

Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления» Кафедра
«Автоматизированные системы обработки информации и управления»



“Разработка интернет-приложений ”

«Python-классы»

Лабораторная работа № 3

Студент группы ИУ5 -53 _____ Юркевич А. Д.

Преподаватель _____ Гапанюк Е.Ю.

Москва 2017

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/024c1dc9f92f1ff733c63a07e447ab51>

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

Код программы: Classes.py

```
import requests
import
base_client import
json import
datetime
import matplotlib.pyplot as plt

class GetId(base_client.BaseClient):
    BASE_URL = "https://api.vk.com/method/"
    method = 'users.get'
    http_method = 'get'

    # Отправка запроса к VK API def
    _get_data(self, method, http_method):
        screen_name = input()
        response = requests.get(base_client.BaseClient.generate_url(self,
        GetId.method), params={'user_ids': screen_name}) return
        self.response_handler(response)

    # Обработка ответа от VK API def
    response_handler(self, response):
        if response.status_code == requests.codes.ok:
            res = str(response.json()[0]['response'])[0]['uid']
return res else:
        print("Error!")

class FriendsAnalytics(base_client.BaseClient):
    BASE_URL = "https://api.vk.com/method/"
    method = 'friends.get'
    http_method = 'post'
    user_id = None

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

    # Отправка запроса к VK API def
    _get_data(self, method, http_method):
        data = {'user_id': self.user_id, 'count': '5000', 'fields': 'bdate'}
        response = requests.post(base_client.BaseClient.generate_url(self,
        self.method), data=data) return self.response_handler(response)

    # Обработка ответа от VK API def
    response_handler(self, response):
        if response.status_code == requests.codes.ok:
            return response else:
                print("Error!")
                def Diagram(self, response):
```

```

if response.json()['response'] is not None: # Есть ли друзья
    blist = list()
    for person in response.json()['response']:
        if person.get('bdate') is None:      # Человек с не пустой
            датой
                continue
            else:
                try:

blist.append(datetime.datetime.strptime(person['bdate'], '%d.%m.%Y'))
except:
    continue

today = datetime.datetime.today()
agelist = list()
for b in blist:
    age = today.year
    - b.year - 1
    if today.month <= b.month:
        if today.day <= b.day:
            age += 1
        agelist.append(age)
agelist.sort()
pred = agelist[0]
count = []
age = []
age.append(pred)
count.append(0)

i = 0

for a in agelist:
    if a==pred:
        count[i] += 1
    else:
        age.append(a)
        count.append(1)
        pred = a
        i += 1

plt.title("friend")
plt.xlabel("age")
plt.ylabel("count")
# представляем точки (x,y) кружочками диаметра
10 plt.plot(age, count, 'r')

# Сетка на фоне для улучшения восприятия
plt.grid(True, linestyle='-', color='0.75')

plt.show()
'''temp = agelist[0]
print(agelist[0], end='\t')
for a in agelist:
    if a != temp:
        print()
        print(a, end='\t')
        print('#', end='')
        temp = a
    else:

```

```

        print('#', end='')

'''

    # Запуск клиента    def
execute(self):# -> object:
res = self._get_data(self.method, http_method=self.http_method) if res ==
    "Error!":
        print("Error!")
        return res

    # Печать диаграммы
    self.Diagram(res)

```

Main.py

```

import requests import
datetime
import json
import base_client

import classes

a = classes.GetId().execute()
print(a) b =
classes.FriendsAnalytics(a)
b.execute()

```

Base_client.py

```

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):# -> object: return
    self._get_data(self.method,
        http_method=self.http_method
    )

```

Результаты работы:



