Національний університет біоресурсів і природокористування України

Програмні інструменти для створення та роботи з проектом Python

Основні поняття мови

Лектор: к.ф.-м.н., доцент Кириченко Віктор Вікторович

Основні задачі, які розв'язуються за допомогою мови Python

Веб-розробка

Створення веб-додатків та сайтів за допомогою Django, Flask, FastAPI

Аналіз даних та Data Science

Обробка, аналіз та візуалізація даних з використанням Pandas, NumPy, Matplotlib

Штучний інтелект та машинне навчання

Розробка моделей ШІ за допомогою TensorFlow, PyTorch, scikit-learn

Автоматизація задач

Скрипти для автоматизації рутинних операцій та тестування

Створення графічних інтерфейсів

Розробка десктопних додатків з Tkinter, PyQt, Kivy

Розробка ігор

Створення ігор та візуалізацій з Pygame, Panda3D

Переваги мови Python

Простота синтаксису

Легкий для вивчення та використання

Читабельність коду

Зрозумілий та структурований синтаксис

Велика стандартна бібліотека

Багато вбудованих функцій та модулів

Кросплатформенність

Працює на Windows, macOS, Linux

Активна спільнота

Підтримка та велика кількість ресурсів

Популярність та перспективність

Високий попит на ринку праці

Огляд сучасних інтегрованих середовищ розробки (IDE) для Python

PyCharm

Професійне IDE з потужними інструментами для розробки на Python

Підтримка Django, Flask, відлагодження

Visual Studio Code

Легкий редактор коду з розширеннями для Python

Безкоштовний, кросплатформний

IDLE

Стандартне середовище розробки, що входить до Python

Простий у використанні для початківців

Thonny

IDE, спеціально розроблений для навчання програмуванню

Вбудований відлагодник та візуалізація

Spyder

Наукове середовище для аналізу даних

Інтеграція з NumPy, SciPy, Matplotlib

Jupyter Notebook

Інтерактивне середовище для створення документів

Ідеально для досліджень та візуалізації

Порівняння IDE для Python

IDE	Вартість	Функціональність	Складність використання	Спеціалізація
PyCharm	Платна (Community Edition безкоштовна)	★★★★★	Середня	Загальна розробка, веб-додатки
Visual Studio Code	Безкоштовна	★★★★☆	Низька	Універсальна, гнучка
IDLE	Безкоштовна	★★☆☆☆	Дуже низька	Початківці, базові задачі
Thonny	Безкоштовна	★★☆☆☆	Дуже низька	Навчання програмуванню
Spyder	Безкоштовна	★★★☆☆	Середня	Наукові обчислення, Data Science
Jupyter Notebook	Безкоштовна	★★★★☆	Низька	Дослідження, візуалізація даних

Інтерпретатор Python - що це таке?

Визначення інтерпретатора

Програма, яка читає та виконує код Python рядок за рядком
Перетворює код у **байт-код** та виконує його у віртуальній машині Python

Як працює інтерпретатор Python

- **Лексичний аналіз** - розбиття коду на токени
- **Синтаксичний аналіз** - перевірка структури коду
- **Компіляція** - створення байт-коду
- **Виконання** - інтерпретація байт-коду віртуальною машиною

Переваги інтерпретованої мови

- **Платформна незалежність** - код працює на будь-якій системі з інтерпретатором
- **Динамічна типізація** - гнучкість у роботі з даними
- **Інтерактивність** - можливість виконувати код по частинах
- **Простота налагодження** - зрозумілі повідомлення про помилки

Встановлення інтерпретатора Python

1

Завантаження з офіційного сайту

- Перейдіть на **python.org**
- Оберіть розділ **Downloads**
- Завантажте останню стабільну версію
- Для Windows - **Windows installer (64-bit)**

2

Процес встановлення на Windows

- Запустіть **.exe файл**
- Оберіть **Install Now**
- **Обов'язково** позначте "Add Python to PATH"
- Дочекайтеся завершення встановлення

3

Перевірка встановлення

- Відкрийте **командний рядок**
- Введіть команду: **python --version**
- Перевірте версію Python
- Запустіть інтерпретатор: **python**

```
PS C:\Users\Viktor> python --version
Python 3.13.7
PS C:\Users\Viktor> pip --version
pip 25.2 from C:\Program Files\Python313\Lib\site-packages\pip (python 3.13)
PS C:\Users\Viktor>
```

Особливості роботи з інтерпретатором Python у терміналі

Запуск інтерпретатора

- Відкрийте **командний рядок** (cmd, PowerShell)
- Введіть команду: `python` або `py`
- Для виходу: `exit()` або `quit()`
- Або натисніть **Ctrl+Z** та Enter

```
C:\Users\User> python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023,
10:12:12)
[MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

Інтерактивний режим

- **REPL**: Read-Eval-Print Loop
- Виконання коду **рядок за рядком**
- Миттєві **результати**
- Ідеально для **тестування** коду

