

# Посібник з програмування на Python

- Розділ 1. Введення в Python
  - Мова програмування Python
  - Встановлення та перша програма на Windows
  - Встановлення та перша програма на Mac OS
  - Встановлення та перша програма на Linux
  - Керування версіями Python на Windows, MacOS та Linux
  - Перша програма в PyCharm
  - Python у Visual Studio
- Розділ 2. Основи Python
  - Введення у написання програм
  - Змінні та типи даних
  - Консольне введення та виведення
  - Арифметичні операції з числами
  - Порозрядні операції з числами
  - Умовні вирази
  - Умовна конструкція if
  - Цикли
  - Функції
  - Параметри функції
  - Оператор return та повернення результату з функції
  - Функція як тип, параметр та результат іншої функції
  - Лямбда-вирази
  - Перетворення типів
  - Область видимості змінних
  - Замикання
  - Декоратори
- Розділ 3. Об'єктно-орієнтоване програмування
  - Класи та об'єкти
  - Інкапсуляція, атрибути та властивості
  - успадкування
  - Перевизначення функціоналу базового класу
  - Атрибути класів та статичні методи
  - Клас об'єкта. Строкове представлення об'єкта
  - Перевантаження операторів
  - Абстрактные классы и методы
- Розділ 4. Обробка помилок та винятків

- Конструкція try...except...finally
- except та обробка різних типів винятків
- Генерація винятків та створення своїх типів винятків
- Розділ 5. Списки, кортежі та словники
  - Список
  - Кортежі
  - Діапазони
  - Словники
  - Безліч
  - List comprehension
  - Упаковка та розпакування
  - Упаковка та розпакування у параметрах функцій
- Розділ 6. Модулі
  - Визначення та підключення модулів
  - Генерація байткоду модулів
  - Модуль random
  - Математичні функції та модуль math
  - Модуль locale
  - Модуль decimal
  - Модуль dataclass. Data-класи
- Розділ 7. Рядки
  - Робота з рядками
  - Основні методи рядків
  - Форматування
- Розділ 8. Pattern matching
  - Конструкція match
  - Кортежі в pattern matching
  - Массивы в pattern matching
  - Словники в pattern matching
  - Класи в pattern matching
  - guards або обмеження шаблонів
  - Установка псевдонимів и паттерн AS
- Розділ 9. Робота з файлами
  - Відкриття та закриття файлів
  - Текстові файли
  - Файли CSV
  - Бинарные файлы
  - Модуль shelve
  - Модуль OS та робота з файловою системою

- Програма підрахунку слів
- Запис та читання архівних zip-файлів
- Розділ 10. Робота з датами та часом
  - Модуль datetime
  - Операции с датами

## Розділ 1. Введення в Python

### Мова програмування Python

Python представляє популярну високорівневу мову програмування, яка призначена для створення додатків різних типів. Це і веб-програми, і ігри, і настільні програми, і робота з базами даних. Досить велике поширення пітон отримав у галузі машинного навчання та досліджень штучного інтелекту.

Вперше мова Python була анонсована 1991 року голландським розробником Гвідо Ван Россумом. З того часу ця мова пройшла великий шлях розвитку. У 2000 році було видано версію 2.0, а в 2008 році - версію 3.0. Незважаючи на такі великі проміжки між версіями постійно виходять підверсії. Так, поточною актуальною версією на момент написання цього матеріалу є 3.13, яка вийшла у жовтні 2024 року.

Основні особливості мови програмування Python:

- Скриптова мова. Код програм визначається як скриптів.
- Підтримка різних парадигм програмування, у тому числі об'єктно-орієнтованої та функціональної парадигм.
- Інтерпретація програм. Для роботи зі скриптами необхідний інтерпретатор, який запускає та виконує скрипт.

Виконання програми на Python виглядає так. Спочатку ми пишемо в текстовому редакторі скрипт з набором виразів цією мовою програмування. Передаємо цей скрипт виконання інтерпретатору. Інтерпретатор трансліює код у проміжний байткод, а потім віртуальна машина переводить отриманий байткод на набір інструкцій, що виконуються операційною системою.

Тут варто зазначити, що хоча формально трансляція інтерпретатором вихідного коду в байткод і переведення байткоду віртуальною машиною в набір машинних команд представляють два різні процеси, але фактично вони об'єднані в інтерпретаторі.



- Портативність та платформонезалежність. Не має значення, яка у нас операційна система – Windows, Mac OS, Linux, нам достатньо написати скрипт, який запускатиметься на всіх цих ОС за наявності інтерпретатора
- Автоматичне керування пам'яті
- Динамічна типізація

Python - дуже проста програма, вона має короткий і в той же час досить простий і зрозумілий синтаксис. Відповідно його легко вивчати, і власне це одна з причин, через яку він є однією з найпопулярніших мов програмування саме для навчання. Зокрема, у 2014 році він був визнаний найпопулярнішою мовою програмування для навчання у США.

Python також популярний не тільки у сфері навчання, а й у написанні конкретних програм у тому числі комерційного характеру. Немалою мірою тому для цієї мови написано багато бібліотек, які ми можемо використовувати.

Крім того, у цієї мови програмування дуже велике співтовариство програмістів, в інтернеті можна знайти з цієї мови безліч корисних матеріалів, прикладів, отримати кваліфіковану допомогу фахівців.

## Пакети та бібліотеки

Інтерпретатор Python супроводжується достатнім функціоналом, який дозволяє створювати програми цією мовою. Проте цього функціоналу може виявитися замало низки завдань. Але через велику спільноту розробників цією мовою по всьому світу також є велика екосистема різних пакетів і бібліотек, які можна використовувати для

різних цілей. У [розділі мови Python](#) на сайті [METANIT.COM](#) будуть розглянуті деякі з цих бібліотек. Перелічу основні їх.

Для створення графічних програм:

- [Tkinter](#)
- PyQt/PySide
- wxPython
- DearPyGui
- EasyGUI

Для створення мобільних додатків:

- Kivy
- Toga

Для створення веб-застосунків:

- [Django](#)
- Flask
- [FastAPI](#)
- Pylons
- Bottle
- CherryPy
- TurboGears
- Nagare

Для автоматизації процесів:

- Selenium (для тестування веб-додатків)
- Flask
- FastAPI
- Pylons
- Bottle
- CherryPy
- TurboGears
- Nagare
- robotframework
- pywinauto
- Lettuce
- Behave
- Requests

Для роботи з різними типами файлів:

- OpenPyXL (Excel)
- lxml (XML)
- ReportLab/borb (PDF)
- pdfcrow / PyPDF2 (PDF)
- Pandas (CSV та Excel)

Для машинного навчання, штучного інтелекту, Data Science:

- Pandas
- SciPy
- PyTorch
- Matplotlib
- Theano
- Tensorflow
- OpenCV
- Scikit-Learn
- Keras
- NumPy

Для візуалізації:

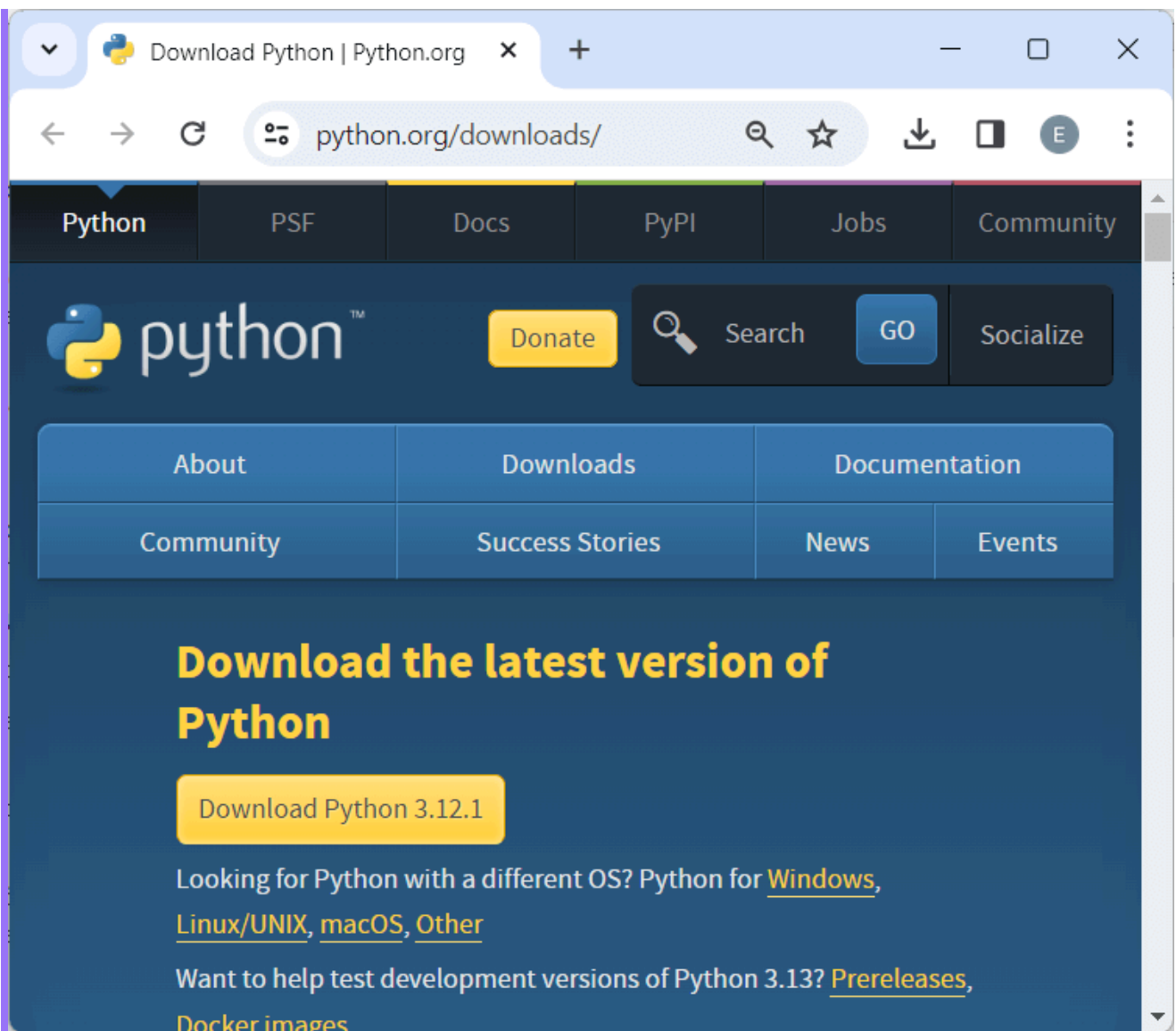
- Matplotlib
- Seaborn
- Plotly
- Bokeh
- Altair
- HoloViews

Встановлення та перша програма на Windows

## Встановлення

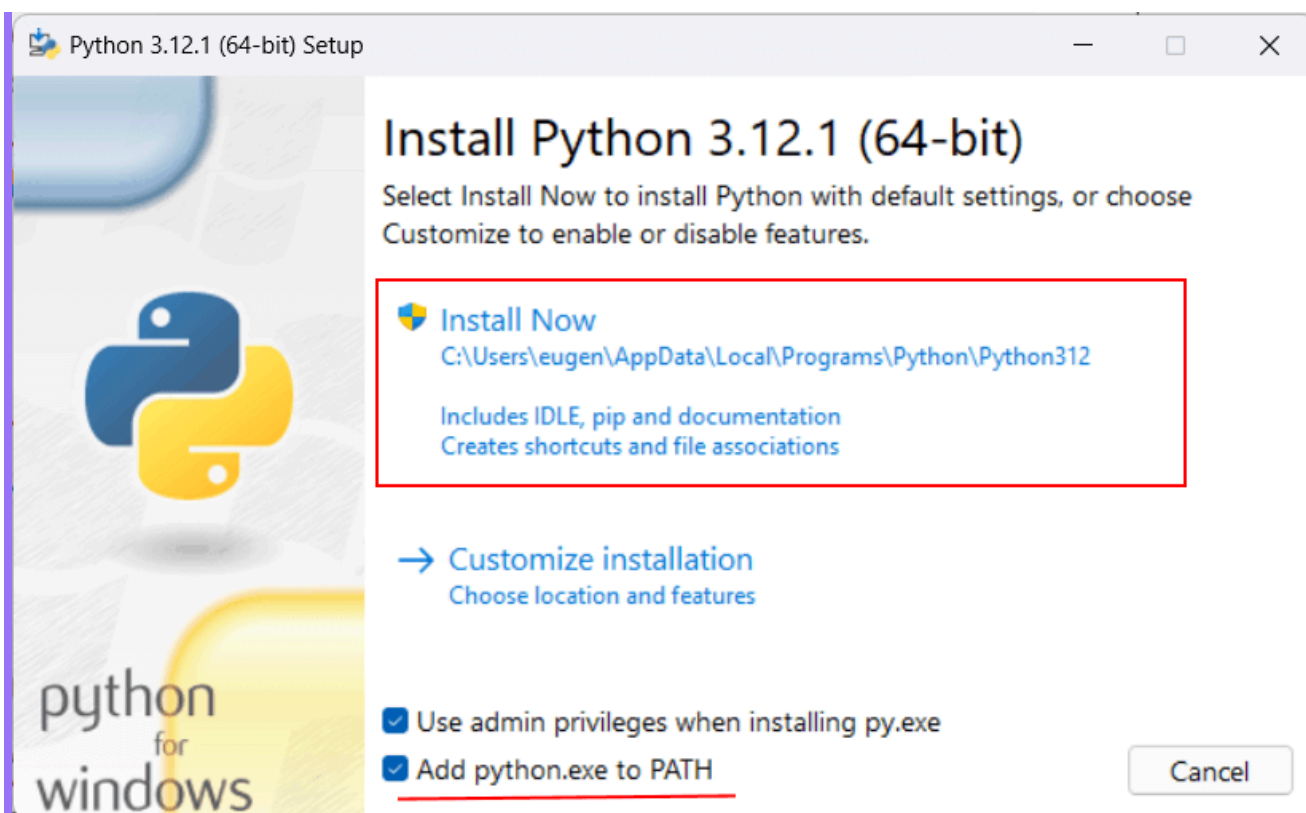
Для створення програм на Python нам знадобиться інтерпретатор. Для його встановлення перейдемо на сторінку

<https://www.python.org/downloads/> та знайдемо посилання на завантаження останньої версії мови:



Після натискання на кнопку буде завантажено відповідною поточною ОС установщик Python. Слід враховувати, що Windows 7 і попередні версії не підтримуються.

На ОС Windows під час запуску інсталятора запускає вікно майстра установки:



Тут ми можемо задати шлях, яким встановлюватиметься інтерпретатор. Залишимо його за умовчанням, тобто

`C:\Users[ім'я_користувача]\AppData\Local\Programs\Python\Python312\`.

Крім того, в самому низу відзначимо прапорець "Add Python 3.12 to PATH", щоб додати шлях до інтерпретатора у змінні середовища.

Після цього ми можемо перевірити інсталяцію Python та його версію, запустивши в командному рядку/терміналі команду `python --version`

```
C:\Users\Nikita>python --version
Python 3.12.1
C:\Users\Nikita>
```

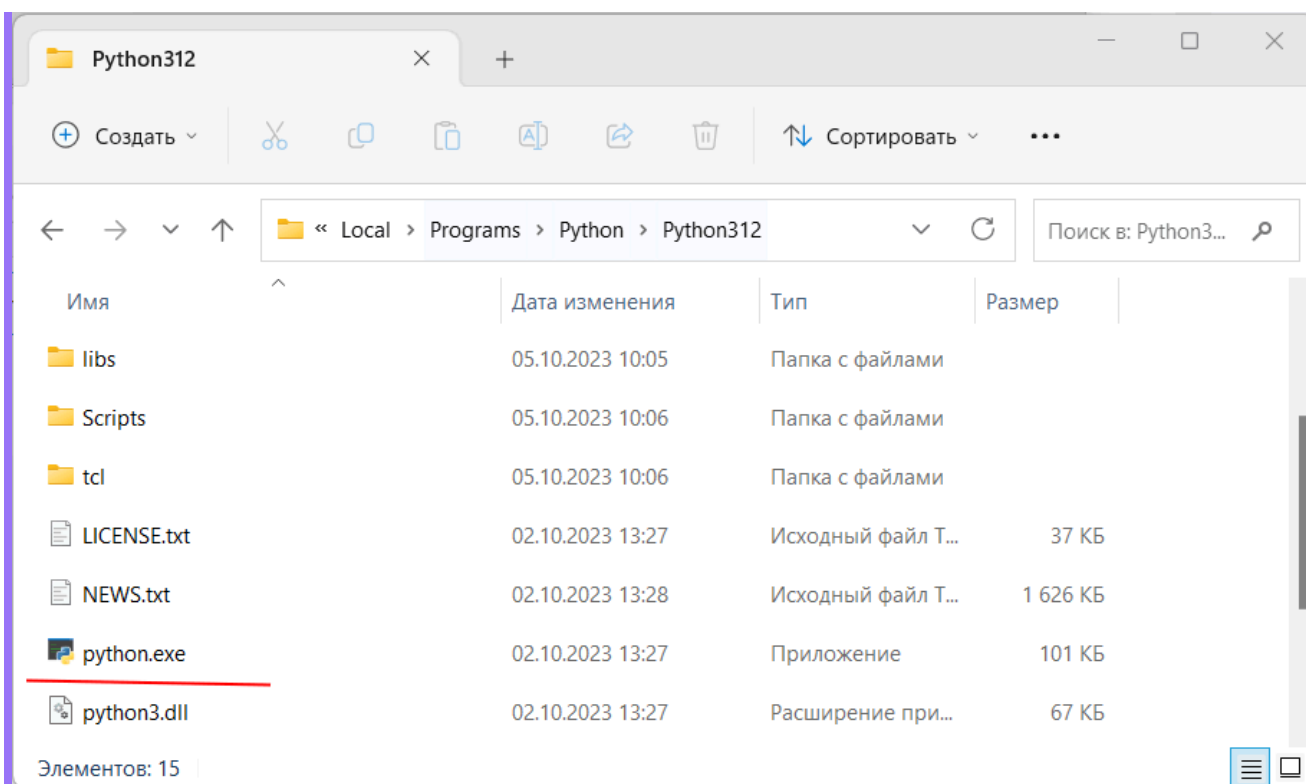
## Запуск інтерпретатора

Після встановлення інтерпретатора, як було описано в попередній темі, ми можемо почати створювати програми на Python. Отже, створимо першу просту програму.

Якщо при установці не було змінено адресу, то на Windows Python за замовчуванням встановлюється шляхом

`C:\Users[ім'я_користувача]\AppData\Local\Programs\Python\Python[номер_версії]\` і представляє файл під назвою `python.exe`.





Запустимо інтерпретатор і введемо до нього наступний рядок:

```
print("hello world")
```

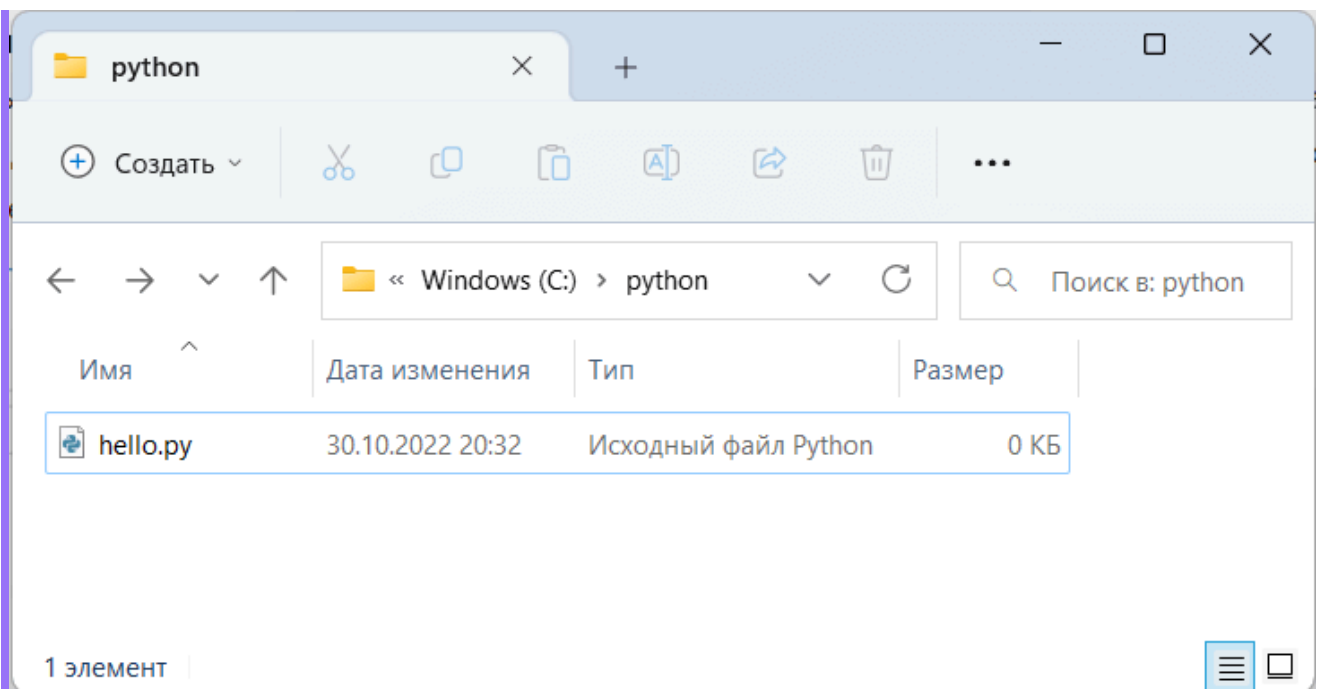
І консоль виведе рядок "hello world":

```
python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64
bit (AMD64)] on win32
Тип "help", "copyright", "кредити" або "license" для більше інформації.
>>> print("hello world")
hello world
>>>
```

Для цієї програми використовувалася функція `print()`, яка виводить рядок на консоль.

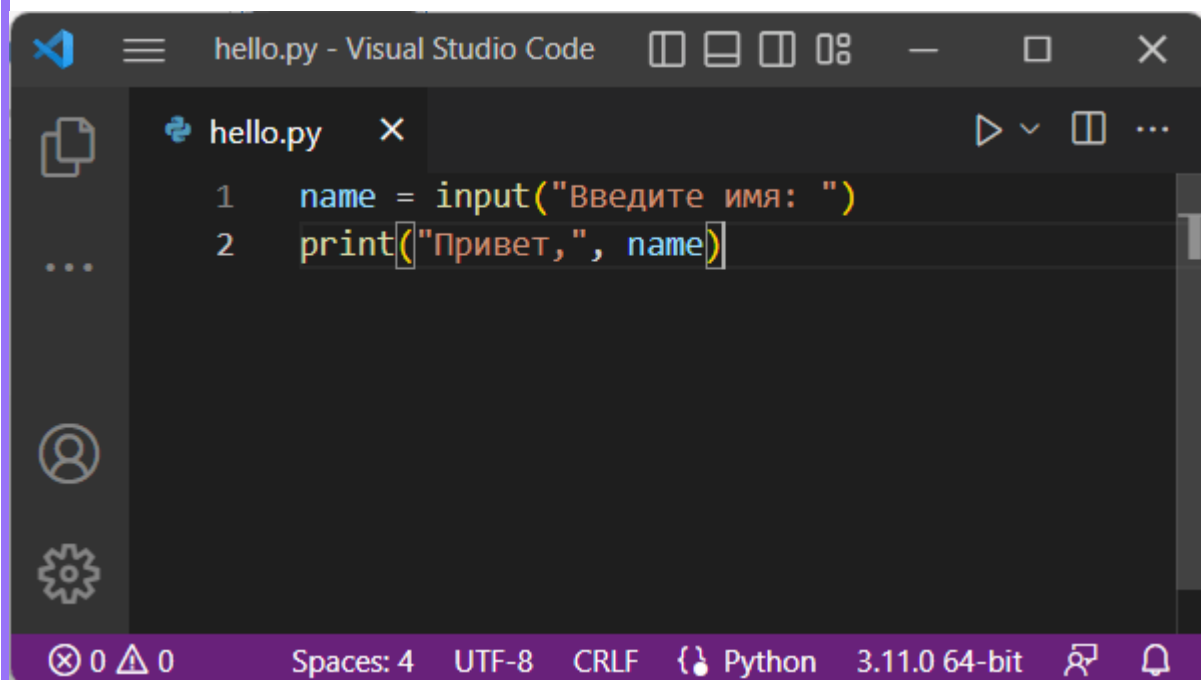
## Створення файлу програми

Насправді програми зазвичай визначаються у зовнішніх файлах-скриптах і потім передаються інтерпретатору на виконання. Тому створимо файл програми. Для цього на диску C або в іншому місці файлової системи визначимо для скриптів папку `python`. А в цій папці створимо новий текстовий файл, який назовемо `hello.py`. За замовчуванням файли з кодом Python, як правило, мають розширення `py`.



Відкриємо цей файл у будь-якому текстовому редакторі і додамо до нього наступний код:

```
name = input("Введіть ім'я: ")
print("Привіт", name)
```



Скрипт складається із двох рядків. Перший рядок за допомогою функції `input()` чекає на введення користувача свого імені. Введене ім'я потім потрапляє до змінної `name`.

Другий рядок за допомогою функції `print()` виводить вітання разом із введеним ім'ям.

Тепер запусимо командний рядок/термінал і за допомогою команди `cd` перейдемо до папки, де знаходиться файл із вихідним кодом `hello.py` (наприклад, у моєму випадку це папка `C:\python`).

```
cd c:\python
```

Далі спочатку введемо повний шлях до інтерпретатора, потім повний шлях до файлу скрипта. Наприклад, у моєму випадку в консоль треба буде вести:

```
C:\Users\Nikita\AppData\Local\Programs\Python\Python312\python.exe hello.py
```

Але якщо при установці була вказана опція "Add Python 3.12 to PATH" , тобто шлях до інтерпретатора Python був доданий до змінних середовищ, то замість повного шляху до інтерпретатора можна просто написати python:

```
python hello.py
```

Або навіть можна скоротити:

```
py hello.py
```

Варіанти з обома способами запуску:

```
Microsoft Windows [Version 10.0.22621.2361]
(c) Корпорація Майкрософт (Microsoft Corporation). Усі права захищені.
C:\Users\Nikita>cd c:\python
c:\python>C:\Users\Nikita\AppData\Local\Programs\Python\Python312\python.exe
hello.py
Введіть ім'я: Nikita
Привіт, Nikita
c:\python>python hello.py
Введіть назву: Tom
Привіт, Tom
c:\python>py hello.py
Введіть ім'я: Bob
Привіт, Bob
c:\python>
```

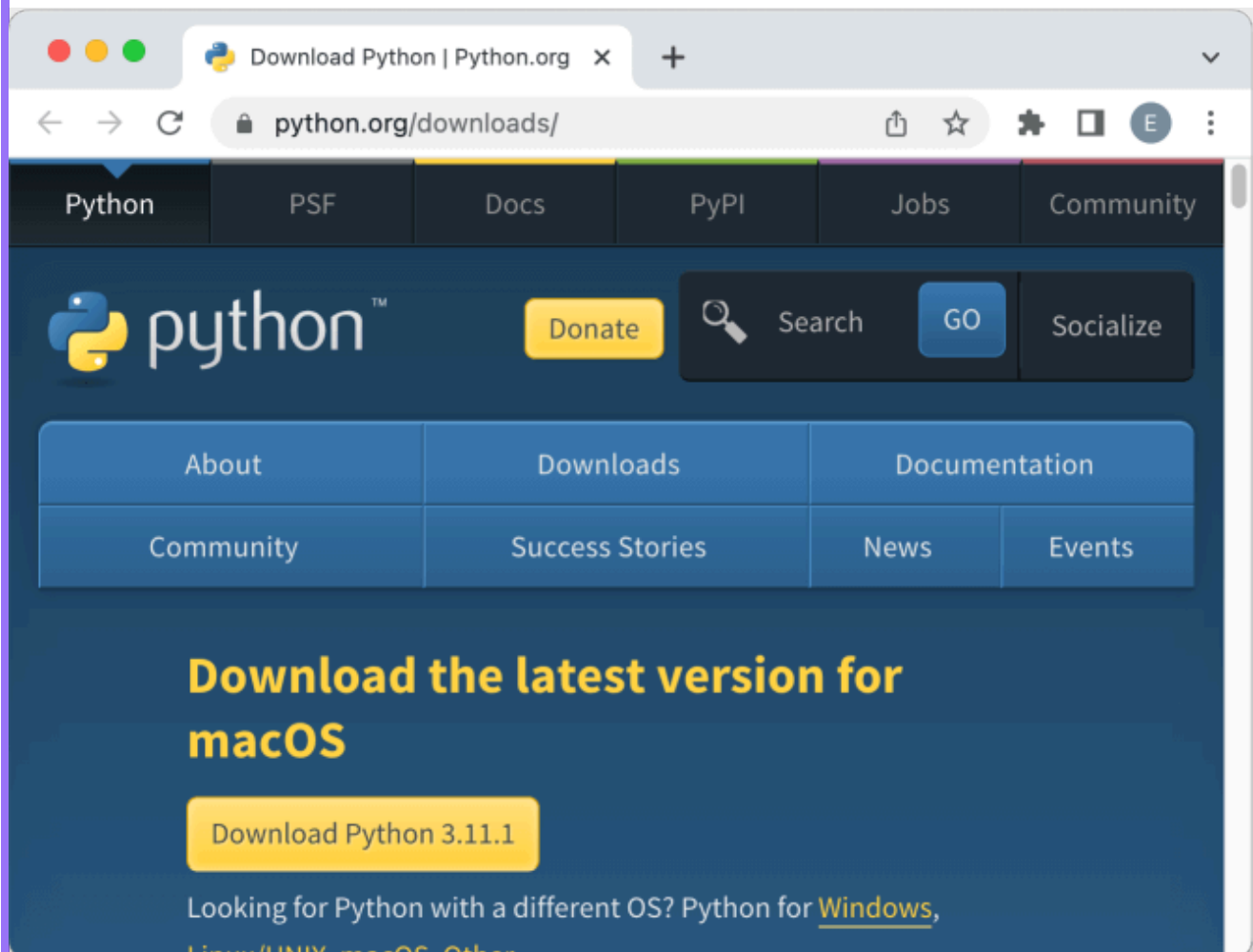
У результаті програма виведе запрошення до введення імені, та був привітання.

