

ВСТУП

Jupyter Notebook – неймовірно потужний інструмент для інтерактивної розробки та подання проектів в області наук про дані. Це зручний інструмент для створення аналітичних звітів, що дозволяє одночасно зберігати разом код, формули, зображення, коментарі, результати і графіки.

Блокнот (notebook) представляє код і результат його виведення в єдиному документі, який об'єднує візуалізацію, розмічений текст, математичні рівняння та інші мультимедійні дані. Цей інтуїтивно зрозумілий робочий процес сприяє ітеративній і швидкій розробці, що робить ноутбуки все більш популярним вибором для подання даних і їх аналізу.

Мета: отримати навички у встановленні, налаштуванні та основах роботи з Jupyter Notebook.

1. ТЕОРЕТИЧНІ ОСНОВИ

В цій лабораторній роботі ви дізнаєтеся, як налаштувати Jupyter Notebooks на локальному комп'ютері, а також як почати використовувати його в ваших проектах.

Почнемо з визначення: що таке «notebook» (блокнот)? Це потужний інструмент для розробки та подання проектів Data Science в інтерактивному вигляді. Він об'єднує код і результат виведення в одному документі, що містить текст, математичні рівняння і візуалізацію. Такий покроковий підхід забезпечує швидкий, послідовний процес розробки, оскільки висновок для кожного блоку показується відразу.

Проект Jupyter є наступником більш раннього проекту IPython Notebook, який вперше був опублікований в якості прототипу в 2010 році. Хоча Jupyter Notebooks можна використовувати з багатьма різними мовами програмування (назва Jupyter є акронімом від назв мов програмування Julia, Python та R), саме Python, є найбільш поширеним варіантом використання.

Найпростіше почати роботу з Jupyter Notebooks, встановивши дистрибутив Anaconda.

Anaconda є найбільш широко використовуваним дистрибутивом

Python для роботи з даними і поставляється з попередньо встановленими найбільш популярними бібліотеками та інструментами. Деякі з найбільших бібліотек Python, включених в Anaconda, включають NumPy, Pandas, Matplotlib.

Повний список бібліотек, можна переглянути за посиланням: <https://docs.anaconda.com/anaconda/packages/pkg-docs/>.

Встановлення дистрибутиву Anaconda дозволяє приступити до роботи без зайвого управління установками або занепокоєння про залежності і проблеми установки, що можуть бути пов'язані з операційною системою.

1.1. Встановлення Anaconda

1. Завантажте останню версію Anaconda за посиланням <https://www.anaconda.com/products/individual>.

2. Встановіть Anaconda керуючись інструкціями.

Після встановлення буде відкрито вікно Anaconda Navigator (рис. 1.1):

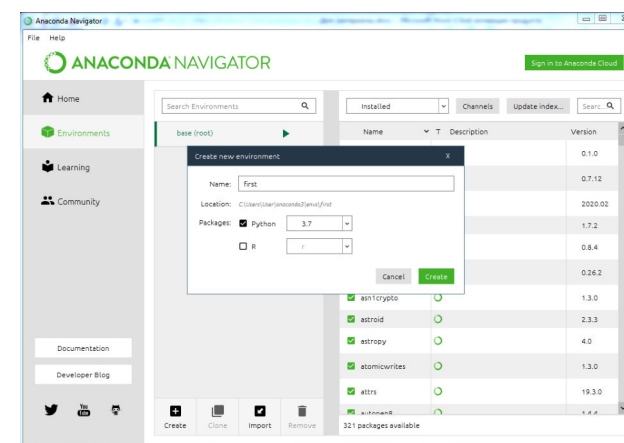


Рисунок 1.1 – Вікно Anaconda Navigator

Для зручності роботи рекомендується створити окреме оточення – це дозволить уникнути конфліктів версій (наприклад, коли потрібна інша версія Python або певної бібліотеки). Для цього в Anaconda Prompt

потрібно запустити команду:

```
conda create --name first python=3.7
```

де **first** – назва оточення.

Можна також вказати потрібну версію Python. Після створення оточення, його необхідно активувати командою

```
conda activate first
```

Створити нове оточення можна також і на вкладці Environments, натиснувши на Create в Anaconda Navigator.

Створення оточення може зайняти кілька хвилин. Перемикайтесь між оточеннями можна у випадяючому списку Applications on на вкладці Environments в Anaconda Navigator (рис. 1.2):

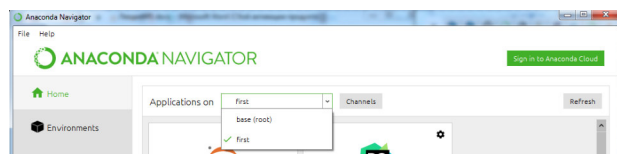


Рисунок 1.2 – Перемикання між створеними оточеннями в Anaconda Navigator

Далі потрібно встановити і запустити в Jupyter Notebook в обраному оточенні (рис. 1.3).

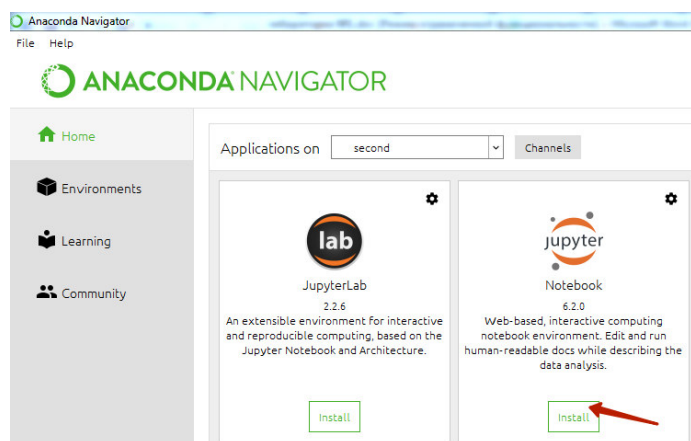


Рисунок 1.3 – Встановлення Jupyter

Можна також запустити блокнот із меню Пуск в групі застосунків Anaconda

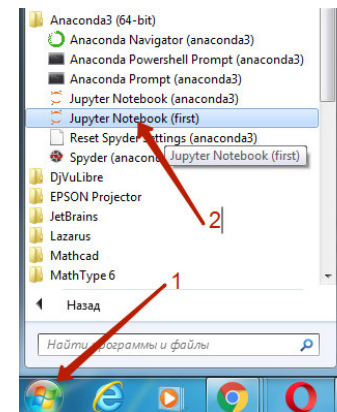


Рисунок 1.4 – Запуск Jupyter Notebook із меню Пуск

1.2. Встановлення Jupyter із командного рядка

1. Встановити Python – скачати файл завантаження з сайту <https://www.python.org/> та запустити файл. Для подальшого зручного програмування, встановіть прапорець Add Python to PATH (рис. 1.5)

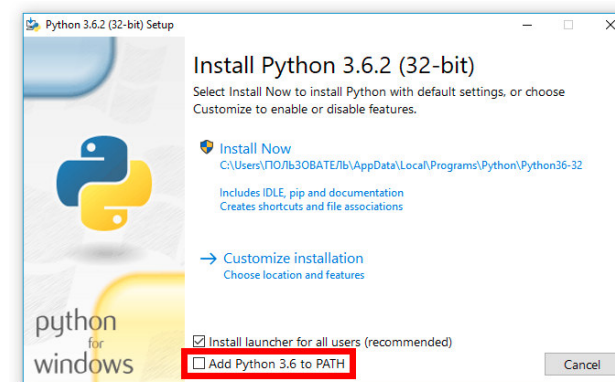


Рисунок 1.5 – Встановлення Python

3. Встановити Jupyter Notebook, задаванням у командному рядку **pip install jupyter**

4. Для запуску блокнота, ведіть команду **jupyter notebook**

Примітка: якщо під час інсталяції Python не було встановлено прапорець Add Python to PATH, можна додати шлях до папки, в якій знаходиться файл python.exe вручну:

Панель управління → Система и безопасность → Система → Дополнительные параметры системы, вкладка «Дополнительно», кнопка «Переменные среды». У відкритому вікні Переменные среды додати шлях у змінну Path (рис. 1.6)

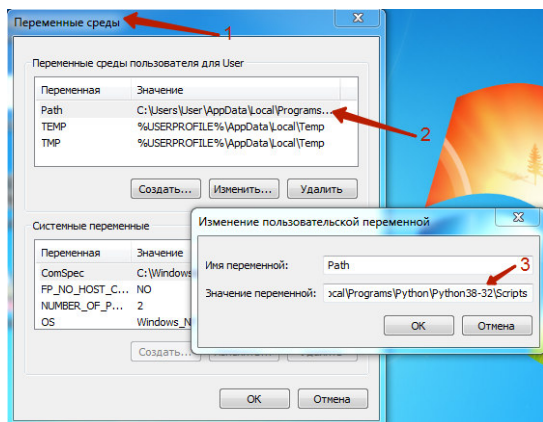


Рисунок 1.6 – Додавання шляху до python.exe вручну

1.3. Jupyter Lab

Подальшим розвитком Jupyter Notebook став Jupyter Lab – гнучке, інтегроване, легко розширюване середовище, що підтримує одночасну роботу з декількома блокнотами Jupyter, текстовими файлами, датасетами, терміналами та іншими компонентами. Аналогічно класичним IDE в JupyterLab можна впорядковувати документи в робочій області в зручному порядку за допомогою

вкладок і роздільників.

Встановлення Jupyter Lab

1. За допомогою conda

conda install -c conda-forge jupyterlab

2. За допомогою pip

pip install jupyterlab

Виклик Jupyter Lab із Anaconda Prompt

jupyter lab

Про роботу в Jupyter Lab можна прочитати за посиланням: <https://jupyterlab.readthedocs.io/en/stable/index.html>.

1.4. Створення першого блокнота

Після запуску блокнота в браузері відкриється нова вкладка, що буде виглядати наступним чином (рис. 1.7):

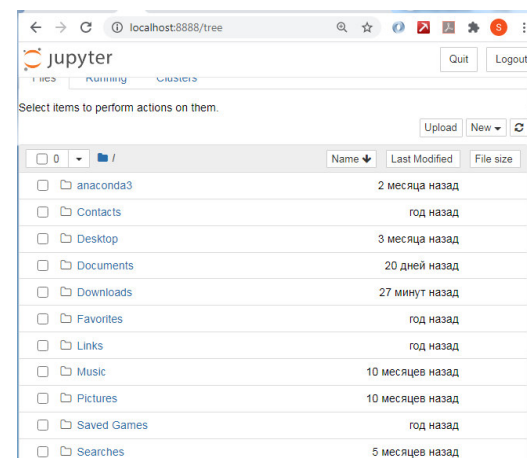


Рисунок 1.7 – Панель навігації Jupyter Notebook

На рис. 1.7 наведено панель інструментів ноутбука, що створена для управління ноутбуками Jupyter.

Майте на увазі, що панель управління може надати вам доступ тільки до файлів і підпапок, що містяться в каталозі запуску Jupyter; проте каталог запуску може бути змінений.

1.5. Зміна стартового каталогу Jupyter Notebook

При запуску із панелі Пуск:

1. Клацнути мишею Пуск → Все программы → Anaconda.
2. Правую кнопкою миші клацнути на назві потрібного блокноту та обрати пункт «Свойства».
3. Замінити в полі «Объект» %USERPROFILE% на шлях до потрібної папки (або додати в кінець рядка), наприклад, D:\py (рис. 1.8).

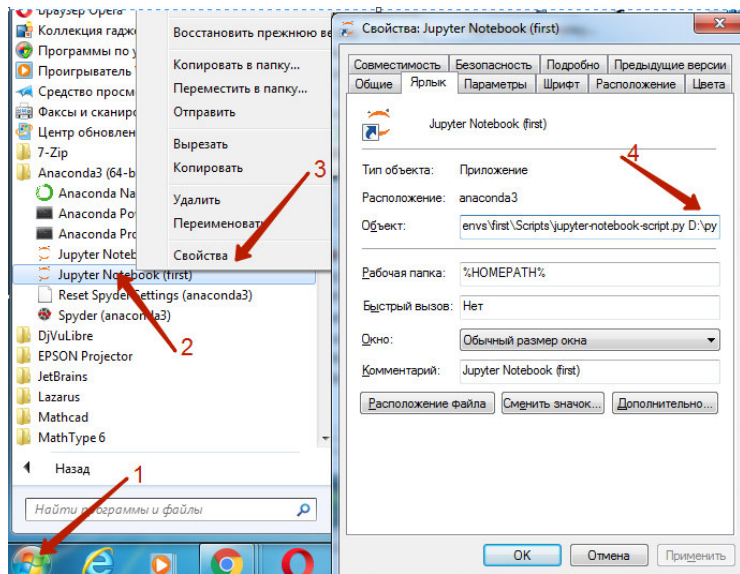


Рисунок 1.8 – Зміна стартового каталогу Jupyter Notebook за допомогою зміни властивостей ярлика

При запуску з терміналу (вікна cmd або anaconda prompt)

Спосіб 1 (працює якщо не налаштований шлях в

jupyter_notebook_config.py)

1a. Перейти в потрібну папку, використовуючи команду cd (наприклад, **cd /d d:\py**).

1b. Визвати команду Jupyter Notebook.

Спосіб 2. Ввести команду

jupyter notebook --notebook-dir="повний_шлях_до_папки"

Інші способи можна переглянути за посиланням <https://stackoverflow.com/questions/35254852/how-to-change-the-jupyter-start-up-folder>.

1.6. Початок роботи з Jupyter Notebook

Jupyter Notebook буде відкрито в браузері, що встановлено в системі за замовченням. URL-адреса в адресному рядку схожа на <http://localhost:8888/tree>. Localhost не є веб-сайтом, але вказує, що контент обслуговується з вашого локального комп'ютера. Ноутбуки і панель інструментів Jupyter – це веб-додатки, при цьому Jupyter запускає локальний веб сервер Python для обслуговування цих додатків в веб-браузері, що робить його практично незалежним від платформи і відкриває можливості для більш зручного обміну в Інтернеті.

Для початку роботи натисніть на кнопку New і оберіть Python3 (рис. 1.9)

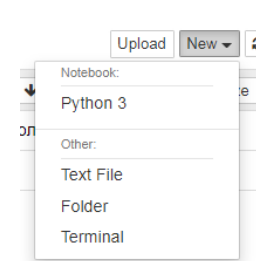


Рисунок 1.9 – Створення нового об'єкту

Ваш перший блокнот Jupyter відкриється в новій вкладці (рис. 1.10). До того ж кожен блокнот використовує свою вкладку, тому

ви можете відкривати кілька блокнотів одночасно.

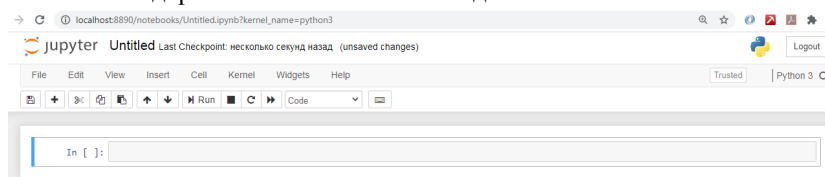


Рисунок 1.10 – Початковий вигляд створеного нового блокноту Jupyter

Якщо ви перейдете назад на панель інструментів (рис. 1.11), ви побачите новий файл Untitled.ipynb та зелений колір піктограми і тексту Running, який говорить про те, що ваш ноутбук працює.



Рисунок 1.11 – Вигляд панелі інструментів Jupyter при запуску блокноту

1.7. Файл ipynb

Буде корисно зрозуміти, чим є файл Jupyter Notebook насправді. По суті, кожен файл .ipynb є текстовим файлом, який описує вміст вашого блокноту у форматі JSON. Кожна комірка та її вміст, включаючи вкладення зображень, які були перетворені в рядки тексту, перераховані в ньому разом з деякими метаданими. Ви можете редагувати їх самостійно (тільки в разі, коли ви точно знаєте, що робите) вибравши Edit → Edit Notebook Metadata в рядку меню в блокноті.

Можна подивитися текст файлу в панелі навігації, натиснувши на кнопку Edit.

1.8. Перейменування файлу

Двічі клацнувши на назві файлу можна її змінити (рис. 1.12).



Рисунок 1.12 – Перейменування блокноту

1.9. Ядра (kernels) та комірки (cells)

- kernel (ядро) – це «обчислювальний двигун», що виконує код, який міститься в документі ноутбука.

- cell (комірка) – це контейнер для тексту, що буде відображатися в блокноті, або код, який буде виконуватися ядром ноутбука.

Комірка (Cell)

Комірки утворюють структуру ноутбука. На скріншоті нового блокнота (рис. 1.10) це поле з синім контуром.

Є два основних типи комірок, які ми розглянемо:

- Комірка коду (Code) містить код, який повинен бути виконаний в ядрі, і відображає його виведення нижче.

- Комірка Markdown містить текст, відформатований з використанням мови розмітки Markdown, і відображає його вивід на місці під час запуску.

Коли ви запустите комірку коду, її вивід буде відображено нижче, а мітка зліва зміниться з In [] на In [1]. Ви завжди можете визначити різницю між кодом і комірками Markdown, тому що комірки коду мають мітку In [n] зліва (n – номер запуску), а комірки Markdown – не мають.

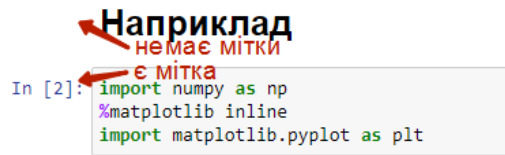


Рисунок 1.13 – Відмінність між комірками коду та Markdown

Частина «In» мітки просто короткий запис слова Input, а номер мітки вказує, коли комірка була виконана в ядрі – в нашому випадку комірка була виконана першою. Запустіть комірку знову, і мітка зміниться на In [2], тому що тепер комірка була другою, запущеною в ядрі.

Введемо наступний код в комірку:

```
In [ ]: import time
        time.sleep(3)
```

Ця комірка не виводить нічого, але для її виконання потрібно три секунди. При цьому Jupyter показує, що комірка в даний момент працює, змінивши мітку на In [*] (рис. 1.14).

```
In [*]: import time
        time.sleep(3) # сон у 3 секунди
```

Рисунок 1.14 – Вигляд комірки, код якої виконується ядром

Як правило, дані у вихідні комірки надходять з будь-яких текстових даних, спеціально надрукованих під час запуску комірок, а також із значення останнього рядка в комірці, будь то змінна, виклик функції або щось інше (рис. 1.15)

```
In [3]: def say_hello(name):
        return 'Hello, {}'.format(name)
        say_hello('Jupyter')

Out[3]: 'Hello, Jupyter!'
```

Рисунок 1.15 – Приклад виклику функції із комірки

1.10. Режими ноутбука

Блокноти Jupyter мають два різні режими EDIT і COMMAND.

Режим редагування EDIT. Щоб увійти в режим EDIT, натисніть клавішу Enter на клавіатурі або клацніть клітинку. Режим редагування можна визначити за зеленою рамкою навколо комірки із зеленим лівим полем (рис. 1.16). Коли ви перебуваєте в режимі редагування, ви можете вводити текст в клітинки.

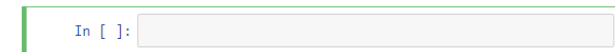


Рисунок 1.16 – Поточна комірка в режимі редагування

Командний режим COMMAND. Щоб увійти в режим COMMAND, натисніть Esc або клацніть де-небудь за межами комірки. Ви побачите сіру рамку навколо комірки із синім лівим краєм (рис. 1.17). Коли ви перебуваєте в командному режимі, ви можете редагувати свій блокнот, але не можете вводити клітинки.

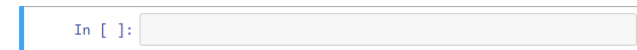


Рисунок 1.17 – Поточна комірка в командному режимі

1.11. Палітра команд

Натиснувши піктограму з клавіатурою, або Ctrl+Shift+P, відкривається палітра клавіатурних команд (рис 1.18).