```
>>> print("Hello, World!")
Hello, World!
>>> x = 5 + 3
>>> print(x)
8
>>>
```

Виконання скриптів

- Створіть файл з розширенням **.py**
- Для запуску: `python script.py`
- Для запуску з параметрами: `python script.py arg1 arg2`
- Доступ до аргументів через **sys.argv**

```
C:\Users\User> python hello.py
Hello from script!
C:\Users\User> python calc.py 5 3
Result: 8
C:\Users\User>
```

Основні команди

- `help()` - довідка
- `dir(object)` - атрибути об'єкта
- `type(var)` - тип змінної
- `import module` - імпорт модуля
- `__name__` - ім'я поточного модуля

```
>>> import math
>>> dir(math)[:5]
['__doc__', '__loader__', '__name__',
'__package__', '__spec__']
>>> math.sqrt(16)
4.0
>>>
```

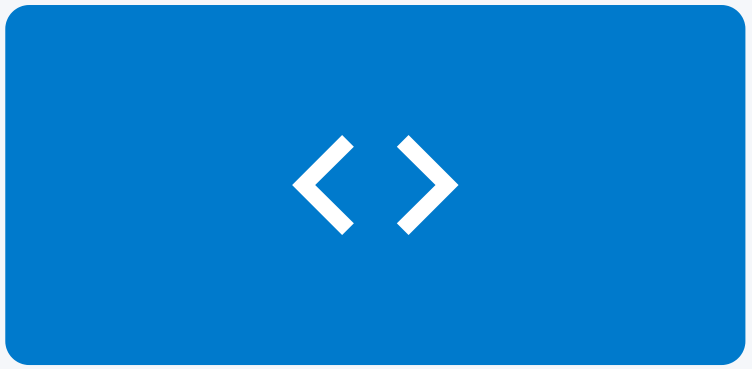
IDE Visual Studio Code - огляд

Переваги VS Code для Python

- Безкоштовний та відкритий код
- Кросплатформність (Windows, macOS, Linux)
- Швидкість роботи та низькі системні вимоги
- Потужна система розширень
- Інтеграція з системами контролю версій

Основні функції

- Інтелектуальне автодоповнення коду
- Вбудований відлагоджувач
- Підсвічування синтаксису
- Інтеграція з терміналом
- Підтримка Jupyter Notebook
- Робота з віртуальними оточеннями



Системні вимоги

ОС: Windows 7 SP1+, macOS 10.11+, Linux
Процесор: 1.6 GHz або швидший
RAM: 2 ГБ (рекомендовано 4+ ГБ)
Місце на диску: 200 МБ
Відеокарта: Підтримка апаратного прискорення

Необхідні розширення

Python - основне розширення для роботи з Python
Pylance - покращене автодоповнення
Jupyter - для роботи з Jupyter Notebook

Встановлення та налаштування Visual Studio Code для Python

1

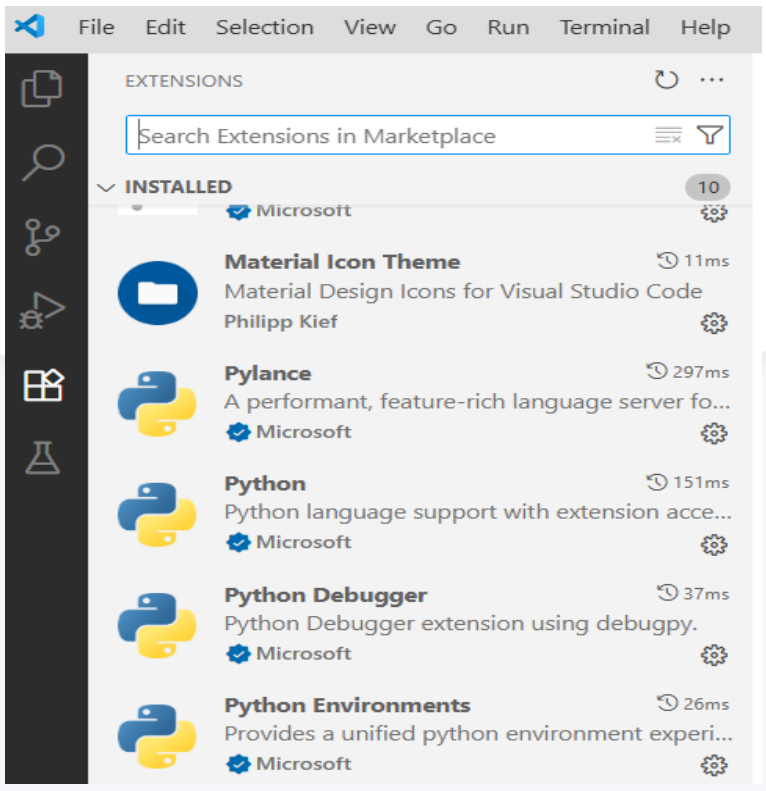
Завантаження та встановлення VS Code

- Перейдіть на **code.visualstudio.com**
- Завантажте версію для вашої ОС
- Запустіть **інсталятор**
- Дотримуйтесь інструкцій на екрані

2

Встановлення розширення Python

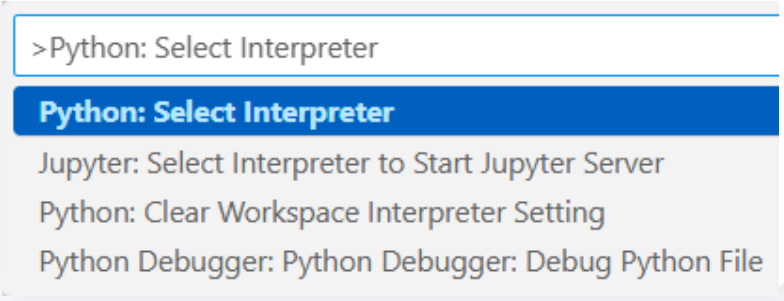
- Відкрийте **VS Code**
- Перейдіть до розділу **Розширення** (Ctrl+Shift+X)
- Знайдіть **Python** від Microsoft
- Натисніть **Встановити**



3

Налаштування середовища

- Відкрийте **палітру команд** (Ctrl+Shift+P)
- Введіть **Python: Select Interpreter**
- Оберіть інтерпретатор Python
- Створіть або відкрийте файл з розширенням **.py**



Віртуальні оточення в Python

Що таке віртуальне оточення

Ізольоване середовище для роботи з Python
Незалежний простір зі своїми:

- Встановленими пакетами
- Конфігураціями
- Версіями Python

Навіщо потрібні віртуальні оточення

- **Ізоляція проектів** - кожен проект має свої залежності
- **Уникнення конфліктів** між версіями пакетів
- **Чистота системи** - глобальне середовище залишається незмінним
- **Портативність** - легке розгортання проектів

Типи віртуальних оточень

- **venv** - стандартний модуль Python 3.3+
- **virtualenv** - зовнішній пакет з розширеними можливостями
- **conda** - для управління пакетами та оточеннями
- **pipenv** - поєднання pip та virtualenv

Налаштування віртуального оточення в терміналі Windows

Команди для створення віртуального оточення

`python -m venv myenv` Створення оточення **myenv**

`python -m venv C:\path\to\myenv` Створення в **конкретній папці**

