Міністерство освіти та науки України

Національний університет водного господарства та природокористування

Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

04-03-388М

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт №1-3 з дисципліни **«Машинне навчання в інтелектуальних та робототехнічних системах»** для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» денної та заочної форм навчання

Рекомендовано науково-методичною радою з якості ННІ ЕАВГ
Протокол № 8 від 23 квітня 2024 р.

Рівне – 2024

Методичні вказівки до виконання лабораторних робіт №1-3 з дисципліни «**Машинне навчання в інтелектуальних та робототехнічних системах**» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» денної та заочної форм навчання. [Електронне видання] / Сафоник А. П., Присяжнюк О. В. – Рівне : НУВГП, 2024. – 39 с.

Укладачі:
Сафоник А. П. д.т.н., професор кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій;
Присяжнюк О. В., к.т.н., доцент кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Відповідальний за випуск: Древецький В. В., д.т.н., професор, завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Керівник освітньої програми «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»: Рудик А. В., д.т.н., професор кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

Зміст

Лабораторна робота 1. Встановлення Python та Anaconda

1.1. Мета роботи

Навчитися встановлювати Python та Anaconda.

1.2. Теоретичні відомості

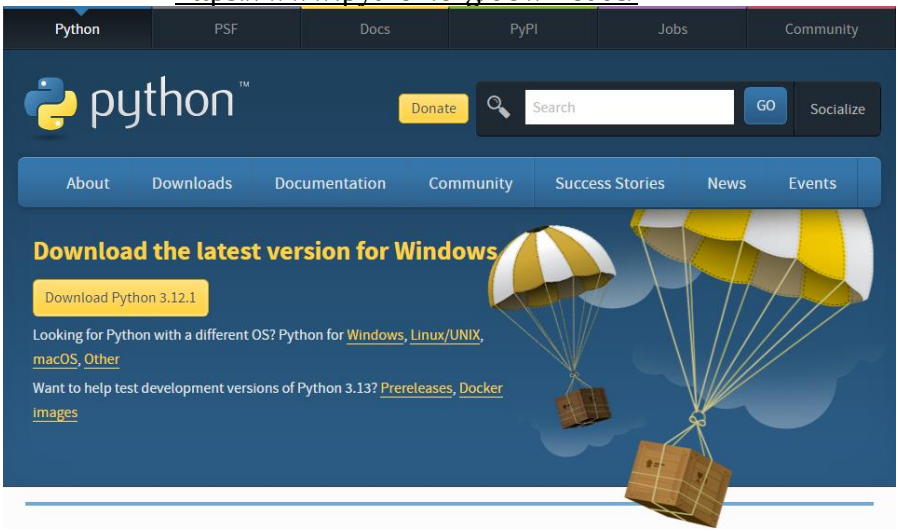
Python (найчастіше вживане прочитання – «Па́йтон», запозичено назву з британського шоу Монті Пайтон) – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Anaconda – це вільно та відкрито розповсюджуваний дистрибутив різних програмних продуктів, зокрема, мов програмування Python та R. Платформа спеціалізується на «наукових обчисленнях»: наука про дані, застосуванні методів машинного навчання, широкомасштабна обробка даних, передбачувальна аналітика тощо. Використання платформи має на меті спрощення управління пакетами та їх розгортання. Версіями пакунків керує система управління пакетами Conda.

1.3. Порядок виконання роботи

1. Встановлення Python

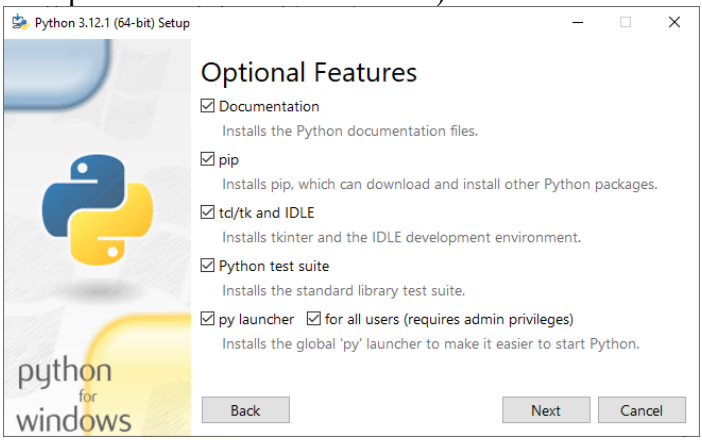
Щоб встановити інтерпретатор Python на свій комп'ютер, перше, що потрібно зробити - це завантажити дистрибутив. Завантажити його можна з офіційного сайту, перейшовши за посиланням https://www.python.org/downloads/



1. Запустіть завантажений інсталяційний файл.
2. Виберіть спосіб встановлення.

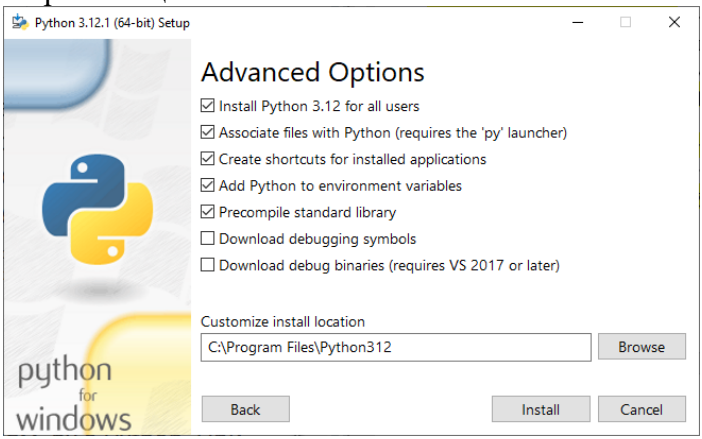
Це вікно пропонує два варіанти: «Install Now» і «Customize installation». Якщо ви виберете Install Now, Python буде встановлений в теці за вказаним шляхом. Крім самого інтерпретатора, буде встановлено IDLE (інтегроване середовище розробки), pip (менеджер пакетів) і документація, а також будуть створені відповідні ярлики і встановлені асоціації з файлами, які мають розширення .py. Customize installation - це нестандартний варіант встановлення. Параметр Add python to PATH необхідний для того, щоб дозволити інтерпретатору запускатися без вказівки повного шляху до виконуваного файлу при роботі в командному рядку.

3. Перевірте необхідні параметри встановлення (доступні, коли ви обираєте Customize installation)



На цьому кроці пропонується позначити доповнення, які будуть встановлені з інтерпретатором Python. Рекомендуємо вибрати всі варіанти.

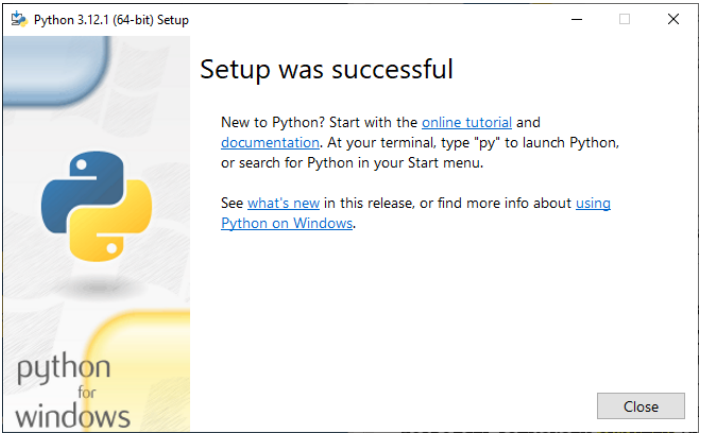
4. Виберіть місце встановлення.



Крім вказівки шляху, це вікно дозволяє внести додаткові зміни в процес встановлення за допомогою наступних опцій:

- Install for all users – Встановити для всіх користувачів. Якщо ви не виберете цей варіант, буде запропоновано встановити в папку користувача, що встановлює інтерпретатор.
- Associate files with Python – Пов'язати файли з розширенням .py з Python. Вибір цього параметра внесе зміни до Windows, які дозволять запускати скрипти Python подвійним клацанням миші.
- Create shortcuts for installed applications – Створити ярлики для запуску програм.
- Add Python to environment variables – Додати шляхи до інтерпретатора Python у змінну PATH.
- Precompile standard library – Провести прекомпіляцію стандартної бібліотеки.
- Останні два пункти пов'язані з завантаженням компонентів для налагодження, встановлювати їх не будемо.

5. Після успішного встановлення ви отримаєте наступне повідомлення.

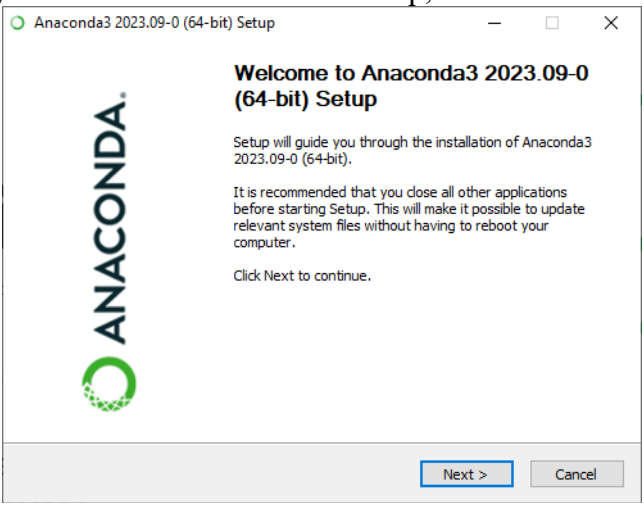


2. Встановлення Анаконди

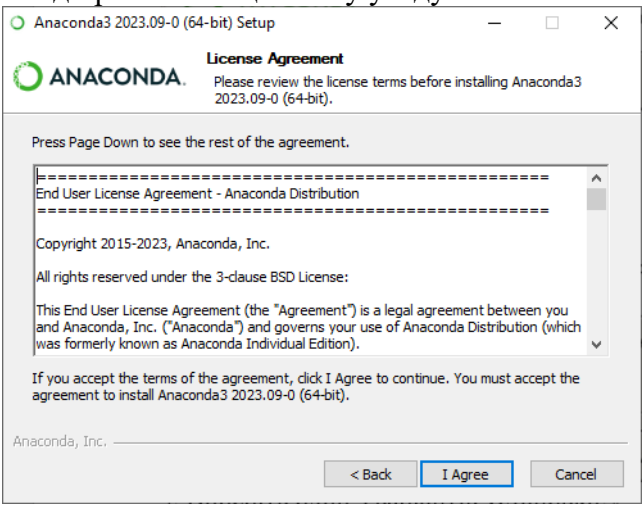
Щоб встановити пакет Anaconda, спочатку потрібно завантажити дистрибутив https://www.anaconda.com/download.

Є варіанти для Windows, Linux і macOS.

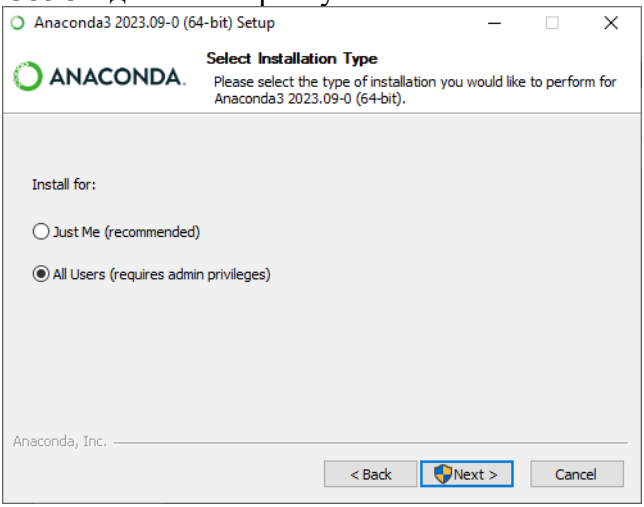
1. Запустіть завантажений інсталятор, натисніть «Next.



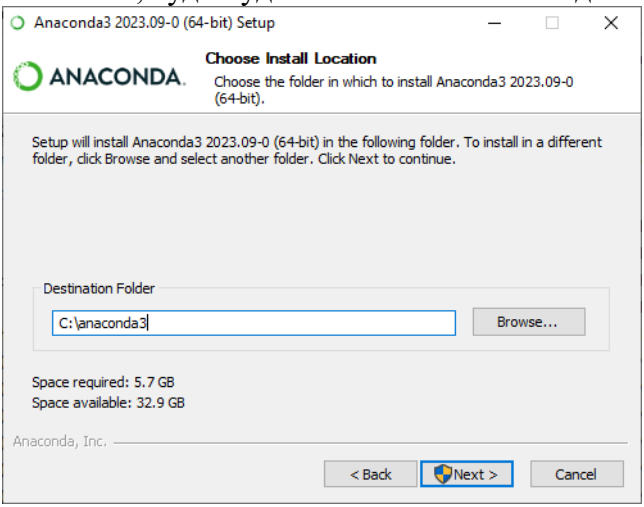
2. Далі слід прийняти ліцензійну угоду.



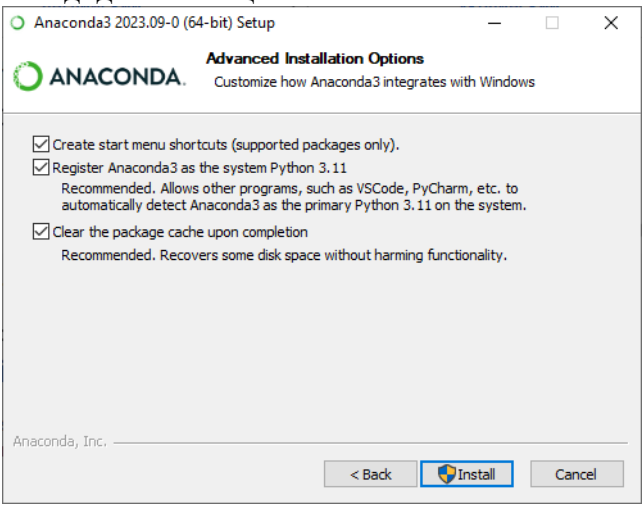
3. Виберіть один з варіантів встановлення:
   - Just Me – тільки для користувача, який почав встановлення;
   - All Users – для всіх користувачів.



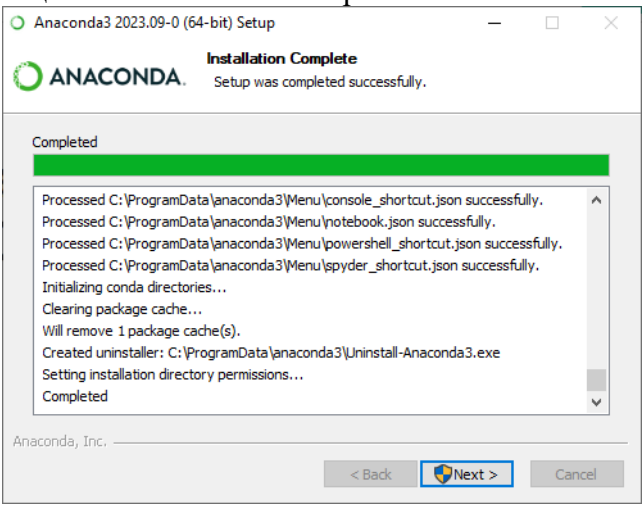
4. Вкажіть шлях, куди буде встановлено Анаконда.



5. Вкажіть додаткові опції та закінчіть встановлення



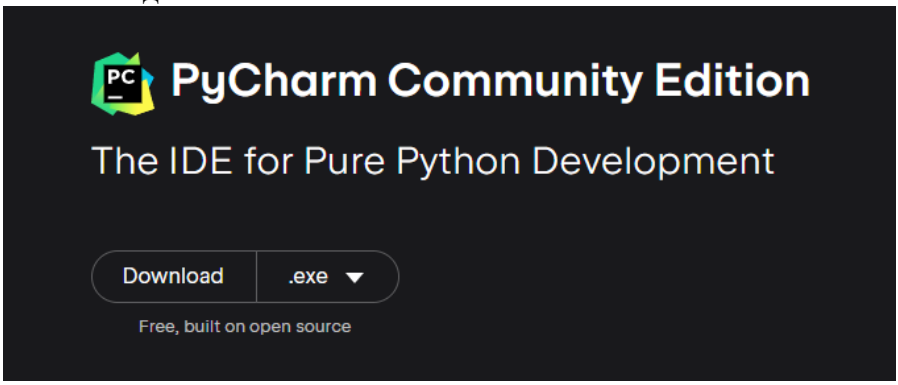
6. Після цього на ваш комп'ютер встановиться Anaconda.