Приклади команд:

- Запустити вибрану комірку: Ctrl + Enter;
- Запустити комірку та вставити нову комірку нижче: Alt + Enter;
- Запустити комірку та перейти до клітинки нижче: Shift + Enter;
- Вставте клітинку зверху: Esc+A;
- Вставте комірку внизу: Esc+B;
- Вирізати виділені клітинки: Esc+X;
- Копіювати виділені комірки: Esc+C.

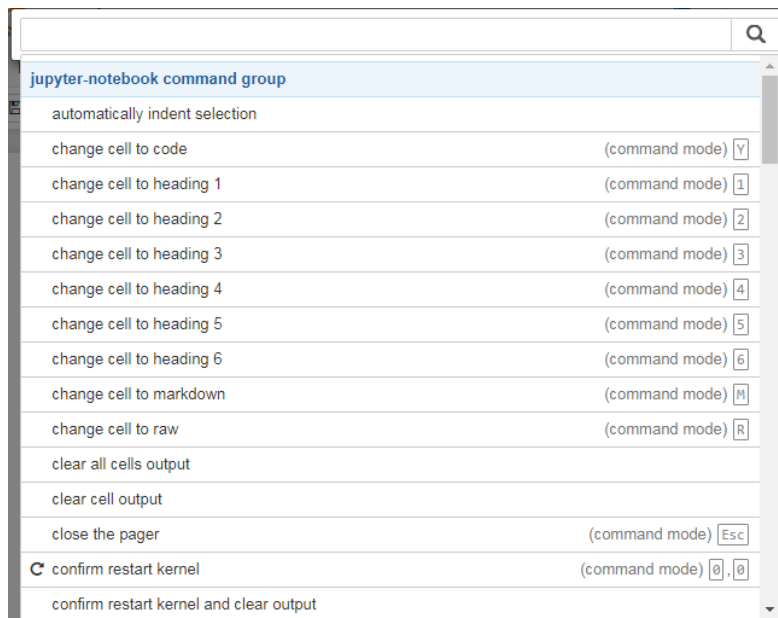


Рисунок 1.18 – Палітра команд Jupyter Notebook

1.12. Як поділитись блокнотом

Однією із важливих функцій Jupyter Notebook, що є особливо помічною при віддаленій роботі в команді – це можливість бути відображеною за допомогою GitHub Google Colab. Але перед тим, як поділитися створеним блокнотом, рекомендується привести його до більш охайного вигляду. Для цього необхідно спочатку очистити всі виведення Cell → All output → Clear, а потім запустити ядро для всіх клітинок Kernel → Restart & Run All. Така послідовність дій призведе до послідовного виконання коду в комірках ноутбука, починаючи з In [1].

Далі завантажте блокнот на GitHub, використовуючи Git Bash, GitHub Desktop, або ж просто завантажте через браузер. GitHub має вбудовані інструменти для рендерингу блокнотів Jupyter. Але іноді трапляються деякі вади відображення.

Для вірного рендерингу можна скористатися сайтом nbviewer (<https://nbviewer.jupyter.org/>), де у відповідному полі ввести адресу репозиторію GitHub і у відкритому вікні репозиторію обрати необхідний файл. Але nbviewer не виконує код блокнотів, а тільки відображає вхід та вихід збережених ноутбуків як веб-сторінку.

Для виконання коду ноутбуку використовується сторонній веб-сервіс Binder, що відкриває середовище виконання. Щоб завантажити ноутбук в mybinder.org необхідно натиснути кнопку Execute on Binder на панелі навігації (рис. 1.19).



Рисунок 1.19 – Палітра команд nbviewer

Крім Binder код ноутбука, що завантажено на GitHub можна також відкрити в Google Colab. Для цього можна скористатися розширенням Open in Colab для Google Chrome, або ж просто додати спереду до адреси блокноту на GitHub адресу:

https://colab.research.google.com/github/шлях_до_блокноту_на_GitHub.

Наприклад, щоб відкрити блокнот https://github.com/svniko/python/blob/master/Lecture_1_ua.ipynb в Google Colab, необхідно в адресному рядку ввести https://colab.research.google.com/github/svniko/python/blob/master/Lecture_1_ua.ipynb.

2. ОСНОВИ РОБОТИ В JUPYTER NOTEBOOK

Комірки коду можуть бути використані як звичайний калькулятор:

```
In [1]: 2 + 5
Out[2]: 7
```

Або ж в них можуть бути використані оператори (рис. 2.1).

```
In [3]: for i in range(3):
        print('Hello')

Hello
Hello
Hello
```

Рисунок 2.1 – Використання операторів в комірках Jupyter Notebook

Для виведення результатів команда `print()` не є обов'язковою. Але без команди `print()` здійснюється виведення тільки значення останньої змінної або результат дії останньої команди в комірці (рис. 2.2).

```
In [5]:
a = [2**x for x in range(11)]
a

Out[5]:
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
```

