except Exception:
        print('Не удалось получить друзей пользователя')
        exit()

    for x in friends:
        if check_bdatt(x):
            bdate_arr = x["bdate"].split('.')
            bdate = datetime.date(int(bdate_arr[2]),int(bdate_arr[1]),int(bdate_arr[0]))
            diff = int((now - bdate).days/365)
            if diff not in old_arr.keys():
                old_arr[diff]=0
            old_arr[diff]=old_arr[diff]+1

    new_arr=[]

    for key in sorted(old_arr):
        print(str(key) + ' ' + '#'.join(' ' for i in range(old_arr[key] + 1)))
        new_arr.append(key)

import matplotlib.pyplot as plt

plt.hist(
    new_arr,
    25
)

plt.show()
```

Результаты работы программы:

