

**Олексій Васильєв**

**Програмування в PYTHON**  
**Теорія і практика**

Навчальний посібник

Київ  
Видавництво Ліра-К  
2023

УДК 004.43Python(075.8)

B19

*Рецензенти*

*Чалий Олександр Васильович*, доктор фізико-математичних наук, професор, член-кореспондент НАПН України, завідувач кафедри медичної і біологічної фізики та інформатики Національного медичного університету ім. О.О. Богомольця.

*Кондратенко Сергій Вікторович*, доктор фізико-математичних наук, професор кафедри оптики фізичного факультету Київського національного університету імені Тараса Шевченка.

*Гаврюшенко Дмитро Анатолійович*, доктор фізико-математичних наук, професор, завідувач кафедри молекулярної фізики фізичного факультету Київського національного університету імені Тараса Шевченка.

**Васильєв О.М.**

**B19** Програмування в Python. Теорія і практика : навч. посіб. Київ : Видавництво Ліра-К, 2023. 462 с.  
ISBN 978-617-520-513-6

Посібник присвячено мові програмування Python. У ньому розглядається широке коло тем, в тому числі принципи створення програм, робота з даними, керуючі інструкції, базові оператори, обговорюються принципи об'єктно-орієнтованого програмування. Також читач дізнається про оброблення помилок, реалізацію потоків, створення програм з графічним інтерфейсом.

Видання містить велику кількість прикладів і буде цікавою та корисною студентам, викладачам, а також усім, хто вивчає мову програмування Python.

**УДК 510.5:004.4](075.8)**

**ISBN 978-617-520-513-6**

© Васильєв О.М., 2023

© Видавництво Ліра-К, 2023

# ЗМІСТ

## **ВСТУП. Книга про мову програмування Python**

Мова Python.....	3
Особливості книги.....	4
Програмне забезпечення.....	5
Про автора.....	9
Зворотній зв'язок.....	9
Подяки.....	9

## **РОЗДІЛ 1. Знайомство з Python**

Перша програма.....	10
Використання різних середовищ розроблення.....	15
Знайомство зі змінними.....	21
Уведення значення в програму.....	25
Функція eval().....	29
Знайомство зі списками.....	31
Знайомство з умовним оператором.....	37
Знайомство з оператором циклу.....	39
Знайомство з функціями.....	43
Резюме.....	46
Завдання для самостійної роботи.....	47

## **РОЗДІЛ 2. Основні операції**

Оператор циклу while.....	48
Оператор циклу for.....	56
Умовний оператор if.....	62
Тернарний оператор.....	73
Оброблення виняткових ситуацій.....	78
Резюме.....	87
Завдання для самостійної роботи.....	88

## **РОЗДІЛ 3. Списки та кортежі**

Знайомство з кортежами.....	90
Основні операції зі списками та кортежами.....	95
Створення вибірки на основі списків та кортежів.....	101
Вкладені списки та кортежі.....	108
Копіювання списків і кортежів.....	113
Функції і методи для роботи зі списками.....	117
Резюме.....	124
Завдання для самостійної роботи.....	125

**РОЗДІЛ 4. Множини та словники**

Знайомство з множинами .....	127
Операції з множинами .....	131
Приклади використання множин.....	138
Знайомство зі словниками.....	143
Операції зі словниками.....	149
Резюме .....	155
Завдання для самостійної роботи .....	155

**РОЗДІЛ 5. Робота з текстом**

Текстові літерали.....	157
Основні операції з текстом.....	168
Методи для роботи з текстом.....	171
Приклади роботи з текстом.....	180
Резюме .....	184
Завдання для самостійної роботи .....	185

**РОЗДІЛ 6. Функції**

Оголошення й виклик функції.....	187
Іменовані аргументи функції.....	195
Механізм передачі аргументів.....	196
Значення аргументів за замовчуванням.....	199
Функції з довільною кількістю аргументів.....	202
Локальні й глобальні змінні.....	204
Вкладені функції.....	206
Лямбда-функції.....	208
Функція як аргумент і результат.....	210
Рекурсія .....	213
Декоратори функцій.....	215
Функції-генератори .....	218
Анотації й документування в функціях.....	221
Резюме .....	224
Завдання для самостійної роботи .....	226