Рисунок 2.2 – Виведення результату без використання команди `print()`

Однак, зверніть увагу на те, що з `print()` є мітка `Out[]` для поточної комірки, а без `print()` такої мітки немає (рис. 2.3).

```
In [8]: a=[]
        for i in range(11):
            a.append(2**i)
        a

Out[8]: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
        мітка

In [9]: a=[]
        for i in range(11):
            a.append(2**i)
        print(a)

[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
        мітка відсутня
```

Рисунок 2.3 – Наявність і відсутність мітки `Out[]` в залежності від використання команди `print()`

Змінивши тип комірок на Markdown можна створювати

розмічений текст. Наприклад, додавши у комірку текст, представлений на рис. 2.4, і, запустивши комірку, отримаємо результат, що представлений на рис. 2.5. Зверніть увагу на те, що в результаті отримуємо активні гіперпосилання, виділені синім підкресленим шрифтом.

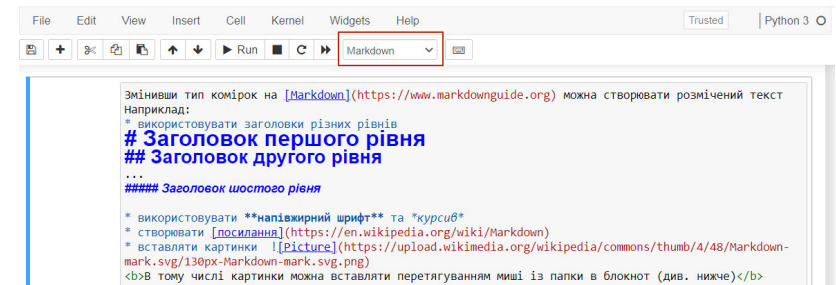


Рисунок 2.4 – Введення в комірку тексту на мові Markdown

Змінивши тип комірок на [Markdown](#) можна створювати розмічений текст
Наприклад:

- використовувати заголовки різних рівнів

Заголовок першого рівня

Заголовок другого рівня

...

Заголовок шостого рівня

- використовувати **напівжирний шрифт** та *курсив*
- створювати [посилання](#)
- вставляти картинки



В тому числі картинки можна вставляти перетягуванням миші із папки в блокнот (див. нижче)

Рисунок 2.5 – Результат рендерингу тексту, написаного на мові Markdown

Мова Markdown дозволяє легко створювати форматовані таблиці. Наступний текст в комірці Markdown:

	This		is		a head	
	-----		-----		-----	
	this		is		a row 1	
	this		is		a row 2	
	this		is		a row 3	

буде відображено як представлено на рис. 2.6.

This	is	a head
this	is	a row 1
this	is	a row 2
this	is	a row 3

Рисунок 2.6 – Результат рендерингу таблиці на мові Markdown

До комірок Markdown також можна додавати формули в форматі LaTeX.

Наприклад, додавши текст (рис. 2.7) в комірку і запустивши її на виконання, отримаємо результат, що представлено на рис. 2.8:

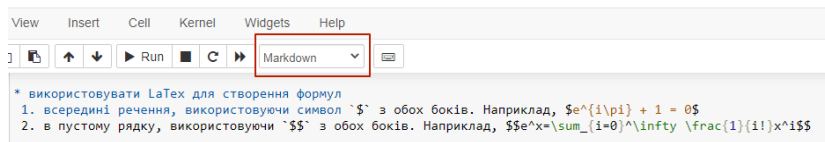


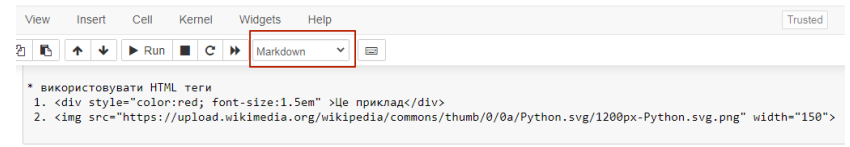
Рисунок 2.7 – Набір формул в форматі LaTeX

- використовувати LaTeX для створення формул
 1. всередині речення, використовуючи символ `$` з обох боків. Наприклад, $e^{i\pi} + 1 = 0$
 2. в пустому рядку, використовуючи `$$` з обох боків. Наприклад,

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

Рисунок 2.8 – Результат рендерингу формул в форматі LaTeX

Всередині Markdown комірки можуть також бути використані HTML теги (рис. 2.9).



а)

- використовувати HTML теги
 1. **Це приклад**
 - 2.



б)

Рисунок 2.9 – Використання HTML тегів в комітках Markdown: а) введений текст з тегами, б) результат рендерингу

Ми розглядаємо використання Jupyter Notebook для мови Python. Але є можливість запуску коду на інших мовах програмування.

Наприклад, запустивши код, позначений `%%HTML`, отримаємо виведення, відформатоване з використанням тегів HTML (рис. 2.10).

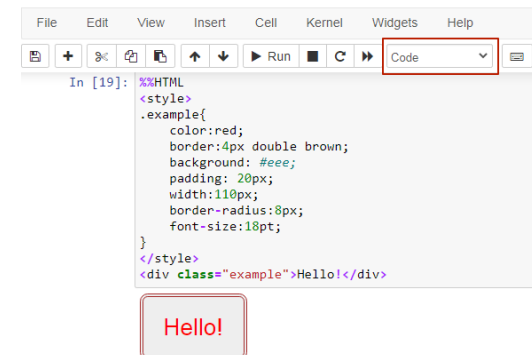


Рисунок 2.10 – Використання HTML тегів в комітках Code

Таким же чином можна вставити код, наприклад, на мові JavaScript (рис. 2.11)

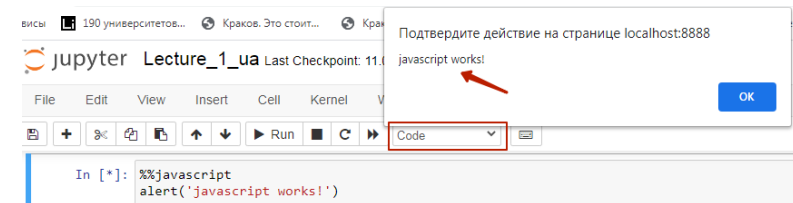


Рисунок 2.11 – Використання мови JavaScript в комітках Jupyter Notebook

Використовуючи команду `%%HTML`, можна вставляти в ноутбук рисунки, створені командами тегу `<svg>` (рис. 2.12)

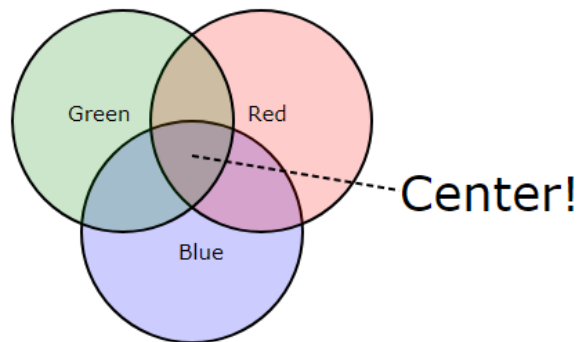
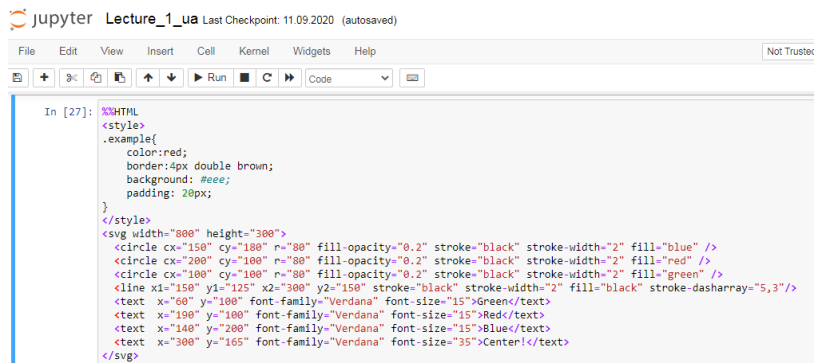


Рисунок 2.12 – Код і результат створення векторного рисунка в форматі .svg

3. ПОСТАНОВА ЗАДАЧІ

1. Встановити Anaconda.
2. Створити оточення для Python 3.
3. Налаштувати поточну активну папку.
4. За допомогою мови Markdown та HTML тегів для країни, відповідно до свого номеру варіанту (див. табл. 3.1), ввести інформацію щодо площини, населення, державного устрою, видатних персоналій, пам'яток культури та природи тощо. При оформленні інформації повинні бути використані:
 - заголовки різних рівнів;
 - різні види накреслення символів (курсив, напівжирний тощо);
 - символи різних кольорів;
 - рамки навколо тексту;
 - таблиці;
 - вкладені списки;
 - картинки за допомогою посилань та вбудовані;
 - гіперпосилання;
 - за допомогою HTML тегу `<svg>` нарисувати флаг країни.
5. Розробити функцію для реалізації алгоритму, відповідно до номеру свого варіанту з табл. 4.2. Для створеної функції реалізувати 2-3 тестові приклади в комітках ноутбуку. Для всіх варіантів організувати контроль значень, що вводяться. Наприклад:

```
In [155]: task4()
          3 4 23 5 23 str 3 4
Out[155]: 'Помилка. У списку присутнє нечислове значення'
```

Рисунок 3.1 – Виклик функції

6. Розмістити створений блокнот на GitHub.

4. ВАРІАНТИ ЗАВДАНЬ НА ЛАБОРАТОРНУ РОБОТУ

Таблиця 4.1 – Пояснення до завдання 4

Варіант	Назва країни для завдання 4
1	Греція
2	Буркіна Фасо
3	Гайана
4	Джибуті
5	Гана
6	Йорданія
7	Ісландія
8	Грузія
9	Чеська республіка
10	Норвегія

Таблиця 4.2 – Пояснення до завдання 5

Варіант	Завдання 5
1	Реалізувати гіпотезу Гольдбаха, що стверджує, що будь-яке парне число (окрім 2) може бути представлене у вигляді суми двох простих чисел. Приклади: In [1] task_1() 6 Out [1] Число 6 можна представити як 3+3 In [2] task_1() 28 Out [2] Число 28 можна представити як 5+23
2	Реалізувати алгоритму Евкліда для знаходження найбільшого спільного дільника за допомогою ділення та віднімання. Приклади: In [3] task_2() 16 36

