МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчёт о лабораторной работе №1 по дисциплине технологии распознавания образов

Выполнила:
Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

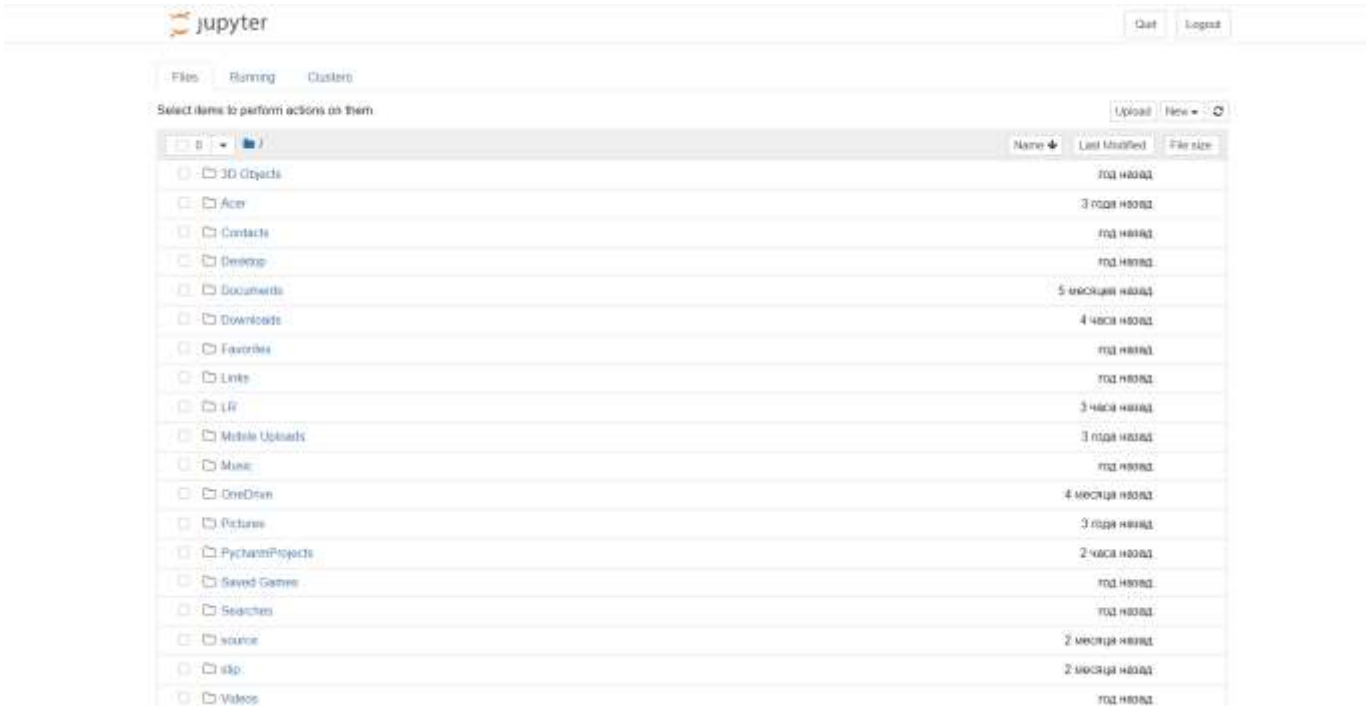


Рисунок 1 – Пример установки и запуска ноутбука

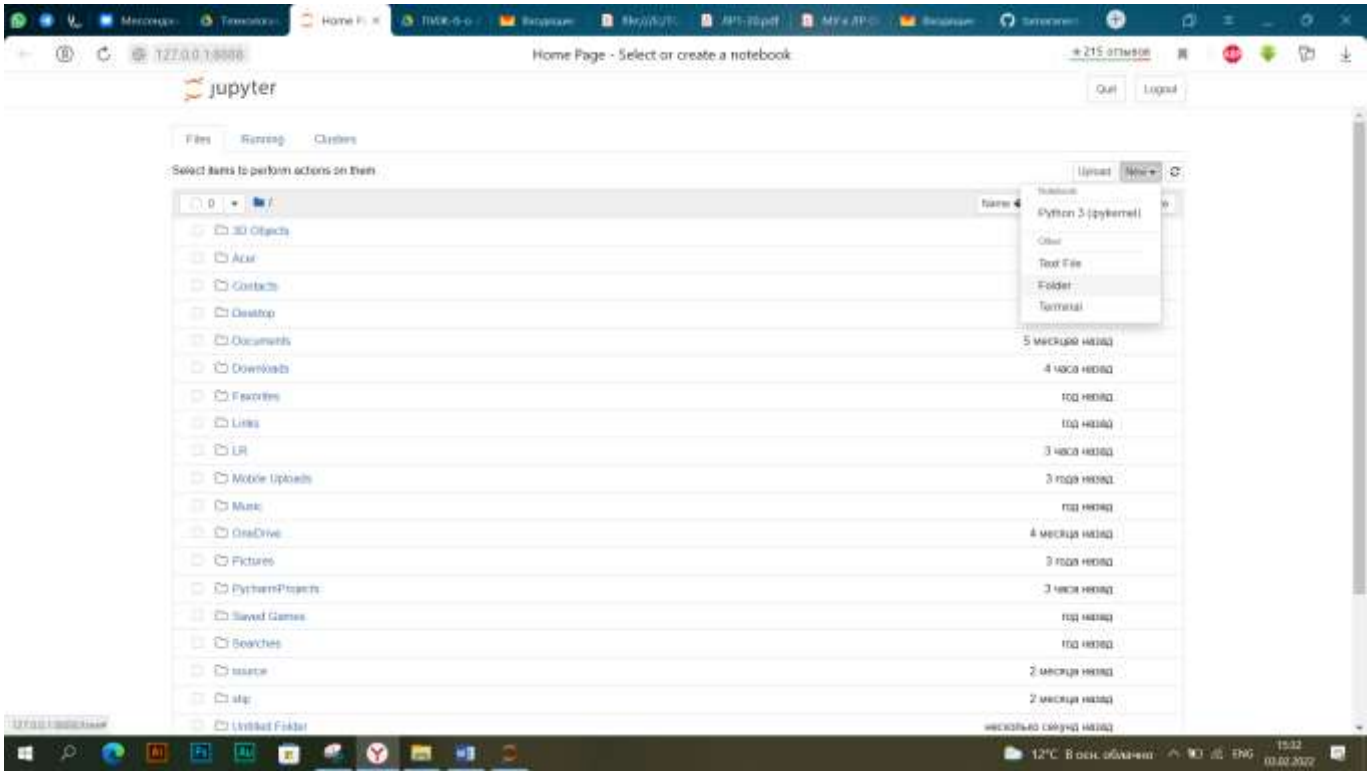


Рисунок 2 – Пример создания папки

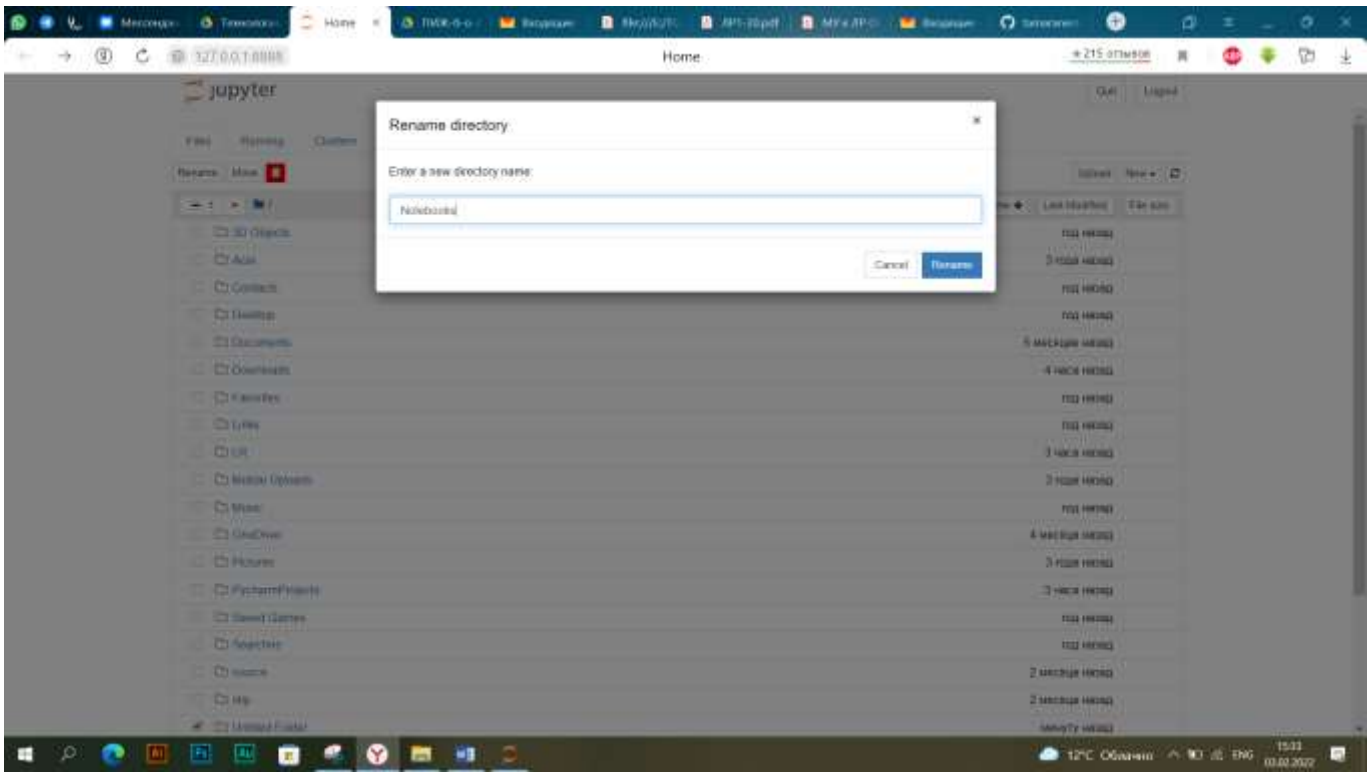


Рисунок 3 – Пример задания имени папки

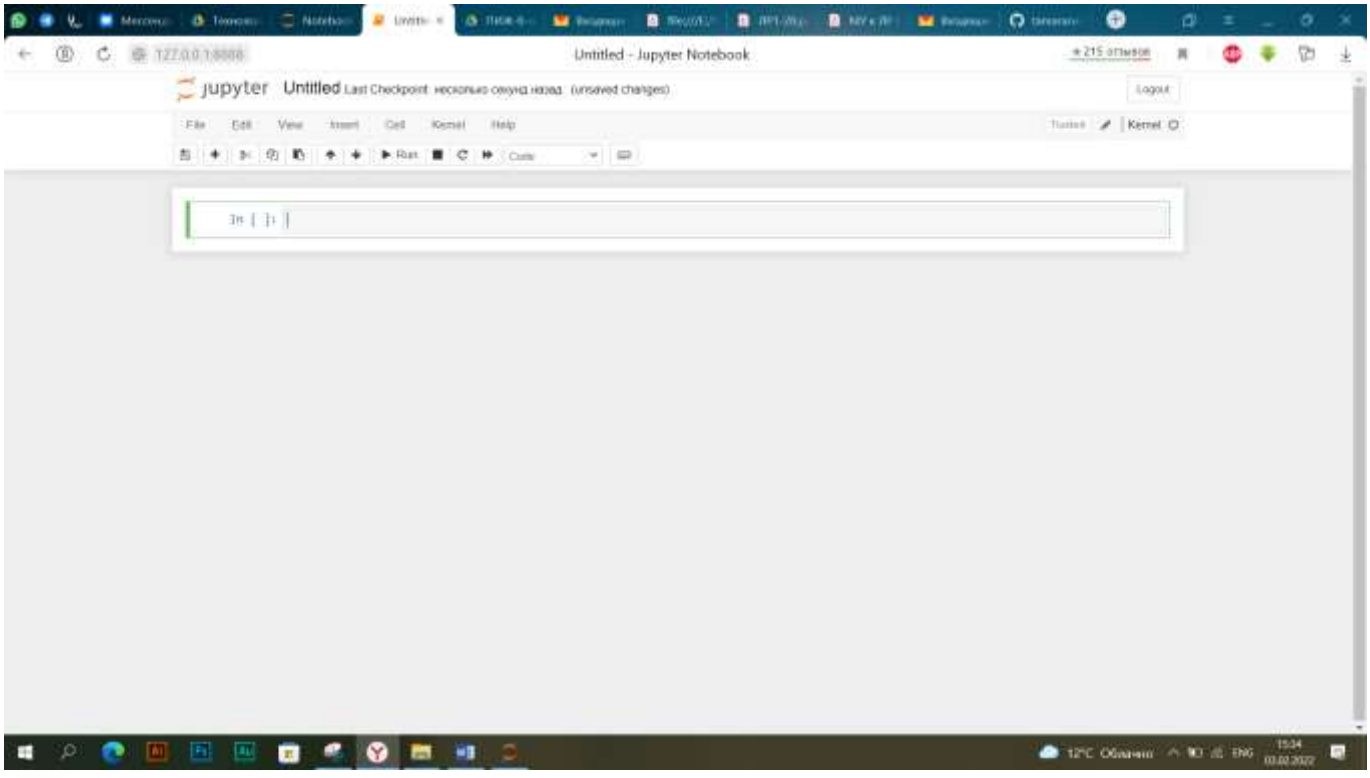


Рисунок 4 – Пример рабочей области

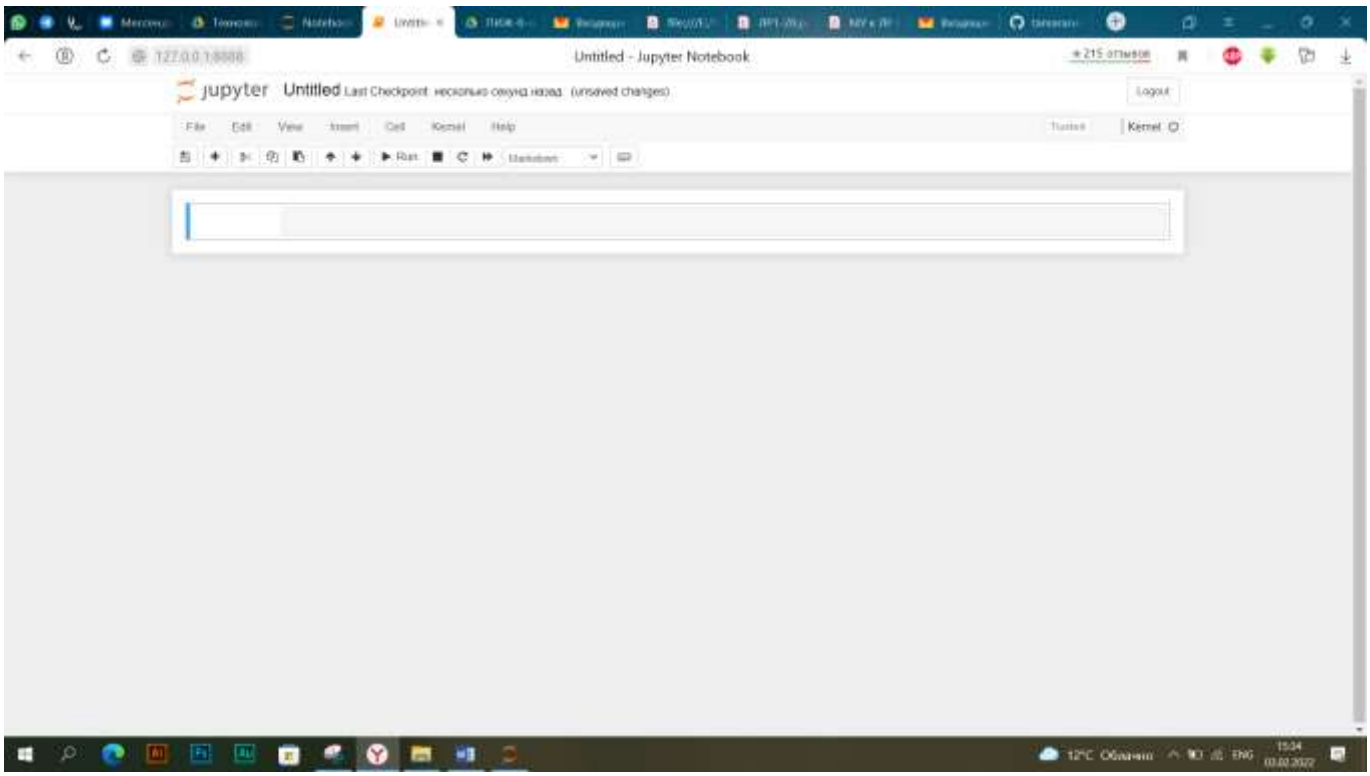


Рисунок 5 – Пример изменения типа ячейки с «Code» на «Markdown»

