

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

«Работа с IPython и Jupyter Notebook»

Отчет по лабораторной работе № 3.1

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2023г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель работы: исследовать базовые возможности интерактивных оболочек IPython и Jupyter Notebook для языка программирования Python.

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * **Repository name ***

Yana-Kh / TRO-Lab-1 ✓

Great repository names are short and memorable. Need inspiration? How about **super-duper-telegram?**

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

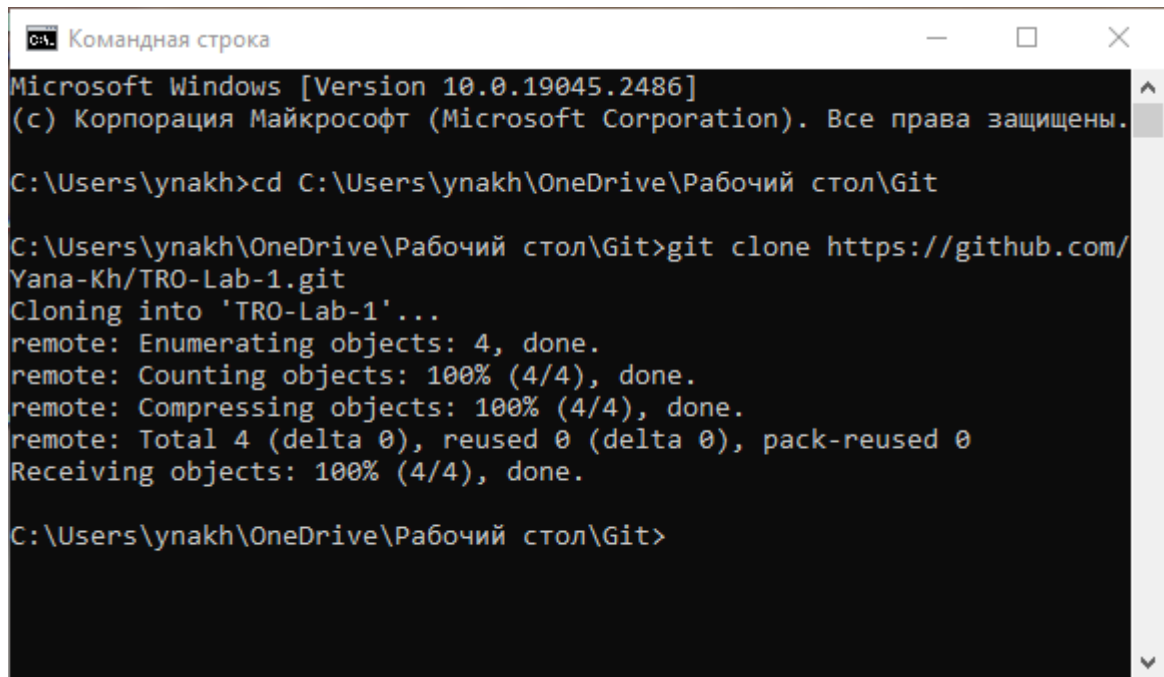
License: MIT License ▼

You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.



```
Командная строка
Microsoft Windows [Version 10.0.19045.2486]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

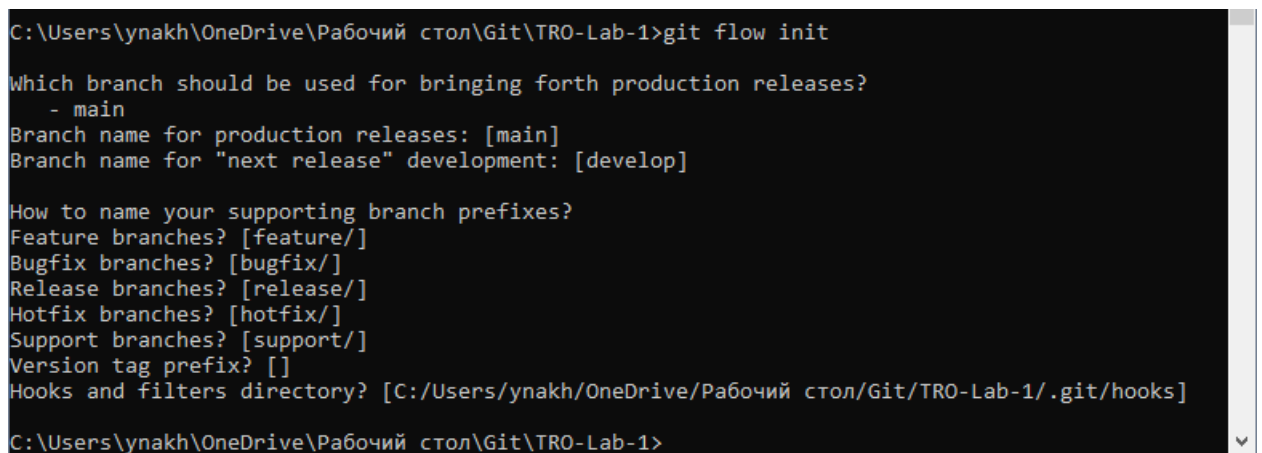
C:\Users\ynakh>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git

C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/
Yana-Kh/TRO-Lab-1.git
Cloning into 'TRO-Lab-1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\Users\ynakh\OneDrive\Рабочий стол\Git>
```

