

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 «Программирование на Python»

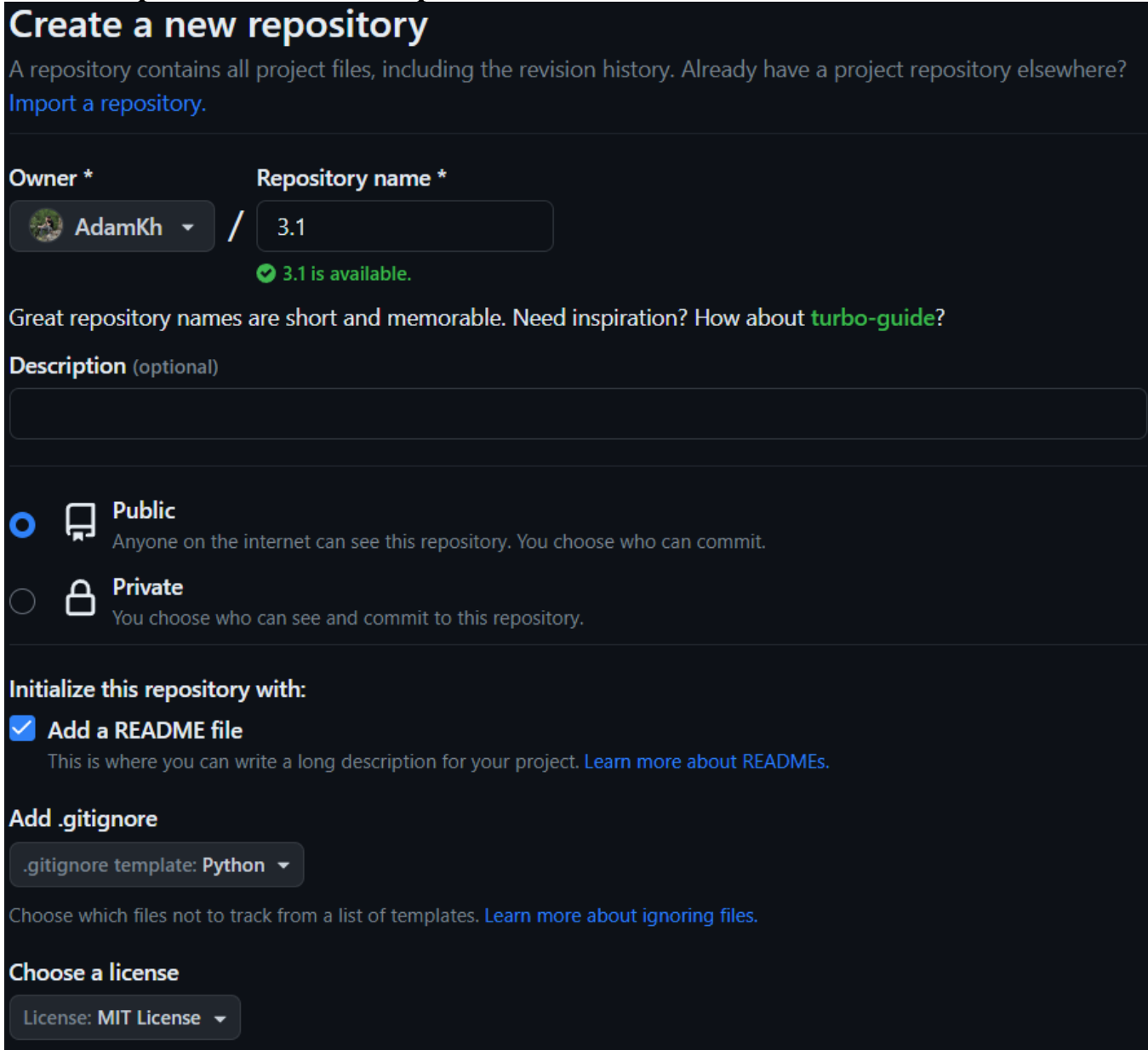
Выполнил студент группы ИВТ-б-о-21-1

Хашиев Адам М.

Проверил Воронкин Р.А. _____

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


Порядок выполнения работы:



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 ***

 AdamKh / 3.1

✔ 3.1 is available.

Great repository names are short and memorable. Need inspiration? How about [turbo-guide?](#)

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:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

```
C:\Users\adam_kh\Desktop\4sem\Анализ данных (Python)>git clone https://github.com/AdamKh/3.1.git
Cloning into '3.1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\adam_kh\Desktop\4sem\Анализ данных (Python)>
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\adam_kh\Desktop\4sem\Анализ данных (Python)>git flow init
Initialized empty Git repository in C:/Users/adam_kh/Desktop/4sem/Анализ данных (Python)/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
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? [C:/Users/adam_kh/Desktop/4sem/Анализ данных (Python)/.git/hooks]
C:\Users\adam_kh\Desktop\4sem\Анализ данных (Python)>
```

Рисунок 3 - Ветвление по модели git-flow

4. Проработать примеры лабораторной работы.

Пример 1. Выставьте свойство “Code”, введите в ячейке “2 + 3” без кавычек и нажмите Ctrl+Enter или Shift+Enter, в первом случае введенный вами код будет выполнен интерпретатором Python, во втором – будет выполнен код и создана новая ячейка. Если у вас получилось это сделать, выполните еще несколько примеров.

```
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
Test while
```

Рисунок 4 - Результат выполнения примера 1

Пример 2. По умолчанию, графики не выводятся в рабочее поле ноутбука. Для того, чтобы графики отображались, необходимо ввести и выполнить следующую команду: `%%matplotlib inline`. Пример вывода графика.

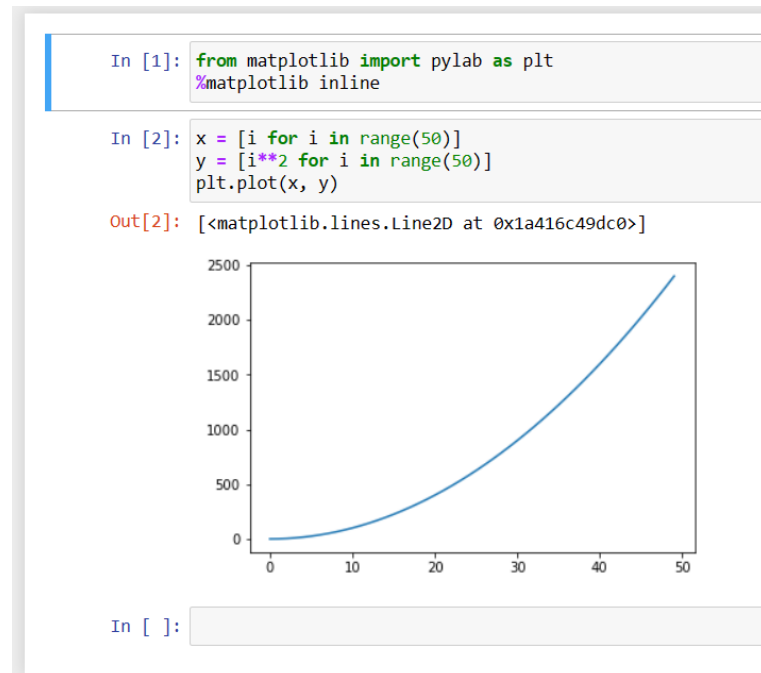


Рисунок 5 - Результат выполнения примера 2

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

Пример 3. `%%time` позволяет получить информацию о времени работы кода в рамках одной ячейки.

```
In [1]: %%time
        import time
        for i in range(50):
            time.sleep(0.1)

        Wall time: 5.49 s

In [ ]:
```

Рисунок 6 - Результат выполнения примера 3

Пример 4. `%timeit` запускает переданный ей код 100000 раз (по умолчанию) и выводит информацию о среднем значении трех наиболее быстрых прогонов.

```
In [1]: %timeit x = [(i**10) for i in range(10)]
2.4 µs ± 37.3 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

In [ ]:
```

Рисунок 7 - Результат выполнения примера 4

5. Создать ноутбук, в котором выполнить решение вычислительной задачи.