Варіант	Завдання 5
	Out [3] Найбільший спільний дільник 16 і 36 – це 4 In [4] task_2() 12 54 Out [4] Найбільший спільний дільник 12 і 54 – це 6
3	Вводиться список цілих чисел. Знайти число, або кортеж чисел, що буде знаходитися рівно в середині списку, якщо список відсортувати. Список вводиться в рядок через пробіл. Приклади: In [5] task_3() 2 4 5 1 7 Out [5] 4 – це середина відсортованого списку In [6] task_3() 7 3 1 7 4 2 8 2 Out [6] (3, 4) – це середина відсортованого списку
4	Реалізувати сортування методом бульбашки. Підрахувати скільки разів було виконано переставлення елементів. Список вводиться в рядок через пробіл. Приклади: In [7] task_4() 5 6 23 8 12 5 Out [7] Було виконано 6 переставлень In [8] task_4() 7 3 1 7 4 2 8 2 Out [8] Було виконано 14 переставлень
5	Вивести k-те за порядком просте число. Число k вводиться під час виконання скрипта. Приклади: In [9] task_5() 5

Варіант	Завдання 5
	Out [9] 5-те просте число - це 11 In [10] task_5() 34 Out [10] 34-те просте число -це 139
6	Ввести 2 числа a і b. Знайти всі прості числа на відрізку $[a, b]$. Приклади: In [11]: task_6() 3 15 Out[12]: [3, 5, 7, 11, 13] In [13]: task_6() 6 24 Out[13]: [7, 11, 13, 17, 19, 23]
7	Реалізувати алгоритм переведення чисел (цілих та з рухомою комою) з десяткової системи числення в двійкову. Приклади: In [14]: task_7() 32 Out[14]: '32 у двійковій системі буде 0 100000' In [15]: task_7() -10.25 Out[15]: '-10.25 у двійковій системі буде 1 1010.010'
8	Реалізувати гном'яче сортування для введеного списку елементів. Вивід значень від найбільшого до найменшого. Приклад: In [16]: task_8() 5 3 78 3 23 7 34 Out[16]: [78, 34, 23, 7, 5, 3, 3]

Варіант	Завдання 5
9	У введеному списку залишити тільки непрості числа, після чого відсортувати їх за зменшенням. Приклади: In [17] task_9() 3 12 15 7 11 35 18 Out [17] [35, 18, 15, 12] In [18] task_9() 0 34 2 1 15 22 4 23 6 Out [18] [34, 22, 15, 6, 4]
10	Реалізувати алгоритм визначення чи є введене число N сумою двох квадратів натуральних чисел. Тобто визначити такі цілі числа a і b, що $a^2+b^2=N$. Якщо таких чисел не існує, вивести відповідне повідомлення. Приклади: In [19] task_10() 45 Out [19] '3^2+6^2=45' In [20] task_10() 21 Out [20] 'Число 21 не можна представити квадратами цілих чисел'

5. ЗМІСТ ЗВІТУ

1. Титульний аркуш.
2. Тема і мета роботи.
3. Лістинги кодірок.
4. Діаграму активності для завдання 5.
5. Скріншоти блокноту Jupiter з вхідними та вихідними даними.
6. Посилання на створений блокнот, розміщений на Github та відображений в nbviewer.
7. Висновок.

Контрольні запитання

1. Що являє собою Jupyter Notebook?
2. Які існують способи встановлення Jupyter Notebook?
3. Як створити окреме оточення для Jupyter Notebook?
4. За допомогою якої команди можна визвати Jupyter Notebook з відповідної папки за допомогою командного рядка?
5. Які режими існують в середовищі Jupyter Notebook?
6. Перерахуйте способи запуску комірки на виконання?
7. Як створити всередині Python Jupyter Notebook комірку з кодом на іншій мові програмування?
8. Що таке мова Markdown і для чого вона використовується?
9. Як створити формулу в новому рядку за допомогою мови LaTeX?
10. Перерахуйте способи перегляду та запуску блокноту Jupyter, розміщеного на GitHub.

СПИСОК ЛІТЕРАТУРИ

1. Project Jupyter. Режим доступу: <https://jupyter.org/>
2. Базовий синтаксис мови Markdown.
<https://www.markdownguide.org/basic-syntax/>
3. Документація по Jupyter Notebook. Режим доступу:
<https://jupyter-notebook.readthedocs.io/en/stable/notebook.html>
4. Уэс Маккинли Python и анализ данных / пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.
5. Плас Дж. Вандер Python для сложных задач: наука о данных и машинное обучение. – СПб. : Питер, 2018. – 576 с.
6. Бенгфорт Бенджамин, Билбро Ребекка, Охеда Тони Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка. – СПб. : Питер, 2019. – 368 с.
7. Навчіться, як використовувати LaTeX в Jupyter Notebook. Режим доступу: <https://towardsdatascience.com/write-markdown-latex-in-the-jupyter-notebook-10985edb91fd>