**РОЗДІЛ 7. Файли та дані**

Числові дані.....	227
Логічні значення.....	238
Дата та час.....	240
Робота з файлами.....	247
Резюме .....	257
Завдання для самостійної роботи .....	258

**РОЗДІЛ 8. Класи та об'єкти**

Концепція класів та об'єктів .....	259
Описання класів і створення об'єктів .....	261
Конструктори й деструктори .....	267
Об'єкт реалізації класу .....	270
Операції з атрибутами класів та об'єктів.....	278
Копіювання об'єктів.....	283
Документування та декоратори .....	285
Використання класів і об'єктів .....	290
Резюме .....	299
Завдання для самостійної роботи .....	301

**РОЗДІЛ 9. Спадкування та спеціальні методи**

Знайомство зі спадкуванням .....	303
Множинне спадкування .....	309
Перевизначення методів при спадкуванні.....	312
Приведення типів .....	323
Перевантаження операторів .....	326
Доступ до атрибутів .....	335
Індексування об'єктів .....	343
Виклик об'єктів .....	346
Ітератори.....	348
Резюме .....	354
Завдання для самостійної роботи .....	354

**РОЗДІЛ 10. Оброблення винятків і потоки**

Принципи оброблення винятків.....	356
Оброблення винятків різних типів .....	360
Використання об'єкта винятку .....	362
Вкладені блоки для оброблення винятків.....	363
Штучне генерування винятків .....	366
Створення класів винятків.....	368
Використання винятків .....	370
Знайомство з потоками .....	377
Взаємодія потоків .....	386
Приклади використання потоків.....	393
Резюме .....	399
Завдання для самостійної роботи .....	400

**РОЗДІЛ 11. Програми з графічним інтерфейсом**

Створення простого вікна.....	402
Вікно з міткою та кнопкою .....	403
Використання текстового поля .....	406
Динамічний список .....	410
Опції, перемикачі та інші компоненти.....	416
Використання меню .....	430
Робота з графікою.....	445
Резюме .....	454
Завдання для самостійної роботи .....	455

<b>ЗАКЛЮЧЕННЯ. Python та програмування .....</b>	<b>457</b>
--	------------

## Вступ

# Книга про мову програмування Python

*Не варто вивчати мову, яка не змінює вашого уявлення про програмування.*

Алан Перліс

На сьогодні існує багато різних мов програмування. Деякі з них популярні, деякі – не надто. Зазвичай популярність мови визначають за кількістю програмістів, які використовують її у своїй роботі на постійній основі, або за запитами роботодавців, які шукають співробітників-програмістів. Довгі роки традиційно популярними є мови програмування Java, C++, C#, JavaScript та PHP. Останнім часом у цій чудовій компанії все частіше згадують мову програмування Python. Навіть більше – за деякими опитуваннями мова Python уже займає лідируючі позиції. Саме цій мові присвячена книга.

## Мова Python

Тенденції такі, що навіть якщо мова Python і не є найпопулярнішою на сьогоднішній день, то все одно немає сумнівів у тому, що масштаби її застосування постійно зростають. Відповідно, збільшується попит на програмістів, які працюють з мовою Python. Така зростаюча популярність мови багато в чому пояснюється її простотою, красою та ефективністю. Спектр задач, які розв'язуються з використанням Python, вражаючий. Тому вивчення Python — вибір розумний і багатообіцяючий.

Чим же примітна мова Python? Що в ній особливого? Відповіді на ці запитання не такі вже й прості. Тим більше що багато залежить від того, з якою мовою ми порівнюватимемо. Серед найважливіших характеристик мови Python можна виділити наступні:

- *Мова інтерпретована.* При першому запуску програми на виконання для неї створюється проміжний код. Саме проміжний код використовується при виконанні програми. Якщо потім у програму вносяться зміни, то при черговому запуску програми створюється новий проміжний код.



### *До уваги*

Мови бувають інтерпретованими та компільованими. Якщо програма компілюється, то на основі вихідного коду створюється виконавчий (машинний) код, який і виконується при запуску програми. Якщо йдеться про інтерпретовану мову, то

програма, написана цією мовою, виконується рядок за рядком, без попередньої компіляції. Існує також проміжний варіант — дещо середнє між компілюванням і інтерпретацією. У такому випадку вихідний код програми перетворюється в проміжний код, який уже потім інтерпретується при виконанні.

