

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с IPython и Jupyter Notebook»

ОТЧЕТ
по лабораторной работе №3.1
дисциплины
«Технологии распознавания образов»

Выполнил:

Зиёдуллаев Жавохир Эркин угли
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

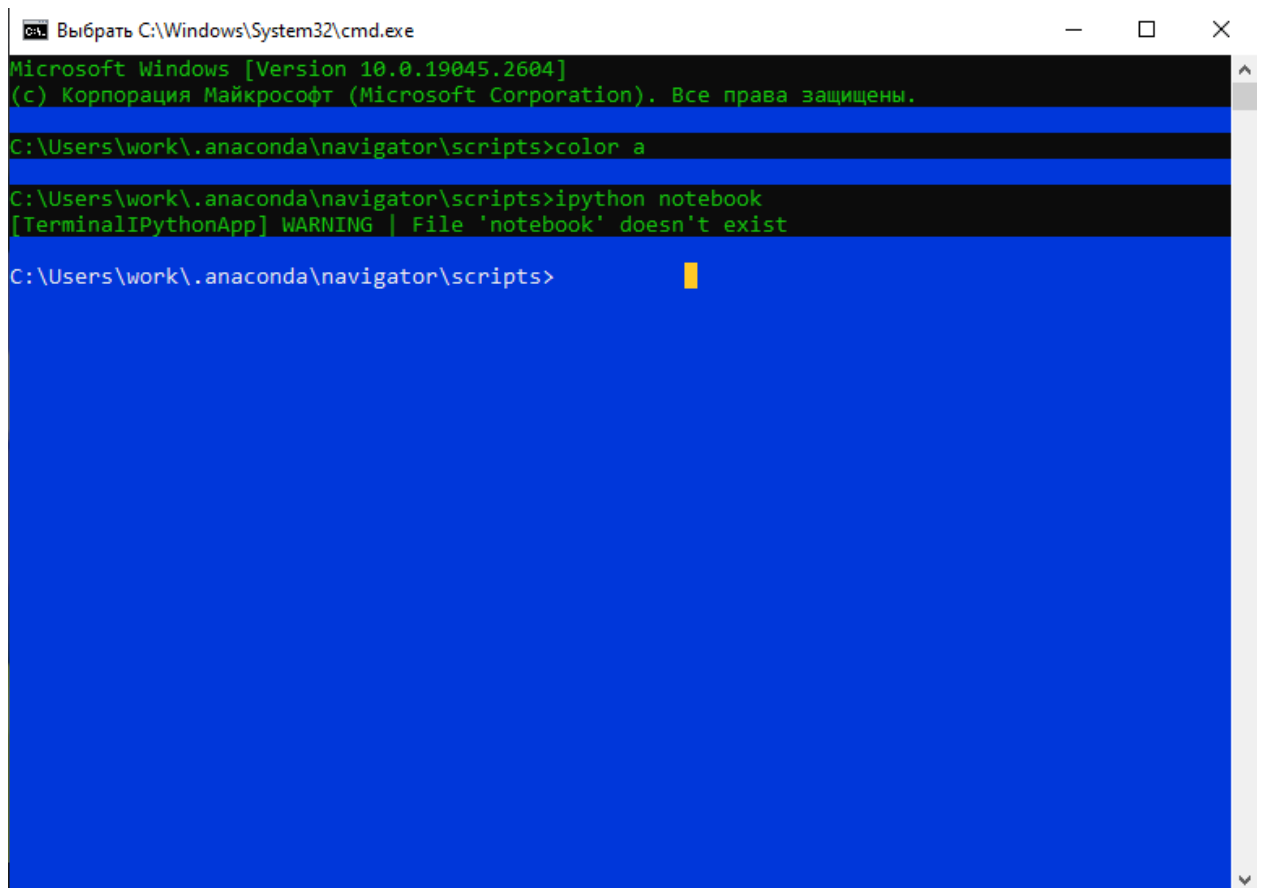
Цель работы:

Исследовать базовые возможности интерактивных оболочек IPython и Jupyter Notebook для языка программирования Python.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: https://github.com/javoxir21/tro_1laba.git



```
cmd. Выбрать C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2604]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\work\.anaconda\navigator\scripts>color a

C:\Users\work\.anaconda\navigator\scripts>ipython notebook
[TerminalIPythonApp] WARNING | File 'notebook' doesn't exist

C:\Users\work\.anaconda\navigator\scripts>
```