Рисунок 2 – Клонирование репозитория

4. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-1>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/ynakh/OneDrive/Рабочий стол/Git/TRO-Lab-1/.git/hooks]

C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-1>
```

Рисунок 3 – Организация репозитория в соответствии с моделью git-flow

5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-1>git add .  
  
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-1>git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Changes to be committed:  
  (use "git restore --staged <file>..." to unstage)  
        modified:   .gitignore  
  
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-1>
```

Рисунок 4 – Дополнение файла .gitignore

6. Проработать примеры лабораторной работы.

```
In [6]: 2 + 3
```

```
Out[6]: 5
```

```
In [4]: a = 5  
        b = 7  
        print(a+b)
```

```
12
```

Рисунок 5 – Пример работы с арифметическими операциями и переменными

```
In [5]: n = 7  
        for i in range(n):  
            print(i*10)
```

```
0  
10  
20  
30  
40  
50  
60
```

```
In [7]: i = 0  
        while True:  
            i +=1  
            if i > 5:  
                break  
            print("Test while")
```

```
Test while  
Test while  
Test while  
Test while  
Test while
```

Рисунок 6 – Пример работы с циклами

```
In [2]: from matplotlib import pylab as plt
        %matplotlib inline
```

```
In [3]: x = [i for i in range(50)]
        y = [i**2 for i in range(50)]
        plt.plot(x,y)
```

```
Out[3]: [<matplotlib.lines.Line2D at 0x2914c11c460>]
```

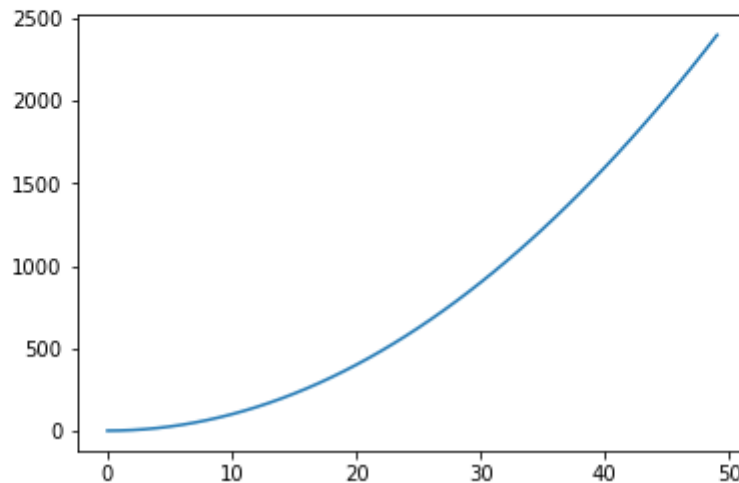


Рисунок 7 – Пример вывода графика

```
In [6]: %env TEST = 5
```

```
env: TEST=5
```

Рисунок 8 – Пример работы с переменными окружения

```
In [9]: %%time
import time
for i in range(50):
    time.sleep(0.1)
```

```
CPU times: total: 0 ns
Wall time: 5.41 s
```

```
In [11]: %timeit x = [(i**10) for i in range(10)]
```

```
1.98 µs ± 3.05 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
```

Рисунок 9 – Пример работы с временем

7. Решить задания в ноутбуках, выполненных преподавателем.