Для інтерпретованих мов властивий більший "демократизм" у питанні описання і оброблення даних. Програми, написані компільованими мовами, характеризуються відносно високою швидкістю виконання.

- У плані синтаксису мова Python проста й лаконічна. Вона не містить надлишкових конструкцій. З іншого боку, мова строга: навіть зайвий пробіл у програмному коді може призвести до помилки.

- Мова Python підтримує парадигму об'єктно-орієнтованого програмування (ООП). При цьому також можна створювати програми, не використовуючи класи та об'єкти.



### *До уваги*

Концепція ООП, яка реалізується в мові Python, може стати сюрпризом для читачів, знайомих з такими мовами програмування, як Java, C++ та C#. Навпаки, ті, хто знайомий з мовою JavaScript, знайдуть для себе деякі знайомі моменти.

- Мова Python зручна для створення застосунків з графічним інтерфейсом.

- Не останнім фактором, який сприяє популярності мови Python, є велика й дружня спільнота розробників, котрі використовують цю мову. Немає нестачі у вільно розповсюджуваних програмних продуктах (зокрема, в середовищах розроблення), які полегшують знайомство з мовою Python і її використання.

Ми представили лише досить загальний і короткий перелік переваг і особливостей мови. У деталі заглибимося в основній частині книги, коли розглядатимемо конкретні приклади й синтаксичні конструкції.

## Особливості книги

Мета книги — навчити читача програмувати мовою Python. Але навчатися можна по-різному. Скажімо, можна слухати лекції в університеті, можна відвідувати курси з програмування, а можна спробувати навчитися самостійно. Останній варіант – найважчий, оскільки зазвичай поруч немає порадника, який міг би підказати або пояснити. Ось саме для цього "складного" випадку в першу чергу й призначена книга. Зрозуміло, що зовсім уникнути "крутих поворотів" при "прокладці маршруту" щодо вивчення мови Python не вийде. Але ми спробуємо їх кількість звести до мінімуму.

Досвід показує, що легше за все різні концепції програмування й підходи засвоюються, коли вони проілюстровані прикладами. Саме така методика використана в книзі. Принципове завдання, яке при цьому вирішується –



донести до читача основну ідею, причому не просто на деякому абстрактному рівні, а на рівні її прикладної реалізації за допомогою програмного коду. Теоретичні відомості наводяться в мінімальному обсязі, але, разом з тим, достатньому для якісного засвоєння матеріалу.

Структура книги така, що в першому розділі наводиться короткий огляд основних синтаксичних конструкцій мови Python. Це дасть змогу читачеві практично відразу, ще до завершення читання книги, приступити до створення нескладних, але цілком функціональних програмних кодів. Цей прийом застосовувався не один раз у книгах, присвячених іншим мовам програмування, і отримав непогані відгуки читачів. Так що є підстави вважати, що він буде корисним і при вивченні мови Python.

Розділи після першого присвячені детальнішому розгляду питань, пов'язаних з ефективним програмуванням у Python. Серед розглянутих тем: робота з даними різних типів, керуючі інструкції, списки та кортежі, множини та словники, робота з текстом, створення функцій, операції з файлами, робота з класами та об'єктами, спадкування та спеціальні методи, оброблення виняткових ситуацій, створення потоків і багато іншого. Останній розділ книги містить корисну інформацію, яка стосується створення застосунків з графічним інтерфейсом (з використанням бібліотеки Tkinter). Для зручності засвоєння матеріалу кожен розділ закінчується коротким резюме, в яке винесені основні положення, розглянуті й обговорені у відповідному розділі. Також кожен розділ містить список задач для самостійної роботи.



### *До уваги*

Матеріал від розділу до розділу ускладнюється поступово. Деякі важливі моменти досить часто повторюються (в різному контексті), особливо в початкових розділах. Іноді одні й ті самі (або схожі) задачі розв'язуються різними методами. Усе це зроблено навмисно. Мета проста – полегшити процес засвоєння інформації та сформуванню основи для розуміння принципів програмування в Python.