1-Запустить Jupyter notebook

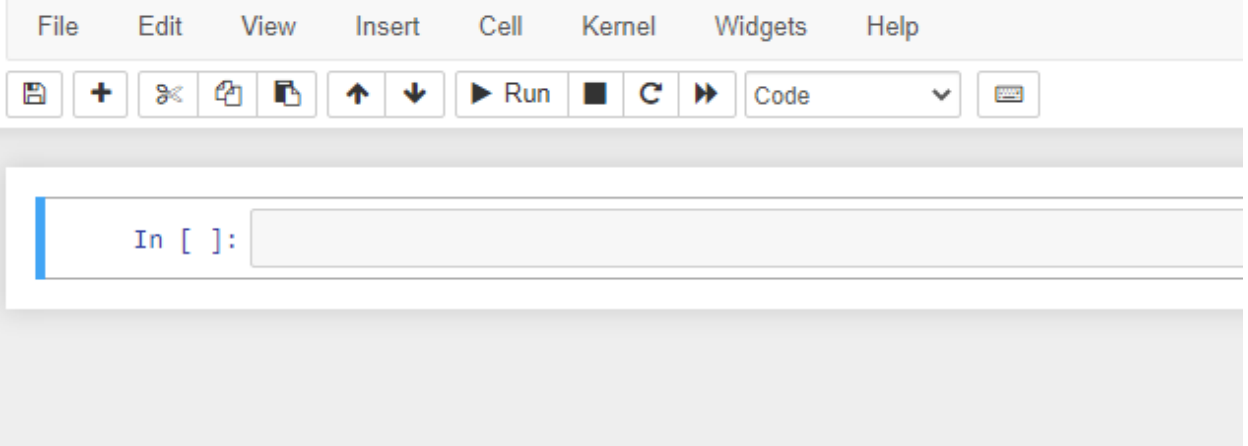


Рисунок -2 Создал notebook



Рисунок -3 Результаты вычисления

jupyter lab-1

Last Checkpoint: 31 minutes ago (autosaved)

Log

File Edit View Insert Cell Kernel Widgets Help

Trusted | Kern

+

⌂

🔍

📄

⬆️

⬆️

▶️ Run

⏏️

⏮️

⏭️

Code

⌵

📄

In [6]:

```

n = 7
for i in range(7):
    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

```

In [ ]:

In [8]: from matplotlib import pylab as plt
%matplotlib inline

In [9]: x = [i for i in range(50)]
        y = [i**2 for i in range(50)]
        plt.plot(x, y)

Out[9]: [<matplotlib.lines.Line2D at 0x25d6a947af0>]

```

x	y
0	0
10	100
20	400
30	900
40	1600
50	2500

Рисунок -4 %matplotlib inline

```

In [10]: %lsmagic

Out[10]: Available line magics:
%alias %alias_magic %autosave %autocall %automagic %autosave %bookmark %cd %clear %cls %colors %conda %config %connect_info %copy %ddir %debug %dhist %dirs %doctest_mode %echo %ed %edit %env %gui %hist %history %killbgscripts %ldir %less %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %macro %magic %matplotlib %mkdir %more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %pip %popd %pprint %precision %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %ren %rep %rerun %reset %reset_selective %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode

Available cell magics:
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javascript %%js %%latex %%markdown %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.

```

```
In [ ]: |
```

Рисунок -5 %lsmagic

```
Automatic is ON, a prefix is not needed for line magic.

