**Московский государственный технический университет им. Н.Э. Баумана**

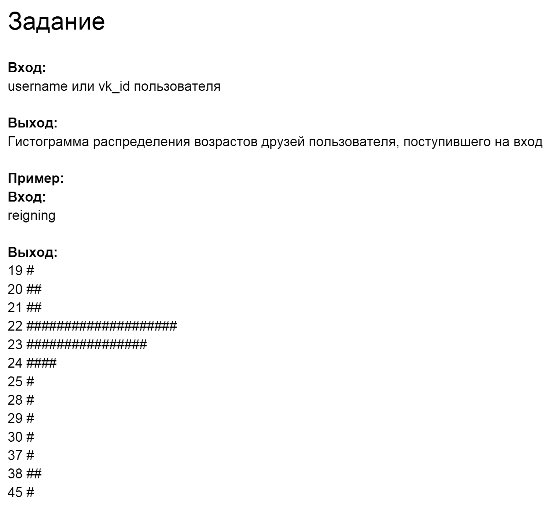
|  |  |
| --- | --- |
| Защищено:  Гапанюк Ю. Е.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. | Демонстрация ЛР:  Гапанюк Ю. Е.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. |

**Отчет по лабораторной работе №3**

**«Python-классы»**

|  |  |
| --- | --- |
|  | ИСПОЛНИТЕЛЬ:  студент группы ИУ5-52  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  Лунев М. А.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. |

Москва, МГТУ - 2016



Листинг

base\_client.py

import requests

class BaseClient:

BASE\_URL = None

method = None

http\_method = None

def get\_params(self):

pass

def get\_json(self):

pass

def get\_headers(self):

pass

def generate\_url(self, method):

return '{0}{1}'.format(self.BASE\_URL, method)

def \_get\_data(self, method, http\_method):

payload = self.get\_params()

response = requests.get(self.BASE\_URL, params=payload)

return self.response\_handler(response)

def response\_handler(self, response):

return response

def execute(self):

return self.\_get\_data(

self.method,

http\_method=self.http\_method

)

friend\_age.py

from base\_client import BaseClient

from datetime import datetime

import requests, json

class FriendsAge(BaseClient):

BASE\_URL = 'https://api.vk.com/method/friends.get'

http\_method = 'GET'

def \_\_init\_\_(self, userId):

self.userId = userId

def get\_params(self):

return {

'user\_id': self.userId ,

'fields': 'bdate'

}

def response\_handler(self, response):

try:

parsed = json.loads(response.text)

list = parsed.get('response')

ages = []

for person in list:

bdate = person.get('bdate')

if bdate is not None:

try:

date = datetime.strptime(bdate, "%d.%m.%Y")

now = datetime.now()

ages.append(int((now - date).days / 365))

except ValueError:

continue

return ages

except:

return None

id\_from\_username.py

from base\_client import BaseClient

import requests, json

class IDFromUsername(BaseClient):

BASE\_URL = 'https://api.vk.com/method/users.get'

http\_method = 'GET'

def get\_params(self):

return {

'user\_ids': self.name

}

def \_\_init\_\_(self, name):

self.name = name

def response\_handler(self, response):

parsed = json.loads(response.text)

try:

return parsed.get('response')[0].get('uid')

except:

return -1

iterators.py

# Итератор для удаления дубликатов

class Unique(object):

def \_\_init\_\_(self, items, \*\*kwargs):

self.items = list(items)

self.max\_size = len(self.items)

self.index = 0

self.set = set()

try:

self.ignore\_case = kwargs['ignore\_case']

except:

self.ignore\_case = False

def \_\_next\_\_(self):

if self.index == self.max\_size:

raise StopIteration

item = self.items[self.index]

if self.ignore\_case is True:

item = item.lower()

while item in self.set:

self.index = self.index + 1

if self.index == self.max\_size:

raise StopIteration

item = self.items[self.index]

if self.ignore\_case is True:

item = item.lower()

self.set.add(item)

return item

def \_\_iter\_\_(self):

return self

main.py

from friends\_age import FriendsAge

from id\_from\_username import IDFromUsername

import matplotlib.pyplot as plt

input = 1 #input()

getID = IDFromUsername(input)

ID = getID.execute()

getAges = FriendsAge(ID)

ages = getAges.execute()

max\_age = max(ages)

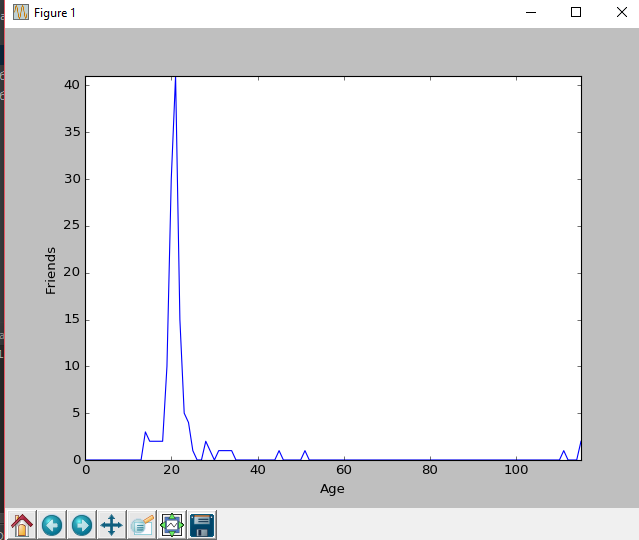
osY = [0] \* (max\_age + 1)

for age in ages:

osY[age] += 1

osX = range(max\_age + 1)

plt.axis([0, max\_age, 0, max(osY)])



plt.xlabel('Age')

plt.ylabel('Friends')

plt.plot(osX, osY)

plt.show()

Проверка