## Програмне забезпечення

Для складання програмних кодів мало знати мову програмування (в нашому випадку Python). Знадобиться ще й певне програмне забезпечення. Яке саме? Не завадила б програма-редактор для набору програмних кодів. Хоча власне програмний код ми можемо набирати хоч у текстовому редакторі, на кшталт Notepad. Для цього нам достатньо створити пустий текстовий документ, увести в нього команди відповідно до правил мови Python, і зберегти з розширенням `.py` (стандартне розширення для файлів з програмами мовою Python).



### До уваги

Окрім розширення `.py`, файли з Python-програмами можуть мати розширення `.pyw`, якщо маємо справу з програмами, в яких використовується графічний інтерфейс (в операційній системі Windows). У файлів, пов'язаних з Python-проектами можуть бути й інші розширення. Наприклад, розширення `.pyc` мають файли зі скомпільованим проміжним кодом (файли з байт-кодом). Оптимізований байт-код зберігається в файлі з розширенням `.pyo`, а розширення `.pyd` використовується для файлів з бінарним кодом динамічних `dll`-бібліотек у операційній системі Windows.

Але навіть якщо ми так вчинимо, цього все-таки буде недостатньо. Нам ще, мінімум, знадобиться програма-інтерпретатор, яка зможе виконувати команди, написані мовою Python. Іншими словами, нам знадобиться спеціальна програма, яка зможе зрозуміти код, який ми написали мовою Python, і виконати цей код. Як уже зазначалося, такі програми називаються *інтерпретаторами*. Тому обійтися зовсім без спеціального програмного забезпечення в нас не вийде. А оскільки програмне забезпечення все одно доведеться встановлювати, то розумно скористатися всім спектром можливостей, доступних розробнику мовою Python. Тим більше що запропоновані для програмування на Python засоби розроблення ефективні й часто безкоштовні.

Найрозумніший підхід при створенні програми мовою Python полягає в тому, щоб використати *інтегроване середовище розроблення* (скорочено *IDE* від *Integrated Development Environment*). Середовище розроблення – це спеціальний застосунок, який дає змогу набирати, налаштовувати і запускати на виконання програмні коди. Фактично, середовище розроблення об'єднує в собі відразу декілька програм. І це досить зручно, оскільки найрізноманітніші завдання, починаючи з набору коду і до налаштування та запуску його на виконання, реалізуються через одну універсальну програму. Як би там не було, але середовище розроблення — це розумно й зручно. Тому загальна рекомендація полягає в тому, щоб використати його. Питання лише в тому, яке саме.

Середовищ розроблення для мови Python доволі багато. Тут ми коротко зупинимося лише на деяких, найпопулярніших (і безкоштовних). Але перед тим, як перейти до обговорення середовищ розроблення, зробимо декілька зауважень відносно усього процесу встановлення програмного забезпечення, необхідного для програмування на Python.

У першу чергу необхідно встановити програму-інтерпретатор (і деякі супутні утиліти). Для цього є сенс перейти на сторінку підтримки мови [www.python.org](http://www.python.org). Сторінка традиційно містить багато корисної інформації. Там, крім іншого, в розділі завантажень **Downloads** (адреса [www.python.org/downloads](http://www.python.org/downloads)) можна знайти призначене для програмування на Python програмне забезпечення. Веб-сторінка з ресурсами, призначеними для завантаження, представлена на рис. В.1.

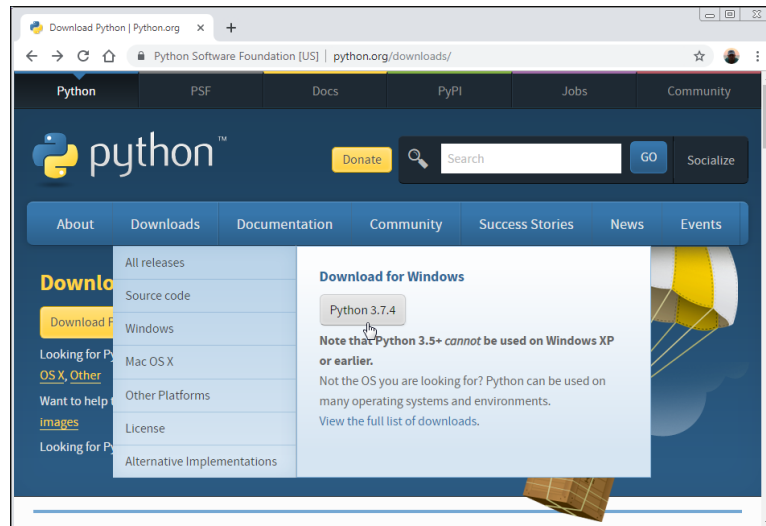


Рис. В.1. Сторінка [www.python.org/downloads](http://www.python.org/downloads) для завантаження програмного забезпечення для програмування на Python