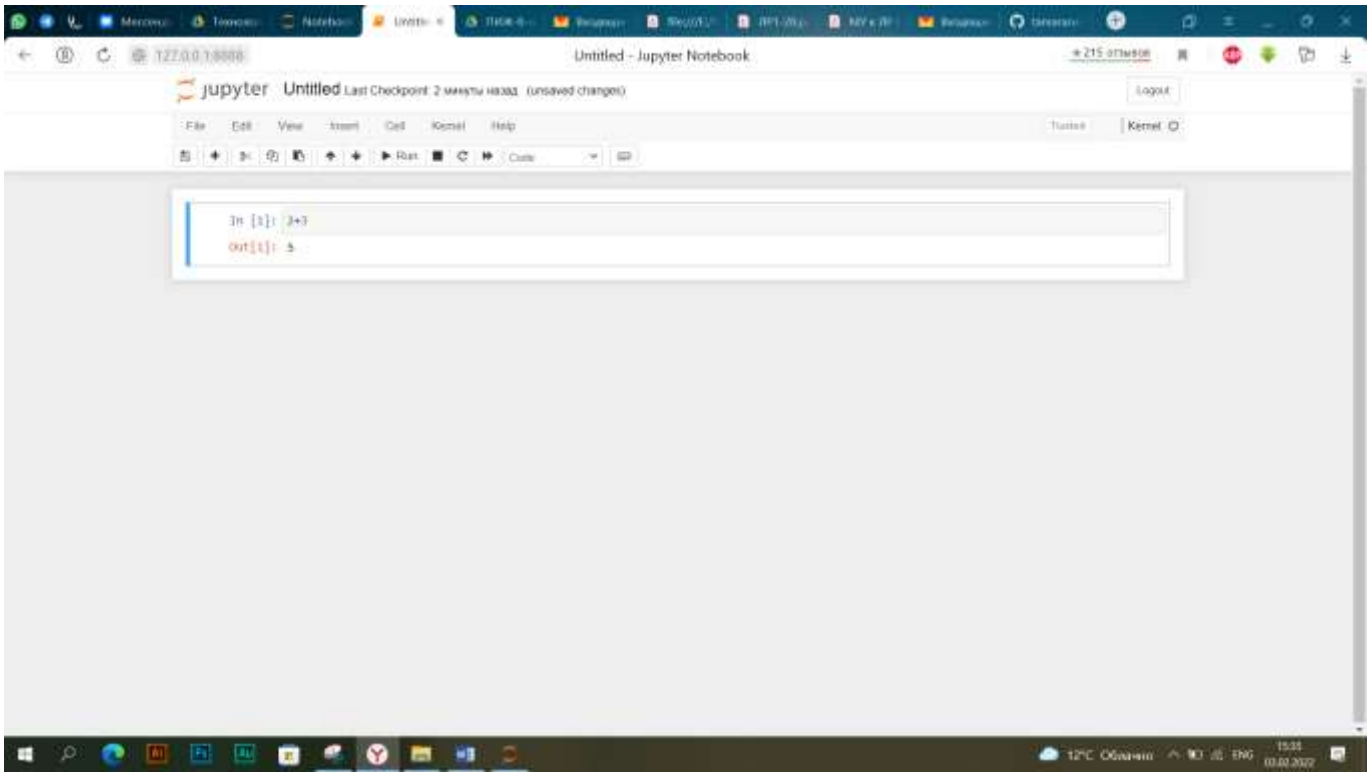


Рисунок 6 – Пример сложения

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

        12
```

Рисунок 7 – Пример сложения переменных и вывод результата

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

        0
        10
        20
        30
        40
        50
        60
```

Рисунок 8 – Пример работы с циклом

```
In [6]: i = 0
        while True:
            i += 1
            if i > 5:
                break
