

**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н. Э. БАУМАНА**

**Лабораторной работы по курсу:**

**«Разработка Интернет Приложений»**

**ЛР4. Python. Функциональные возможности**

Выполнил: Житенев В.Г.

Группа РТ5-41

Преподаватель: Гапанюк Ю.Е.

Москва 2017 г.

Цель работы:

В этой лабораторной работе необходимо познакомиться с модулями и ООП в Python, а также освоить работу с сетью. Кроме того, необходимо создать набор классов для реализации работы с VK API.

Листинг:

**baseclient.py**

**import** requests  
  
  
**class** BaseClient:  
 *# URL vk api* BASE\_URL = **"https://api.vk.com/method/"** *# метод vk api* method = **None** *# GET, POST, ...* http\_method = **None  
  
 def** \_\_init\_\_(self):  
 *# Инициализация* self.success = **True  
  
 def** get\_params(self):  
 **return None  
  
 def** get\_json(self):  
 *# Получение данных POST запроса* **return None  
  
 def** \_get\_data(self, method, http\_method):  
 *# Отправка запроса к VK API* **try**:  
 response = requests.get(self.BASE\_URL + self.method + **"."** + self.http\_method, params=self.get\_params())  
 print(self.BASE\_URL + self.method + **"."** + self.http\_method)  
  
 **except** Exception:  
 **raise** SystemExit(**"Нет ответа от VK"**)  
  
 **return** self.response\_handler(response)  
  
 **def** response\_handler(self, response):  
 *# Обработка ответа от VK API* **return** response  
  
 **def** is\_success(self):  
 *# Проверка найден ли ID* **return** self.success  
  
 **def** execute(self):  
 *# Запуск клиента* **try**:  
 self.success = **True  
 return** self.\_get\_data(  
 self.method,  
 http\_method=self.http\_method  
 )  
 **except** Exception:  
 *#print('Ошибка')* self.success = **False**

**clientVK.py**

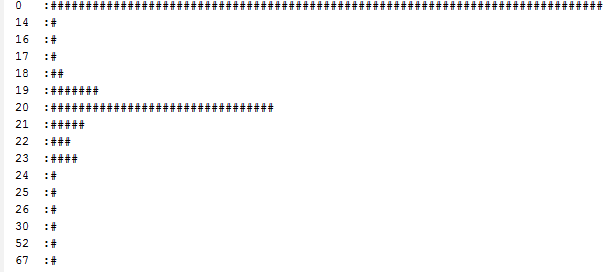
**import** base\_client  
**from** datetime **import** datetime  
  
  
**class** ClientGetID(base\_client.BaseClient):  
 *# метод vk api* method = **"users"** *# GET, POST, ...* http\_method = **"get"  
  
 def** \_\_init\_\_(self, username):  
 *# Инициализация* super().\_\_init\_\_()  
 self.json\_data = **None** self.username = username  
  
 **def** get\_params(self):  
 *# Получение логина* **return** {  
 **"user\_ids"**: self.username  
 }  
  
 **def** response\_handler(self, response):  
 *# Получение ID пользователя* self.json\_data = response.json()  
 print(self.json\_data)  
 *# print("Json data = ", self.json\_data)* **return** self.json\_data[**"response"**][0][**"uid"**]  
  
 **def** get\_json(self):  
 *# Получить json строку* **return** self.json\_data  
  
  
**def** calculate\_age(born, today):  
 *# Вычисление возраста* **return** today.year - born.year - ((today.month, today.day) < (born.month, born.day))  
  
  
**class** ClientGetFriendsAges(base\_client.BaseClient):  
 *# метод vk api* method = **"friends"** *# GET, POST, ...* http\_method = **"get"  
  
 def** \_\_init\_\_(self, user\_id):  
 *# Инициализация* super().\_\_init\_\_()  
 self.json\_data = **None** self.user\_id = user\_id  
  
 **def** get\_params(self):  
 **return** {  
 **"user\_id"**: self.user\_id,  
 **"fields"**: **"bdate"** }  
  
 **def** response\_handler(self, response):  
 *# Получение списка возрастов* self.json\_data = response.json()  
 ages = list()  
 today = datetime.utcnow()  
  
