

**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

Рязанцев.М.Д. « » 2023г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

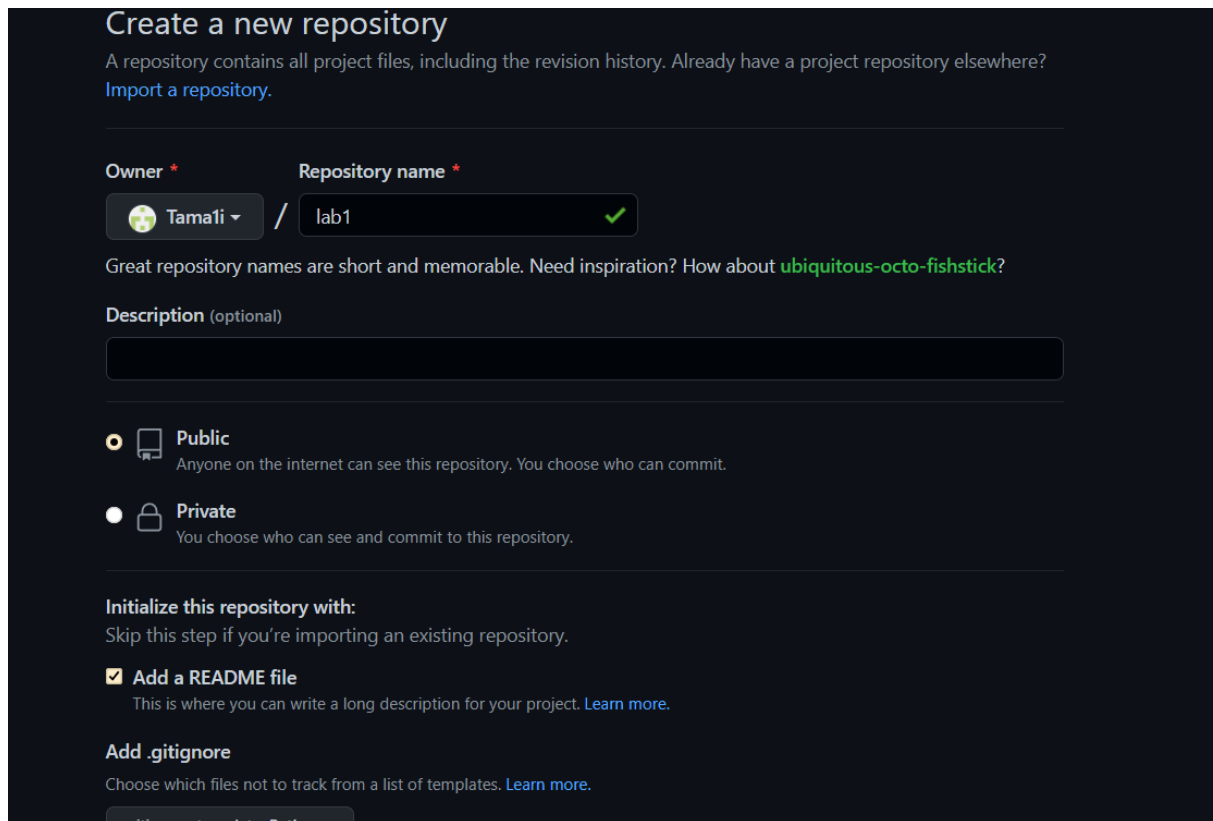
Проверил Воронкин Р.А. _____
(подпись)

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 *

Tama1i / lab1 ✓

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-octo-fishstick?](#)

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.](#)

Рисунок 1 – Создание репозитория

```
D:\2kurs\!22kurs\obraz\git\lab1>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:/2kurs/!22kurs/obraz/git/lab1/.git/hooks]
```

Рисунок 2 – Организация репозитория в соответствии с моделью git-flow

Проработать примеры лабораторной работы.

```
In [1]: 3 + 2
Out[1]: 5

In [2]: a = 5
        b = 7
        print(a + b)
12

In [3]: n = 7
        for i in range(n):
            print(i*10)
0
10
20
30
40
50
60

In [4]: i = 0
        while True:
            i += 1
            if i > 5:
                break
            print("Test while")
Test while
Test while
Test while
Test while
```

Рисунок 5 – проработка примеров

```
In [6]: x = [i for i in range(50)]
        y = [i**2 for i in range(50)]
        plt.plot(x,y)
Out[6]: [matplotlib.lines.Line2D at 0x2d47ecc6490>]
```

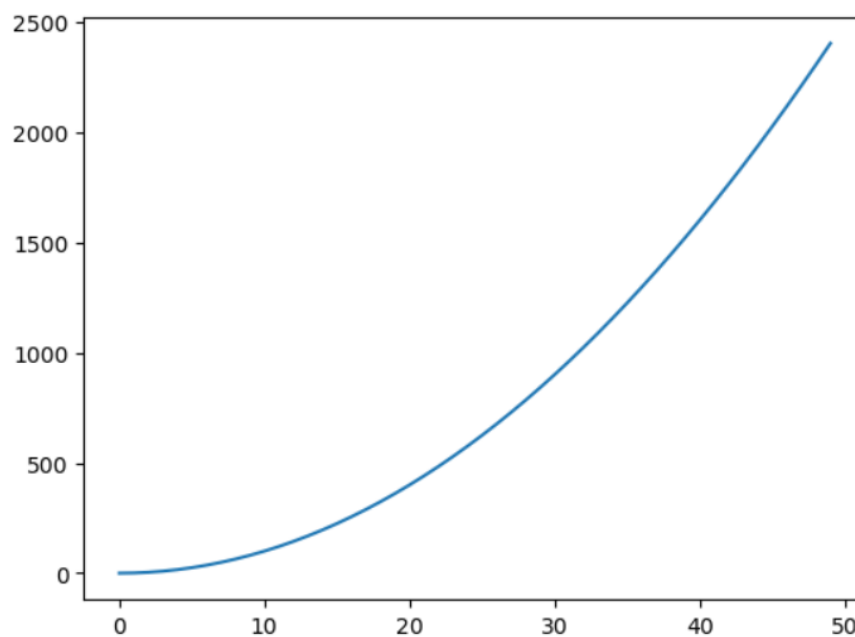


Рисунок 6 – проработка примеров

```

In [7]: lsmagic

Out[7]: Available line magics:
%alias %alias_magic %await %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 %%ppypy %%python %%python2 %%python3 %%ruby %%script %%ssh %%svg %%sx %%system %%time %%timeit %%writefile

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

In [8]: %env TEST = 5

env: TEST=5

In [9]: %run ./lab1.py

-----
OSError                                Traceback (most recent call last)
D:\Anaconda\lib\site-packages\IPython\core\magics\execution.py in run(self, parameter_s, runner, file_finder)
    713         fpath = arg_lst[0]
--> 714         filename = file_finder(fpath)
    715     except IndexError:

D:\Anaconda\lib\site-packages\IPython\utils\path.py in get_py_filename(name, force_win32)
    108     else:
--> 109         raise IOError('File `%r` not found.' % name)
    110

```

Рисунок 7 – проработка примеров

```

In [11]: %%time
import time
for i in range(50):
    time.sleep(0.1)

Wall time: 5.34 s

In [12]: %timeit x = [(i**10) for i in range(10)]

3.71 µs ± 567 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

In [ ]:

```

Рисунок 8 – проработка примеров

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.

$$R_{Ed} = \frac{2 \cdot \hbar^2}{G \cdot m_e \cdot m_p^2}$$

Рисунок 9 – eddington universe radius

```
] : R_Edd=2*constants.hbar**2/(constants.G*constants.m_e*constants.m_p**2)
print(R_Edd)
```

1.3076515370853231e+26

```
] : R_Edd/(constants.light_year*10**9)
```

```
] : 13.82188765312757
```

```
] : |
```

Рисунок 9 – код программы



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

Задание:

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

Пример 1:

Input: 123456

Output: No

Пример 2:

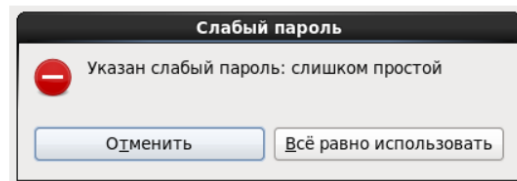
Input: 123042

Output: Yes

```
] : t = int(input("Enter a number:"))
c3 = t // 1000
last3 = t % 1000
if (c3 // 100 + c3 // 10 % 10 + c3 % 10 == last3 // 100 + last3 // 10 % 10 + last3 % 10):
    print("Yes")
else:
    print("No")
```

Enter a number:46700
No

Пароль



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`.

Пусть имя пользователя -- Андрей.

Output: strong

```
In [5]: p = input("Enter a password: ")
n = input("Enter a name: ")
if (p == p.upper() or p == p.lower() or p.isalpha()
    or len(set(p)) < 4 or n.lower() in p.lower()):
    print("weak")
else:
    print("strong")
```

```
Enter a password: jrebi34T
Enter a name: Matvei
strong
```

Числа Фибоначчи

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.

