Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «РИП»

Отчет по лабораторной работе №3

«Python-классы»

Выполнил:
студент группы ИУ5-53
Гатауллин И. И.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2017 г.

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 #
```

Код программы

main.py

```python
import lib.draw_hist as smpl_hist
import lib.friends_class as fr
from lib import user_class as usr
import lib.nice_drawer as nice_hist

if __name__ == '__main__':
    params = {'user_ids': ' id290864283'}

    user = usr.User()
    user.set_params(params)
    user.execute()

    friends = fr.Friends()

    params = {'uid': user.uid,
              'fields': ('bdate')}

    friends.set_params(params=params)
    friends.execute()

    smpl_hist.draw(friends.friends_lst)
    nice_hist.draw(friends.friends_lst)
```

user_class.py

```python
import lib.base_class as bc
from lib import request_exception as exp


class User(bc.BaseClient):

    method = 'users.get'
    def __init__(self):
        super(bc.BaseClient, self).__init__()
        self.uid = None

    # Обработка ответа от VK API
    def response_handler(self, response):
        ret = None
        try:
            data = response.json()

            self.uid = data['response'][0]['uid']
            ret = data['response'][0]
        except:
            raise exp.RequestError('Bad request')
        return ret
```

friends_user.py

```
import datetime
import lib.base_class as bc
from lib import request_exception as exp


class Friends(bc.BaseClient):

    method = 'friends.get'

    def __init__(self):
        super(bc.BaseClient, self).__init__()
        self.friends_lst = []

    def _get_friends_lst(self, data):
        friends_with_full_bdate = []
        for item in data:
            if 'bdate' in item and len(item['bdate'].split('.')) == 3:
                date = datetime.datetime.strptime(item['bdate'],
'%d.%m.%Y').date()

                today = datetime.date.today()

                delta = today - date

                item['age']  = (delta.days // 365)
                friends_with_full_bdate.append(item)
        self.friends_lst = friends_with_full_bdate
        return friends_with_full_bdate

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

        try:
            data = response.json()
            # print(data)
            data = data['response']
        except:
            raise exp.RequestError('Bad request')
        else:
            return self._get_friends_lst(data)
```

draw_hist.py

```
def _find_max(lst):
    max_age = 0
    for item in lst:
        if item['age'] > max_age:
            max_age = item['age']
    # print(max_age)
    return max_age


def draw(lst):
    count_of_ages = [0 for i in range(0, _find_max(lst))]
    for item in lst:
        count_of_ages[item['age']-1] += 1
    for i in range(len(count_of_ages)):
        print(i+1, '#' * count_of_ages[i])
```

nice_drawer.py

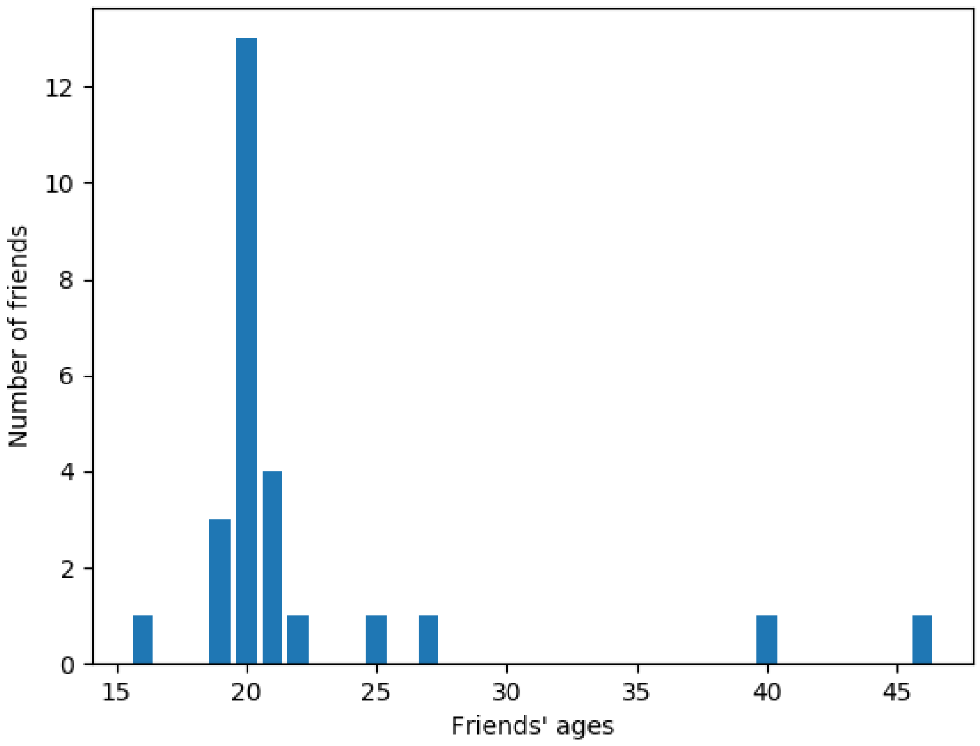
```
import matplotlib.pyplot as plt


def draw(friends_lst):
    ages_count_dict = {}

    for friend in friends_lst:
        if friend['age'] in ages_count_dict:
            ages_count_dict[friend['age']] += 1
        else:
            ages_count_dict[friend['age']] = 1

    x_axis = []
    y_axis = []

    for x, y in ages_count_dict.items():
        x_axis.append(x)
        y_axis.append(y)

    plt.bar(x_axis, y_axis, align='center')

    plt.ylabel('Number of friends')
    plt.xlabel('Friends\' ages')
    plt.show()
```

base_class.py

```
import requests as req


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

    def __init__(self):
        self.params = {}

    def set_params(self, params):
        self.params = params
        #print (self.params)

    # Получение GET параметров запроса
    def get_params(self):
        return self.params
    # 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):

        url = self.generate_url(method)
        response = req.get(url, params=self.get_params())
        # print(response.headers)
        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
        )
```

Результат

```
…
14
15
16 #
17
18
19 ###
20 #############
21 ####
22 #
23
24
…
```



Number of friends
0
2
4
6
8
10
12
15
20
25
30
35
40
45
Friends' ages