class BaseClient: #базовый класс

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):

response = None

# todo выполнить запрос

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

)

import requests ## импорт необходимых библиотек

import json

import time

from collections import Counter

class vk\_api(BaseClient): #наследуемся от базового класса

user\_id,params,username = None,'',None # ввод используемых переменных

age=[]

c=[]

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

self.BASE\_URL='http://api.vk.com/method/' #объявление ссылки на метод

self.http\_method='GET' #объявление метода

self.username=username

def get\_data(self,method,params): #функция получения информации (об аккаунте)

self.method = method

r=requests.get(self.generate\_url(self.method),params)

data = r.json()

return data

def get\_param(self, \*\*kwargs): #функция формирования запроса

for key in kwargs:

self.params = self.params+"&"+key+"="+str(kwargs[key]) #str т.к. int-->str

return self.params

def get\_id(self): #получаем ТОЛЬКО id пользователя (за только отвечает # ["response"][0]['uid'])

return self.get\_data("users.get",self.get\_param(uids=self.username,v='3.0'))["response"][0]['uid']

def get\_friends(self): #функция поиска друзей

friends = self.get\_data("friends.get",self.get\_param(user\_id=self.get\_id(),fields="bdate",v='3.0'))

for friend in friends.get('response'): #идем по словарю приписанному к ключу "response'

if friend.get("bdate"):

date = friend["bdate"].split(".") #разделение

if len(date)>2: #проверка даты

self.age.append(int((int(time.time())-int(time.mktime(time.strptime(friend["bdate"], '%d.%m.%Y'))))/31536000)) #вычисляем текущий возраст

self.age = dict(Counter(self.age)) #словарь возрастов - "возраст" :"кол-во раз"

return self.age

def print\_sharp(self,count): #функция выводящая в строчку count-решеток

sharp = '';

for i in range(count):

sharp = sharp + "#"

return sharp

def print\_age(self): #вывод возрастов из словаря и приписка к ним решеток с помощью print\_sharp

for i in sorted(self.age): #+сортиртировка словаря

print("Age {} : {}".format(i,self.print\_sharp(self.age[i])))

###################################################################

# выполнение

vk = vk\_api('id96115000')

print(vk.get\_id())

print(vk.get\_friends())

vk.print\_age()

#результат

96115000

{14: 6, 15: 12, 16: 29, 17: 21, 18: 27, 19: 69, 20: 44, 21: 15, 22: 7, 23: 15, 24: 12, 25: 10, 26: 17, 27: 18, 28: 26, 29: 11, 30: 12, 31: 8, 32: 12, 33: 4, 34: 3, 35: 2, 36: 2, 37: 2, 38: 2, 41: 3, 42: 1, 46: 2, 48: 1, 56: 2, 58: 1, 89: 2, 90: 2, 95: 1, 100: 1, 101: 1, 111: 1, 112: 1, 113: 2, 114: 3, 115: 10}

Age 14 : ######

Age 15 : ############

Age 16 : #############################

Age 17 : #####################

Age 18 : ###########################

Age 19 : #####################################################################

Age 20 : ############################################

Age 21 : ###############

Age 22 : #######

Age 23 : ###############

Age 24 : ############

Age 25 : ##########

Age 26 : #################

Age 27 : ##################

Age 28 : ##########################

Age 29 : ###########

Age 30 : ############

Age 31 : ########

Age 32 : ############

Age 33 : ####

Age 34 : ###

Age 35 : ##

Age 36 : ##

Age 37 : ##

Age 38 : ##

Age 41 : ###

Age 42 : #

Age 46 : ##

Age 48 : #

Age 56 : ##

Age 58 : #

Age 89 : ##

Age 90 : ##

Age 95 : #

Age 100 : #

Age 101 : #

Age 111 : #

Age 112 : #

Age 113 : ##

Age 114 : ###

Age 115 : ##########