            print("Test while")

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

Рисунок 9 – Пример вывода текста через цикл

Рисунок 10 – Пример переименования ноутбука

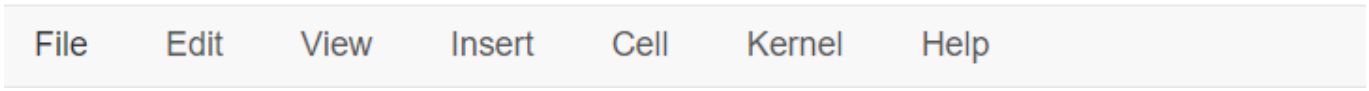


Рисунок 11 – Пример элементов интерфейса

Рисунок 12 – Пример панели инструментов

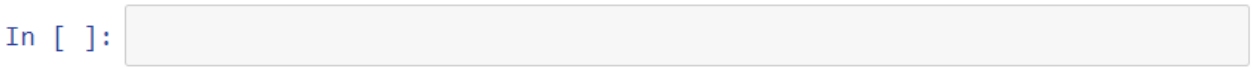


Рисунок 13 – Пример рабочего поля с ячейками

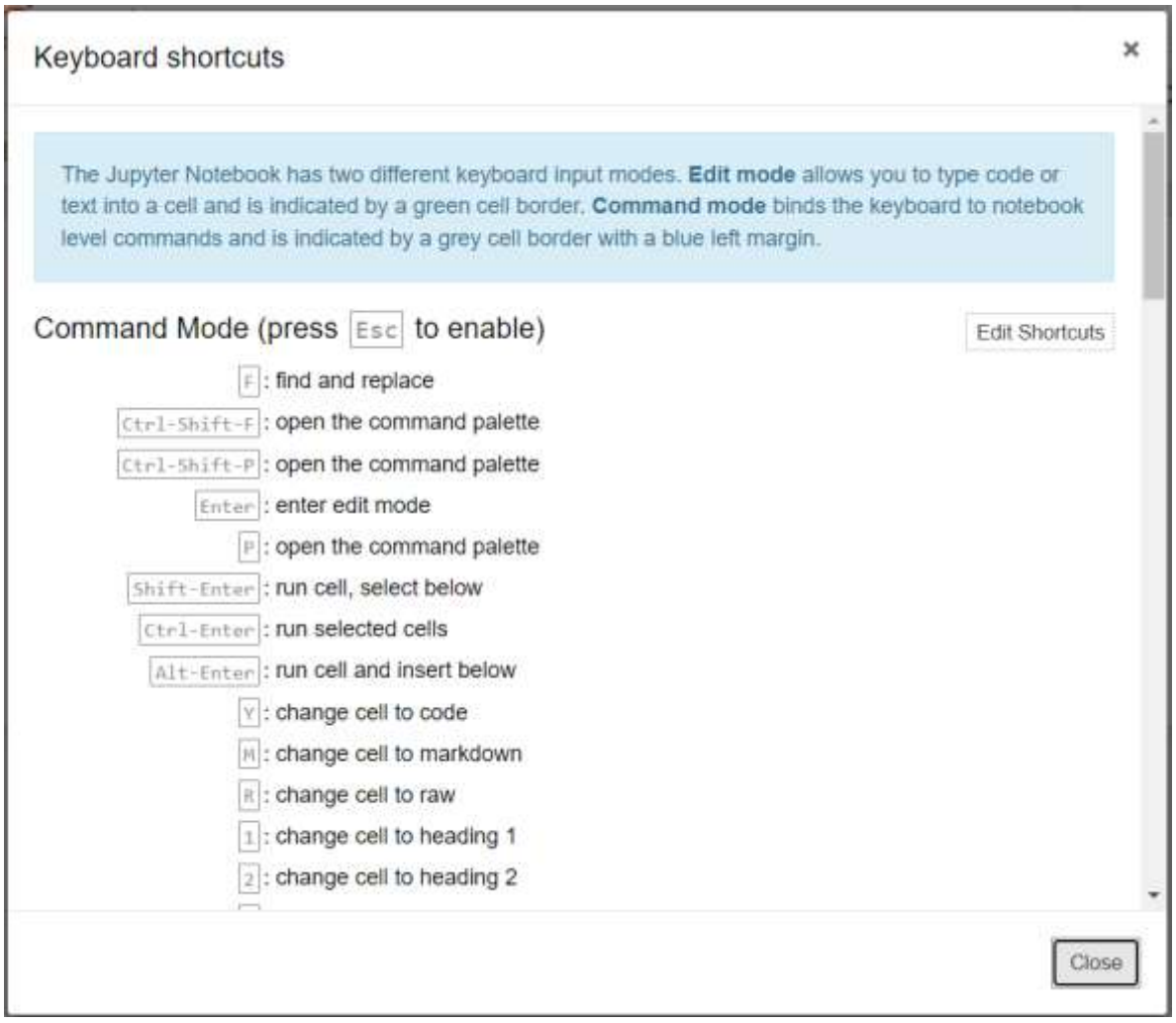


Рисунок 14 – Пример Keyboard Shortcuts