Пример 1:

Input: 3

Output: 1 1 2

Пример 2:

Input: 10

Output: 1 1 2 3 5 8 13 21 34 55

```
In [6]: v = int(input("vvod "))
a, b = 0, 1
print(b)
for i in range(v):
    s = a + b
    a = b
    b = s
    print(b)
```

```
vvod 3
1
1
2
3
```

с помощью функции print можно вывести данные в одну строку

```
In [60]: import csv
from math import sqrt

with open('summer-products-with-rating-and-performance_2020-08.csv', encoding = 'utf-8') as csvfile:
    d = csv.reader(csvfile, delimiter=',')
    t1 = []
    t2 = []
    for row in d:
        if row[0] == "index":
            continue
        t1.append(float(row[8]))
        t2.append(float(row[9]))
```

среднее значение

```
In [61]: sr1 = sum(t1) / len(t1)
sr2 = sum(t2) / len(t2)
print(f"Среднее значение 1 {sr1}")
print(f"Среднее значение 2 : {sr2}")
```

```
Среднее значение 1 3.8208963763509174
Среднее значение 2 : 889.659249841068
```

```
In [62]: v1 = sum((el-sr1)**2 for el in t1) / len(t1)
st_v = sqrt(v1)
v2 = sum((elem-sr2)**2 for elem in t2) / len(t2)
st_b = sqrt(v2)
print(sum((el-sr1)**2 for el in t1))
print(len(t1))
print(f"Стандартное отклонение 1: {st_v}")
```

```
In [63]: import numpy as np
import pandas as pd
data = pd.read_csv('summer-products-with-rating-and-performance_2020-08.csv')
data.std()
```

C:\Users\ryaza\AppData\Local\Temp\ipykernel_11788\3721584870.py:4: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
Out[63]: index          454.230301
price           3.932030
retail_price     30.357863
units_sold      9356.539302
uses_ad_boosts   0.495639
rating           0.515374
rating_count     1983.928834
rating_five_count 980.203270
rating_four_count 400.516231
rating_three_count 311.690656
rating_two_count  151.343933
rating_one_count  214.075544
badges_count      0.340709
badge_local_product 0.134565
badge_product_quality 0.262472
badge_fast_shipping 0.112075
product_variation_inventory 21.353137
shipping_option_price 1.024371
shipping_is_express 0.050379
countries_shipped_to 20.301203
inventory_total    2.562799
has_urgency_banner 0.000000
merchant_rating_count 78474.455607
merchant_rating    0.204768
```

```
In [64]: sum_ab = 0
sum_square = 0

for i, el in enumerate(t1):
    sum_ab += el * t2[i]
    sum_square += el**2

size = len(t1)
k_lin = (size * sum_ab - sum(t1) * sum(t2))/(size * sum_square - sum(t1)**2)
b_lin = sr1 - sr2 * k_lin
func_val = []

for el in t1:
    func_val.append(k_lin * el + b_lin)

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

Уравнение линейной зависимости: y = 208.30507951826667x + -185316.7198859288
```

```
In [65]: cor_chisl = 0
for i, el in enumerate(t1):
    cor_chisl += (el - sr1)*(t2[i] - sr2)
sqr_diff_vol = sum((el-sr1)**2 for el in t1)
sqr_diff_bag = sum((el-sr2)**2 for el in t2)
r_xy = cor_chisl / sqrt(sqr_diff_vol * sqr_diff_bag)
print(f"Коэффициент парной корреляции: {r_xy}")

Коэффициент парной корреляции: 0.05411229308648011
```

Рисунок 10 – проработка заданий выданных преподавателем

Необходимо скомбинировать продукты питания так, чтобы их концентрация была оптимальной и соответствовала нормам содержания витаминов в пище.

Обозначим оптимальную концентрацию (количество единиц) для продукта 1 как x1, для продукта 2 – как x2 и так далее. Так как мы будем смешивать продукты, то для каждого витамина (столбца таблицы) можно просто просуммировать значения, по всем продуктам. Учитывая, что сбалансированная диета должна включать 170 единиц витамина А, то, используя данные из столбца «Витамин А», составим уравнение:

Продукт	Витамин А	Витамин В	Витамин С	Витамин D	Витамин Е
1	1	10	1	2	2
2	9	1	0	1	1
3	2	2	5	1	2
4	1	1	1	2	13
5	1	1	1	9	2

```
In [1]: import numpy as np
from scipy.linalg import solve

In [2]: A = np.array(
    [
        [1, 9, 2, 1, 1],
        [10, 1, 2, 1, 1],
        [1, 0, 5, 1, 1],
        [2, 1, 1, 2, 9],
        [2, 1, 2, 13, 2],
    ]
)

In [3]: b = np.array([170, 180, 140, 180, 350]).reshape((5, 1))

In [4]: x = solve(A, b)

In [5]: x
Out[5]: array([[10.],
               [10.],
               [20.],
               [20.],
               [10.]])

In [ ]:
```

Рисунок 11 – выполнение индивидуальной задачи

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?

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

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.