**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «РИП»

Отчет по лабораторной работе №3

## **«**Python-классы**»**

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

**import** lib.draw\_hist **as** drawer  
**import** lib.friends\_class **as** fr  
**from** lib **import** user\_class **as** usr  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
  
 params = {**'user\_ids'**: **'ryleva97'**}  
  
 user = usr.User()  
 user.set\_params(params)  
 user.execute()  
 *#print(user.uid)* friends = fr.Friends()  
  
 params = {**'uid'**: user.uid,  
 **'fields'**: (**'bdate'**)}  
  
 friends.set\_params(params=params)  
 friends.execute()  
  
 drawer.draw(friends.friends\_lst)

**user\_class.py**

**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** *# self.params = {'user\_ids': 'academeg'}  
  
 # Обработка ответа от VK API* **def** response\_handler(self, response):  
 ret = **None  
 try**:  
 data = response.json()  
 self.uid = data[**'response'**][0][**'uid'**]  
 *#print(self.uid)* ret = data[**'response'**][0]  
 *#print(ret)* **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  
 *#print(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'**]  
 *#print(data)* **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])

**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  
 )