Встановлення та перша програма на Mac OS

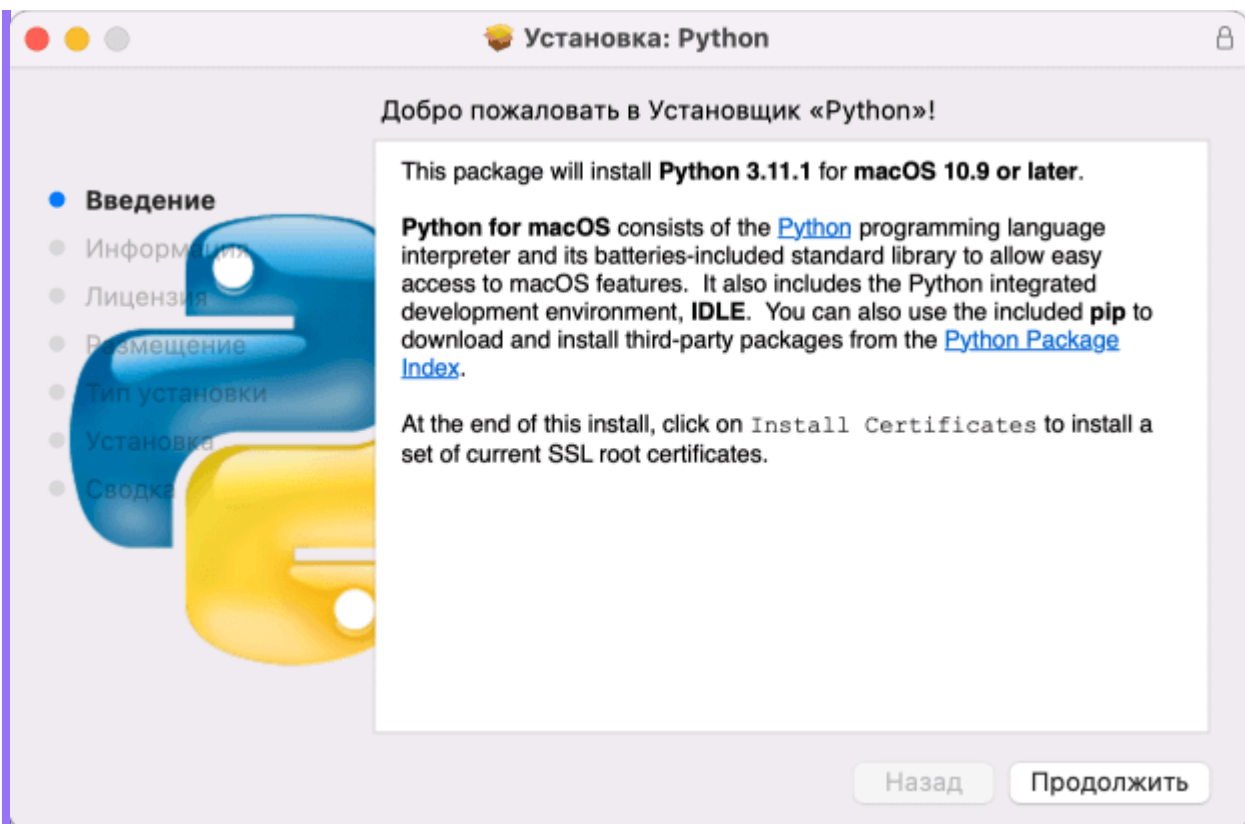
## Встановлення

Для створення програм на Python нам знадобиться інтерпретатор. Для його встановлення перейдемо на сторінку

<https://www.python.org/downloads/> та знайдемо посилання на завантаження останньої версії мови:

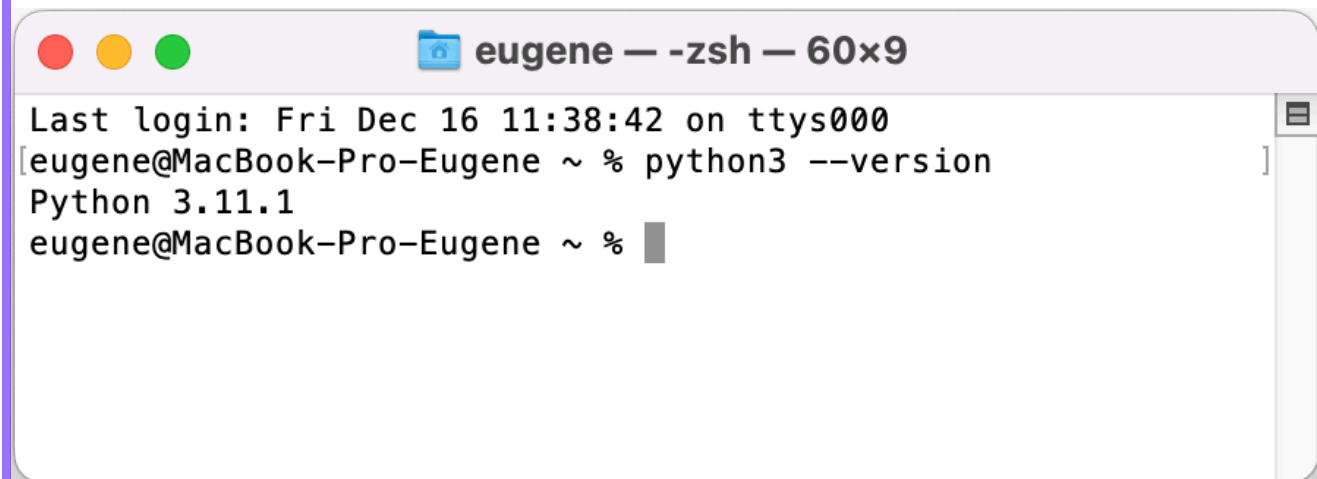


Якщо поточна ОС - Mac OS, за адресою <https://www.python.org/downloads/> буде запропоновано завантажити графічний інсталятор для MacOS. Завантажимо, запустимо його та виконаємо покрокову установку:



Для звернення до інтерпретатора Python на MacOS застосовується команда `python3` . Наприклад, після встановлення інтерпретатора перевіримо його версію командою

```
python3 --version
```



## Перша програма

Спочатку визначимо де на жорсткому диску для скриптів папку `python` . А в цій папці створимо новий текстовий файл, який назовемо `hello.py` . За замовчуванням файли з кодом Python, як правило, мають розширення `py` .

Відкриємо цей файл у будь-якому текстовому редакторі і додамо до нього наступний код:

```
name = input("Your name: ")
print("Hello, ", name)
```



Скрипт складається із двох рядків. Перший рядок за допомогою функції `input()` чекає на введення користувача свого імені. Введене ім'я потім потрапляє до змінної `name`.

Другий рядок за допомогою функції `print()` виводить вітання разом із введеним ім'ям.

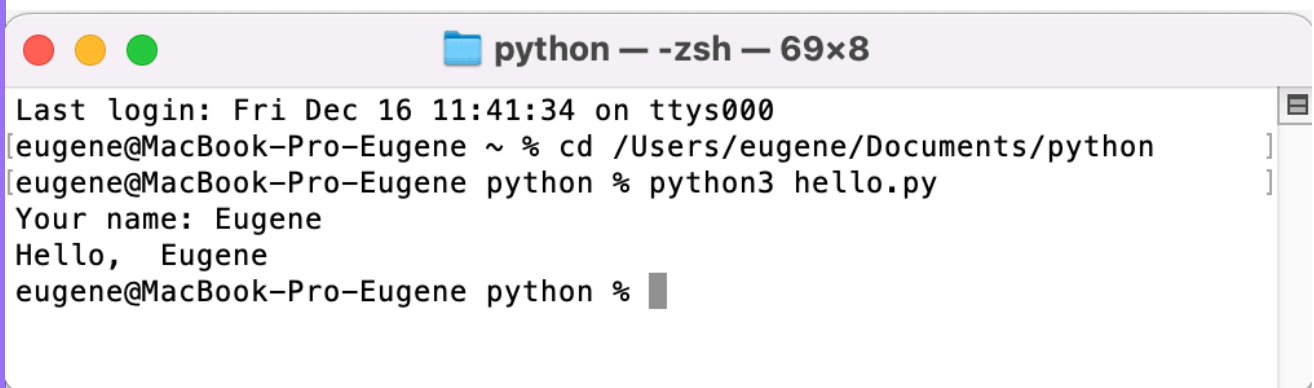
Тепер запусимо командний рядок/термінал і за допомогою команди `cd` перейдемо до папки, де знаходиться файл із вихідним кодом `hello.py` (наприклад, у моєму випадку це папка `/Users/Nikita/Documents/python`).

```
cd /Users/Nikita/Documents/python
```

Далі для виконання скрипту `hello.py` передамо його інтерпретатору `python`:

```
python3 hello.py
```

У результаті програма виведе запрошення до введення імені, та був привітання.

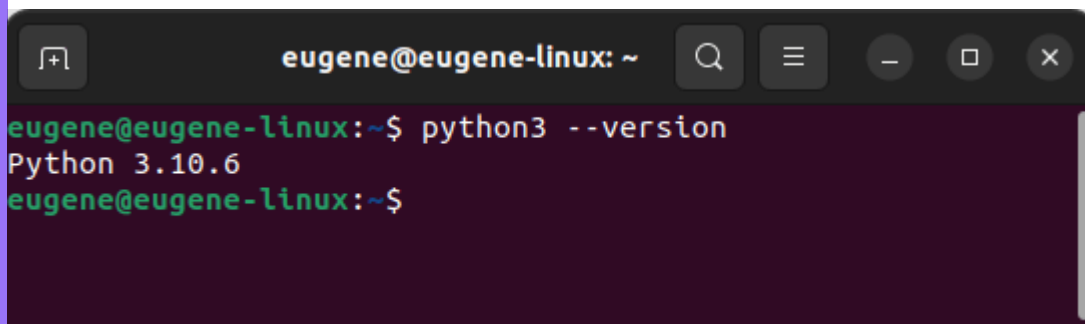


Встановлення та перша програма на Linux

## Встановлення

Для створення програм на Python нам знадобиться інтерпретатор. Варто зазначити, що в деяких дистрибутивах Linux (наприклад, в Ubuntu) Python може бути встановлений за умовчанням. Для перевірки версії Python у терміналі потрібно виконати наступну команду

```
python3 --version
```

A terminal window titled 'eugene@eugene-linux: ~' with standard window controls. The prompt is 'eugene@eugene-linux:~\$'. The command 'python3 --version' has been entered and executed, resulting in the output 'Python 3.10.6'. The prompt is now 'eugene@eugene-linux:~\$'.

Якщо Python встановлений, вона відобразить версію інтерпретатора.