 **for** friend **in** self.json\_data[**"response"**]:  
 date\_of\_birth = friend.get(**"bdate"**)  
 print (date\_of\_birth)  
  
 **try**:  
 date\_of\_birth = datetime.strptime(date\_of\_birth, **"%d.%m.%Y"**)  
 **except** (ValueError, TypeError):  
 print(**"Ошибка формата даты"**)  
 date\_of\_birth = datetime.today()  
 *# continue* ages.append(calculate\_age(date\_of\_birth, today))  
  
 **return** ages  
  
 **def** get\_json(self):  
 **return** self.json\_data()

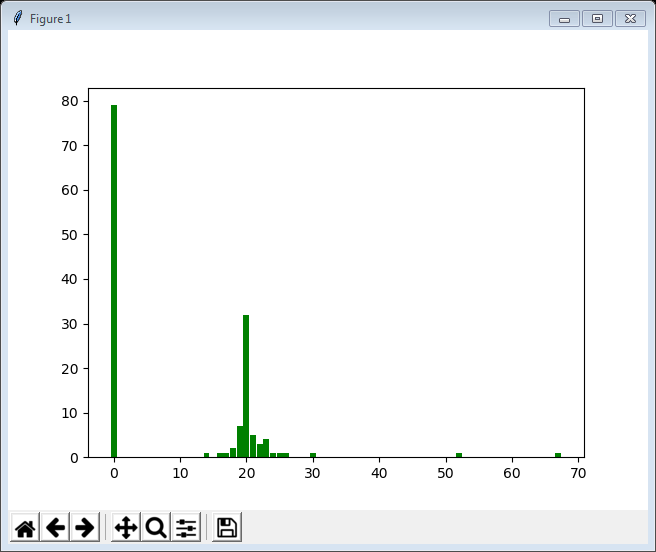
**main.py**

**from** clientVK **import** \*  
**from** histogram **import** Hist  
**import** matplotlib.pyplot **as** plt  
  
  
**def** main():  
 username = input(**'Введите логин VK: '**)  
  
 client\_get\_id = ClientGetID(username)  
 user\_id = client\_get\_id.execute()  
  
 **if** client\_get\_id.is\_success():  
 print(**"ID: "**, user\_id)  
 **else**:  
 print(**'Не существует пользователя с таким ID.'**)  
 **return** 0  
  
 *# find age list* friends\_ages\_list = ClientGetFriendsAges(user\_id).execute()  
 **if not** friends\_ages\_list:  
 print(**'Друзей не найдено'**)  
 **return** 0  
 **else**:  
 print(**"Ages: "**, friends\_ages\_list)  
 *# write gist* username\_friend\_gist = Hist(friends\_ages\_list)  
 username\_friend\_gist.printGist()  
  
 *# show gist* title = **"Гистограмма"** title\_x = **"Возраст"** title\_y = **"Количество друзей"** username\_friend\_gist.showBar(title, title\_x, title\_y)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"** : main()

**histogram.py**

**import** matplotlib.pyplot **as** plt  
  
  
**class** Hist:  
 *# данные гистаграммы* \_age\_dictionary = dict()  
  
 **def** \_\_init\_\_(self, age\_list):  
 self.\_ages\_list = sorted(age\_list)  
 **for** value **in** self.\_ages\_list:  
 self.\_age\_dictionary.update(  
 {value: self.\_age\_dictionary.get(value, 0) + 1})  
  
 **def** get\_data(self):  
 **return** self.\_ages\_list  
  
 **def** printGist(self):  
 **for** age, count **in** self.\_age\_dictionary.items(): *# dict.items возвращает пары* print(str(age).ljust(4) + **":"** + **"#"** \* count)  
  
 **def** showBar(self, title1, title\_x, title\_y):  
 plt.bar(list(self.\_age\_dictionary.keys()),  
 self.\_age\_dictionary.values(), color=**'g'**, width=0.9, linewidth=20)  
 plt.show()

****

****

**Ключевой особенностью программы является обработка пользователей с некорректной датой рождения на страницы в ВК (отсутствует год рождения или дата отсутствует полностью). Такие пользователи получают «нулевой» год рождения, как показано на гистограмме**