Факультет «Информатика и системы управления» Кафедра «Системы обработки информации и управления»



Лабораторные работы по курсу:

## «Разработка Интернет Приложений»

## ЛР3. Python-классы

Исполнитель:

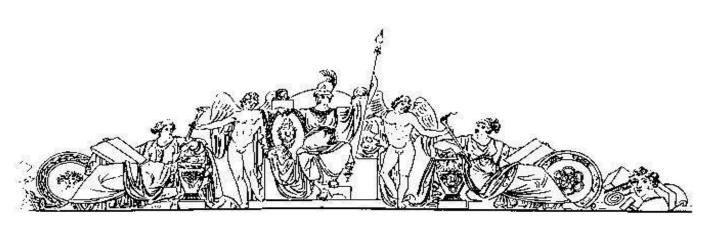
Студент группы РТ5-51

Рымарь А. А.

Преподаватель:

Гапанюк Ю. Е.

«\_\_\_»\_\_\_



#### 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:
```

)

```
base_client.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
   # Получение GET параметров
   запроса def get params(self):
       return None
    # Получение данных POST
   запроса def get json(self):
       return None
    # Получение НТТР 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):
        # Выполнение запроса
        response = requests.get(self.BASE URL + method + "." +
                                http method, params=self.get params())
        if response.status code != 200:
            print('Error')
            input()
        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
```

### ClientGetFriendsAges.py:

```
from datetime import datetime
from BaseClient import BaseClient
class ClientGetFriendsAges(BaseClient):
    # метод vk api
   method = "friends"
    # id пользователя
   user_id = None
    # GET, POST, ...
   http method = "get"
    json data = None
    def init (self, user id):
        self.user id = user id
    def get params(self):
        return {
            "user id": self.user id,
            "fields": "bdate"
    def calculate age(self, born, today):
        if today is None:
            today = datetime.utcnow()
        return today.year - born.year - ((today.month, today.day) <</pre>
(born.month, born.day))
    def response handler(self, response):
        self.json data = response.json()
        ages = list()
        today = datetime.utcnow()
        date tmp = None
        for friend in self.json_data["response"]:
            date tmp = friend.get("bdate")
            if date tmp is None or len(date tmp) < 6:</pre>
                continue
                      ages.append(self.calculate age(datetime.strptime(date tmp,
                                          "%d.%m.\(\overline{8}Y\)"), today))
        return ages
    def get json(self):
        return self.json data
       ClientGetID.py:
from BaseClient import BaseClient
class ClientGetID(BaseClient):
    # метод vk api
   method = "users"
    # GET, POST, ...
   http_method = "get"
    json_data = None
    #Создание экземпляра
    def init (self, username):
        self.username = username
    # Получение GET параметров
    запроса def get params(self):
        return {
```

```
"user ids": self.username
        }
    # Обработка ответа от VK API
    def response handler(self, response):
        self.json_data = response.json()
        ids = list()
        try:
            ids.append(self.json data["response"][0]["uid"])
        except Exception:
            print('Id не найден')
            return None
        return ids
    # Получение данных POST
    запроса def get json(self):
        return self.json data
                 gist.py:
import matplotlib.pyplot as plt
class Gist:
    # данные гистаграммы
    data = None
    _data_sorting = dict()
    def init__(self, date):
        \frac{--}{\text{self.}} \frac{--}{\text{data}} = date
        for number in date:
            self. data sorting.update({number: self. data sorting.get(number, 0) + 1})
    def get data(self):
        return self._data
    def print hist(self):
        str out = ""
        for age, stat in self._data_sorting.items():
            str_out += str(age).ljust(4) + str().ljust(stat, '#') +
        '\n' print(str_out)
        plt.hist(self. data)
        plt.xlabel('BospacT')
        plt.ylabel('Кол-во людей')
        plt.title('Гистограмма распределения возрастов друзей')
        plt.show()
       main.py:
from ClientGetID import ClientGetID
from ClientGetFriendsAges import
ClientGetFriendsAges from Gist import Gist
friends id arr = None
while friends id arr is None:
    input username = input()
    friends_id_arr = ClientGetID(input_username).execute()
friends ages = ClientGetFriendsAges(friends id arr).execute()
if len(friends_ages) == 0:
   print('Heт друзей :(')
    input()
```

```
else:
    print("ID: ", friends_id_arr)
    print("Ages: ", friends_ages)

Gist(friends_ages).print_hist()
    main()
```

# Вывод:

16# 17 # 18 #### 19 ######## 20 ############### 21 ####### 22 ####### 23 # 24 ## 25 # 26 ### 27 ### 28 ## 29 ## 30 # 33 # 38# 47 # 97# 103 # 105 #