Однак навіть якщо Python встановлений, його версія може бути не останньою. Для встановлення останньої доступної версії Python виконаємо таку команду:

```
sudo apt-get update && sudo apt-get install python3
```

Якщо потрібно встановити не останню доступну, а певну версію, то вказується також підверсія Python. Наприклад, встановлення версії Python 3.10:

```
sudo apt-get install python3.10
```

Відповідно, встановлення версії Python 3.11:

```
sudo apt-get install python3.11
```

## Запуск інтерпретатора

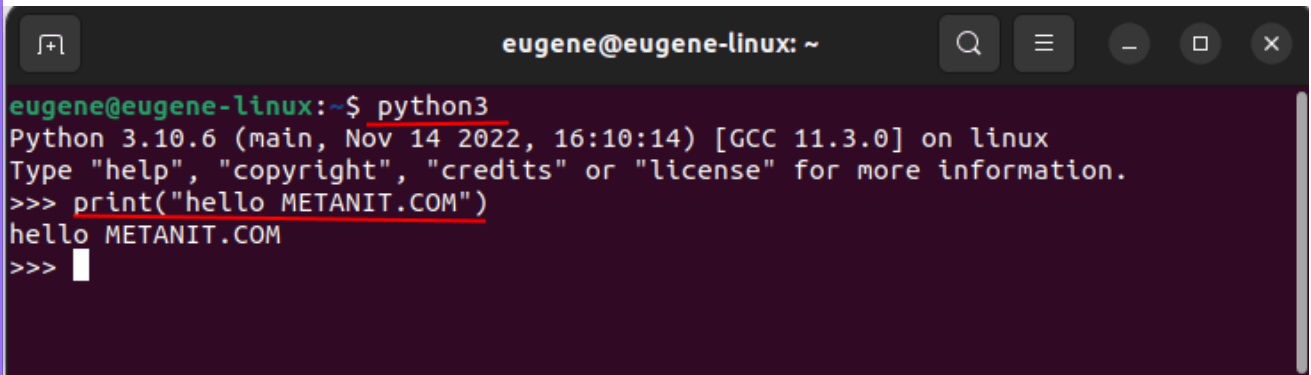
Після встановлення інтерпретатора ми можемо почати створювати програми на Python. Отже, створимо першу просту програму. Для цього введемо в терміналі

```
python3
```

В результаті запускається інтерпретатор Python. Введемо до нього наступний рядок:

```
print("hello METANIT.COM")
```

І консоль виведе рядок "hello METANIT.COM":

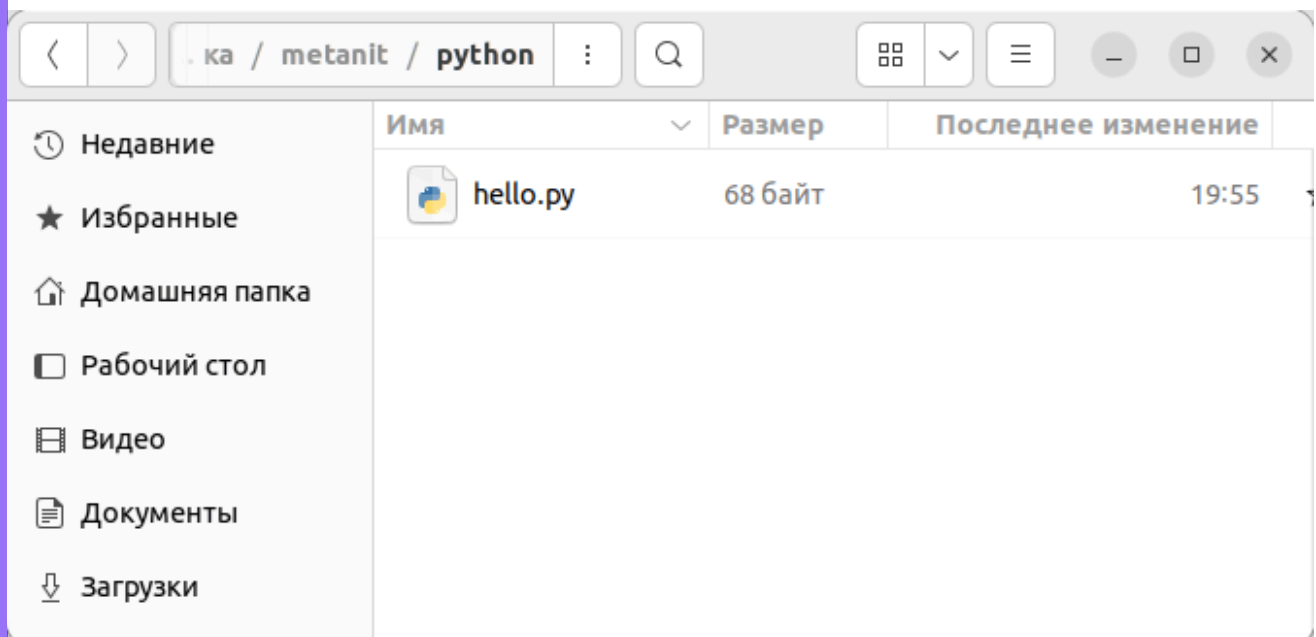


```
eugene@eugene-linux: ~  
eugene@eugene-linux:~$ python3  
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("hello METANIT.COM")  
hello METANIT.COM  
>>> 
```

Для цієї програми використовувалася функція `print()` , яка виводить рядок на консоль.

## Створення файлу програми

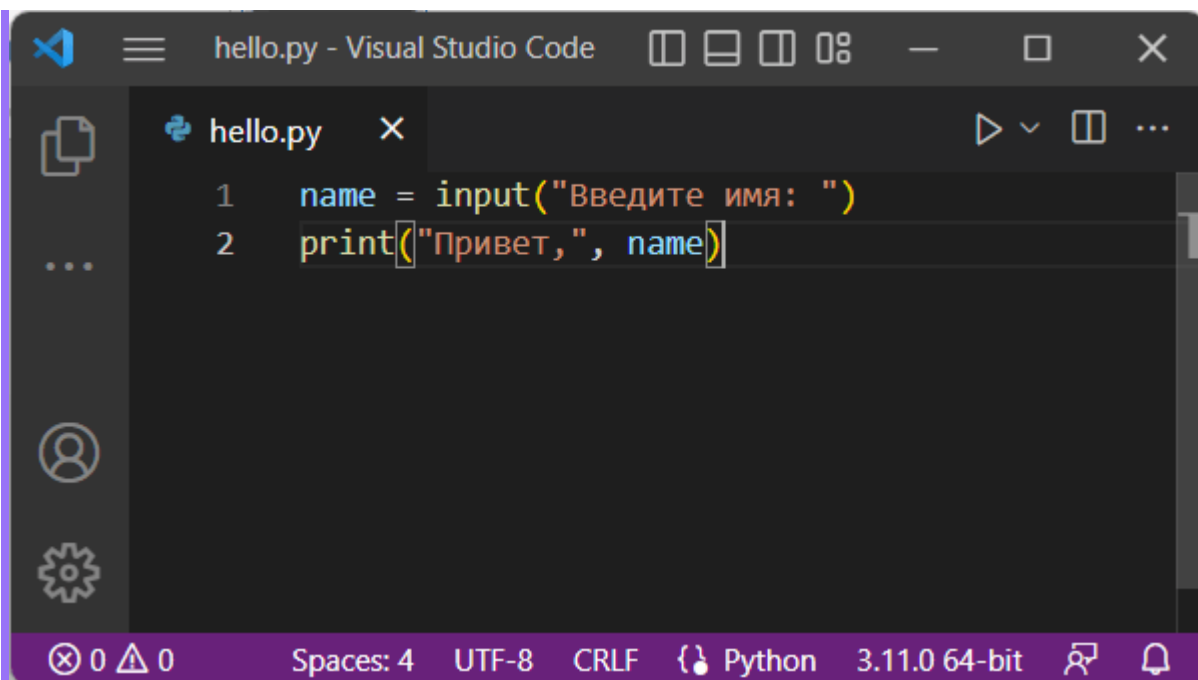
Насправді програми зазвичай визначаються у зовнішніх файлах-скриптах і потім передаються інтерпретатору на виконання. Тому створимо файл програми. Для цього визначимо для скриптів папку `python` . А в цій папці створимо новий текстовий файл, який назовемо `hello.py` . За замовчуванням файли з кодом Python, як правило, мають розширення `py` .



Відкриємо цей файл у будь-якому текстовому редакторі і додамо до нього наступний код:

```
name = input("Введіть ім'я: ")  
print("Привіт", name)
```





```
hello.py - Visual Studio Code
1  name = input("Введіть ім'я: ")
2  print("Привет,", name)
```

Spaces: 4 UTF-8 CRLF Python 3.11.0 64-bit

Скрипт складається із двох рядків. Перший рядок за допомогою функції `input()` чекає на введення користувача свого імені. Введене ім'я потім потрапляє до змінної `name`.

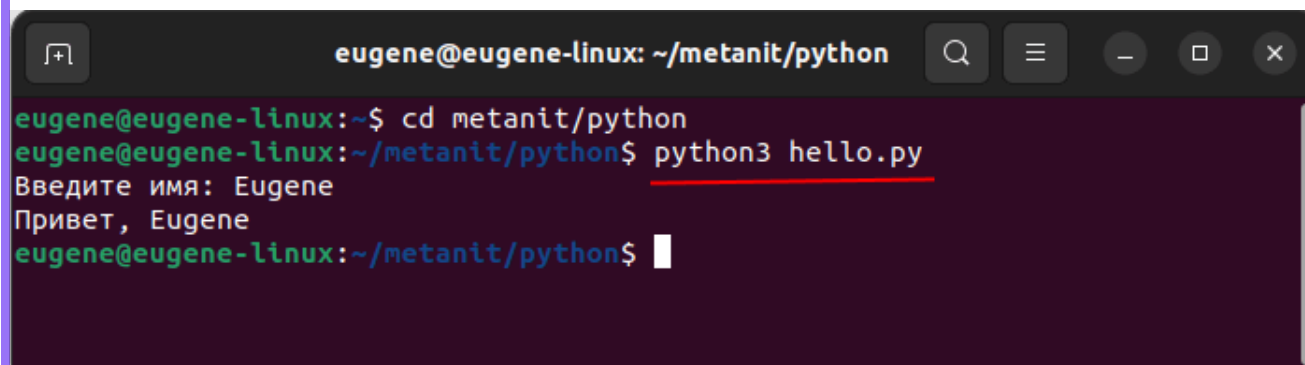
Другий рядок за допомогою функції `print()` виводить вітання разом із введеним ім'ям.

Тепер запустимо термінал та за допомогою команди `cd` перейдемо до папки, де знаходиться файл з вихідним кодом `hello.py` (наприклад, у моєму випадку це папка `metanit/python`

в каталозі поточного користувача). І потім виконаємо код `hello.py` за допомогою наступної команди

```
python3 hello.py
```

У результаті програма виведе запрошення до введення імені, та був привітання.



```
eugene@eugene-linux: ~/metanit/python
eugene@eugene-linux:~$ cd metanit/python
eugene@eugene-linux:~/metanit/python$ python3 hello.py
Введіть ім'я: Eugene
Привет, Eugene
eugene@eugene-linux:~/metanit/python$
```

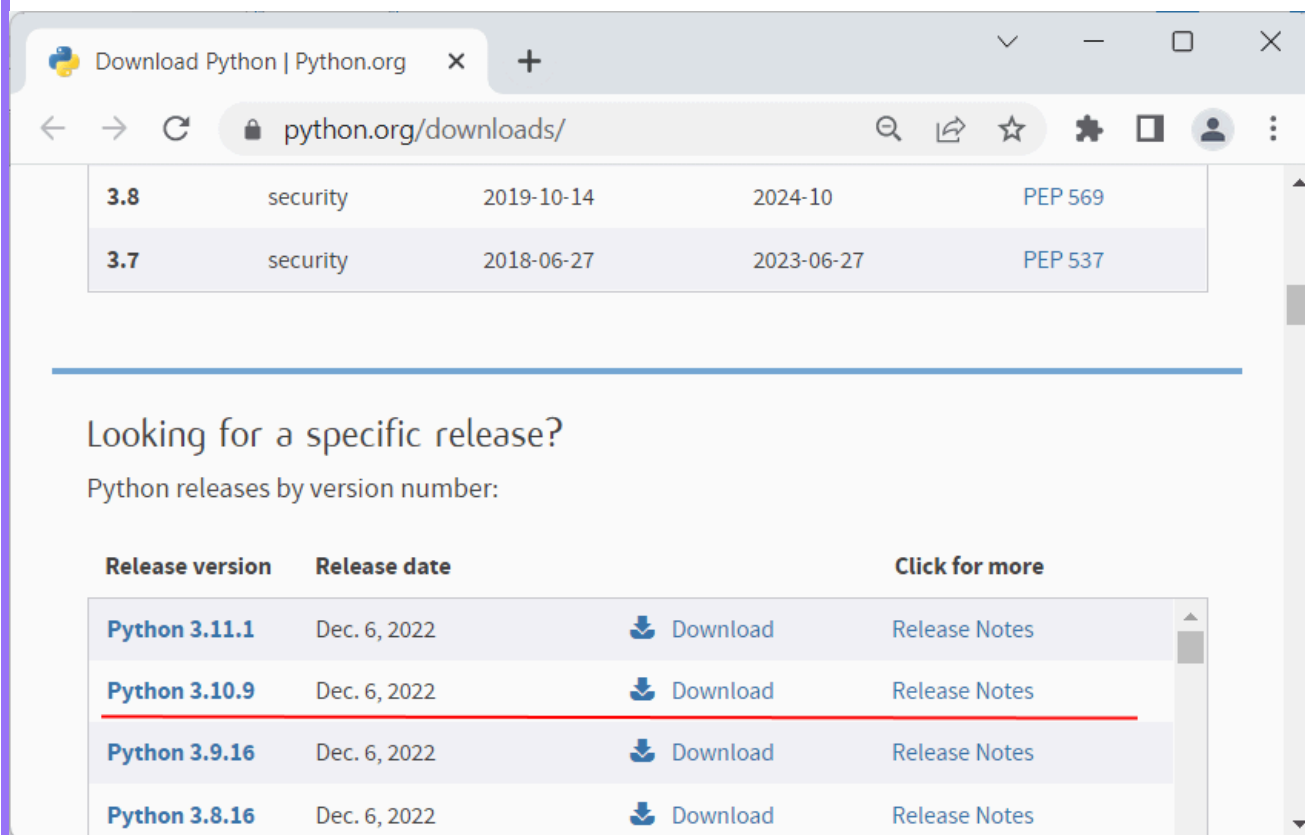
## Керування версіями Python на Windows, MacOS та Linux

На одній робочій машині одночасно можна встановити кілька версій Python. Це буває корисно, коли йде робота з деякими зовнішніми бібліотеками, які підтримують різні версії python, або з якихось інших причин нам треба використовувати кілька різних версій. Наприклад, на момент написання статті останньою та актуальною є версія

Python 3.11 . Але, скажімо, потрібно також встановити версію 3.10 , як у цьому випадку керувати окремими версіями Python?