Задание №1:

- 1) Определите число `ticket_number` — шестизначный номер билета;
- 2) Напишите код, который по шестизначному номеру `ticket_number` билетика проверяет, является ли он счастливым;
- 3) Если номер счастливый, выведите строку `Yes`, иначе — `No`.

```
In [1]: ticket_number = int(input("Enter a number:"))
first3 = ticket_number // 1000
last3 = ticket_number % 1000
if (first3 // 100 + first3 // 10 % 10 + first3 % 10 == last3 // 100 + last3 // 10 % 10 + last3 % 10):
    print("Yes")
else:
    print("No")

Enter a number:567345
No
```

Рисунок 10 – Решение примера

Задание №2:

Пусть пароль может содержать только латинские буквы, знаки препинания и цифры.

Пароль считается надёжным, если удовлетворяет следующим условиям:

- содержит буквы в разных регистрах;
- содержит цифры;
- содержит не менее 4 уникальных символов;
- не содержит ваше имя латиницей, записанное буквами любых регистров (anna, iVan, ...).

Иначе пароль считается слабым.

- 1) Определите строку `password` — придуманный вами пароль;
- 2) Напишите код, который по паролю `password` проверяет, является ли он надёжным;
- 3) Если пароль надёжный, выведите строку `strong`, иначе — `weak`.

```
In [82]: password = input("Enter a password: ")
name = input("Enter a name: ")
if (password == password.upper() or password == password.lower() or password.isalpha()
    or len(set(password)) < 4 or name.lower() in password.lower()):
    print("weak")
else:
    print("strong")

Enter a password: an12dRei
Enter a name: andrey
strong
```

Рисунок 11 – Решение примера

Задание №3:

- 1) Определите число amount — количество чисел Фибоначчи, которые надо вывести;
- 2) Напишите код, который выводит первые amount чисел Фибоначчи.

```
In [6]: amount = int(input("Enter a numbe: "))
a, b = 0, 1
print(b)
for i in range(amount):
    sum = a + b
    a = b
    b = sum
    print(b)

Enter a numbe: 6
1
1
2
3
5
8
13
```

Рисунок 12 – Решение примера

Задание №4:

На сайте <https://www.kaggle.com/> выберите любой набор данных в формате CSV и проведите для него маленькое исследование: загрузите данные из набора с использованием стандартного модуля csv, посмотрите средние значения и стандартные отклонения двух выбранных числовых атрибутов, найдите [методом наименьших квадратов](#) уравнение линейной

зависимости, связывающей один числовой атрибут с другим. Для оценки заданной зависимости найдите [коэффициент парной корреляции](#), сделайте соответствующие выводы.

Импорт библиотек:

Чтение файла с данными и вывод столбцов:

```
import csv
from math import sqrt

with open('avocado.csv', 'r', newline='') as csvf:
    data = csv.reader(csvf, delimiter=',')
    total_vol = []
    total_bags = []
    for row in data:
        if row[3] == "Total_Volume":
            continue
        total_vol.append(float(row[3]))
        total_bags.append(float(row[7]))
```

Рисунок 13 – Решение примера

Подсчитаем среднее значение в 2-х столбцах

```
vol = sum(total_vol) / len(total_vol)
bag = sum(total_bags) / len(total_bags)
print(f"Среднее значение Total_Volume {vol}")
print(f"Среднее значение Total_Bags : {bag}")
```

```
Среднее значение Total_Volume 850644.0130089332
Среднее значение Total_Bags : 239639.20205983953
```

Рисунок 14 – Решение примера


```

: v1 = sum((el-vol)**2 for el in total_vol) / len(total_vol)
  st_v = sqrt(v1)
  v2 = sum((elem-bag)**2 for elem in total_bags) / len(total_bags)
  st_b = sqrt(v2)
  print(sum((el-vol)**2 for el in total_vol))
  print(len(total_vol))
  print(f"Стандартное отклонение Total_Volume: {st_v}")
  print(f"Стандартное отклонение Total_Bags: {st_b}")

```

2.176434493218307e+17

18249

Стандартное отклонение Total_Volume: 3453450.731237387

Стандартное отклонение Total_Bags: 986215.3770258684

Рисунок 15 – Решение примера

Составим уравнение линейной зависимости

```

: sum_ab = 0
  sum_square = 0

  for i, el in enumerate(total_vol):
      sum_ab += el * total_bags[i]
      sum_square += el**2

  size = len(total_vol)
  k_lin = (size * sum_ab - sum(total_vol) * sum(total_bags)) / (size * sum_square - sum(total_vol)**2)
  b_lin = vol - bag * k_lin
  func_val = []

  for el in total_vol:
      func_val.append(k_lin * el + b_lin)

  print(f"Уравнение линейной зависимости: y = {k_lin}x + {b_lin}")