```
In [4]: from matplotlib import pylab as plt
        %matplotlib inline

In [5]: x = [i for i in range(50)]
        y = [i**2 for i in range(50)]
        plt.plot(x, y)

Out[5]: [<matplotlib.lines.Line2D at 0x1e603776f10>]
```

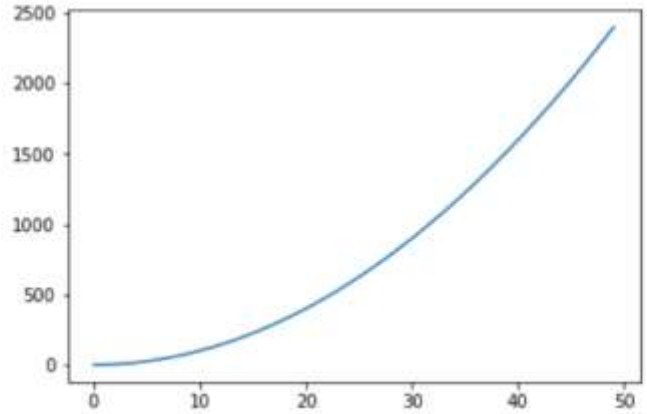


Рисунок 15 – Пример вывода графика

```
In [15]: %lsmagic

Out[15]: Available line magics:
         %alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave
         %bookmark  %cd  %clear  %cls  %colors  %conda  %config  %connect_inf
         o  %copy  %ddir  %debug  %dhist  %dirs  %doctest_mode  %echo  %ed  %
         edit  %env  %gui  %hist  %history  %killbgscripts  %ldir  %less  %lo
         ad  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %logs
         top  %ls  %lsmagic  %macro  %magic  %matplotlib  %mkdir  %more  %not
         ebook  %page  %pastebin  %pdb  %pdef  %pdoc  %pfile  %pinfo  %pinfo2
         %pip  %popd  %pprint  %precision  %prun  %psearch  %psource  %pushd
         %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %rehashx  %rel
         oad_ext  %ren  %rep  %rerun  %reset  %reset_selective  %rmdir  %run
         %save  %sc  %set_env  %store  %sx  %system  %tb  %time  %timeit  %un
         alias  %unload_ext  %who  %who_ls  %whos  %xdel  %xmode

         Available cell magics:
         %%!  %%HTML  %%SVG  %%bash  %%capture  %%cmd  %%debug  %%file  %%htm
         l  %%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.
```

Рисунок 16 – Пример использования %ismagic

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

env: TEST=5
```

Рисунок 17 – Пример использования %env

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

        Wall time: 5.49 s

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

         7.08 µs ± 419 ns per loop (mean ± std. dev. of 7 runs, 100000 loops
         each)
```

Рисунок 18 – Пример использования %%time и %timeit