```
C:\Projects> python -m venv myenv
Successfully created virtual environment at C:\Projects\myenv
```

Активація/деактивація оточення

`myenv\Scripts\activate` **Активація** віртуального оточення

`deactivate` **Деактивація** віртуального оточення

```
C:\Projects> myenv\Scripts\activate
(myenv) C:\Projects>
(myenv) C:\Projects> deactivate
C:\Projects>
```

Встановлення пакетів у віртуальному оточенні

`pip install package_name` Встановлення **пакета**

`pip install -r requirements.txt` Встановлення з **файлу залежностей**

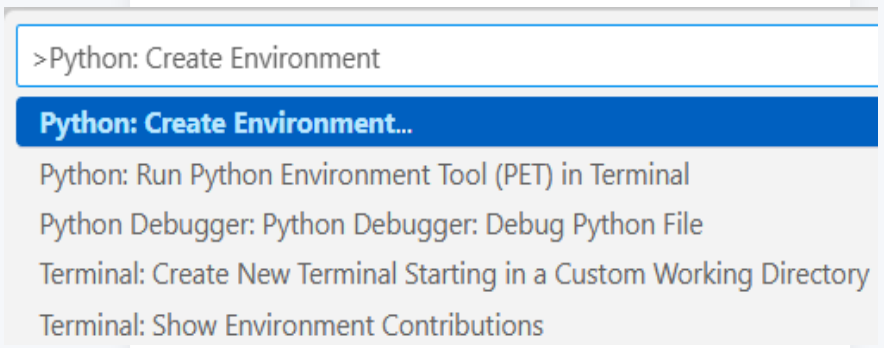
`pip freeze > requirements.txt` Збереження **списку пакетів**