```

Уравнение линейной зависимости: y = 0.2750211065408285x + 784738.1744878748

Рисунок 16 – Решение примера

Найдем коэффициент парной корреляции:

```

: cor_chisl = 0
  for i, el in enumerate(total_vol):
      cor_chisl += (el - vol) * (total_bags[i] - bag)
  sqr_diff_vol = sum((el-vol)**2 for el in total_vol)
  sqr_diff_bag = sum((el-bag)**2 for el in total_bags)
  r_xy = cor_chisl / sqrt(sqr_diff_vol * sqr_diff_bag)
  print(f"Коэффициент парной корреляции: {r_xy}")

```

Коэффициент парной корреляции: 0.9630470824267295

Коэффициент парной корреляции ~ 0.96, что может говорить о неявной зависимости

Рисунок 17 – Решение примера

8. Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.), условие которой предварительно необходимо согласовать с преподавателем.

Вычислительная задача: раскраска графов.

Логика: Первый блок - инициализация и заполнение списка вершин и связей

```
In [2]: amount_v = int(input("Введите количества вершин: "))
A = []
i = 0
while i < amount_v:
    m_smej = list(map(int, input(f"Введите смежные для {i + 1} вершины:").split(' ')))
    A.append(m_smej)
    i += 1
```

```
Введите количества вершин: 9
Введите смежные для 1 вершины:2 5
Введите смежные для 2 вершины:1 5 4 3
Введите смежные для 3 вершины:2 4 6 8
Введите смежные для 4 вершины:2 3 5 6 7
Введите смежные для 5 вершины:1 2 4 7
Введите смежные для 6 вершины:3 4 7 8 9
Введите смежные для 7 вершины:4 5 6 9
Введите смежные для 8 вершины:3 6 9
Введите смежные для 9 вершины:8 6 7
```

Возьмем за основу тот факт, что любой граф можно раскрасить с использованием 4 цветов. Создадим список, который будет хранить в себе списки с вершинами, принадлежащими определенному цвету. Также создадим список использованных вершин, счетчик цветов и флаг, для проверки связей.

```
In [5]: first = []
second = []
third = []
fourth = []
used = []
color = [first, second, third, fourth]
ch_color = 0
flag = 0
```

Рисунок 18 – Решение задачи

Второй блок - вычислительный. Первый цикл меняет "цвет", с которым мы работаем, второй - проходится по вершинам, а третий - по связям вершин

Вначале мы проверяем вершины еще нет в "цвете" или не была ли она уже использована. Если нет, то дальше рассматриваем связи.

Если вершина не имеет связь с вершиной, которая уже в "цвете" - устанавливаем флаг в единицу, в противном случае в ноль и завершаем проверку смежности

После проверки смежности, в зависимости от значения флага, добавляем вершину в "цвет" и список использованных.

```
In [18]: for ch_color in range(len(color)):
    for i, v in enumerate(A):
        flag = 0
        if i + 1 not in used and i + 1 not in color[ch_color]:
            for j in A[i]:
                if j not in color[ch_color]:
                    flag = 1
            else:
                flag = 0
                break
            if flag == 1:
                color[ch_color].append(i + 1)
                used.append(i + 1)
    if ch_color < 4:
        ch_color += 1
```

Рисунок 19 – Решение задачи

```
In [19]: print(f"Вершины входящие в 1-й цвет: {color[0]}")
print(f"Вершины входящие во 2-й цвет: {color[1]}")
print(f"Вершины входящие в 3-й цвет: {color[2]}")
print(f"Вершины входящие в 4-й цвет: {color[3]}")
```

```
Вершины входящие в 1-й цвет: [1, 3, 7]
Вершины входящие во 2-й цвет: [2, 6]
Вершины входящие в 3-й цвет: [4, 8]
Вершины входящие в 4-й цвет: [5, 9]
```

Рисунок 20 – Решение задачи

9. Зафиксируйте сделанные изменения в репозитории.




 Yana_id.ipynb	id
 avocado.csv	id
 lab3.1_yana.ipynb	id

Рисунок 5 – Фиксирование изменений в репозитории

Вопросы для защиты работы

1. Как осуществляется запуск Jupyter notebook?

Jupyter Notebook входит в состав Anaconda. Для запуска Jupyter Notebook перейдите в папку Scripts (она находится внутри каталога, в котором установлена Anaconda) и в командной строке наберите: «ipython notebook»

В результате будет запущена оболочка в браузере

2. Какие существуют типы ячеек в Jupyter notebook?

Ячейки в блокноте Jupyter бывают четырех типов – Code, Markdown и Raw и Headings.

Содержимое в ячейке Code обрабатывается как инструкции на языке программирования, по умолчанию используется Python.

Ячейки Markdown содержат текст, отформатированный с использованием языка markdown. Доступны все виды функций форматирования, такие как выделение текста жирным шрифтом и курсивом, отображение упорядоченного или неупорядоченного списка, отображение табличного содержимого и т.д.

Содержимое Raw ячейки не оценивается ядром notebook.

Headings-ячейка может использоваться для разбивки блокнота на разделы.

3. Как осуществляется работа с ячейками в Jupyter notebook?

Если ваша программа зависла, то можно прервать ее выполнение выбрав на панели меню пункт Kernel -> Interrupt.

Для добавления новой ячейки используйте Insert->Insert Cell Above и Insert->Insert Cell Below.

Для запуска ячейки используете команды из меню Cell, либо следующие сочетания клавиш:

Ctrl+Enter – выполнить содержимое ячейки.

Shift+Enter – выполнить содержимое ячейки и перейти на ячейку ниже.

Alt+Enter – выполнить содержимое ячейки и вставить новую ячейку ниже.

4. Что такое "магические" команды Jupyter notebook? Какие "магические" команды Вы знаете?

Под магией в IPython понимаются дополнительные команды, выполняемые в рамках оболочки, которые облегчают процесс разработки и расширяют ваши возможности. Список доступных магических команд можно получить с помощью команды `%lsmagic`.

Для работы с переменными окружения используется команда `%env`.

Запуск Python кода из `“.py”` файлов, а также из других ноутбуков – файлов с расширением `“.ipynb”`, осуществляется с помощью команды `%run`.

Для измерения времени работы кода используйте `%%time` и `%timeit`.

5. Самостоятельно изучите работу с Jupyter notebook и IDE PyCharm и Visual Studio Code. Приведите основные этапы работы с Jupyter notebook в IDE PyCharm и Visual Studio Code.

Jupyter (ранее IPython Notebook) - это проект с открытым исходным кодом, который позволяет легко комбинировать текст Markdown и исполняемый исходный код Python на одном холсте, называемом notebook. Visual Studio Code поддерживает работу с записными книжками Jupyter как изначально, так и через файлы кода Python. В этом разделе рассматривается встроенная поддержка, доступная для записных книжек Jupyter, и демонстрируется, как:

- Создавайте, открывайте и сохраняйте записные книжки Jupyter
- Работа с ячейками кода Jupyter
- Просмотр, проверка и фильтрация переменных с помощью обозревателя переменных и средства просмотра данных
- Подключение к удаленному серверу Jupyter
- Отладка записной книжки Jupyter

Настройка среды

Для работы с Python в записных книжках Jupyter необходимо активировать среду Anaconda в VS Code или другую среду Python, в которой установлен пакет Jupyter. Чтобы выбрать среду, используйте команду Python: Select Interpreter из палитры команд (Ctrl+Shift+P).

После активации соответствующей среды вы можете создать и открыть записную книжку Jupyter, подключиться к удаленному серверу Jupyter для запуска ячеек кода и экспортировать записную книжку Jupyter в виде файла Python.

Обширная поддержка интеграции Jupyter Notebook в PyCharm позволяет разработчикам создавать, выполнять и отлаживать исходные коды, одновременно изучая их выходные данные.

PyCharm позволяет вносить изменения в исходный документ разными способами. Это включает:

- Редактирование и предварительный просмотр.
- Использование записной книжки как исходного кода с определениями в виде текстов.
- Предоставление предварительных просмотров в реальном времени вместе с отладкой.
- Параметры автосохранения вашего кода.
- Выделение всех типов синтаксических ошибок и ошибок.
- Возможность добавлять комментарии к строкам.
- Возможность одновременного выполнения и предварительного просмотра результатов.
- Разрешения на использование специального отладчика Jupyter Notebook Debugger.
- Распознавайте файлы.ipynb по значку.

Для работы с Python в записных книжках Jupyter необходимо активировать среду Anaconda в VS Code или другую среду Python, в которой

установлен пакет Jupyter. Для выбора среды используйте команду Python: Select Interpreter из командной палитры (Ctrl+Shift+P).

После активации соответствующей среды можно создать и открыть записную книжку Jupyter, подключиться к удаленному серверу Jupyter для запуска ячеек кода и экспортировать записную книжку Jupyter в виде файла Python.