## Windows

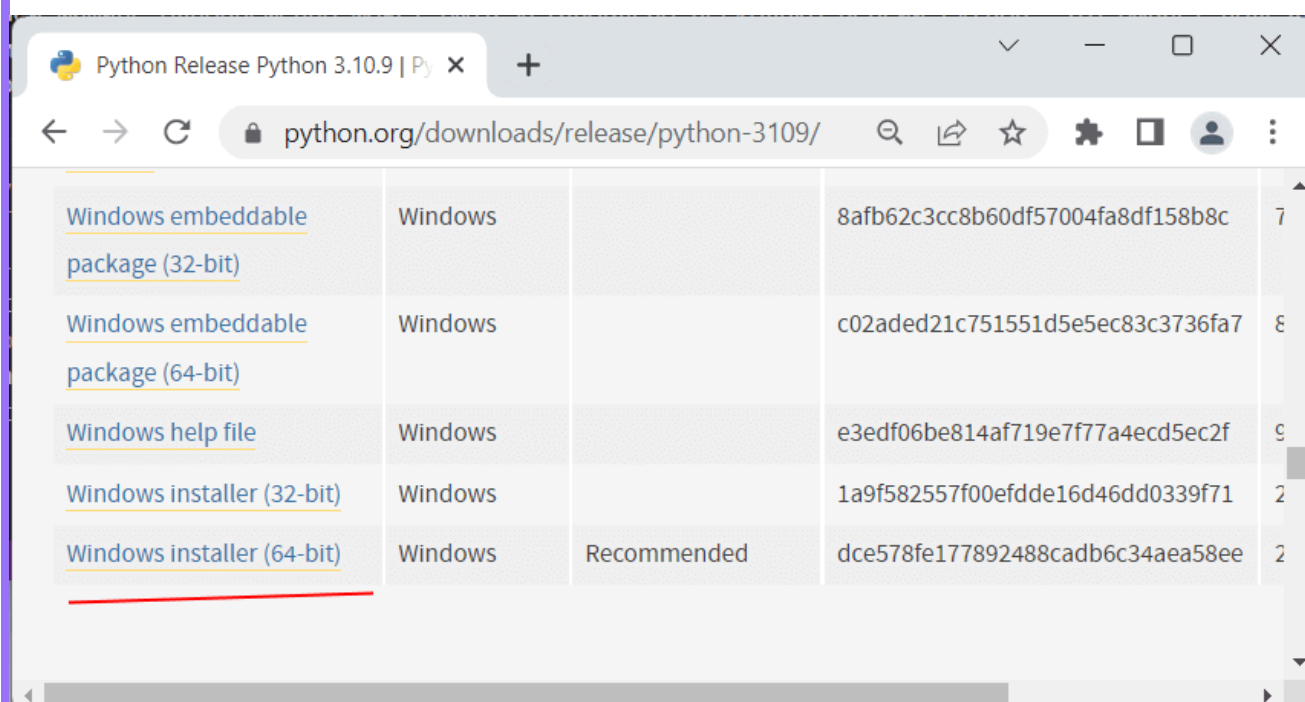
На сторінці завантажень <https://www.python.org/downloads/> ми можемо знайти посилання на потрібну версію:



The screenshot shows the Python.org downloads page. At the top, there's a table of releases with columns for version, security status, release date, and end-of-support date. Below this, there's a section titled "Looking for a specific release?" with the text "Python releases by version number:". This section contains a table with columns for "Release version", "Release date", and "Click for more". The table lists Python 3.11.1, 3.10.9, 3.9.16, and 3.8.16, each with a "Download" link and "Release Notes".

Release version	Release date	Click for more
Python 3.11.1	Dec. 6, 2022	<a href="#">Download</a> <a href="#">Release Notes</a>
Python 3.10.9	Dec. 6, 2022	<a href="#">Download</a> <a href="#">Release Notes</a>
Python 3.9.16	Dec. 6, 2022	<a href="#">Download</a> <a href="#">Release Notes</a>
Python 3.8.16	Dec. 6, 2022	<a href="#">Download</a> <a href="#">Release Notes</a>

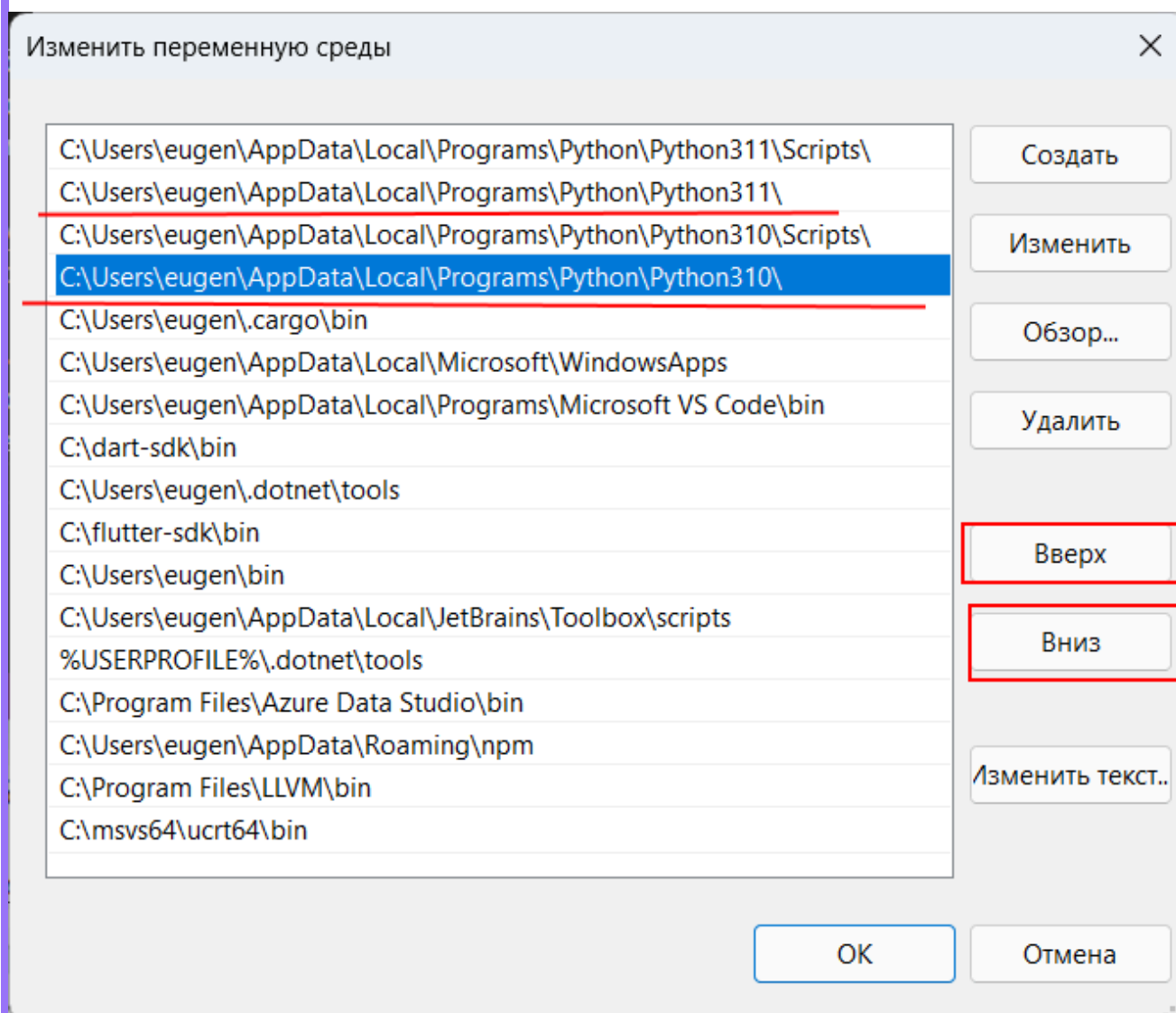
І також завантажити її та встановити:



The screenshot shows the Python.org download page for Python 3.10.9. It lists several download options for Windows, including "Windows embeddable package (32-bit)", "Windows embeddable package (64-bit)", "Windows help file", "Windows installer (32-bit)", and "Windows installer (64-bit)". The "Windows installer (64-bit)" option is highlighted with a red underline.

Download link	Platform	Architecture	SHA256 hash	Size
<a href="#">Windows embeddable package (32-bit)</a>	Windows		8afb62c3cc8b60df57004fa8df158b8c	7
<a href="#">Windows embeddable package (64-bit)</a>	Windows		c02aded21c751551d5e5ec83c3736fa7	8
<a href="#">Windows help file</a>	Windows		e3edf06be814af719e7f77a4ecd5ec2f	9
<a href="#">Windows installer (32-bit)</a>	Windows		1a9f582557f00efdde16d46dd0339f71	2
<a href="#">Windows installer (64-bit)</a>	Windows	Recommended	dce578fe177892488cadb6c34aea58ee	2

Щоб при використанні інтерпретатора Python не прописувати до нього весь шлях, додамо при встановленні в змінні середовища. Але тут треба враховувати, що в змінних середовищах може міститися кілька шляхів до різних інтерпретаторів Python:



Та версія Python, яка знаходиться вище, буде стандартною версією. За допомогою кнопки "Вгору" можна потрібну нам версію перемістити на початок, зробивши версією за замовчуванням. Наприклад, у моєму випадку це версія 3.11. Відповідно, якщо я введу в терміналі команду

```
python --version
```

або

```
py --version
```

то консоль відобразить версію 3.11:

```
C:\python>python --version  
Python 3.11.0
```

Для звернення до версії 3.10 (та для всіх інших версій) необхідно використовувати вказувати номер версії:

```
C:\python>py -3.10 --version  
Python 3.10.9
```

наприклад, виконання скрипту `hello.py` за допомогою версії 3.10:

```
py -3.10 hello.py
```

Так само можна викликати й інші версії Python.

## MacOS

На MacOS можна встановити різні версії, наприклад, завантаживши з офіційного сайту пакет інстальатора для певної версії.

Для звернення до певної версії Python на MacOS вказуємо явно підверсію у форматі `python3.[номер_подверсии]`. Наприклад, я маю версію Python 3.10. Перевіримо її версію:

```
python3.10 --version
```

Аналогічно звернення до версії `python3.9` (за умови, якщо вона встановлена)

```
python3.9 --version
```

Наприклад виконання скрипту `hello.py` за допомогою версії `python 3.10`:

```
python3.10 hello.py
```

## Linux

На Linux можна встановити одночасно кілька версій Python. Наприклад, встановлення версій 3.10 та 3.11:

```
sudo apt-get install python3.10  
sudo apt-get install python3.11
```

Однією з версій є стандартна версія. І для звернення до неї достатньо прописати `python3`, наприклад, перевіримо версію за замовчуванням:

```
python3 --version
```

Для звернення до інших версій треба вказувати підверсію:

```
python3.10 --version  
python3.11 --version
```

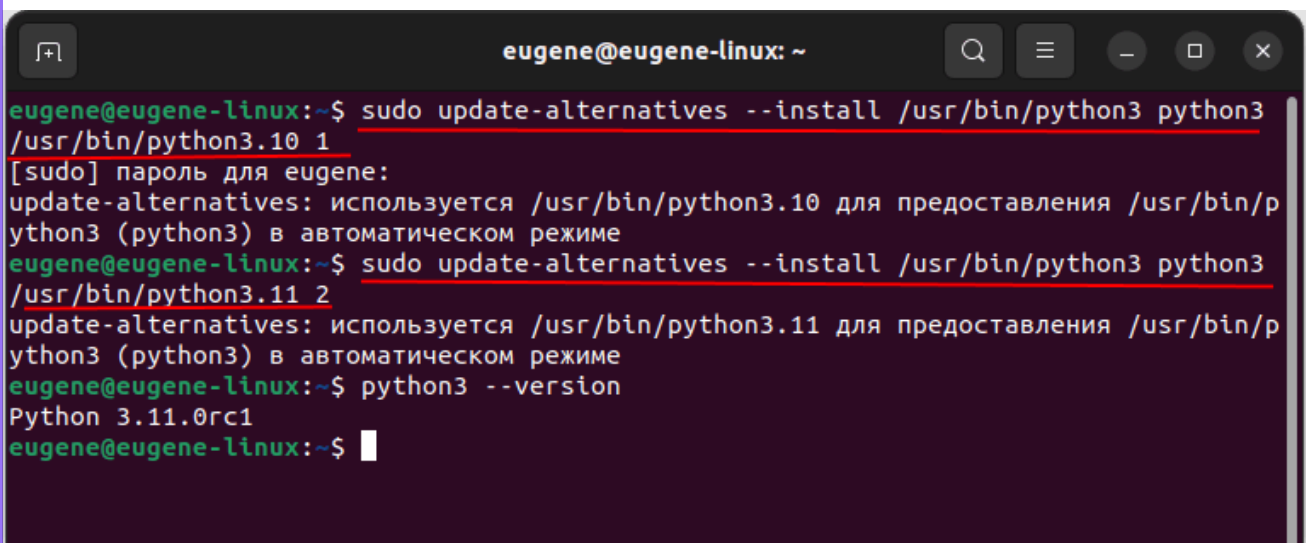
Наприклад, виконання скрипту `hello` за допомогою версії Python 3.10:

```
python3.10 hello.py
```

Але може скластися ситуація, коли нам треба змінити стандартну версію. У цьому випадку використовується команда `update-alternatives` для зв'язування певної версії Python з командою `python3`. Наприклад, ми хочемо встановити як версію за промовчанням Python 3.11. У цьому випадку послідовно виконаємо такі команди:

```
sudo update-alternatives --install /usr/bin/python3 python3  
/usr/bin/python3.10 1  
sudo update-alternatives --install /usr/bin/python3 python3  
/usr/bin/python3.11 2
```

Числа праворуч вказують на пріоритет/стан. Так, для версії 3.11 вказано більший пріоритет, тому при зверненні до використання `python3` буде використовуватися саме версія 3.11 (у моєму випадку це Python 3.11.0rc1)



```
eugene@eugene-linux: ~  
eugene@eugene-linux:~$ sudo update-alternatives --install /usr/bin/python3 python3  
/usr/bin/python3.10 1  
[sudo] пароль для eugene:  
update-alternatives: используется /usr/bin/python3.10 для предоставления /usr/bin/p  
ython3 (python3) в автоматическом режиме  
eugene@eugene-linux:~$ sudo update-alternatives --install /usr/bin/python3 python3  
/usr/bin/python3.11 2  
update-alternatives: используется /usr/bin/python3.11 для предоставления /usr/bin/p  
ython3 (python3) в автоматическом режиме  
eugene@eugene-linux:~$ python3 --version  
Python 3.11.0rc1  
eugene@eugene-linux:~$
```

За допомогою команди

```
sudo update-alternatives --config python3
```

можна змінити стандартну версію

```
eugene@eugene-linux: ~  
eugene@eugene-linux:~$ python3 --version  
Python 3.11.0rc1  
eugene@eugene-linux:~$ sudo update-alternatives --config python3  
Есть 2 варианта для альтернативы python3 (предоставляет /usr/bin/python3).  
  
  Выбор    Путь                Приор Состояние  
-----  
*  0        /usr/bin/python3.11      2      автоматический режим  
  1        /usr/bin/python3.10  1      ручной режим  
  2        /usr/bin/python3.11  2      ручной режим  
  
Press <enter> to keep the current choice[*], or type selection number: 1  
update-alternatives: используется /usr/bin/python3.10 для предоставления /usr/bin/p  
ython3 (python3) в ручном режиме  
eugene@eugene-linux:~$
```

Перша програма в PyCharm

Минулої теми було описано створення найпростішого скрипта мовою Python. Для створення скрипта використовувався текстовий редактор. У моєму випадку це був Notepad. Але є інший спосіб створення програм, який представляє використання різних інтегрованих середовищ розробки або IDE.