In [11]: %env TEST = 5

env: TEST=5

In [ ]: |
```

Рисунок -6 Команда %env

```
In [15]: %%time
import time
for i in range(50):
    time.sleep(0.1)

Wall time: 5.11 s

In [ ]: |

In [ ]: |
```

Рисунок -7 Команда %%time

```
In [16]: %timeit x = [(i**10) for i in range(10)]

4.26 µs ± 111 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

In [ ]: |
```

Рисунок -8 Команда %timeit

Решить задания в ноутбуках, выполненных преподавателем.

Задание 1

Билет считается счастливым, если выполнено следующее условие:
сумма первых трёх цифр номера равна сумме последних трёх цифр.

Задание:

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

```
In [4]: def isLucky(ticket):
        id = str(ticket)
        sum1, sum2 = 0, 0
        if len(id) == 6:
            for index, elem in enumerate(id):
                if index < 3:
                    sum1 += int(elem)
                else:
                    sum2 += int(elem)
            if sum1 == sum2:
                print("Yes")
            else:
                print("No")
        else:
            print("Номер билета должен состоять из 6-ти цифр!")
        ticket_number = 111111
        isLucky(ticket_number)

        ticket_number = 534854
        isLucky(ticket_number)
```

Yes
No

In []:

Рисунок 9 – Решение задания 1

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.

Задание:

- 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 [2]: def isStrong(pwd, nme):
        if len(pwd) > 0:
            diff_reg = False
            contain_numbers = False
            unique_symb = False
            not_a_name = False
            upp, dwn = 0, 0
            for i in pwd:
                if i.isupper() == True:
                    upp += 1
                else:
                    dwn += 1
            if upp > 0 and dwn > 0:
                diff_reg = True
            numbers = ['1', '2', '3', '4', '5', '6', '7', '8', '9']
            for i in pwd:
                if i in numbers:
                    contain_numbers = True
            pwd_lower = pwd.lower()
            if(len(set(pwd_lower))) >= 4:
                unique_symb = True
            if not(nme.lower() in pwd_lower()):
                not_a_name = True
            if diff_reg == True and contain_numbers == True and unique_symb == True and not_a_name == True:
                print("strong")
            else:
                print("weak")
            print("Символы разных регистров: ", diff_reg)
            print("содержит цифры: ", contain_numbers)
            print("Не менее 4-х уникальных символов: ", unique_symb)
            print("Не содержит имени: ", not_a_name)
        else:
            print("Пароль не может быть пустым!")

password = "qwerty1234"
name = "Javokhir"
isStrong(password, name)

password = "qwerty1234Javokhir"
name = "javokhir"
isStrong(password, name)
```

Рисунок 10 – Решение задания 2

Задание 3

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

```
In [5]: def fib(n):
        if n in (1, 2):
            return 1
        return (fib(n - 1) + fib(n - 2))

        amount = 10
        for i in range(1, amount + 1):
            print(fib(i), end=' ')
```

1 1 2 3 5 8 13 21 34 55

Рисунок 11 – Решение задания 3

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.

Набор Данных

Выбранный набор данных содержит статистику по игровым стримам на Twitch.

Для анализа я выбрал два показателя: Пиковое число каналов, стримящих игру и пиковое число зрителей стримов по ней.

Импорт данных:

|

```
import csv
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from math import sqrt

data = []
x = []
y = []

df = pd.read_csv('Twitch_game_data.csv', encoding="cp1252")

with open("Twitch_game_data.csv", encoding='cp1252') as f:
    reader = csv.DictReader(f, dialect='excel')
    for row in reader:
        data.append(row)

for i in data:
    x.append(int(i.get('Peak_viewers')))
    y.append(int(i.get('Peak_channels')))
```

Рисунок 12 – Импорт набора данных

Средние значения показателей:

```
In [5]: def average(lst):
        return sum(lst) / len(lst)

print("Средние значения: ")
print("X: ", average(x))
print("Y: ", average(y))

print("Проверка: ")
print("X: ", np.mean(x))
print("Y: ", np.mean(y))
```

```
Средние значения:
X:  55095.117847222224
Y:  586.7592361111111
Проверка:
X:  55095.117847222224
Y:  586.7592361111111
```

Рисунок 13 – Вычисление средних значений двух показателей

Стандартные отклонения показателей:

```
In [6]: def std_deviation(lst):
        for i in lst:
            return (sum((x-(sum(lst) / len(lst))) ** 2 for x in lst) / len(lst)) ** 0.5

        print("Стандартные отклонения: ")
        print(std_deviation(x))
        print(std_deviation(y))

        print("Проверка: ")
        print(np.std(x))
        print(np.std(y))

Стандартные отклонения:
132908.80199979182
2721.316593865302
Проверка:
132908.80199979153
2721.3165938653
```

Рисунок 14 – Вычисление стандартных отклонений показателей

Уравнение линейной зависимости:

```
In [7]: def linear_reg_eq(lst_x, lst_y):
        y_predicted = []
        sum_sq_xi, x_mult_y = 0, 0
        sum_xi, sum_yi = sum(lst_x), sum(lst_y)
        n = len(x)

        sum_sq_xi = sum([lst_x[i] ** 2 for i in range(n)])
        x_mult_y = sum([lst_x[i] * lst_y[i] for i in range(n)])

        a = ((sum_yi * sum_sq_xi) - (sum_xi * x_mult_y)) / ((n * sum_sq_xi) - (sum_xi * sum_xi))
        b = ((n * x_mult_y) - (sum_xi * sum_yi)) / ((n * sum_sq_xi) - (sum_xi * sum_xi))

        for elem in lst_x:
            y_predicted.append(((a + b) * elem))

        print(a)
        print(b)
        print(f"y = {a} + {b} * x")
        return y_predicted

y_pred = linear_reg_eq(x, y)

5.530787385449832e-10
0.010893441340323405
y = 5.530787385449832e-10 + 0.010893441340323405 * x
```

Рисунок 15 – Составление уравнения линейной зависимости показателей

Коэффициент парной корреляции:

```
In [8]: def pair_corell(lst_x, lst_y):
        cov = 0
        avrg_x, avrg_y = average(lst_x), average(lst_y)

        deviat_x = sum((num - avrg_x)**2 for num in lst_x)
        deviat_y = sum((num - avrg_y)**2 for num in lst_y)

        cov = sum([(num - avrg_x)*(lst_y[index] - avrg_y) for index, num in enumerate(lst_x)])

        cor_coef = cov / sqrt(deviat_x * deviat_y)
        print(f"Коэффициент парной корреляции: {cor_coef}")

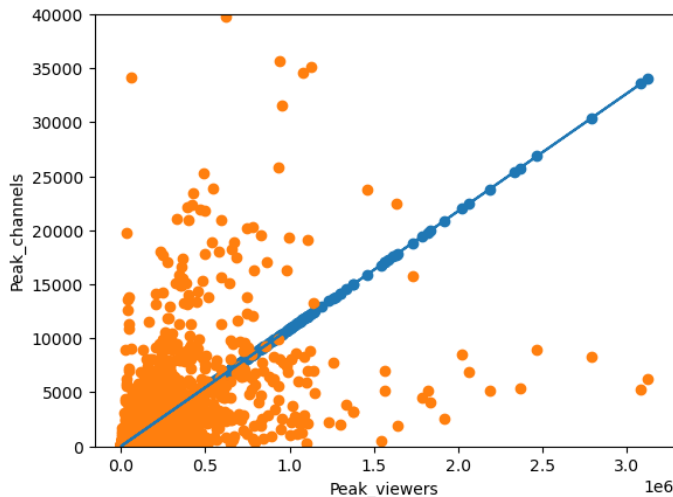
pair_corell(x,y)
print("Проверка: ", df['Peak_viewers'].corr(df['Peak_channels']))

Коэффициент парной корреляции: 0.5320344723804856
Проверка: 0.5320344723804865
```

Рисунок 16 – Определение коэффициента парной корреляции

Визуализация:

```
In [9]: plt.plot(x, y_pred)
plt.scatter(x, y_pred)
plt.scatter(x, y)
plt.ylim(0, 40000)
plt.xlabel('Peak_viewers')
plt.ylabel('Peak_channels')
plt.show()
```



Вывод:

Парный коэффициент корреляции равен 0.53, значит зависимость между показателями - средняя. Показатель пикового числа каналов, стримящих игру мало зависит от пикового числа зрителей со всех стримов по данной игре, так как их, очевидно, намного больше в количестве.

Рисунок 17 – Визуализация результатов и вывод

Вывод: В результате выполнения работы были исследованы базовые возможности оболочек IPython и Jupyter Notebook для языка программирования Python, а также были написаны программы в них.

Контрольные вопросы:

1. Как осуществляется запуск Jupyter Notebook?

В командной строке Anaconda набрать команду: `jupyter notebook`.

2. Какие существуют типы ячеек в Jupyter Notebook?

Существует два вида ячеек: 1) Ячейка кода содержит код, который должен быть выполнен в ядре, и отображает его вывод ниже; 2) Ячейка

Markdown содержит текст, отформатированный с использованием Markdown, и отображает его вывод на месте при запуске.

3. Как осуществляется работа с ячейками в Jupyter Notebook?

После выбора ячейки «Code», можно записать код на языке Python, а затем нажать Ctrl+Enter или Shift+Enter. В первом случае введенный код будет выполнен интерпретатором Python, а во втором – будет создана новая ячейка, которая расположится уровнем ниже.

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

Важной частью функционала Jupyter Notebook является поддержка магии. Под магией понимаются дополнительные команды, выполняемые в рамках оболочки, которые облегчают процесс разработки и расширяют возможности.

Узнать команды: `%lsmagic`

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

Запуск кода с расширением `.ipynb` осуществляется с помощью команды `%run`.

Для измерения времени работы необходимо использовать команды `%%time` и `%timeit`.

`%matplotlib` используется для отображения объектов графиков на экране, ключ после него указывает каким способом отображать график.

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.

PyCharm:

1. Создать новый проект

2. В этом проекте создать новый файл `ipynb`.

3. Если не установлен пакет Jupyter Notebook, появится сообщение об ошибке: «Пакет Jupyter не установлен», и будет опция «Установить пакет jupyter».

4. «Установить пакет jupyter». Это запустит процесс установки, который вы можете просмотреть, щелкнув запущенные процессы в правом нижнем углу окна PyCharm.

5. Можно создать ячейки кода и выполнить их.

6. Чтобы запустить сервер Jupyter, нужно выполнить ячейку кода. По умолчанию сервер Jupyter использует порт 8888 на локальном хосте. Эти конфигурации доступны в окне инструментов сервера. После запуска вы можете просмотреть сервер над окном исходного кода, а рядом с ним вы можете просмотреть ядро, созданное как «Python 2» или «Python 3».

7. Теперь можно получить доступ к вкладке переменных в PyCharm, чтобы увидеть, как значения переменных меняются при выполнении ячеек кода. Можно также установить точки останова в строках кода, а затем щелкнуть значок «Выполнить» и выбрать «Debug Cell» (или использовать сочетание клавиш `Alt+Shift+Enter`), чтобы начать отладку.

Visual Studio Code:

1. Чтобы создать новый Jupyter Notebook можно запустить Command Palette (Ctrl+Shift+P) и ввести new notebook. Первым результатом должен быть Jupyter: Create New Blank Jupyter Notebook. Также, его можно создать, нажав на новый файл .ipynb.

2. Блокноты, созданные VS Code, по умолчанию являются доверенными (trusted). Ставить пометку trust нужно вручную по запросу редактора перед выполнением.

3. После создания блокнота нажать save на верхней части панели инструментов, чтобы сохранить его в рабочем пространстве. Теперь можно экспортировать созданный блокнот как скрипт Python или файл HTML/PDF, используя соответствующую иконку.

4. По умолчанию в новом блокноте появится пустая ячейка. Добавьте в нее код и выполните его с помощью Ctrl+Enter. Эта команда запустит выделенную ячейку. Shift+Enter выполняет то же действие, но при этом создает и выделяет новую ячейку ниже, а Alt+Enter выполняет выделенную, создает еще одну ниже, но при этом сохраняет метку на предыдущей.

Иконка + добавляет новую ячейку для кода, а bin удаляет ее. Чтобы перемещать фрагменты вверх и вниз, пользуемся соответствующими стрелками.

Изменить тип ячейки на markdown довольно просто: просто нажмите на иконку M, расположенную над кодом. Чтобы снова установить значение code, выберите значок {}. Выполнить эти действия также можно с помощью клавиш M и Y.

Также, расширение Jupyter для VS Code поддерживает построчное выполнение кода в ячейке, нажав на кнопку, расположенную рядом с иконкой

Play. Вторая возможность для отладки — можно просто экспортировать блокнот как скрипт Python и работать с ним прямо в отладчике VS Code, не переходя в другую среду.