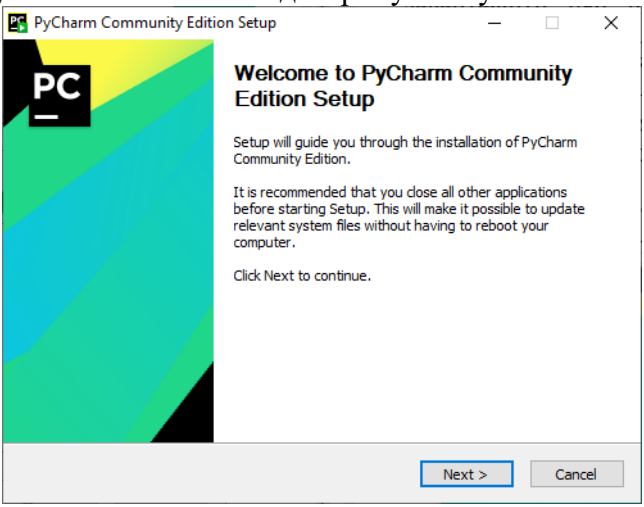
3. Встановлення PyCharm

Завантажте PyCharm з офіційного сайту за посиланням: https://www.jetbrains.com/pycharm/download/

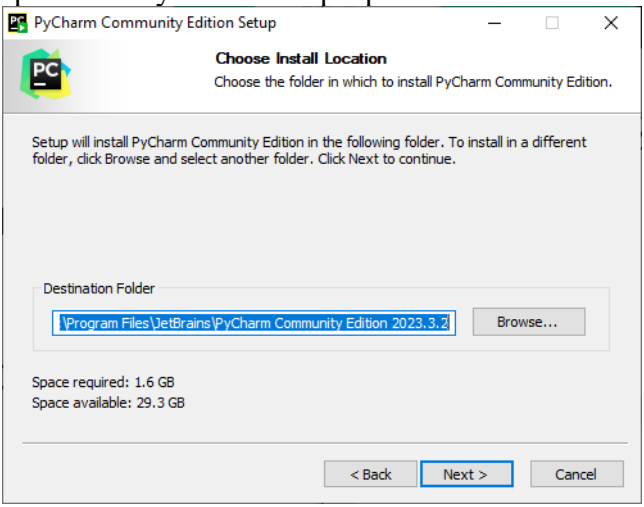
Існує два типи ліцензії PyCharm – Professional і Community Edition. Ми будемо використовувати версію Community, оскільки вона безкоштовна і її функціоналу більш ніж достатньо для наших завдань.



1. Запустіть завантажений дистрибутив PyCharm.



2. Виберіть шлях установки програми.

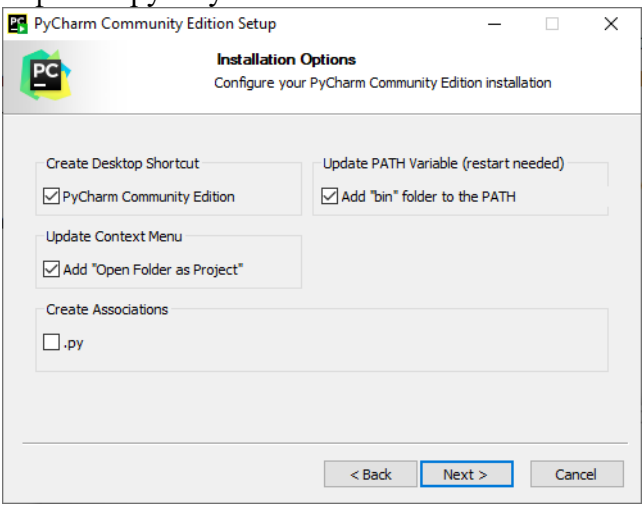


3. Вкажіть потрібні вам опції:

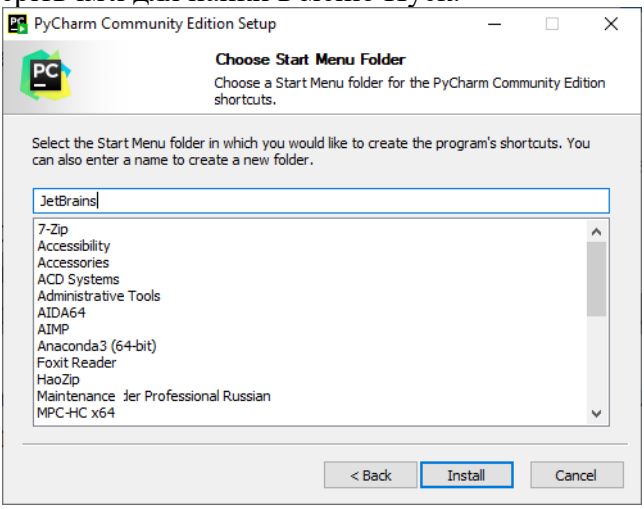
- Create Desktop Shortcut – Створити ярлик на робочому столі
- Update Context Menu: Add "Open Folder as Project" – Оновити контекстне меню, додає опцію до контекстного меню папки, яка дозволить відкрити каталог як проект Pycharm.

• Add "bin" folder to the PATH – Додати папку "bin" до PATH, дозволяє виконувати бінарні інструменти Pycharm/Python з командного рядка, просто написавши їх назву.

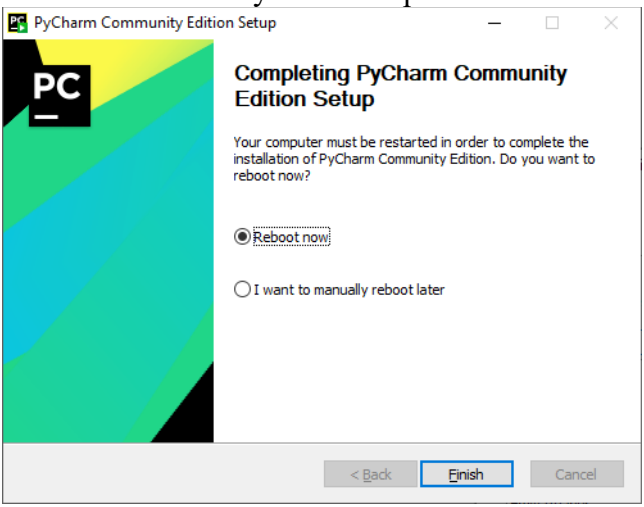
• Create associations: .py – Створити асоціації, якщо ви хочете пов'язати файли .py з PyCharm



4. Виберіть ім'я для папки в меню Пуск.



5. Якщо ви обирали опцію Add "bin" folder to the PATH, то вам запропонують перезавантажити пристрій, після чого pyCharm буде встановлено на вашому комп'ютері.



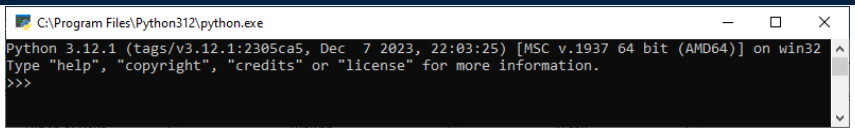
4. Перевірка стану роботи

Тепер перевіримо працездатність всього, що ми встановили.

4.1 Тестування інтерпретатора Python

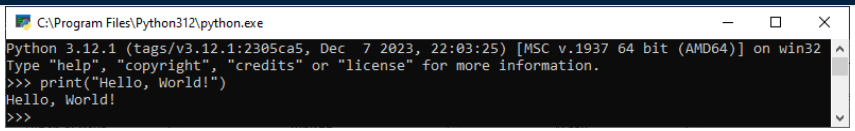
Для початку протестуємо інтерпретатор в командному режимі. Натисніть комбінацію win + R та у командному рядку введіть:

```
python
```



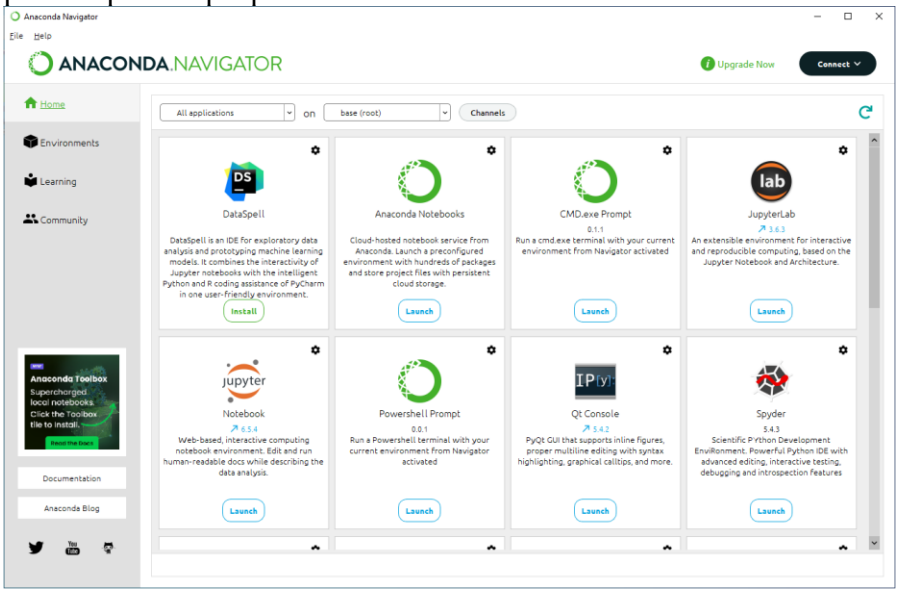
Введіть якийсь код для тесту, наприклад:

```
print("Hello, World!")
```

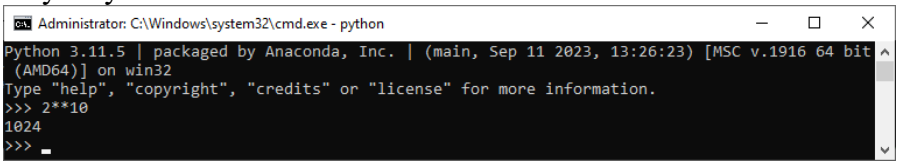


4.2 Перевірка анаконди

Відкрийте Anaconda Navigator з меню Пуск. Тут ви знайдете різні корисні програми.



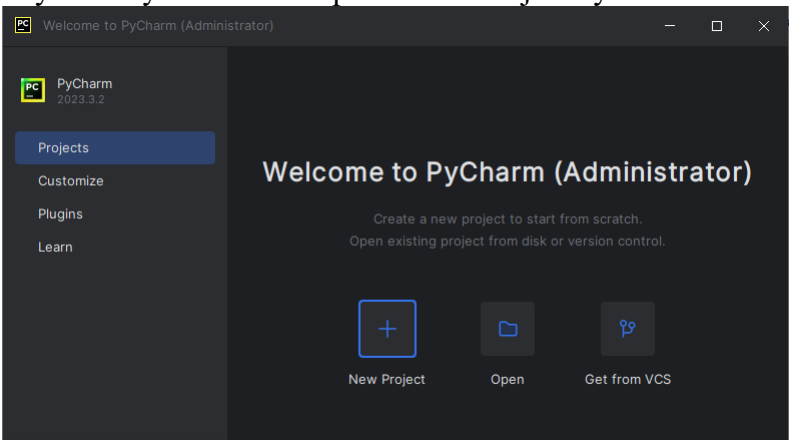
Для перевірки запустимо «CMD.exe Prompt» та повторимо дії з пункту 4.1.



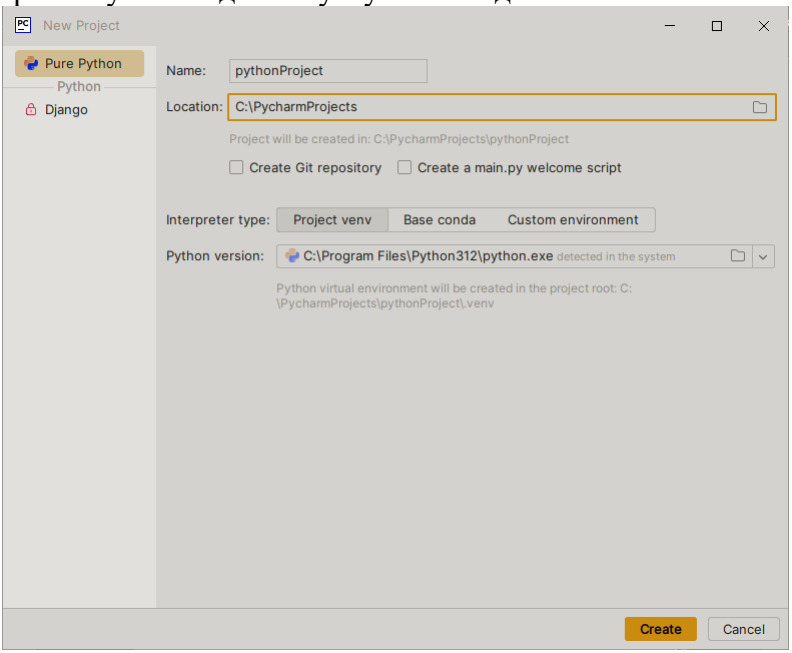
Як бачимо, тут Пайтон має іншу версію, оскільки програми в Анаконді працюють у віртуальному середовищі.

4.3 Перевірка PyCharm

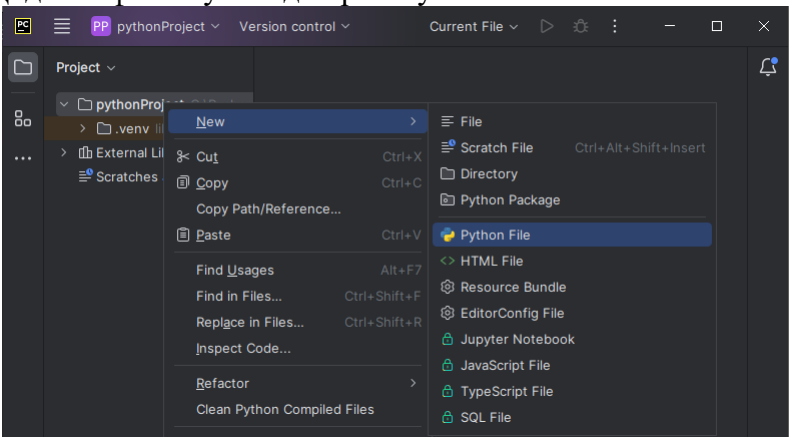
Запустіть PyCharm і виберіть «New Project» у вікні.



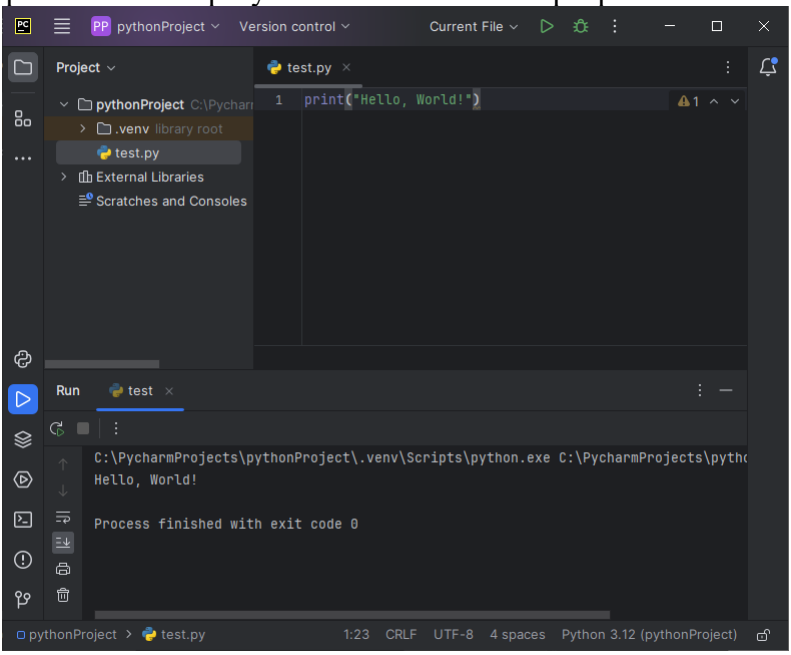
Вкажіть шлях до проекту Python і інтерпретатора, який буде використовуватися для запуску і налагодження.



Додайте файл Python до проекту.



Введіть тестовий код та запустіть скрипт. В результаті має відкритися вікно з результатом виконання програми.



1.4. Контрольні запитання

1. Що таке Python?
2. Що таке Anaconda?
3. Наведіть декілька прикладів програм, що доступні в Anaconda за замовчуванням.
4. Чому версія Python у Anaconda інша?

Лабораторна робота 2. Робота з IPython і Jupyter Notebook

2.1. Мета роботи

Навчитися працювати з IPython і Jupyter Notebook.

2.2. Теоретичні відомості

IPython надає багатий інструментарій, який допоможе вам максимально використати Python в інтерактивному режимі. Базовими компонентами IPython є потужна інтерактивна оболонка для багатого набору функцій і ядро для Jupyter. Ноутбук Jupyter - це графічна веб-обгортка для IPython, яка розширює ідею консольного підходу до інтерактивних обчислень.

Основними відмінними особливостями цієї платформи є: комплексна інтроспекція об'єктів, збереження історії введення протягом усіх сеансів, кешування результатів виведення під час сеансу з автоматично створеними посиланнями, розширювана система «чарівних» команд, журнал сеансів, розширювана обробка синтаксису, доступ до системної оболонки, інтегрований доступ до налагоджувача pdb і профайлера Python.

IPython дозволяє кільком клієнтам підключатися до одного обчислювального ядра і, завдяки своїй архітектурі, може працювати в паралельному кластері.

У Jupyter Notebook можна розробляти, документувати і виконувати додатки на Python, він складається з двох компонентів: веб-додатку, що працює в браузері, і ноутбуків - файлів, в яких можна працювати з вихідним кодом програми, запускати його, вводити і виводити дані і т.д.

Документи, створені у Jupyter Notebook називають Ноутбуками - це файли, що містять вхідні та вихідні дані інтерактивного сеансу. По суті, це запис вашої роботи, але вона дозволяє заново виконати код, присутній на ньому. Ноутбуки можна експортувати в формати PDF, HTML.

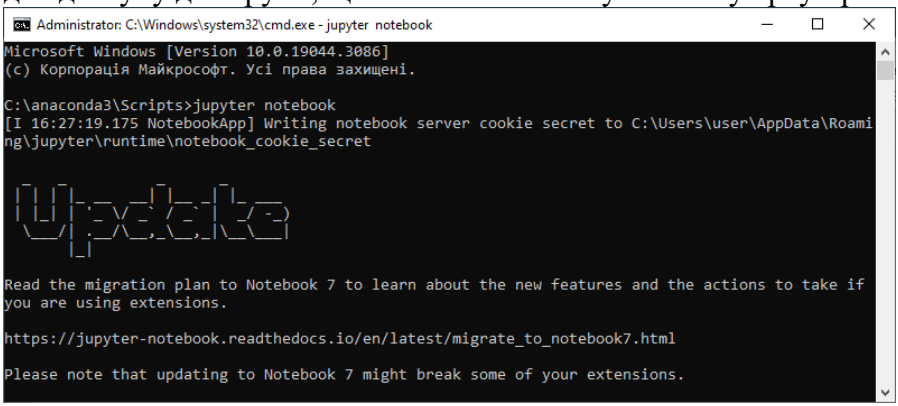
Більш детальну інформацію можна знайти у документації: https://jupyter-notebook.readthedocs.io

2.3. Порядок виконання роботи

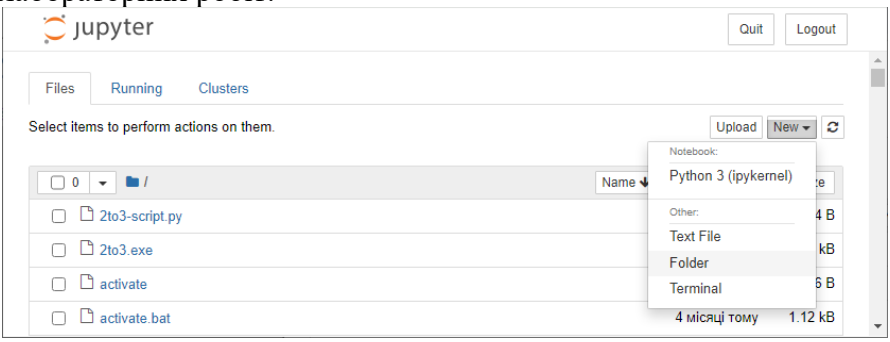
Jupyter можна запустити через Anaconda Navigator (в такому разі, стартовий каталог буде у папці користувача), або одразу через exe файл у папці C:\anaconda3\Scripts (стартовий каталог буде знаходитись у папці Scripts).

Примітка: Тут і далі будемо вважати, що пакет Anaconda встановлений в папці C:\anaconda3.

Бажано запускати jupyter-notebook.exe з правами адміністратора. Відкриється вікно, де будуть вказані посилання для доступу до Jupyter, що автоматично запуститься у браузері.



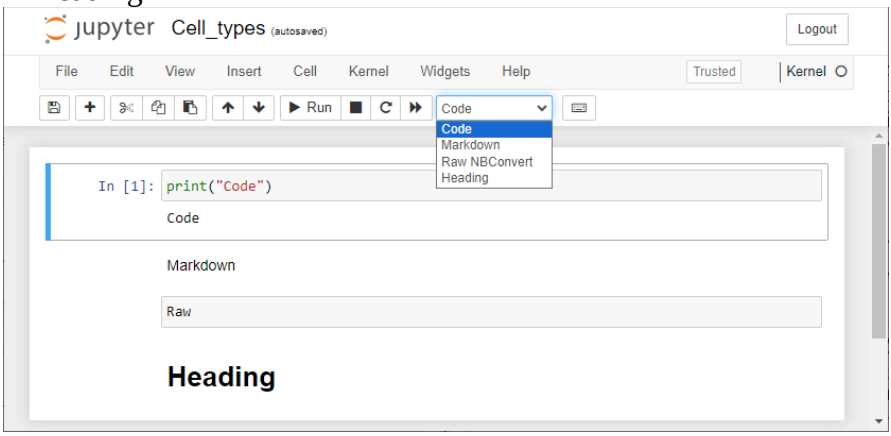
Через New у правій частині вікна створіть папку для лабораторних робіт.



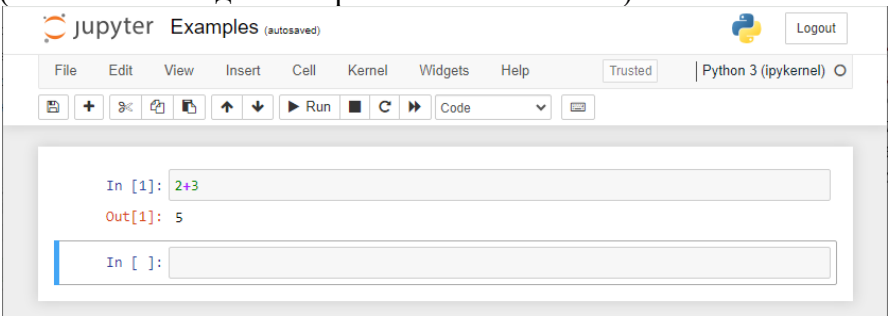
Щоб перейменувати або видалити файли та папки, потрібно виділити її та обрати відповідну дію. Перейменуйте папку на, наприклад, «Jupyter_Notebooks»

Зайдіть у неї та створіть ноутбук за допомогою тієї ж кнопки New.

Текст у ноутбуках вводиться у комірки. Вони можуть бути різних типів: якщо це код Python, то потрібно встановити властивість «Code» на панелі інструментів; для тексту розмітки - «Markdown»; необроблений текст - «Raw NBConvert»; заголовки - «Heading».



Комірки типу «Code», виконують код Python подібно до Python IDLE Shell. Для прикладу, введіть у неї «2+3» та запустіть код через Ctrl + Enter (просто виконається код) або Shift + Enter (виконається код та створиться нова клітинка).



Щоб краще зрозуміти роботу комірок, спробуйте виконати різний код у них.

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

In [2]: n = 10
        k = 5
        print(n*k)
        50

In [6]: print(n*k)
        30

In [4]: import math
        math.comb(n, k)
Out[4]: 252

In [5]: n = (int(input("n = ")))
        k = (int(input("k = ")))
        n = 3
        k = 10

In [7]: while n < k:
            n +=2
            print(n*2)
        10
        14
        18
        22
```

Магія

Важливою частиною функціоналу Jupyter Notebook є підтримка «Магії», що відноситься до додаткових команд оболонки, які спрощують процес розробки і розширюють ваші можливості. Розглянемо деякі з них.

Існує два типи магічних команд:

- **Line Magics**: діють на одному рядку та позначаються префіксом %, після якого йде аргумент без дужок і лапок.
- **Cell Magics**: діють на всю клітинку блокнота, тобто на кілька рядків. Вони позначаються подвійним префіксом %%.

Список доступних магічних команд можна отримати за допомогою команди %lsmagic. Також є схожа до неї команда %quickref, що виводить короткий опис усіх основних функцій.

```
In [1]: %lsmagic

Out[1]: Available line magics:
        %alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark
        %cd  %clear  %cls  %code_wrap  %colors  %conda  %config  %connect_info  %copy
        %ddir  %debug  %dhist  %dirs  %doctest_mode  %echo  %ed  %edit  %env  %gui  %hi
        st  %history  %killbgscripts  %ldir  %less  %load  %load_ext  %loadpy  %logoff
        %logon  %logstart  %logstate  %logstop  %ls  %lsmagic  %macro  %magic  %matplot
        lib  %mkdir  %more  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc  %pfile  %p
        info  %pinfo2  %pip  %popd  %pprint  %precision  %prun  %psearch  %psource  %pu
        shd  %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %rehashx  %reload_ex
        t  %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  %%code_wrap  %%debug  %%file  %%h
        tml  %%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.
```

%whos - виводить усі існуючі змінні.

```
In [2]: num = 3
        s = 'str'
        %whos

        Variable   Type    Data/Info
        ----------------------------
        num        int     3
        s          str     str
```

%run - дозволяє запускати скрипти Python та інші ноутбуки.

```
In [3]: %run Examples.ipynb

        50
        50
        n = 10
        k = 13
        24
        28
```

%%time і %%timeit - вимірюють час виконання коду. Можна використовувати як для всієї клітинки, так і окремого рядка.

```
In [4]: %%time
        import time
        for i in range(1000):
            time.sleep(0.01)

        CPU times: total: 15.6 ms
        Wall time: 10.2 s
```

```
In [5]: %%timeit
        import time
        for i in range(1000):
            time.sleep(0.01)

        10.8 s ± 525 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
In [6]: 1 def fibonacci(n):
        2     if n == 0 or n == 1: return n
        3     return fibonacci(n-1)+fibonacci(n-2)
        4 %timeit fibonacci(10)

        13.3 µs ± 32.2 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

%%html - відтворює клітинку як html.

```
In [7]: 1 %%html
        2 <font size=5 color='blue'>HTML in Jupyter!</font>

          HTML in Jupyter!
```

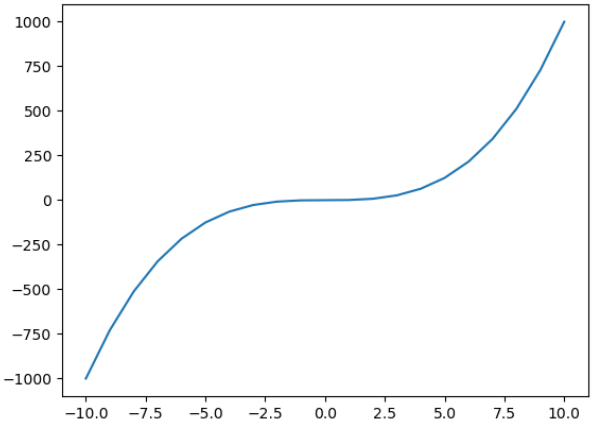
%%javascript або %%js - запускає клітинку як код Javascript.

```
In [8]: 1 %%javascript
        2 element.text("Hello, world!");

        Hello, world!
```

Відображення зображень у ноутбуці

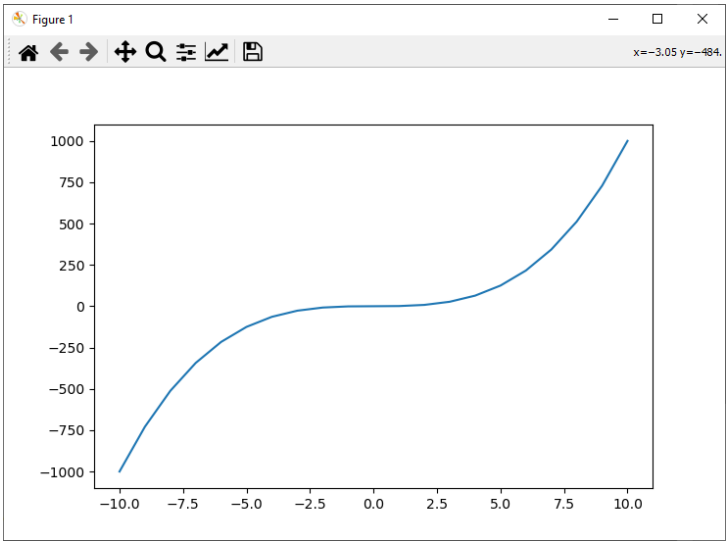
Магічні команди також використовуються і при роботі з зображеннями. Так наприклад %matplotlib з різними аргументами, буде по різному відображати графіки що побудовані за допомогою бібліотеки matplotlib. За замовчуванням, при використанні pylab, діаграми виводяться у клітинці простою картинкою:

```
In [1]: import matplotlib.pylab as plt
        x = [i for i in range(-10, 11)]
        y = [i**3 for i in range(-10, 11)]
        plt.plot(x,y)

Out[1]: [<matplotlib.lines.Line2D at 0x26a40918a10>]
```

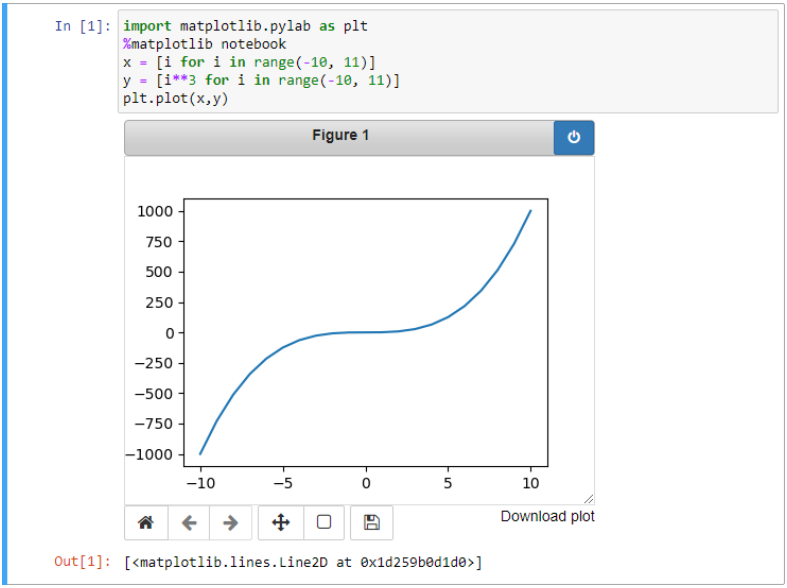


Щоб вивести графік в окремому вікні та мати змогу взаємодіяти з ним, можна використати - %matplotlib qt:

```
In [1]: import matplotlib.pylab as plt
        %matplotlib qt
        x = [i for i in range(-10, 11)]
        y = [i**3 for i in range(-10, 11)]
        plt.plot(x,y)

Out[1]: [<matplotlib.lines.Line2D at 0x24ad7398210>]
```



А щоб взаємодіяти з графіком одразу в клітинці - %matplotlib notebook:



Знайдіть у документації ще декілька магічних команд, та наведіть приклади роботи з ними, коротко пояснивши їх роботу.

2.4. Контрольні запитання

1. Що таке Jupyter Notebook?
2. За яким посиланням можна відкрити Jupyter у браузері?
3. Як виконати код у комірці?
4. Які ви знаєте типи клітинок?
5. В чому відмінність між магічними командами що починаються з % та %%?

Лабораторна робота 3. Лінійна регресія

3.1. Мета роботи

Набути навичок використовувати лінійну регресію для практичних задач.

3.2. Теоретичні відомості

Є множина об'єктів $\mathbb{X}$, і кожному з них потрібно призначити якесь значення. Наприклад, є набір операцій на банківській картці, і необхідно зрозуміти, які з цих операцій були здійснені шахраями. Якщо розділити всі операції на два класи і «нулем» позначити законні дії, а «одиницею» шахрайські, то вийде найпростіше завдання класифікації. Інший приклад: є дані геологорозвідки і потрібно оцінити перспективи різних родовищ. В цьому випадку, використовуючи набір геологічних даних, ваша лінійна модель дозволить, наприклад, оцінити потенційну річну рентабельність шахти. Це приклад задачі регресії. Числа, яким ми хочемо зіставити об'єкти з нашої множини, іноді називаються цілями, або таргетами (від англійського **target**).

Таким чином, задачі класифікації та регресії можна сформулювати як пошук відображення від сукупності об'єктів $\mathbb{X}$ до набору можливих цілей.

Математично проблеми можна описати так:

- **Класифікація**: $X \to \{0,1,\dots,K\}$, де $0,\dots,K$ - номери класів.
- **Регресія**: $X \to R$.

Очевидно, що просто зіставляти якісь об'єкти з якимись числами - справа досить безглузда. Ми хочемо швидко виявити шахраїв або вирішити, де будувати шахту. Тому потрібен якийсь критерій якості. Потрібно знайти відображення, яке найкраще апроксимує справжню відповідність між об'єктами та цілями. Можливих відображень може бути багато, але ми можемо спростити собі завдання і погодитися з тим, що хочемо шукати рішення тільки в якомусь заданому параметризованому сімействі функцій. Найпростіше таке сімейство - лінійні функції виду:

$$y = w_1 x_1 + \dots + w_D x_D + w_0,$$

де y - цільова змінна (**target**), $(x_1, \dots, x_D)$ - вектор, що відповідає об'єкту вибірки (**вектор ознак**), а $w_1, \dots, w_D, w_0$ є параметрами моделі.

Ознаки ще називають фічами (від англійської **features**). Вектор $w = (w_1, \dots, w_D)$ часто називають вектором ваг, оскільки передбачення моделі можна розглядати як зважену суму ознак об'єкта, а число w_0 є вільним коефіцієнтом, або зміщенням. Більш компактно лінійну модель можна записати як:

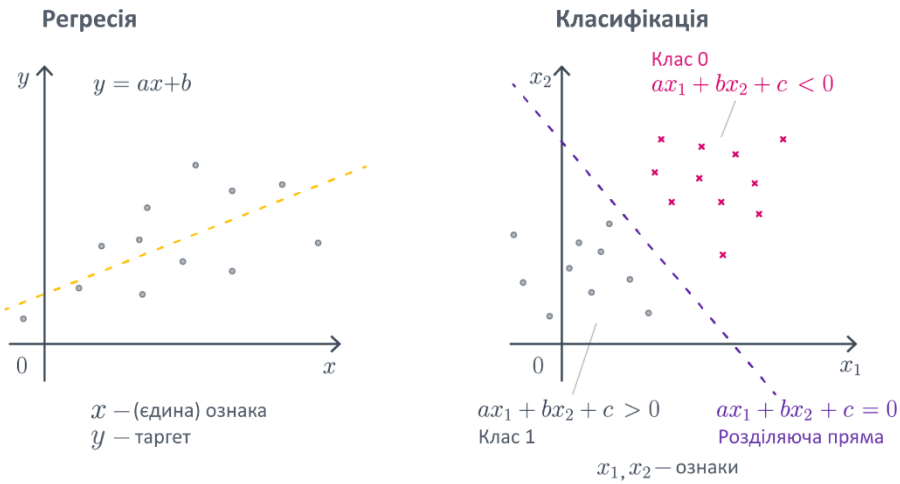
$$y = \langle x, w \rangle + w_0$$

Тепер, коли ми вибрали сімейство функцій, в якому будемо шукати рішення, завдання стало набагато простішим. Ми вже не шукаємо якесь абстрактне відображення, а конкретний вектор:

$$(w_0, w_1, \dots, w_D) \in R^{D+1}$$

Примітка: Для застосування лінійної моделі необхідно, щоб кожен об'єкт вже був представлений вектором числових ознак $x_1, \dots, x_D$. Звичайно, в лінійну модель просто не можна ставити текст або графік, спочатку доведеться придумати для нього числові ознаки. Модель називається лінійною, якщо вона лінійна за цими числовими ознаками.

Подивимося, як буде працювати така модель, якщо $D = 1$. Тобто наші об'єкти мають лише одну числову ознаку, за якою вони відрізняються. Тепер наша лінійна модель буде виглядати досить просто: $y = w_1 x_1 + w_0$. Для задачі регресії ми зараз намагаємося апроксимувати значення y якоюсь лінійною функцією від змінної x. А що означатиме лінійність для задачі класифікації? Згадаємо приклад пошуку шахрайських операцій по картах. Скажімо, ми точно знаємо одну числову змінну - обсяг транзакції. Для бінарної класифікації транзакцій на законні і потенційно шахрайські, будемо шукати так зване **правило поділу**: там, де значення функції позитивне, ми будемо прогнозувати один клас, де негативне - інший. У нашому прикладі найпростішим правилом буде якесь порогове значення обсягу транзакцій, після якого можна позначити транзакцію як підозрілу.



У разі більш високих розмірностей замість прямої буде гіперплощина з аналогічним сенсом.

Питання, над яким варто задуматися. А раптом одна з ознак *категоріальна*, тобто приймає значення з (зазвичай скінченного числа) значень, які не є числами? Наприклад, це може бути пора року, рівень освіти, марка машини і так далі. Як правило, з такими значеннями неможливо виконувати арифметичні операції або результати їх застосування не мають сенсу.

В такому разі потрібно якось закодувати ознаку. Для прикладу візьмемо такий набір даних. Тут є дві такі ознаки - **pet_type** та **color**. Перша приймає 4 різні значення, друга – 5.

	pet_type	color	weight
0	carrot	red	0.600000
1	cat	white	3.000000
2	hamster	brown	0.800000
3	cat	gray	5.000000
4	dog	black	7.000000

Найпростіший спосіб закодувати ознаки через **one-hot encoding**. Це замінить кожну категоріальну ознаку на декілька бінарних, що приймають значення «0» або «1». Таким чином замість категоріальної ознаки **pet_type** ми отримаємо 4 бінарні, що фактично відповідають на питання «ознака приймає це значення?»:

	weight	pet_type_carrot	pet_type_cat	pet_type_dog	pet_type_hamster	color_black	color_brown	color_gray	color_red	color_white
0	0.600000	1	0	0	0	0	0	0	1	0
1	3.000000	0	1	0	0	0	0	0	0	1
2	0.800000	0	0	0	1	0	1	0	0	0
3	5.000000	0	1	0	0	0	0	1	0	0
4	7.000000	0	0	1	0	1	0	0	0	0

При використанні даного методу зазвичай прибирають одну з бінарних ознак, оскільки якщо цього не зробити, сума бінарних ознак створених з однієї категоріальної, дасть стовпець одиниць, що погано впливає на цільову змінну та може призвести до перенавчання моделі.

	weight	pet_type_cat	pet_type_dog	pet_type_hamster	color_brown	color_gray	color_red	color_white
0	0.600000	0	0	0	0	0	1	0
1	3.000000	1	0	0	0	0	0	1
2	0.800000	0	0	1	1	0	0	0
3	5.000000	1	0	0	0	1	0	0
4	7.000000	0	1	0	0	0	0	0

Крім простоти, лінійні моделі мають ще ряд переваг. Наприклад, ми можемо досить легко судити про те, як ті чи інші ознаки впливають на результат. Так, якщо вага w_i позитивна, то з ростом i-тої ознаки таргет в разі регресії збільшиться, а в разі класифікації наш вибір зміститься на користь одного з класів. Значення ваг також має прозору інтерпретацію: чим більше вага w_i, тим «важливіше» i-та ознака для остаточного прогнозу. Тобто, якщо ви побудували лінійну модель, ви цілком можете пояснити замовнику деякі її результати. Якість моделей називається інтерпретованістю. Особливо вона цінується в промислових задачах, в яких ціна помилки висока. Якщо життя людини може залежати від роботи вашої моделі, то дуже важливо

зрозуміти, як модель приймає ті чи інші рішення і якими принципами керується. В той же час не всі методи машинного навчання добре інтерпретуються, наприклад, поведінка штучних нейронних мереж або градієнтний бустинг досить складно інтерпретувати.

При цьому сліпо довіряти вагам лінійних моделей також не варто по ряду причин:

- Лінійні моделі - це ще досить вузький клас функцій, вони добре працюють для невеликих наборів даних і простих завдань. Однак якщо ви вирішуєте більш складну задачу з лінійною моделлю, то, швидше за все, доведеться придумувати додаткові можливості, які є складними функціями від початкових. Пошук таких додаткових можливостей називається feature engineering. Але пошуком таких штучних ознак можна дуже захопитися, так що осмисленість інтерпретації буде сильно залежати від здорового глузду експерта, який побудував модель.
- При наявності приблизної лінійної залежності між ознаками коефіцієнти в лінійній моделі можуть повністю втратити свій фізичний сенс.
- Особливо обережно варто відноситись до тверджень виду «цей коефіцієнт маленький, отже, ця ознака не важлива». По-перше, все залежить від масштабу ознаки: раптом коефіцієнт малий, щоб компенсувати його. По-друге, залежність справді може бути слабкою, але хто знає, в якій ситуації вона виявиться важливою. Такі рішення приймаються на основі даних, наприклад шляхом перевірки статистичного критерію.
- Конкретні значення ваг можуть змінюватися в залежності від навчальної вибірки, хоча зі збільшенням її розміру вони будуть потихеньку сходитися до ваг «кращої» лінійної моделі, яку можна було б побудувати за всіма даними на світі.

3.3. Порядок виконання роботи

Імпортуємо бібліотеку pandas, що призначена для роботи з даними та для прикладу завантажуємо датасет у якому представлена вибірка з ростом і вагою 10 000 студентів чоловічої та жіночої статі. Переглянемо перші 5 елементів:

```
import pandas as pd
data =
pd.read_csv("https://gist.githubusercontent.com/nstokoe/7d4717e96c21b8ad04ec91f361b000cb/raw/bf95a2e30fceb9f2ae990eac8379fc7d844a0196/weight-height.csv")
data.head()
```

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

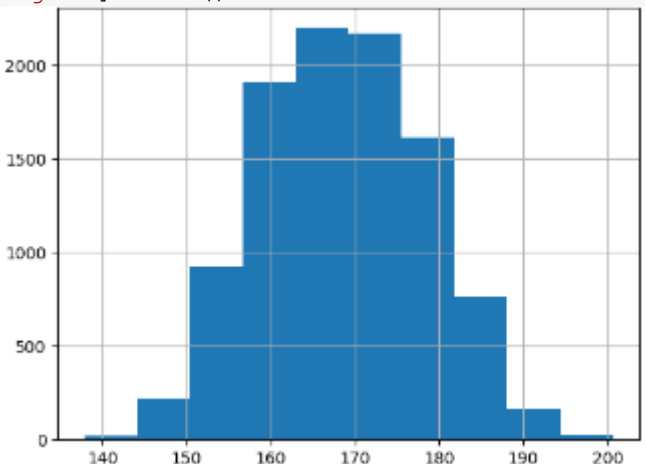
Переведемо дюйми у сантиметри, а фунти у кілограми

```
data['Height']*=2.54
data['Weight']*=0.4536
data.head()
```

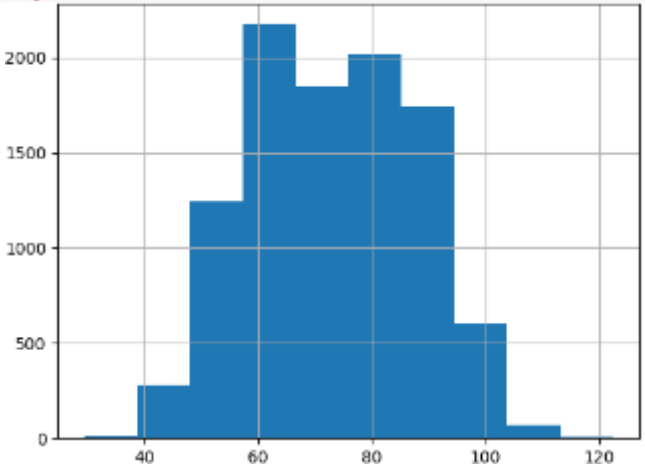
	Gender	Height	Weight
0	Male	187.571423	109.722920
1	Male	174.706036	73.624030
2	Male	188.239668	96.499252
3	Male	182.196685	99.811265
4	Male	177.499761	93.600270

Переглянемо розподіл даних:

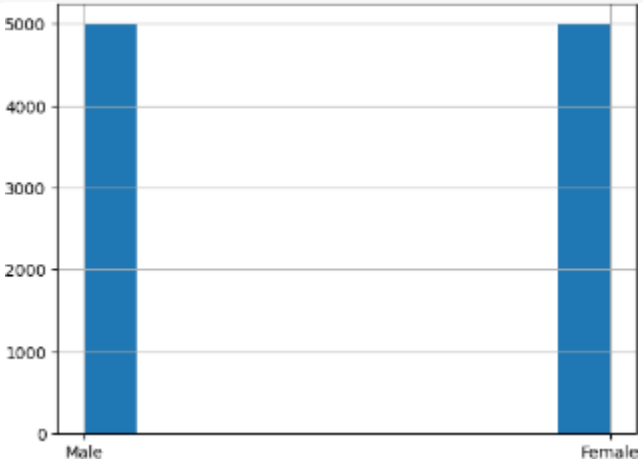
```
data['Height'].hist()
```




```
data['Weight'].hist()
```



```
data['Gender'].hist()
```



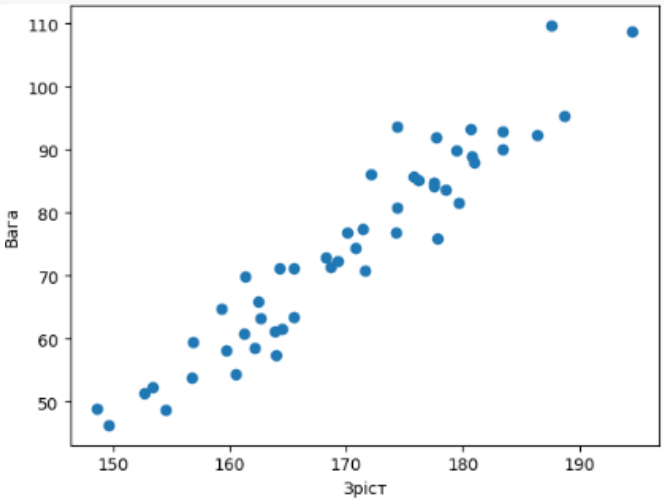
Оскільки дані мають забагато елементів, скоротимо вибірку, взявши кожен 200+N-ий елемент (де N - номер варіанту) та виведемо структуру даних

```
N = 0
Df = data[::N+200]


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 9800
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Gender  50 non-null     object 
 1   Height  50 non-null     float64
 2   Weight  50 non-null     float64
dtypes: float64(2), object(1)
memory usage: 1.3+ KB
```

Тепер можна перейти до задання входу і виходу моделі та побудови точкового графіку їх залежності:

```
X = df[['Height']]
y = df['Weight']

import matplotlib.pyplot as plt
plt.scatter(X, y)
plt.xlabel("Зріст")
plt.ylabel("Вага")
plt.show()
```



Створимо модель парної лінійної регресії виду $y = wx + b$

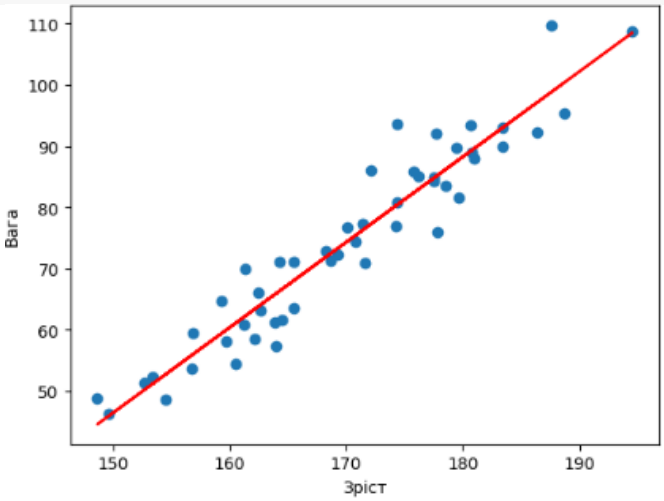
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X, y)
w = model.coef_
b = model.intercept_
w,b
```

```
(array([1.39252407]), -162.42219744720774)
```

$y = 1.393x - 162.422$, де x - зріст в см, y - вага в кг.

Знайдемо передбачення моделі та нарисуємо графік вихідних даних і побудованої моделі:

```
y_pred = model.predict(X)

X = df[['Height']]
y = df['Weight']

plt.scatter(X, y)
plt.plot(X, y_pred,'red')

plt.xlabel("Зріст")
plt.ylabel("Вага")
plt.show()
```



Перевіримо точність побудованої моделі:

```
model.score(X,y)
```

```
0.9095341533562858
```

Позначимо класи змінної Gender як 0 і 1 та побудуємо множинну регресію виду $y = w_1x_1 + w_2x_2 + b$:

```
X1 = df.drop(['Weight'], axis=1)
X1['Gender'] = pd.factorize(X1['Gender'])[0]

model1 = LinearRegression()
model1.fit(X1, y)
w = model1.coef_
b = model1.intercept_
w,b
```

```
(array([-6.25690744,  1.18517264]), -124.03451070427762)
```

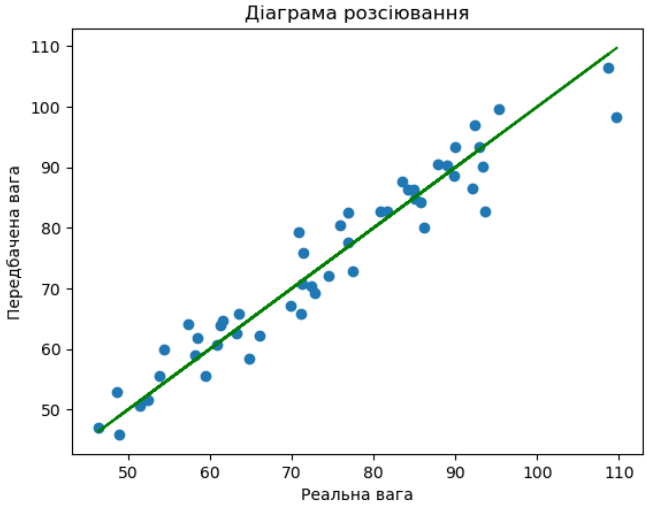
$y = -6.257x_1 + 1.185x_2 - 124.035$, де x_1 - стать, x_2 - зріст в см, y - вага в кг.

Перевіримо точність новопобудованої моделі, знайдемо її передбачення та нарисуємо діаграму розсіювання вихідних та передбачених значень ваги:

```
model1.score(X1,y)
```

```
0.9294561135335533
```

```
y_pred1 = model1.predict(X1)

plt.scatter(y, y_pred1)
plt.plot(y, y,'green')

plt.xlabel("Реальна вага")
plt.ylabel("Передбачена вага")
plt.title('Діаграма розсіювання')
plt.show()
```



3.4. Контрольні запитання

1. Що таке Лінійна регресія та Класифікація?
2. Наведіть функцію лінійної регресії.
3. Який метод можна використати для обробки категоріальних ознак? Поясніть принцип роботи.
4. Які ознаки були визначені для навчання моделі?
5. Що показує графік розсіювання?