```
In [1]: ticket_number = int(input("Введите число: "))

        Введите число: 102300

In [2]: if 999999 >= ticket_number >= 100000:
            if ((ticket_number % 10) + ((ticket_number // 10) % 10) + ((ticket_number // 100) % 10) ==
                    ((ticket_number // 1000) % 10) + ((ticket_number // 10000) % 10) + ((ticket_number // 100000) % 10)):
                print("Yes")
            else: print("No")
        else: print("Error")

        Yes
```

Рисунок 18 – Пример решения задания №1

```
In [1]: ticket_number = int(input("Введите число: "))

        Введите число: 102300

In [2]: if 999999 >= ticket_number >= 100000:
            if ((ticket_number % 10) + ((ticket_number // 10) % 10) + ((ticket_number // 100) % 10) ==
                    ((ticket_number // 1000) % 10) + ((ticket_number // 10000) % 10) + ((ticket_number // 100000) % 10)):
                print("Yes")
            else: print("No")
        else: print("Error")

        Yes
```

Рисунок 19 – Пример решения задания №2

```
In [1]: password = str(input())

        Andrei8

In [2]: import re
        if ((re.search('\d+', password)) and  (re.search(r'[andrei]', password))
            and (re.search(r'[ANDREI]', password)) and (password.find('Andrey')) and len(password)>4):
            password.lower()
            if (password.find("andrei")) == -1:
                print ("Weak")
        else: print ("Srting")

        Weak
```

Рисунок 20 – Пример решения задания №2

```
In [1]: amount = int(input())

        10

In [2]: 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
        13
        21
        34
        55
```

Рисунок 21 – Пример решения задания №3

In [2]:
```
import csv
from math import sqrt

with open('covid.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    effect_covid_2015 = []
    effect_covid_2021 = []
    for row in data:
        if row[4] == "China" and row[0] == "Exports" and row[1] == "2015":
            row[9] = int(row[9]) / 10000000
            effect_covid_2015.append(int(row[9]))
        if row[4] == "China" and row[0] == "Exports" and row[1] == "2021":
            row[9] = int(row[9]) / 10000000
            effect_covid_2021.append(int(row[9]))
    income_2015 = sum(effect_covid_2015)
    income_2021 = sum(effect_covid_2021)
    print("Совокупный доход от экспорта в Китае за 2015 год составил "
        f"{income_2015}$")
    print("Совокупный доход от экспорта в Китае за 2021 год составил "
         f"{income_2021}$")
```

```
Совокупный доход от экспорта в Китае за 2015 год составил 2465760000000$
Совокупный доход от экспорта в Китае за 2021 год составил 5675150000000$
```

Таким образом, можно сделать вывод, что в 2021 году Китай заработал на экспорте товаров в два раза больше, чем до пандемии.

Посчитаем количество дней экспортируемого товара в 2015 году и найдём среднее количество товара экспортируемого за день. Тоже самое сделаем с экспортом в 2021 году.

In [3]:
```
operation_export_2015 = len(effect_covid_2015)
operation_export_2021 = len(effect_covid_2021)
print(f"Китай произвёл {operation_export_2015} операций экспорта в 2015 году")
print(f"Китай произвёл {operation_export_2021} операций экспорта в 2021 году")

average_income_per_operation_2015 = round(income_2015 / operation_export_2015)
average_income_per_operation_2021 = round(income_2021 / operation_export_2021)
print(f"{average_income_per_operation_2015}$ средний доход за операцию в 2015 году")
print(f"{average_income_per_operation_2021}$ средний доход за операцию в 2021 году")
```

```
Китай произвёл 2876 операций экспорта в 2015 году
Китай произвёл 2789 операций экспорта в 2021 году
860000000$ средний доход за операцию в 2015 году
2030000000$ средний доход за операцию в 2021 году
```

Таким образом, можно сделать вывод, что количество операций экспорта уменьшилось, но доход вырос. Соответственно и средний доход за одну операцию вырос.

Для того чтобы посчитать среднее отклонение выбранных величин создадим две промежуточные переменные, которые будут содержать в себе сумму квадратов разности для каждого элемента из списка, поделённую на количество элементов.

In [4]:
```
var1 = sum((elem-average_income_per_operation_2015)**2 for elem in effect_covid_2015) / operation_export_2015
st_dev_2015 = round(sqrt(var1))
var2 = sum((elem-average_income_per_operation_2021)**2 for elem in effect_covid_2021) / operation_export_2021
st_dev_2021 = round(sqrt(var2))
print(f"Стандартное отклонение коэффициента экспорта в 2015 году: {st_dev_2015}")
print(f"Стандартное отклонение коэффициента экспорта в 2021 году: {st_dev_2021}")
```

