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

Работа с IPython и Jupyter Notebook.

по дисциплине «Технологии программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-21-1

Лысенко И.А. « »_____20__г.

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. Создал репозиторий на GitHub: https://github.com/IsSveshuD/lab_3_1.git .

2. Проработал примеры:

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

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

```
12
```

```
In [4]: n = 7
        for i in range(n):
            print(i*10)
```

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

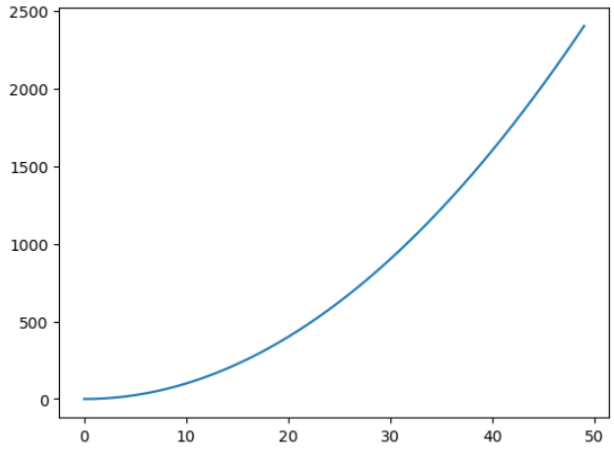
```
In [5]: i = 0
        while True:
            i += 1
            if i > 5:
                break
            print("Test while")
```

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

```
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 0x25e0aaac790>]
```



```
In [11]: %lsmagic

Out[11]: 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

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

In [37]: %env TEST = 5

env: TEST=5

In [39]: %%time
import time
for i in range(50):
    time.sleep(0.1)

CPU times: total: 0 ns
Wall time: 5.46 s

In [40]: %timeit x = [(i**10) for i in range(10)]

3.75 µs ± 303 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

Рисунок 1 – Примеры

3. Выполнил индивидуальное задание. Решил задачу по теории вероятности.

```
В партии 10 % нестандартных деталей. Наудачу отобраны четыре детали. Написать
биномиальный закон распределения дискретной случайной величины X – числа
нестандартных деталей среди четырёх отобранных и построить многоугольник
распределения.
```

```
In [2]: import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import binom

pb = binom(n = 4, p = 0.1)

x = np.arange(0,5)
pmf = pb.pmf(x)
a = [round(i, 4) for i in pmf]
print(x)
print(a)
plt.plot(x , pb.pmf(x))
plt.ylabel('P(x)')
plt.xlabel('x')
plt.show()
```

```
[0 1 2 3 4]
[0.6561, 0.2916, 0.0486, 0.0036, 0.0001]
```

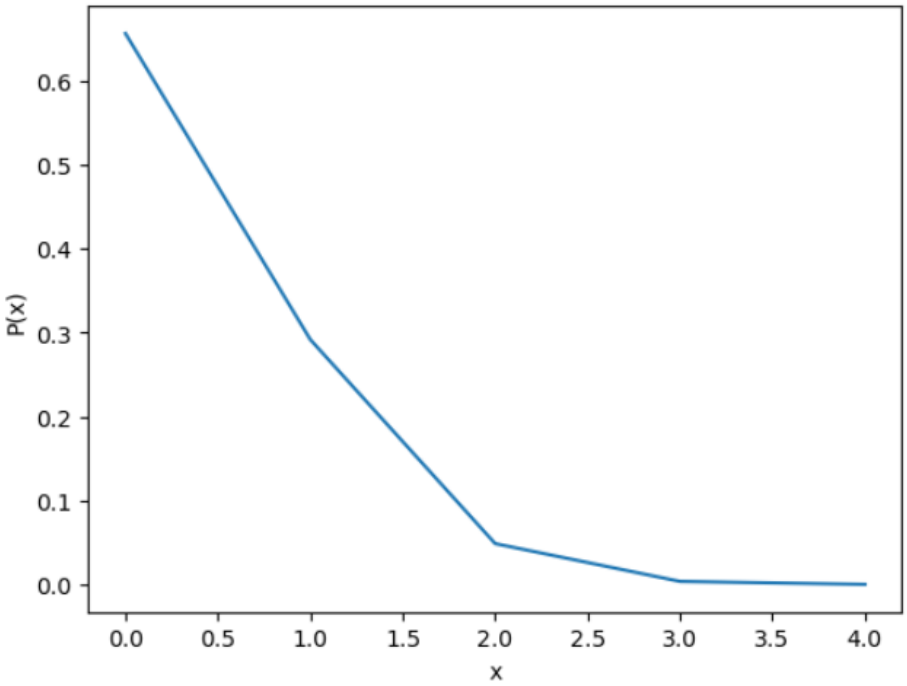
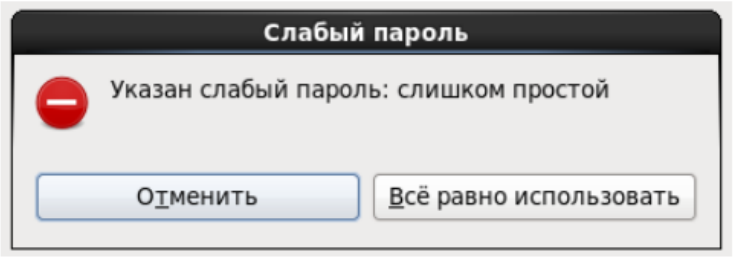


Рисунок 2 – Индивидуальное задание

4. Выполнил задачи от преподавателя:

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 латинницей, 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`.

```
password = Strelec39!
```

```
# проверяем условия надежности пароля
if any(char.isdigit() for char in password) and \
   any(char.islower() for char in password) and \
   any(char.isupper() for char in password) and \
   len(set(password)) >= 4 and \
   not any(word.lower() in password.lower() for word in ["ivan"]):
       print("strong")
else:
    print("weak")
```

```
weak
```

Числа Фибоначчи

Как известно, числа Фибоначчи — это последовательность чисел, каждое из которых равно сумме двух предыдущих (первые два числа равны 1):

$$1, 1, 2, 3, 5, 8, 13, \ldots$$

Задание:

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

```
amount = 10 # any number
```

```
fibonacci = [1, 1] # задаем начальные значения для последовательности

for i in range(2, amount):
    next_num = fibonacci[i-1] + fibonacci[i-2]
    fibonacci.append(next_num)

# выводим результат
print(*fibonacci, sep=' ')
```

```
1 1 2 3 5 8 13 21 34 55
```

Время исследований

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.

```python
import csv
import statistics
import numpy as np
from matplotlib import pylab as plt
%matplotlib inline

with open('cardata.csv', newline='') as file:
    reader = csv.DictReader(file)
    data = [row for row in reader]

prices = [float(row['price (eur)']) for row in data]
mileage = [float(row['mileage (kms)']) for row in data]

#Нахождение средних значений и средних отклонений.
print("Средняя цена:", statistics.mean(prices))
print("Стандартное отклонение цены:", statistics.stdev(prices))
print("Средний пробег:", statistics.mean(mileage))
print("Стандартное отклонение пробега:", statistics.stdev(mileage))

x = np.array(mileage)
y = np.array(prices)
#Уравнение линейной зависимости
coeffs = np.polyfit(x, y, 1)
equation = np.poly1d(coeffs)
print("Уравнение линейной зависимости:", equation)

r = np.corrcoef(x, y)[0, 1]
print("Парная корреляция:", r)
```

```
Средняя цена: 15973.38383838384
Стандартное отклонение цены: 6984.338370017161
Средний пробег: 83228.83333333333
Стандартное отклонение пробега: 46202.11858874543
Уравнение линейной зависимости:
-0.02984 x + 1.846e+04
Парная корреляция: -0.19738045972064197
```

Для иисследования был взят набор данных, содержащий информацию о различных подержанных автомобилях, которые были выставлены на продажу на www.flexicar.es веб-сайт провинции Барселона в апреле 2022 года.

Результаты исследования: Средняя цена автомобиля составляет около 16 тысяч евро, с отклонением в 7 тысяч. Средний пробег автомобиля составляет около 83 тысяч киллометров, с отклонением в 46 тысяч. Уравнение линейной зависимости между ценой и пробегом следующее: -0,0298x + 18460 Коэффициент кореляции: -0,2. Это значит, что корреляция между ценой и пробегом слабая и отрицательная, т.е чем выше пробег автомобиля, тем ниже его цена.

Отсюда можно сделать вывод, что цена автомобиля в провинции Барселона зависит от пробега, и эта зависимость линейная и имеет слабую отрицательную корреляцию.

Рисунок 3 – Задачи от преподавателя.

Ответы на вопросы:

1. Как осуществляется запуск Jupyter notebook?

Jupyter Notebook входит в состав Anaconda. Для запуска Jupyter Notebook необходимо перейти в папку Scripts (она находится внутри каталога, в котором установлена Anaconda) и в командной строке набрать:

> ipython notebook - В результате будет запущена оболочка в браузере.

2. Какие существуют типы ячеек в Jupyter notebook?

Code – для кода, Markdown – для текста.

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 следующиесочетания 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.

Available line magics:

%alias %alias_magic %autoawait %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

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

PyCharm:

– Сначала вы должны создать новый проект.

– В этом проекте создайте новый файл ipynb, выбрав File> New…> Jupyter Notebook. Это должно открыть новый файл записной книжки.

– Если у вас не установлен пакет Jupyter Notebook, над вновь открытым файлом ipynb появится сообщение об ошибке и предложение «Установить пакет jupyter» рядом с ним.

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

– Чтобы начать изучение Jupyter Notebook в PyCharm, создайте ячейки кода и выполните их.

– Выполните ячейку кода, чтобы запустить сервер Jupyter. По умолчанию сервер Jupyter использует порт 8888 по умолчанию 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».

– Теперь вы можете получить доступ к вкладке переменных в PyCharm, чтобы увидеть, как значения ваших переменных меняются при выполнении ячеек кода. Это помогает при отладке.

Visual Studio Code:

– Если у вас еще нет существующего файла Jupyter Notebook, откройте VS Code Command Palette с помощью сочетания клавиш CTRL+SHIFT+P (Windows) или Command+SHIFT+P (macOS) и запустите команду «Python: Create Blank New Jupyter Notebook».

– Если у вас уже есть файл Jupyter Notebook, это так же просто, как просто открыть этот файл в VS Code. Он автоматически откроется с новым редактором Jupyter.

Вывод: были исследованы базовые возможности интерактивных оболочек IPython и Jupyter Notebook для языка программирования Python.