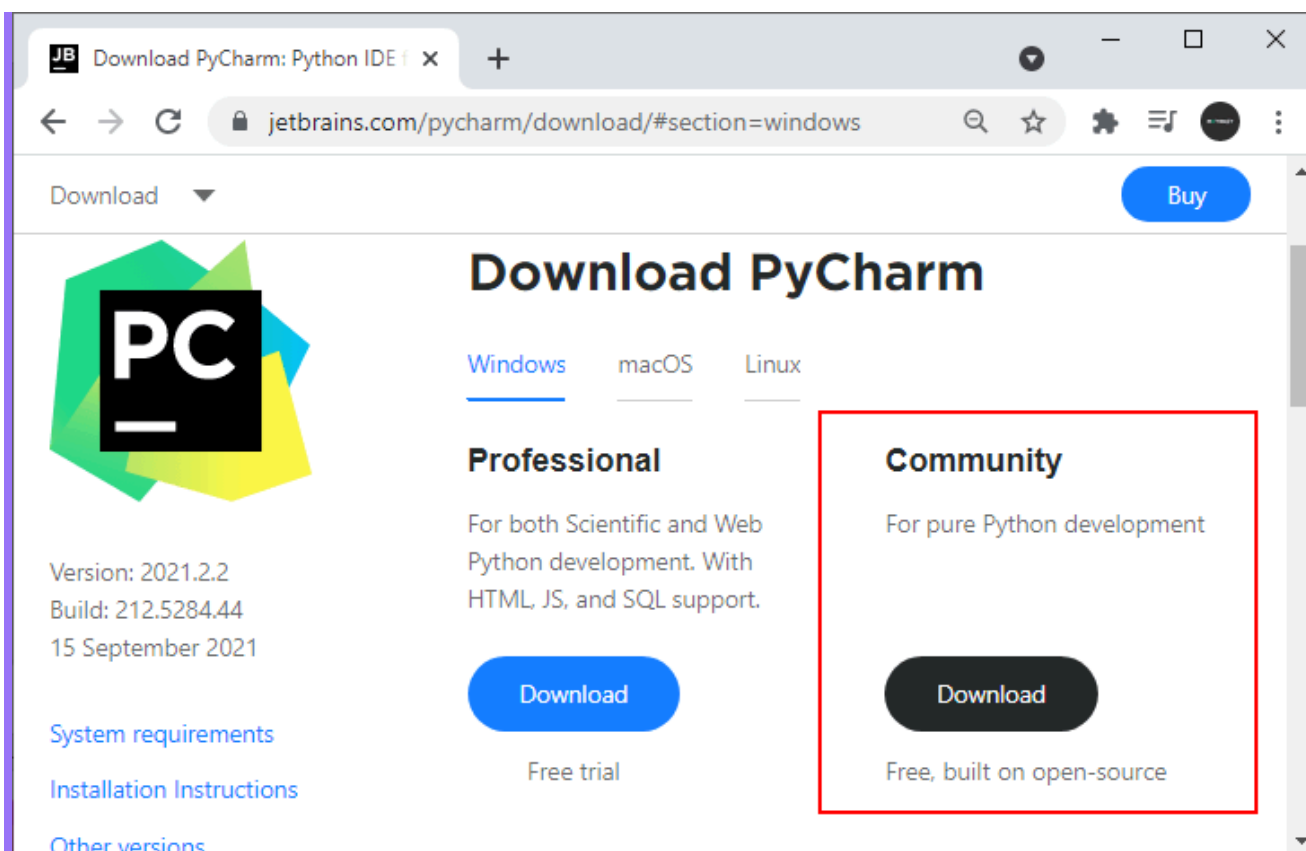
IDE надають нам текстовий редактор для набору коду, але на відміну від стандартних текстових редакторів, IDE також забезпечує повноцінне підсвічування синтаксису, автодоповнення або інтелектуальну підказку коду, можливість відразу виконати створений скрипт, а також багато іншого.

Для Python можна використовувати різні середовища розробки, але однією з найпопулярніших серед них є середовище PyCharm, створене компанією JetBrains. Це середовище динамічно розвивається, постійно оновлюється і доступне найбільш поширених операційних систем - Windows, MacOS, Linux.

Щоправда, вона має одне важливе обмеження. А саме вона доступна у двох основних варіантах: платний випуск Professional та безкоштовний Community. Багато базових можливостей доступні і в безкоштовному випуску Community. У той же час ряд можливостей, наприклад, веб-розробка доступні тільки в платному Professional.

У нашому випадку скористаємось безкоштовним випуском Community. Для цього перейдемо на

[сторінку завантаження](#) та завантажимо інсталяційний файл PyCharm Community.



JB Download PyCharm: Python IDE x +

jetbrains.com/pycharm/download/#section=windows

Download ▾ Buy

# Download PyCharm

[Windows](#) [macOS](#) [Linux](#)

## Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free trial

## Community

For pure Python development

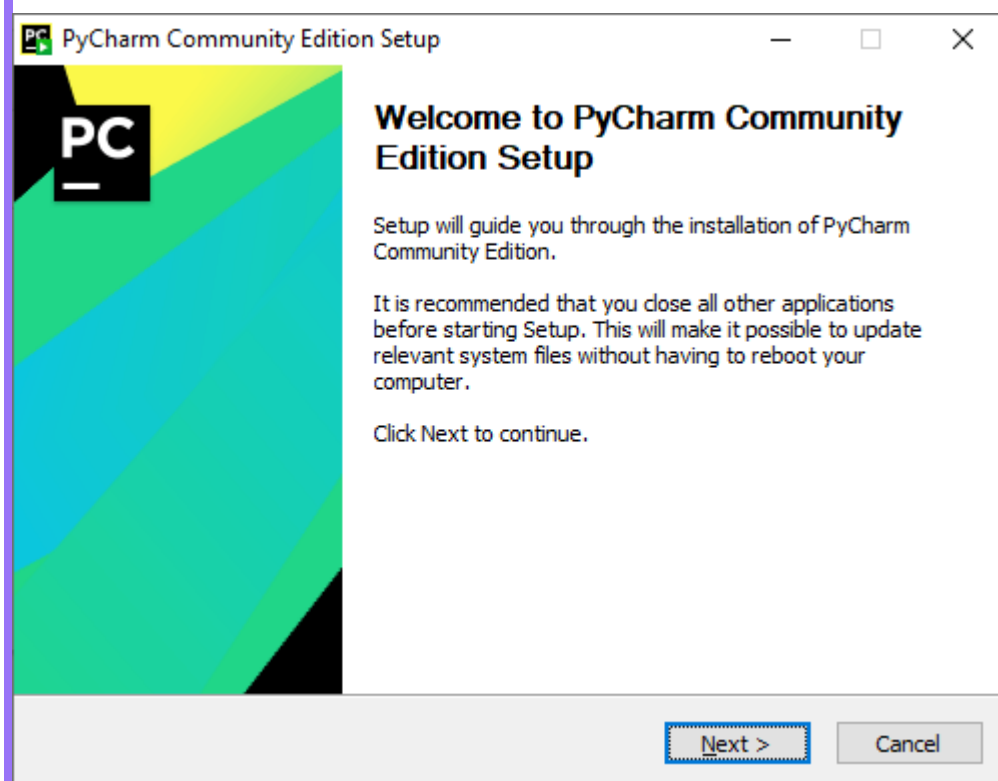
[Download](#)

Free, built on open-source

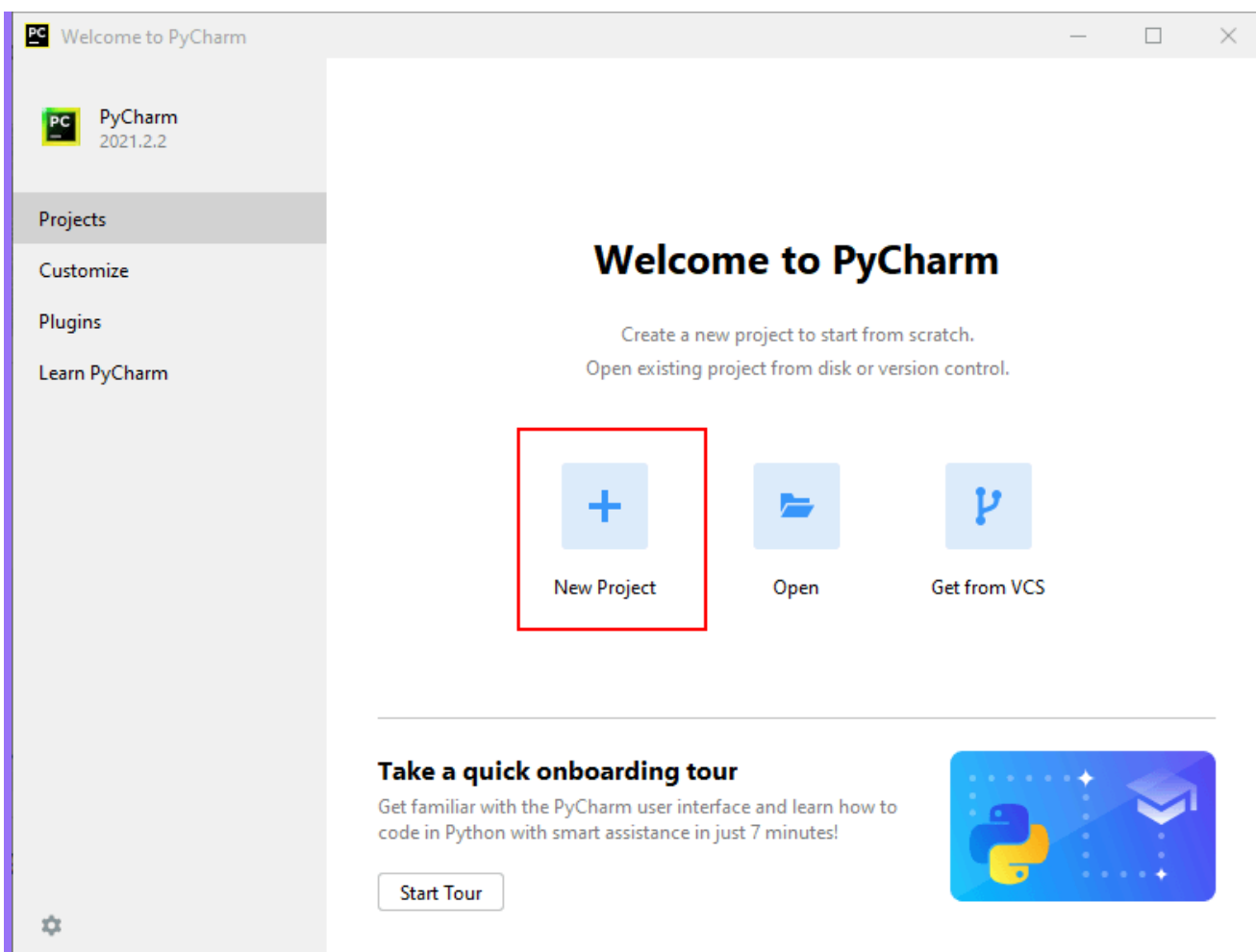
Version: 2021.2.2  
Build: 212.5284.44  
15 September 2021

[System requirements](#)  
[Installation Instructions](#)  
[Other versions](#)

Після завантаження виконаємо його встановлення.

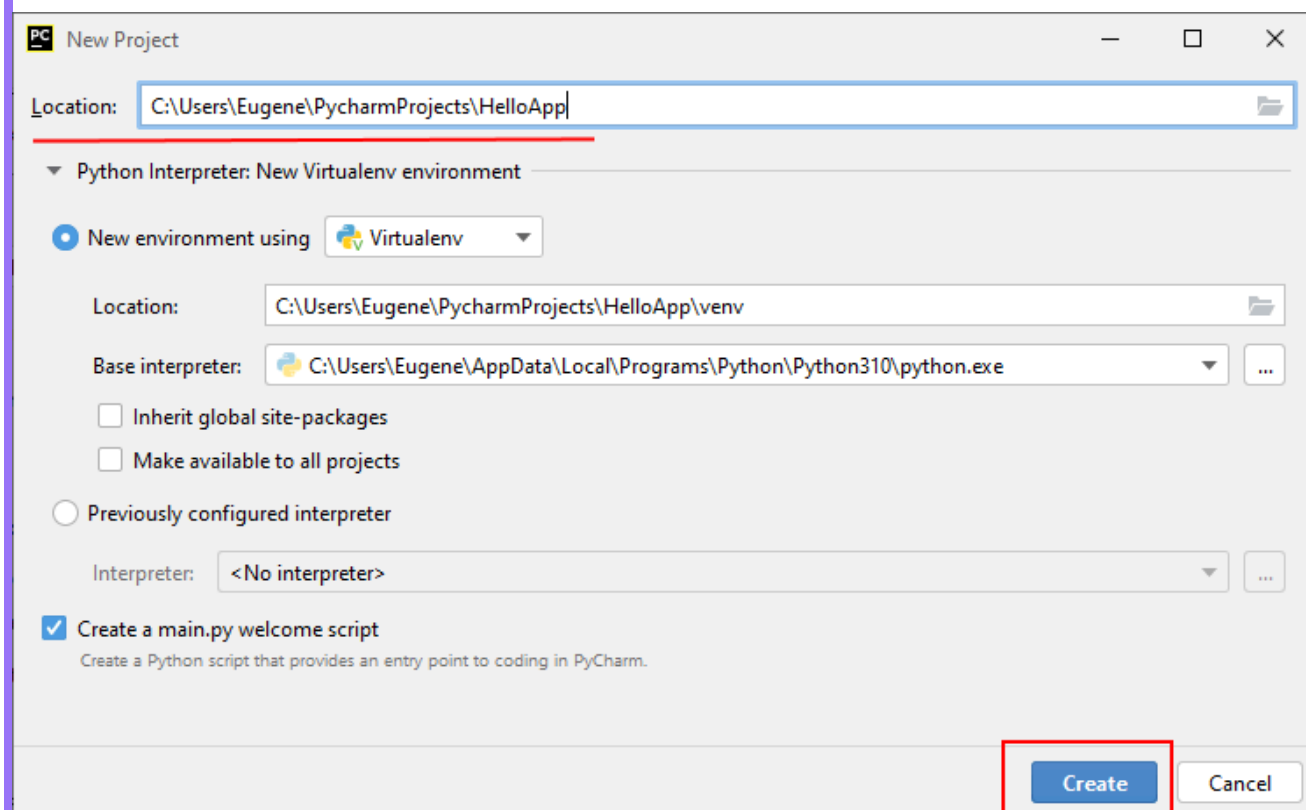


Після завершення встановлення запустимо програму. При першому запуску відкривається початкове вікно:



Створимо проект і для цього оберемо пункт New Project .