```
Стандартное отклонение коэффициента экспорта в 2015 году: 159
Стандартное отклонение коэффициента экспорта в 2021 году: 361
```

Для того чтобый найти коэффициенты уравнений линейной зависимости посчитаем сумму произведений элементов из двух списков, а также сумму квадратов элементов из списка коэффициентов экспорта за 2015 и 2021 год. Затем посчитам по формуле коэфициент k и b. После выведим их на экран, а также значения функции для каждой точки списка коэффициентов за 2015 и 2021 год.

In [5]:
```
sum_ab = 0
sum_square = 0
raz = len(effect_covid_2015) - len(effect_covid_2021)
del effect_covid_2015[:raz]
for idx, elem in enumerate(effect_covid_2015):
    sum_ab += elem * effect_covid_2021[idx]
    sum_square = elem**2 + effect_covid_2021[idx]**2
sum_2015 = sum(effect_covid_2015)
sum_2021 = sum(effect_covid_2021)
operation = len(effect_covid_2015)
av_2015 = round(sum_2015 / operation)
av_2021 = round(sum_2021 / operation)
k_lin = ((operation * sum_ab - sum_2015) * sum_2021)/(operation * sum_square - sum_2015**2)
b_lin = av_2015 - av_2021 * k_lin
func_val_1 = []
for elem in effect_covid_2015:
    func_val_1.append(k_lin * elem + b_lin)
print(f"Уравнение линейной зависимости: y = {k_lin}x + {b_lin}")
print("Значение функции на кривой методом наименьших квадратов: ")
for val in func_val_1:
    print(val)
```

```
-478725100.79837894
-487123788.26852584
-495522475.73867273
-499721819.4737463
-503921163.20881987
-516519194.4140403
-529117225.6192608
-537515913.0894079
-545914600.5595548
-554313288.0297017
-558512631.7647753
-566911319.2349224
-579509350.4401429
-596306725.3804369
-608904756.5856574
-613104100.3207307
-625702131.5259514
-638300162.7311718
-642499506.4662452
```

Для рассчёта коэффициента парной корреляции разделим формулу на числитель и знаменатель. Для числителя посчитаем произведение разностей элементов каждого из списков со средним значением этих списков. В знаменателе считаем произведение средних квадратических отклонений.

In [7]:
```
cor_chisl = 0
for idx, elem in enumerate(effect_covid_2015):
    cor_chisl += (elem - average_income_per_operation_2015)*(effect_covid_2015[idx] - average_income_per_operation_2021)
sqr_diff_2015 = sum((elem-average_income_per_operation_2015)**2 for elem in effect_covid_2015)
sqr_diff_2021 = sum((elem-average_income_per_operation_2021)**2 for elem in effect_covid_2021)
r_xy = cor_chisl / sqrt(sqr_diff_2015 * sqr_diff_2021)
print(f"Коэффициент парной корреляции: {r_xy}")
```

```
Коэффициент парной корреляции: 0.4468005676842572
```

Значение 0,44 означает полную положительную зависимость, иными словами, между наблюдаемыми переменными имеется точная линейная зависимость с отрицательным или положительным коэффициентом.

Рисунок 22 – Пример решения задания №4

2. Вопросы для защиты

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?

%ismagic – список доступных магических команд

%env – для работы с переменными окружения

%run – для запуска файлов с расширением «.py»

%%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.

IDE, которая играет важную роль при разговоре о Python, — это Jupyter Notebook. Ранее известный как IPython Notebook, Jupyter Notebook особенно важен для придания формы тому, что Дональд Кнут, ученый-компьютерщик из Стэнфорда, назвал «грамотным программированием».

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.

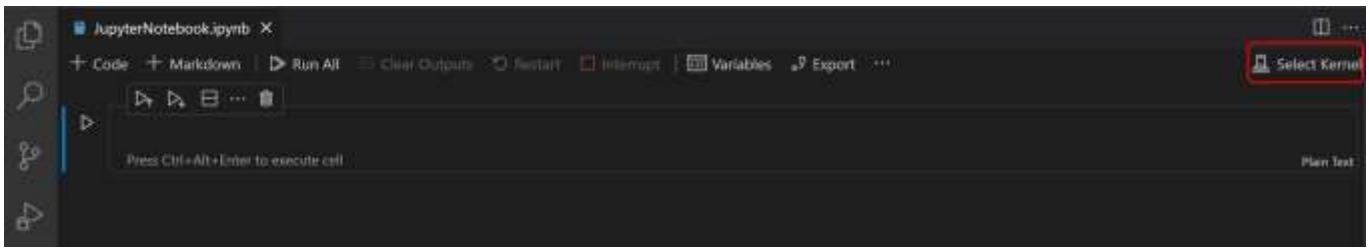
Обширная поддержка интеграции Jupyter Notebook в PyCharm позволяет разработчикам создавать, выполнять и отлаживать исходные коды, одновременно изучая их выходные данные.

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.



Ссылка на репозиторий: https://github.com/tamaranesterenko/TRO_LR_1