Первый автомобиль стоит на светофоре, а второй движется со скоростью 30 м/с. Какое расстояние между автомобилями будет через 20 секунд, если первый автомобиль начинает движение с постоянным ускорением 2 м/с²?

Индивидуальное задание

Первый автомобиль стоит на светофоре, а второй движется со скоростью 30 м/с. Какое расстояние между автомобилями будет через 20 секунд, если первый автомобиль начинает движение с постоянным ускорением 2 м/с²?

```
Ввод [1]: import matplotlib.pyplot as plt
executed in 856ms, finished 02:10:24 2023-04-28

Ввод [2]: # Исходные данные
v2 = 30 # Скорость второго автомобиля, м/с
a1 = 2 # Ускорение первого автомобиля, м/с²
t = 20 # Время, сек
executed in 10ms, finished 02:10:32 2023-04-28

Ввод [4]: # Рассчитываем расстояния
s2 = v2 * t
s1 = 0.5 * a1 * t**2
d = s2 - s1
executed in 9ms, finished 02:10:47 2023-04-28

Ввод [5]: # Выводим результат
print(f'Расстояние между автомобилями через {t} секунд: {d} м')

# Строим график
time = range(0, 25)
distances = [v2 * t - 0.5 * a1 * t**2 for t in time]
executed in 12ms, finished 02:11:02 2023-04-28

Расстояние между автомобилями через 20 секунд: 200.0 м
```

```
Ввод [6]: plt.plot(time, distances)
plt.xlabel('Время, сек')
plt.ylabel('Расстояние между автомобилями, м')
plt.title('Зависимость расстояния между автомобилями от времени')
plt.show()
executed in 190ms, finished 02:11:14 2023-04-28
```

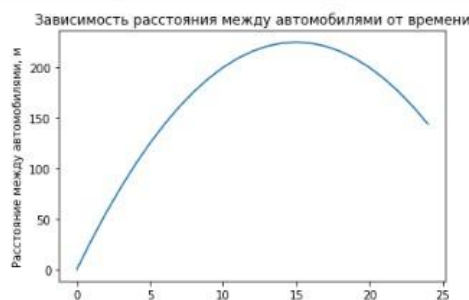


Рисунок 8 - Результат выполнения индивидуального задания

Контрольные вопросы:

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?

Если это код Python, то на панели инструментов нужно выставить свойство “Code”.

Если это Markdown текст – выставить “Markdown”.

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 следующие сочетания 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?

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

Для работы с переменными окружения используется команда %env.

Запуск Python кода из “.py” файлов, а также из других ноутбуков – файлов с расширением “.ipynb”, осуществляется с помощью команды %run.

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

`%timeit` запускает переданный ей код 100000 раз (по умолчанию) и выводит информацию о среднем значении трех наиболее быстрых прогонов.

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. Сначала вы должны создать новый проект.
2. В этом проекте создайте новый файл `ipynb`, выбрав `File> New...> Jupyter Notebook`. Это должно открыть новый файл записной книжки.
3. Если у вас не установлен пакет Jupyter Notebook, над вновь открытым файлом `ipynb` появится сообщение об ошибке. Сообщение об ошибке гласит: «Пакет Jupyter не установлен», и у вас будет опция «Установить пакет jupyter» рядом с ним.
4. Нажмите «Установить пакет jupyter». Это запустит процесс установки, который вы можете просмотреть, щелкнув запущенные процессы в правом нижнем углу окна PyCharm.
5. Чтобы начать изучение Jupyter Notebook в PyCharm, создайте ячейки кода и выполните их.
6. 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».
7. 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. Вы также можете 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

- 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 нативным редактором Jupyter.

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