```
(myenv) C:\Projects> pip install numpy
Collecting numpy
Successfully installed numpy-1.24.3
```

Налаштування віртуального оточення в Visual Studio Code

Створення віртуального оточення в VS Code

1. Відкрийте **Command Palette** (Ctrl+Shift+P)
2. Введіть **Python: Create Environment**
3. Оберіть тип оточення (**Venv** або **Conda**)
4. Вкажіть **інтерпретатор** для використання

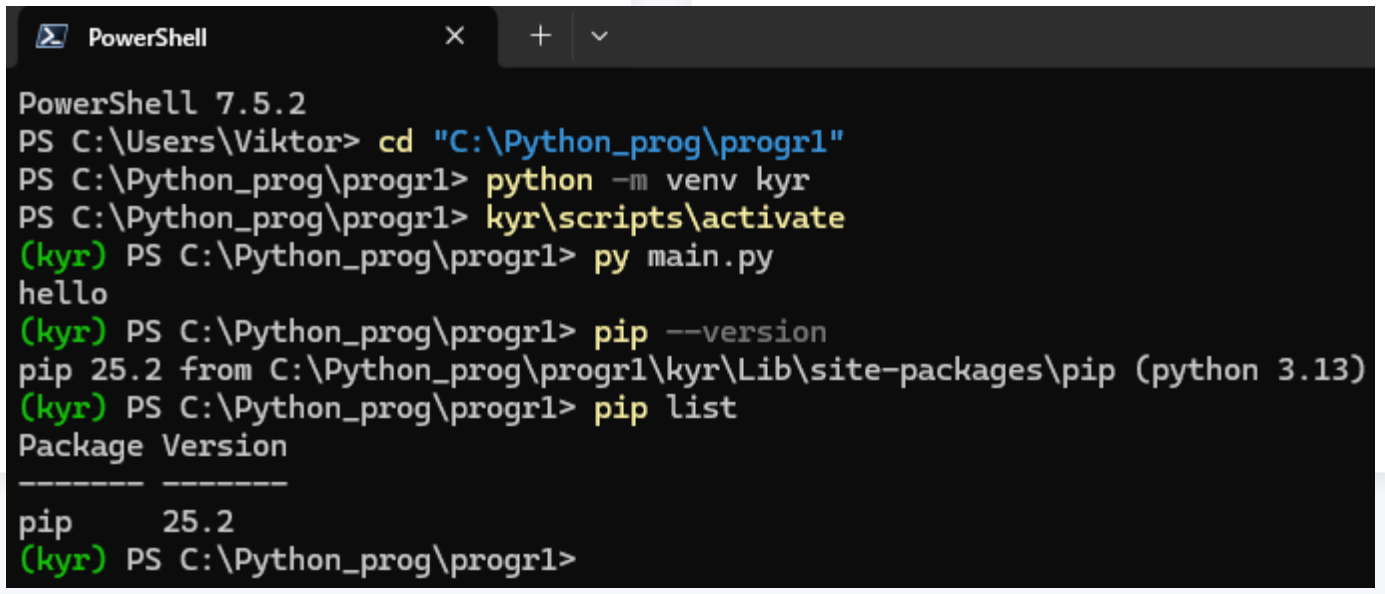


Вибір інтерпретатора

1. Натисніть на версію Python у **нижній панелі** VS Code
2. Або через **Command Palette**: **Python: Select Interpreter**
3. Оберіть потрібний **інтерпретатор** зі списку
4. VS Code автоматично **активує** вибране оточення

Робота з залежностями

1. Відкрийте **термінал** у VS Code (Ctrl+`)
2. Використовуйте **pip install package_name** для встановлення
3. Створіть **requirements.txt** для фіксації залежностей
4. Використовуйте **pip install -r requirements.txt** для відновлення



Створення проекту та простої програми на Python

Структура проекту

- **my_project/** - головна папка проекту
- **main.py** - головний файл програми
- **modules/** - папка з модулями
- **requirements.txt** - залежності проекту
- **README.md** - документація проекту

Створення простої програми "Hello, World!"

Створіть файл **main.py** та додайте код:

```
# Проста програма Hello, World!
print("Hello, World!")
```

Для запуску програми:

```
python main.py
```

Результат виконання:

```
Hello, World!
```

Основні конструкції мови

- Змінні та типи даних
- Умовні оператори
- Цикли
- Функції
- Списки та кортежі
- Словники

Приклад простої програми

```
# Програма для розрахунку площі прямокутника
def calculate_area(width, height):
    return width * height

# Введення даних від користувача
w = float(input("Введіть ширину: "))
h = float(input("Введіть висоту: "))

# Розрахунок та вивід результату
area = calculate_area(w, h)
print(f"Площа прямокутника: {area}")
```