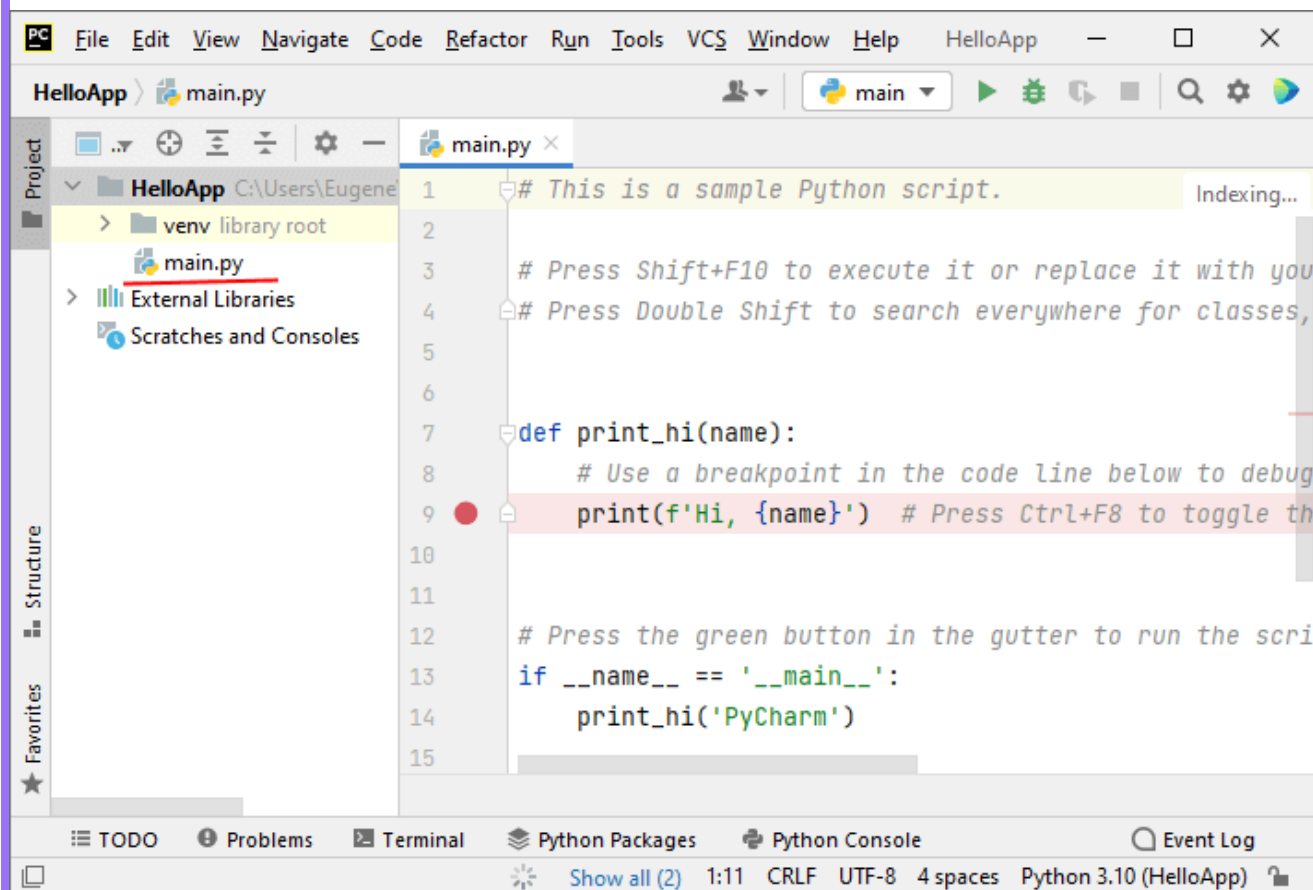
Далі нам відкриється вікно для налаштування проекту. У полі Location необхідно вказати шлях до проекту. У моєму випадку проект буде розміщено в папці HelloApp. Власне назва папки і буде назвою проекту.





Крім шляху до проекту, всі інші налаштування залишимо за замовчуванням і натиснемо на кнопку Create для створення проекту.

Після цього буде створено порожній проект:

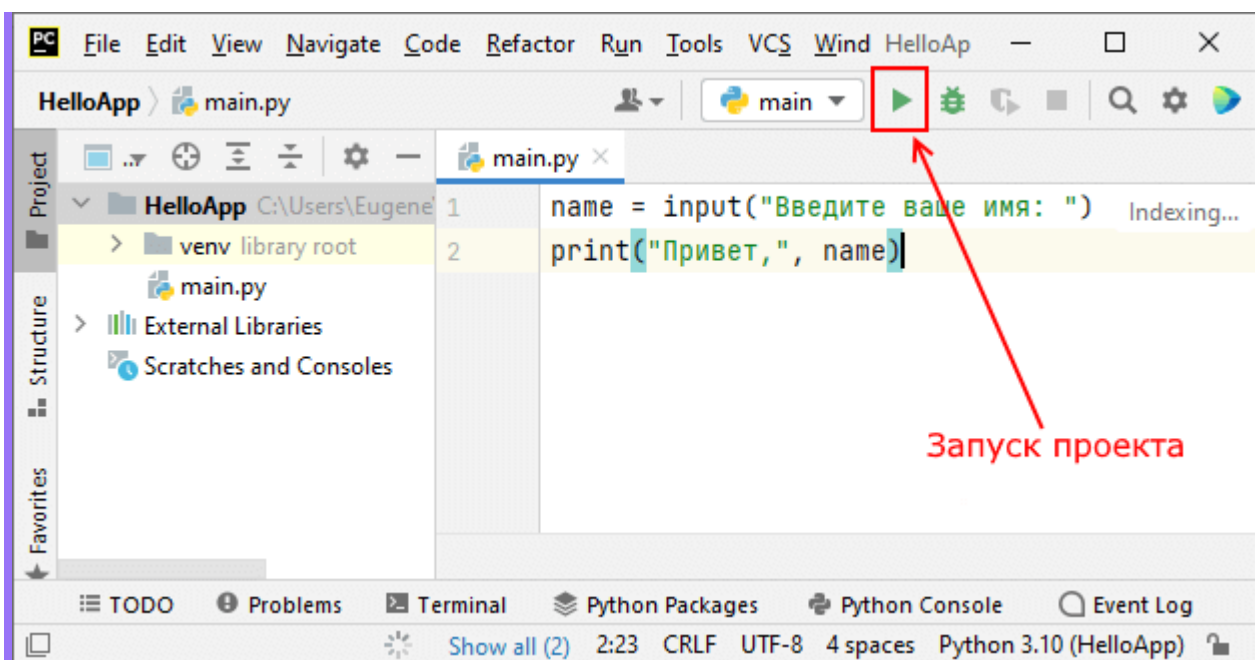


У центрі середовища буде відкрито файл main.py з деяким вмістом за промовчанням.

Тепер створимо найпростішу програму. Для цього змінимо код файлу main.py так:

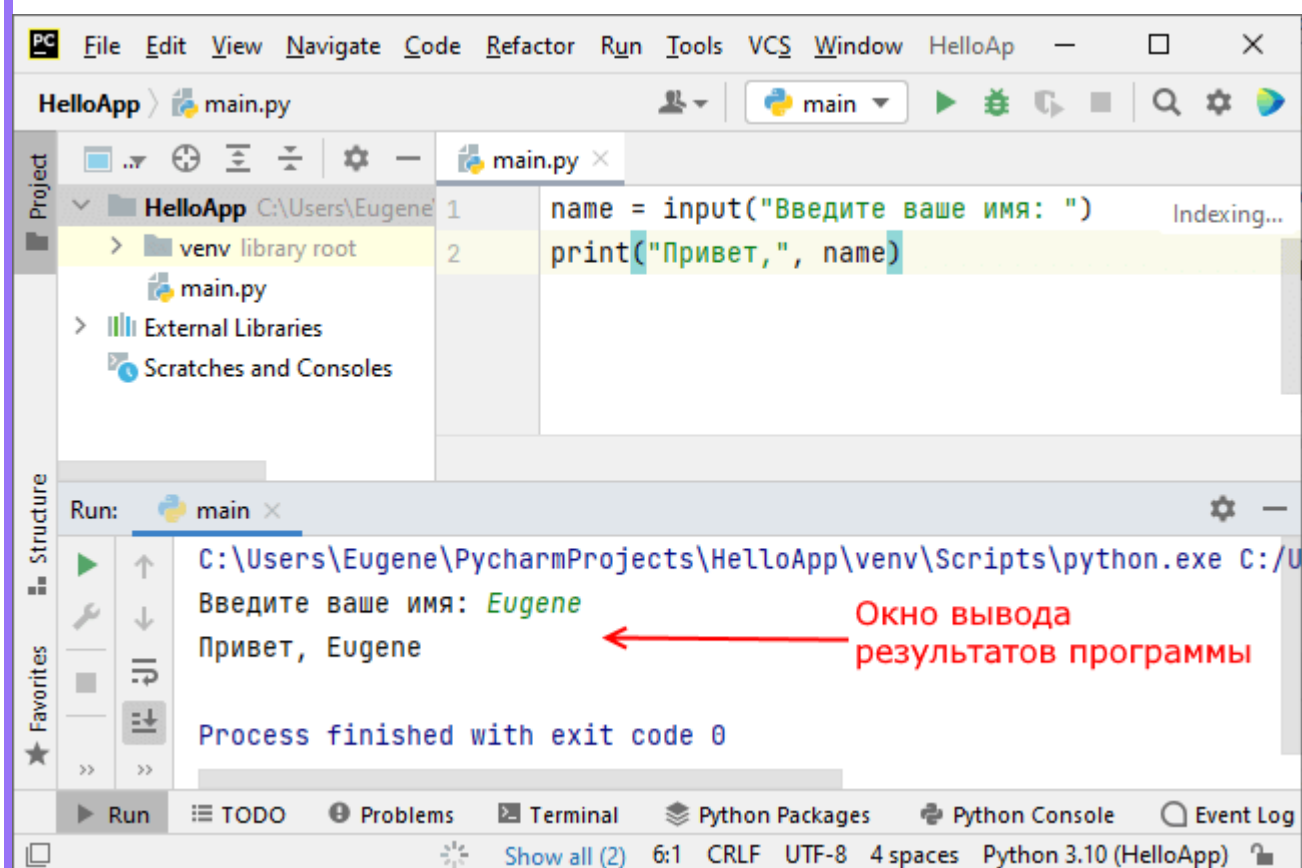
```
name = input("Введіть ваше ім'я: ")
print("Привіт", name)
```

Для запуску скрипту натиснемо на зелену стрілку в панелі інструментів програми:



Також для запуску можна перейти в меню Run і там натиснути на підпункт Run 'main' )

Після цього внизу IDE відобразиться вікно виведення, де потрібно буде ввести ім'я і де після цього буде виведено привітання:

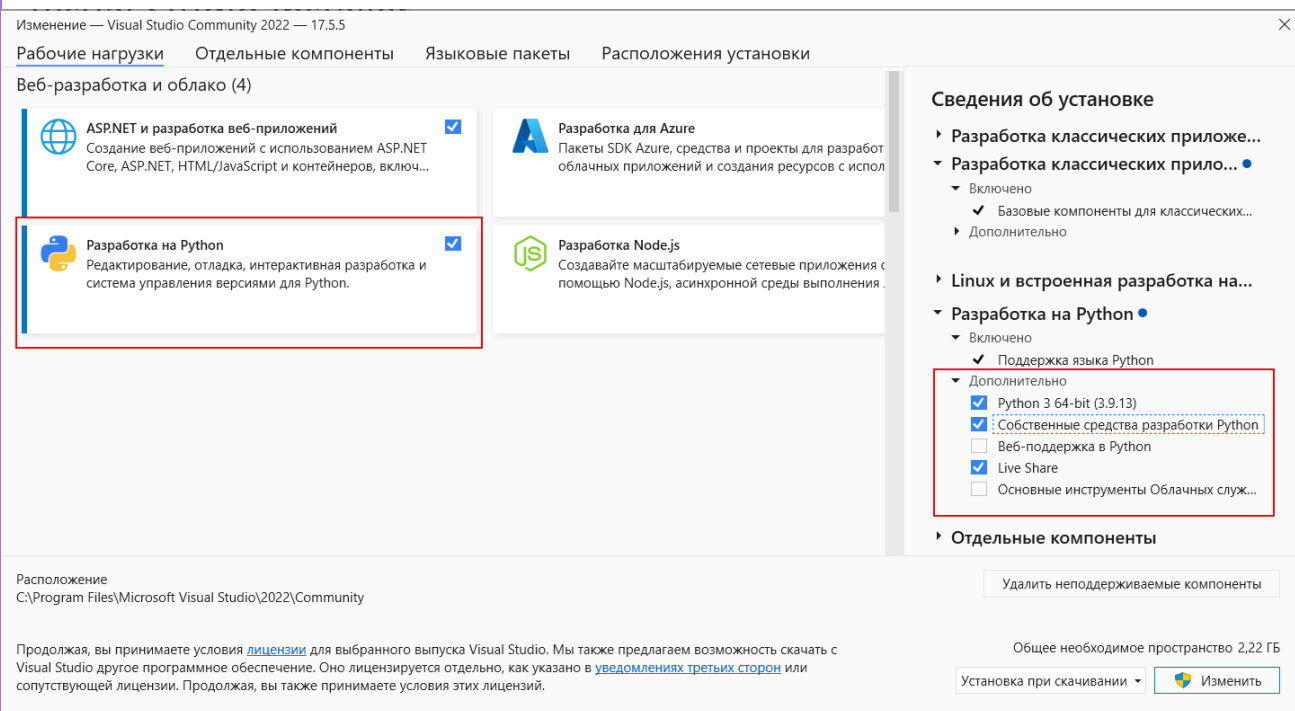


## Python y Visual Studio

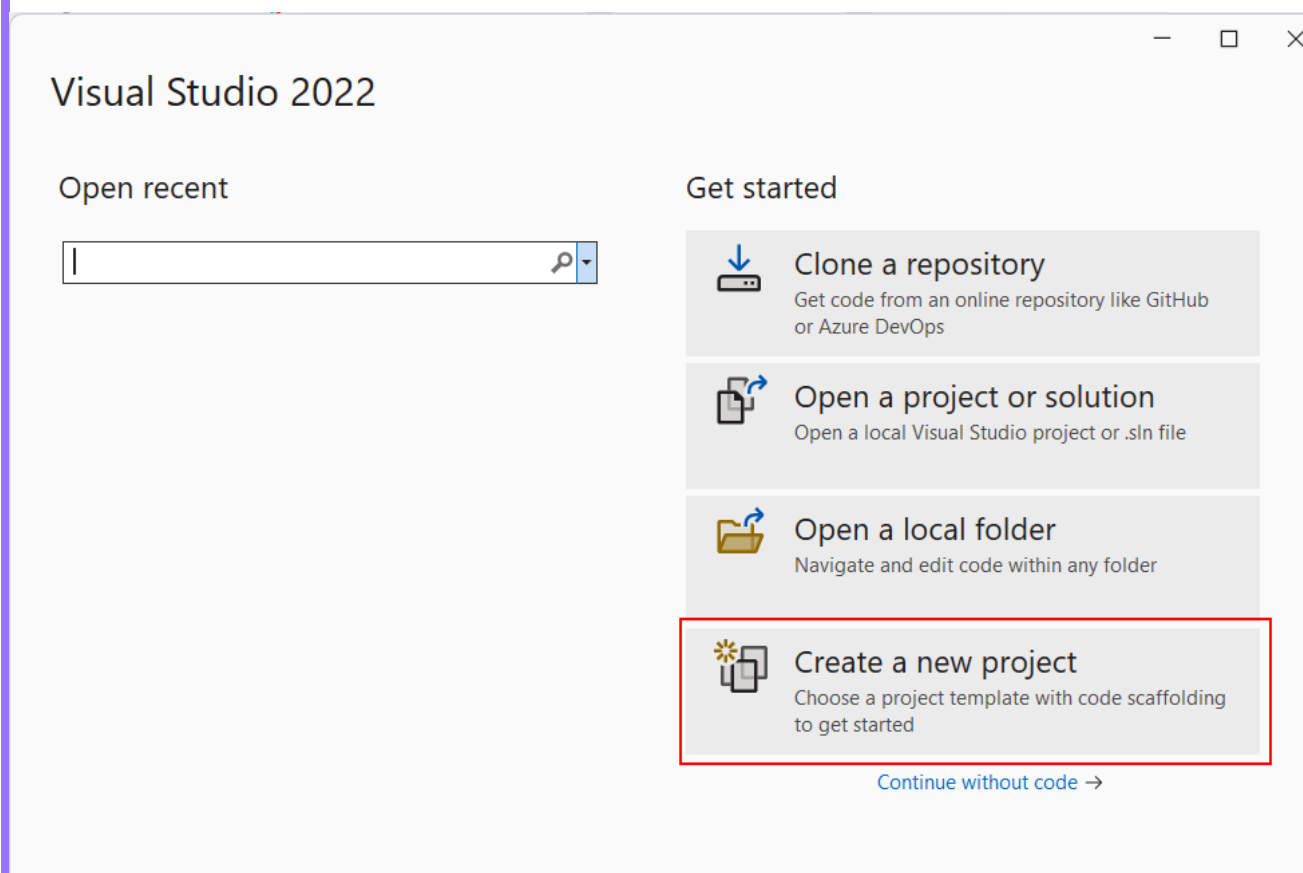
Одним із середовищ розробки, що дозволяє працювати з Python, є Visual Studio. Перевагою даної IDE порівняно, скажімо, з PyCharm, слід зазначити насамперед те, що в її безкоштовній редакції Visual Studio Community безкоштовно доступні ряд функцій і можливостей, які в тому ж PyCharm доступні лише платній версії Professional

Edition. Наприклад, це веб-розробка, у тому числі за допомогою різних фреймворків. У той же час засоби для розробки на Python у Visual Studio доступні поки що тільки у версії для Windows.

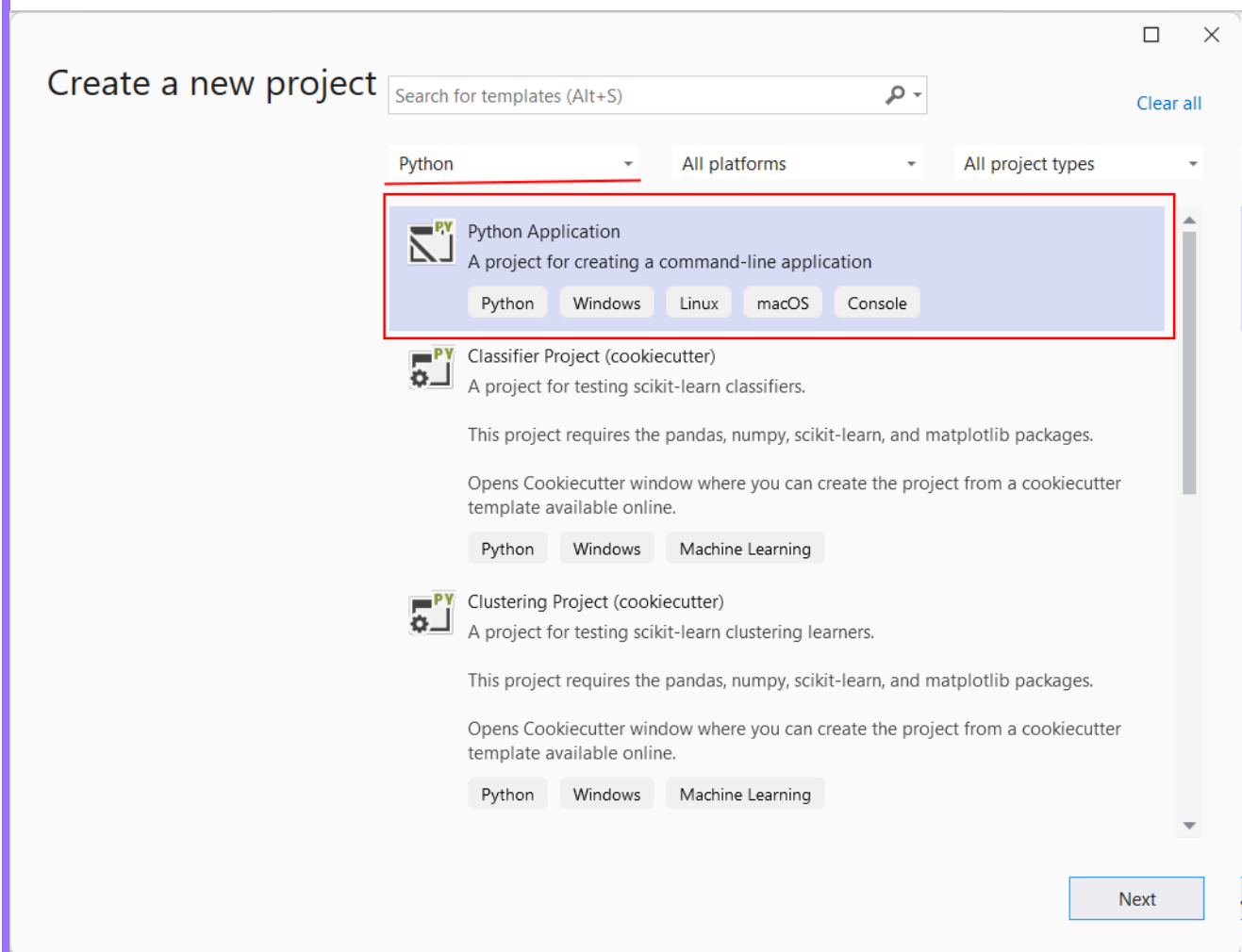
Отже, завантажимо інсталяційний файл Visual Studio Community за посиланням <https://visualstudio.microsoft.com/ru/vs/community/> . Після запуску інсталяційного файлу виберемо серед опцій пункт Розробка на Python :



Після інсталяції Visual Studio запустимо її і у вікні програми оберемо Create a new project :



Далі у вікні створення нового проекту виберемо шаблон Python Application :



На наступному вікні вкажемо назву та шлях до проекту. Наприклад, у моєму випадку проект називатиметься "HelloApp":

□

×

## Configure your new project

Python Application   Python   Windows   Linux   macOS   Console

Project name

Location

...

Solution

Solution name ⓘ

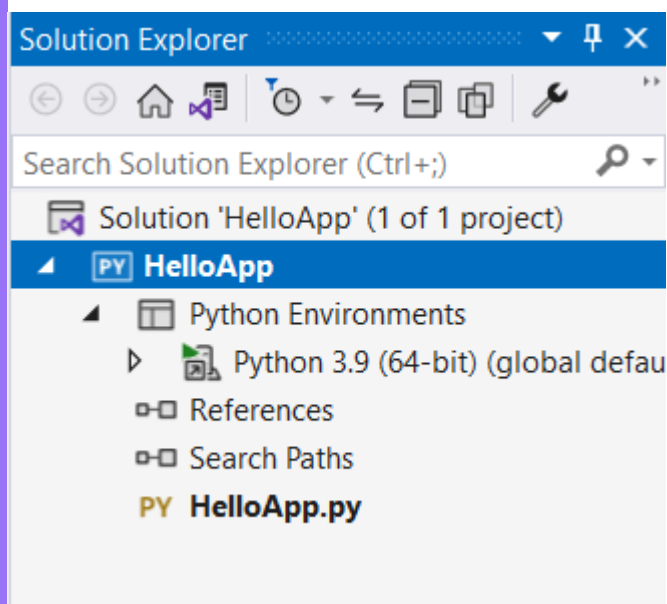
☐ Place solution and project in the same directory

Project will be created in "C:\Users\eugen\source\repos\Python\HelloApp\HelloApp\"

Back

Create

Натисніть кнопку Create, і Visual Studio створить новий проект:



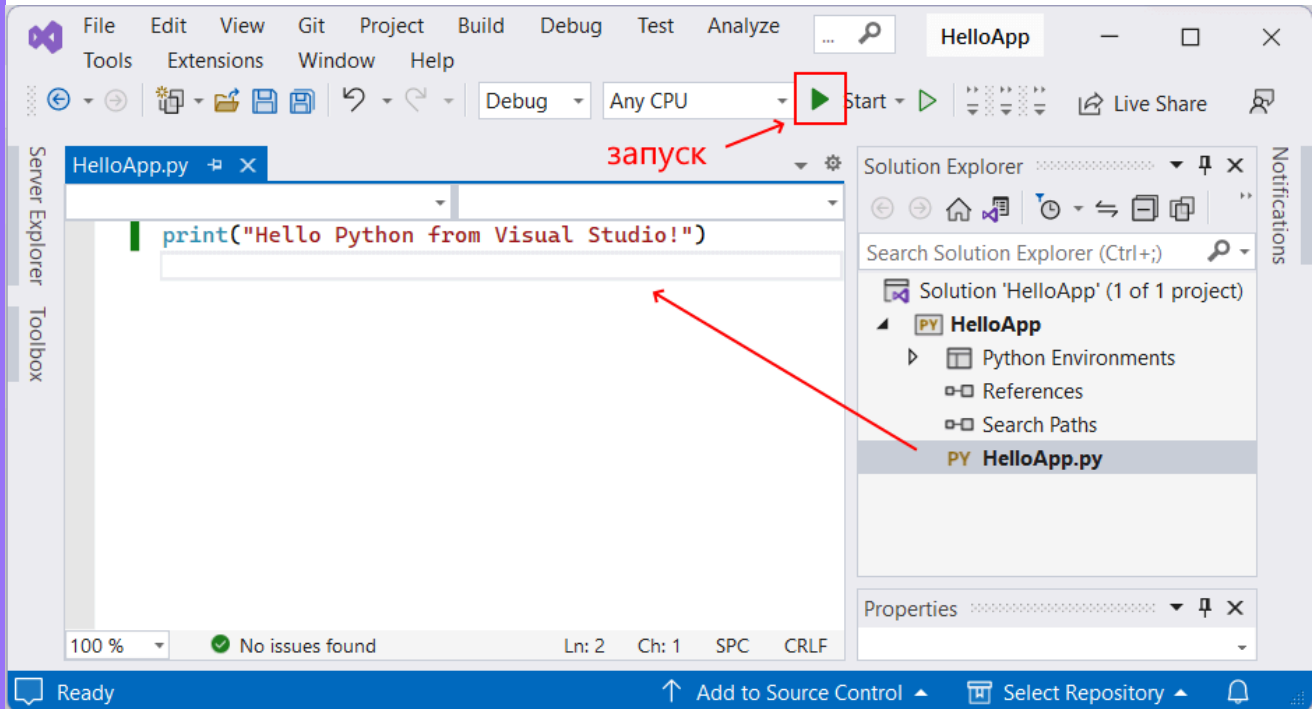
Праворуч у вікні Solution Explorer (Обозреватель рішень) можна побачити структуру проекту. За замовчуванням ми можемо побачити такі елементи:

- Python Environments : тут можна побачити всі використовувані середовища, зокрема тут можна версію Python, яка використовується.
- References : цей вузол містить всі зовнішні залежності, які використовуються поточним проектом
- Search Paths : цей вузол дозволяє вказати шляхи пошуку для модулів Python
- HelloApp.py : власне файл Python з вихідним кодом

За промовчанням у Visual Studio вже відкрито файл HelloApp.py, але він поки що порожній. Додамо до нього наступний рядок:

```
print("Hello Python from Visual Studio!")
```

І потім на панелі інструментів натиснемо на зелену стрілочку для запуску:



В результаті запуску відобразиться консоль, яка виведе потрібний рядок:

