**Московский Государственный Технический Университет им. Н.Э. Баумана**

**Разработка интернет-приложений**

Отчёт по лабораторной работе №3

«Python классы»

Выполнил:

студент группы ИУ5-51

Корзин Алексей

# Цель работы

В лабораторной работе необходимо создать набор классов для реализации работы с VK API.

# Листинг программы

## Модуль derived\_client.py

**import** requests

**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):

resp = requests.get(self.BASE\_URL + self.method, params=self.get\_params())

**return** self.response\_handler(resp)

*# 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

)

## Модуль get\_user\_id.py

**import** json **import** base\_client

**class** GetUserId(base\_client.BaseClient): BASE\_URL = **"https://api.vk.com/method/"** method = **"users.get"** http\_method = **"GET"** user\_ids = **""**

**def** get\_params(self):

**return** {

**"user\_ids"**: self.user\_ids

} **def** get\_json(self, data): **return** json.dumps(data)

**def** response\_handler(self, response):

resp = response.json()

**return** resp

## Модуль derived\_client.py

**import** base\_client **import** json **import** get\_user\_id **import** sys

**import** matplotlib.pyplot **as** plt **import** numpy **as** np **from** \_datetime **import** datetime

**def** calculate\_age(born):

today = datetime.utcnow()

**return** today.year - born.year -((today.month, today.day) < (born.month, born.day))

**def** draw\_distribution(array, draw\_hist):

*#find min and max in ages to determine range of distribution* min = array[0] max = array[0] **for** i **in** range(1, len(array)):

**if** array[i] < min: min = array[i] **if** array[i] > max: max = array[i]

*#initialize and count number of people in each age* distribution = {} **for** i **in** range(min, max + 1):

distribution[i] = 0 **for** i **in** range(0, len(array)):

*#key in dict is age* distribution[array[i]] += 1 *#find max count in distribution* dmax = 0 **for** i **in** distribution:

**if** distribution[i] > dmax: dmax = distribution[i]

*#x/dmax, where x is length of the highest column* relation = 80/dmax;

*#normalize distribution by relation* x = [] y = [] **for** i **in** distribution:

distribution[i] \*= relation

distribution[i] = int(round(distribution[i], 0))

x.append(i)

y.append(distribution[i]) **if**(draw\_hist): plt.hist([array], range(min, max + 1)) plt.show() **else**:

*#print distribution* **for** i **in** distribution: sys.stdout.write(str(i)) sys.stdout.write(**" "**) **for** i **in** range(0, distribution[i]):

sys.stdout.write(**"#"**) print(**''**);

**class** GetFriends(base\_client.BaseClient): BASE\_URL = **"https://api.vk.com/method/"** method = **"friends.get"** http\_method = **"GET"** user\_id = **""**

**def** get\_params(self):

**return** {

**"user\_id"**: self.user\_id, **"fields"**: **"bdate"**

} **def** get\_json(self, data): **return** json.dumps(data)

**def** response\_handler(self, response): resp = json.loads(response.text) **return** resp

g = get\_user\_id.GetUserId()

g.user\_ids = **"durov"** user = g.execute() print(user) uid = user[**"response"**][0][**"uid"**] print(uid) c = GetFriends()

c.user\_id = uid friends = c.execute() print(friends)

ages = [] **for** f **in** friends[**"response"**]:

**if "bdate" in** f:

**if** len(f[**"bdate"**]) > 5:

ages.append(calculate\_age(datetime.strptime(f[**"bdate"**], **"%d.%m.%Y"**)))

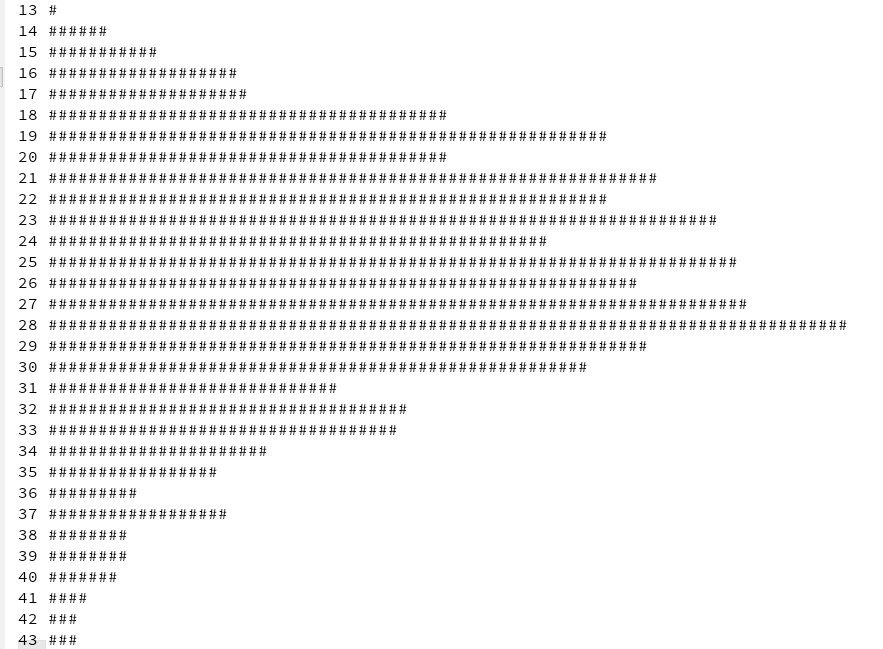
print(ages)

draw\_distribution(ages, **False**)

plt.show()

# Результат работы

Никнейм пользователя: oblomoff (3000+ друзей) В режиме вывода в консоль:



В режиме построения графика:

