

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 3.1
по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Трушева В. О. .«__»_____2023г.

Подпись студента_____

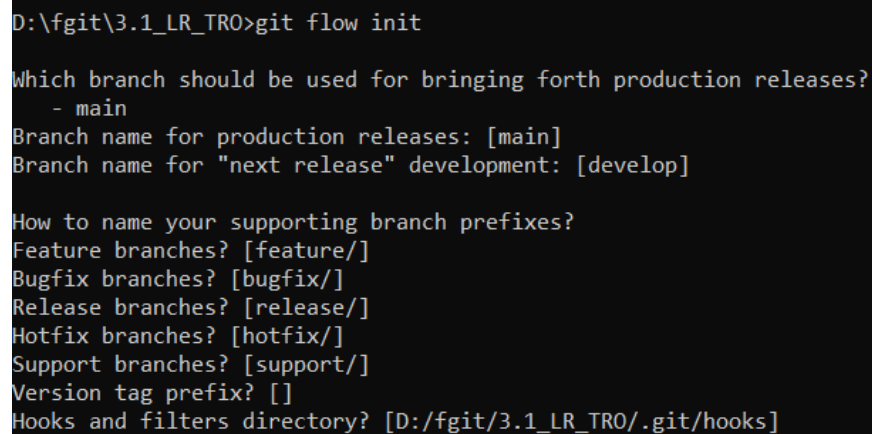
Работа защищена « __ »_____20__г.

Проверила Воронкин Р.А. _____
(подпись)

Ставрополь 2022

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

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).
3. Выполните клонирование созданного репозитория на рабочий компьютер.
4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.



```
D:\fgit\3.1_LR_TRO>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? [D:/fgit/3.1_LR_TRO/.git/hooks]
```

Рисунок 1 – Модель ветвления git-flow

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

```
In [1]: 2 + 2
```

```
Out[1]: 4
```

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

```
12
```

```
In [6]: 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 2 – 1 пример

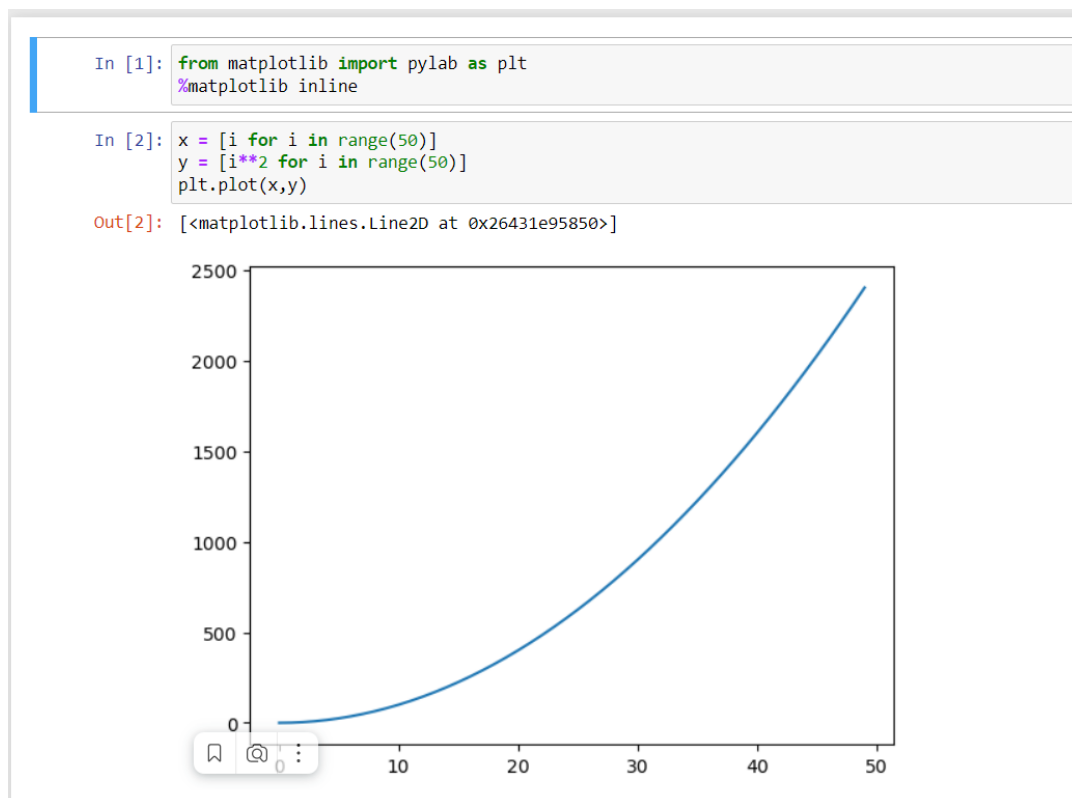


Рисунок 3 – 2 пример

```

In [2]: %lsmagic

Out[2]: Available line magics:
%alias %alias_magic %autoawait %autocall %automagic %autosave %bookmark %cd %clear %cls %colors %conda %config %co
nnect_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 %matpl
otlib %mkdir %more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %pip %popd %pprint %precisio
n %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 %unl
oad_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

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

In [3]: %env TEST = 5

env: TEST=5

In [7]: %%time
import time
for i in range(50):
    time.sleep(0.1)

Wall time: 5.52 s

In [8]: %timeit x = [(i**10) for i in range(10)]

3.74 µs ± 48.1 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

```

Рисунок 4 – 3 пример

7. Решить задания в ноутбуках, выполненных преподавателем.

Условие 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` билетика проверяет, является ли он счастливым;
- 3) Если номер счастливый, выведите строку `Yes`, иначе — `No`.

```
In [4]: ticket_number = int(input("ticket number of 6 digits: "))
        ticket number of 6 digits: 45645644

In [6]: sum1 = ticket_number // 100000 + ticket_number // 10000 % 10 + ticket_number // 1000 % 10
        sum2 = ticket_number % 1000 // 100 + ticket_number % 100 // 10 + ticket_number % 10
        if (ticket_number // 10000000) < 1:
            if sum1 == sum2:
                print("Yes")
            else:
                print("No")
        else:
            print("Incorrect number of digits")
        incorrect number of digits
```

Рисунок 5 – Код программы 1 задания

Условие 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 [15]: password = input("Enter password: ")  
name = input("Enter name: ")
```

```
Enter password: an12dRei  
Enter name: andrei
```

```
In [16]: 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")
```

```
strong
```

Рисунок 6 – Код программы 2 задания

Условие 3. Как известно, числа Фибоначчи — это последовательность чисел, каждое из которых равно сумме двух предыдущих (первые два числа равны 1): 1,1,2,3,5,8,13,...

Задание:

1) Определите число amount — количество чисел Фибоначчи, которые надо вывести;

2) Напишите код, который выводит первые amount чисел Фибоначчи.

```
In [27]: amount = int(input("Enter amount: "))
```

Enter amount: 6

```
In [28]: fib1 = 1
fib2 = 1
print(fib1)
print(fib2)
i = 0

while i < amount - 2:
    fib_sum = fib1 + fib2
    fib1 = fib2
    fib2 = fib_sum
    i = i + 1
    print(fib2)
```

1
1
2
3
5
8

Рисунок 7 – Код программы 3 задания

Условие 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.

```

In [7]: import csv
        from math import sqrt

        with open('data_food.csv', 'r', newline='') as csvfile:
            data = csv.reader(csvfile, delimiter=',')
            salary = []
            weight = []
            for row in data:
                if row[7] == "Male":
                    weight.append(float(row[3]))
                    salary.append(float(row[6]))

In [10]: sr_weight = sum(weight) / len(weight)
         sr_salary = sum(salary) / len(salary)
         print(f"Среднее значение веса мужчин: {sr_weight:.4f}")
         print(f"Среднее значение заработной платы мужчин: {sr_salary:.4f}")

Среднее значение веса мужчин: 74.4965
Среднее значение заработной платы мужчин: 45649.9397

In [11]: weight_otk = sqrt(sum((elem-sr_weight)**2 for elem in weight) / len(weight))
         salary_otk = sqrt(sum((elem-sr_salary)**2 for elem in salary) / len(salary))
         print("Стандартное отклонение коэффициента веса мужчин: ")
         print(f"{weight_otk:.4f}")
         print("Стандартное отклонение заработной платы мужчин: ")
         print(f"{salary_otk:.4f}")

Стандартное отклонение коэффициента веса мужчин: 17.5647
Стандартное отклонение заработной платы мужчин: 56383.8126

```

Рисунок 8 – Код программы 4 задания

```

In [12]: sum_elem = 0
         sum_square = 0
         for index, elem in enumerate(weight):
             sum_elem += elem * salary[index]
             sum_square += elem**2
         a = (len(weight) * sum_elem - sum(weight) * sum(salary)) / (len(weight) * sum_square - sum(weight)**2)
         b = sr_salary - sr_weight * a
         func_elem = []
         for elem in weight:
             func_elem.append(a * elem + b)
         print(f"Уравнение линейной зависимости: y = {a}x + {b}")
         print("Значения функции на кривой методом наименьших квадратов: ")
         for val in func_elem:
             print(val)

Уравнение линейной зависимости: y = 727.534138541161x + -8548.83285092099
Значения функции на кривой методом наименьших квадратов:
44561.159262583766
42378.55684696028
44197.392193313186
46016.22753966609
46016.22753966609
61794.44444903047

```

Рисунок 9 – Код программы 4 задания

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.

Алгоритм Дейкстры

Алгоритм Дейкстры находит минимальные веса, которые могут быть достигнуты при переходе стартовой вершины v во все остальные вершины.

```
In [4]: matr = ((0, 3, 1, 3, 0, 0),
               (3, 0, 4, 0, 0, 0),
               (1, 4, 0, 0, 7, 5),
               (3, 0, 0, 0, 0, 2),
               (0, 7, 0, 0, 0, 4),
               (0, 0, 5, 2, 4, 0))
```

```
In [21]: import math

def смеj_v(v, matr):
    for i, weight in enumerate(matr[v]):
        if weight > 0:
            yield i
```

функция-генератор `смеj_v` возвращает смежные вершины для данной вершины v . В цикле перебирается строка матрицы смежности. И если вес больше 0, то возвращается номер вершины. А в цикле ниже, где используется данная функция, перебирается данный генератор. Поэтому используется ключевое слово `yield`, которое используется примерно как `return` — отличие в том, что функция вернёт генератор. При повторном вызове такой функции выполнение продолжается с оператора `yield`, на котором работа была прервана.

```
In [22]: def arg_min(l, s):
        amin = -1
        m = max(l) # максимальное значение в l ()
        for i, t in enumerate(l):
            if t < m and i not in s:
                m = t
                amin = i
        return amin
```

Функция возвращает вершину с минимальным весом. Перебираются значения в строке, где было найдено максимальное значение l . И находим минимальное значение ($t < m$) для тех вершин, которые не рассмотрели (i not in s).

```
In [23]: l = len(matr) # число вершин в графе
```

Рисунок 10 – Индивидуально задание

```
In [23]: n = len(matr) # число вершин в графе
        l = [math.inf]*n # последняя строка в таблице
        v = 0 # стартовая вершина
        s = {v} # просмотренные вершины
        l[v] = 0 # нулевой вес стартовой вершины

        while v != -1: # цикл, пока не просмотрим все вершины
            for j in смеj_v(v, matr): # перебираем все связанные вершины с вершиной v
                if j not in s: # если вершина не просмотрена
                    w = l[v] + matr[v][j]
                    if w < l[j]:
                        l[j] = w
            v = arg_min(l, s) # выбираем следующий узел с наименьшим весом
            if v > 0: # выбрана очередная вершина
                s.add(v) # добавляем новую вершину в рассмотренные
        print(l)

[0, 3, 1, 3, 8, 5]
```

Рисунок 11 – Индивидуально задание

9. Зафиксируйте сделанные изменения в репозитории.

10. Выполните слияние ветки для разработки с веткой `main` (`master`).

11. Отправьте сделанные изменения на сервер GitHub.

Вопросы для защиты работы

1. Как осуществляется запуск Jupyter notebook?

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

2. Какие существуют типы ячеек в Jupyter notebook?

Code

Markdown

Raw NBConvert

Heading

3. Как осуществляется работа с ячейками в Jupyter notebook?

Перед первой строкой написано `In []`. Это ключевое слово значит, что дальше будет ввод. Попробуйте написать простое выражение вывода.

Вывод должен отобразиться прямо в notebook. Это и позволяет заниматься программированием в интерактивном формате, имея возможность отслеживать вывод каждого шага.

Также обратите внимание на то, что `In []` изменилась и вместе нее теперь `In[1]`. Число в скобках означает порядок, в котором эта ячейка будет запущена. В первой цифра 1, потому что она была первой запущенной ячейкой. Каждую ячейку можно запускать индивидуально и цифры в скобках будут менять соответственно.

Если есть несколько ячеек, то между ними можно делиться переменными и импортами. Это позволяет проще разбивать весь код на связанные блоки, не создавая переменную каждый раз. Главное убедиться

в запуске ячеек в правильном порядке, чтобы переменные не использовались до того, как были созданы.

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

Эти встроенные команды IPython упрощают решение задач по анализу данных с помощью Python, а также обеспечивают упрощенное взаимодействие Python с операционными системами, другими языками программирования или ядрами.

`%ismagic` – список доступных магических команд

`%env` – для работы с переменными окружения

`%run` – для запуска файлов с расширением «.ру»

`%%time` – позволяет получить информацию о времени работы кода в рамках одной ячейки

`%timeit` – запускает переданный ей код 1000000 (по умолчанию) и выводит информацию среднем значении трёх наиболее быстрых прогонах

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.

Грамотное программирование — это стандартная форма программирования, ориентированная на удобочитаемость кода. Это позволяет программистам придавать форму логическим единицам своего кода, значению этих единиц кода и их результатам. Скомпилированный блокнот представляет код как законченный и понятный мыслительный процесс и его технологическое воплощение.

Для поддержки грамотного программирования в Jupyter Notebook есть множество доступных инструментов, которые обеспечивают полную

свободу редактирования кода с его соответствующей поддерживающей прозой.

Начиная с базового уровня, записные книжки (файлы, в которых написан код) могут разделять код на «ячейки». Ячейки позволяют легко различать определенные функции.

Помимо ячеек кода, доступны ячейки разметки, в которых легко ввести описание кода, значение или результаты. Возможности редактирования ячеек разметки безграничны; вы можете поиграть с текстовыми форматами, изображениями и даже математическими уравнениями и диаграммами.

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.