**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Системы обработки информации и управления»

Кафедра ИУ5. Курс «РИП»

Отчет по лабораторным работе №3

«Python-классы»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-53 |  | преподаватель каф. ИУ5 |
| Белков А.Д. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2017 г.

**Задание и порядок выполнения ЛР №3**

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

**Код программы**

**base.py**

import requests  
  
  
class VkRequest:  
 # URL vk api  
 BASE\_URL = 'https://api.vk.com/method/'  
 # метод vk api  
 method = None  
 # GET, POST, ...  
 http\_method = None  
 parameters = ()  
  
 # Получение GET параметров запроса  
 def get\_params(self):  
 return self.parameters  
  
 # Получение данных POST запроса  
 def get\_json(self):  
 return self.response.json()  
  
 # Получение HTTP заголовков  
 def get\_headers(self):  
 return self.response.headers  
  
 # 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):  
 try:  
 if http\_method == 'get':  
 self.response = requests.get(self.generate\_url(method), params=self.get\_params())  
 except ConnectionError:  
 print('Неудачная попытка отправки запроса')  
 else:  
 return self.response\_handler(self.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  
 )

**get\_friends.py**

from base import \*  
  
  
class GetFriends(VkRequest):  
 method = 'friends.get'  
 http\_method = 'get'  
  
 def \_\_init\_\_(self, user\_id):  
 self.parameters = (('user\_id', user\_id), ('fields', 'bdate'))

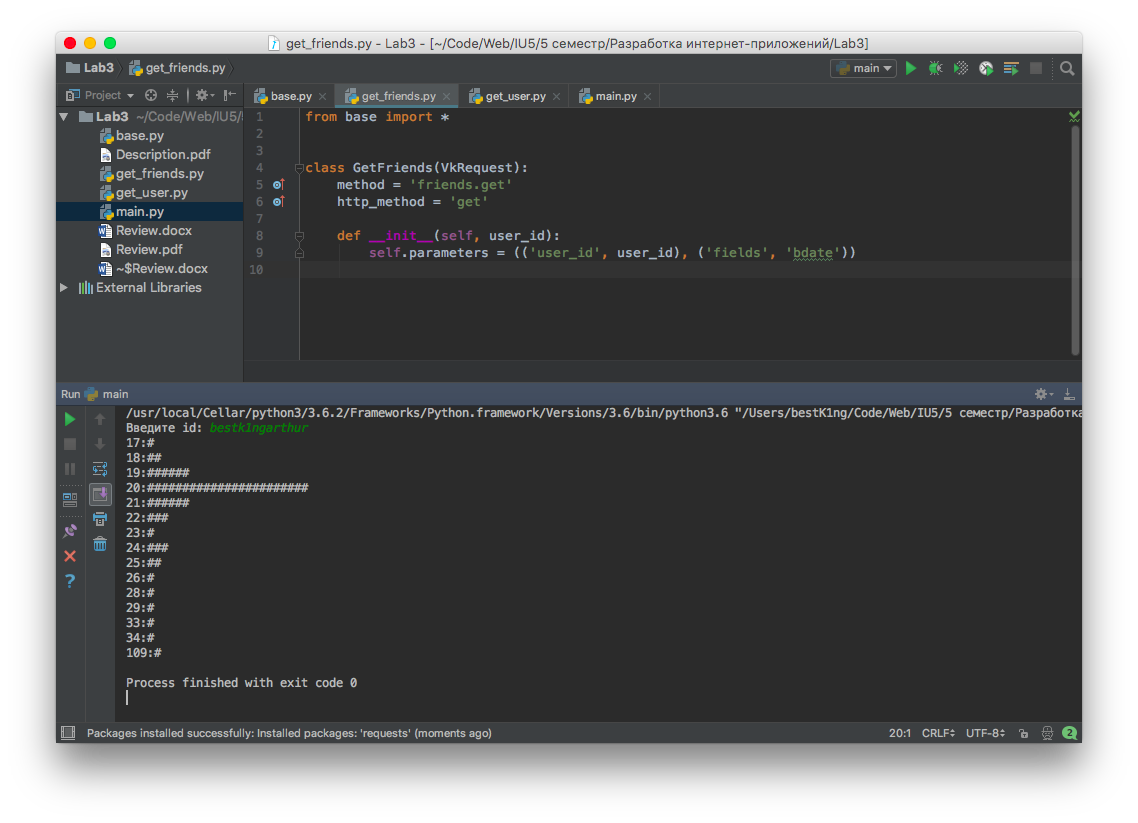
**get\_user.py**

from base import \*  
  
  
class GetUser(VkRequest):  
 method = 'users.get'  
 http\_method = 'get'  
  
 def \_\_init\_\_(self, username):  
 self.parameters = (('user\_ids', username), ('fields', 'bdate'))

**main.py**

import datetime  
from get\_friends import \*  
from get\_user import \*  
  
  
def make\_chart(friends):  
 chart = {}  
  
 for friend in friends:  
 date\_birth = friend['bdate']  
 date = datetime.datetime.strptime(date\_birth, '%d.%m.%Y')  
 years = int((datetime.datetime.now() - date).days / 365.25)  
  
 if years in chart:  
 chart[years] += 1  
 else:  
 chart[years] = 1  
 return chart  
  
  
def draw\_chart(chart):  
 keys = sorted(chart.keys())  
 for i in keys:  
 print("{}:{}".format(i, chart[i] \* '#'))  
  
  
def load\_with\_id(user\_id):  
 final\_friends = []  
  
 request = GetFriends(user\_id)  
 try:  
 request.execute()  
 friends = request.get\_json()['response']  
 except KeyError:  
 print('Incorrect id or username')  
 else:  
 for friend in friends:  
 if 'bdate' in friend:  
 if len(friend['bdate'].split('.')) > 2:  
 final\_friends.append(friend)  
  
 chart = make\_chart(final\_friends)  
 draw\_chart(chart)  
  
  
def load\_with\_username(username):  
 request = GetUser(username)  
 try:  
 request.execute()  
 user = request.get\_json()['response']  
 except KeyError:  
 print('Incorrect id or username')  
 else:  
 user\_id = user[0]['uid']  
 load\_with\_id(user\_id)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 user\_input = input("Введите id: ")  
  
 if user\_input.isdigit():  
 user\_id = int(user\_input)  
 load\_with\_id(user\_id)  
 else:  
 username = user\_input  
 load\_with\_username(username)

**Скришоты выполнения**

****