Слід завантажити відповідні файли й виконати встановлення. Процес установлення простий та інтуїтивно зрозумілий, тому особливих коментарів не вимагає і зазвичай проходить без проблем. Слід зазначити, що в цьому випадку автоматично буде встановлене й середовище розроблення, яке називається *IDLE*. Це просте й надійне середовище, яке цілком придатне для ефективної роботи з програмними кодами мовою Python.



### До уваги

Методи роботи з середовищем *IDLE*, так само як і з іншими середовищами розроблення, стисло описані в першому розділі.

Якщо читача з якихось причин середовище *IDLE* не влаштовує, можна скористатися іншим середовищем. Добре, що вибір досить великий.



### До уваги

Зазвичай середовища розроблення встановлюються без інтерпретатора, тому рекомендується спочатку встановити інтерпретатор (наприклад, завантаживши файли з сайту [www.python.org](http://www.python.org)), а вже після цього встановлювати середовище розроблення. У такому випадку налаштування середовища, пов'язані з інтерпретатором, скоріш за все, будуть виконані автоматично.

Компанія JetBrains пропонує для розробників на Python середовище розроблення, яке називається PyCharm. Інформацію про це середовище розроблення (а також про інші програмні продукти компанії JetBrains) представлено на сайті [www.jetbrains.com](http://www.jetbrains.com). На рис. В.2 на сторінці підтримки відкритий розділ **Tools**, в якому є посилання на завантаження файлів для встановлення середовища розроблення PyCharm.

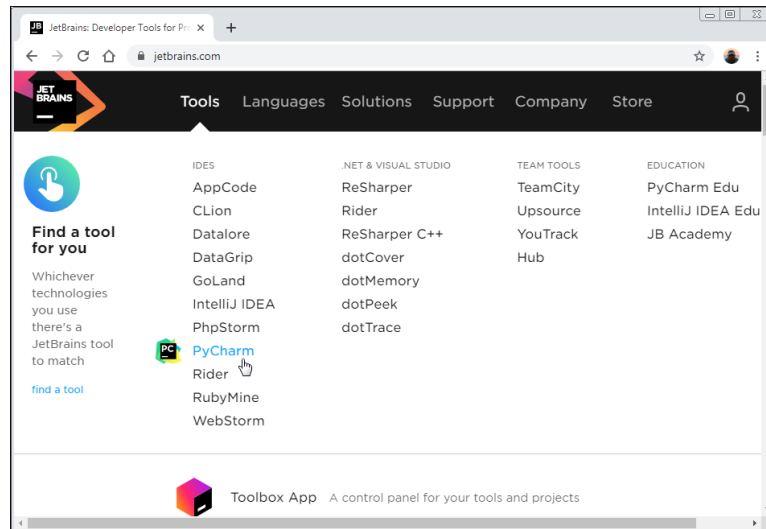


Рис. В.2. Сторінка [www.jetbrains.com](http://www.jetbrains.com) з посиланням для завантаження файлів для встановлення середовища розроблення PyCharm

Процес встановлення середовища досить простий. Це зручне й ефективно середовище розроблення. Правда, процес створення застосунків (у порівнянні з тим, як все відбувається при використанні інших середовищ) може видатися дещо заплутаним, хоча це, безумовно, суб'єктивна думка. Разом з тим, середовище PyCharm є, мабуть, оптимальним вибором при роботі з Python.

Досить зручним і функціональним є середовище розроблення Visual Studio Code (продукт компанії Microsoft). На рис. В.3 показано вікно браузера, в якому відкрита сторінка <https://code.visualstudio.com> підтримки проекту.

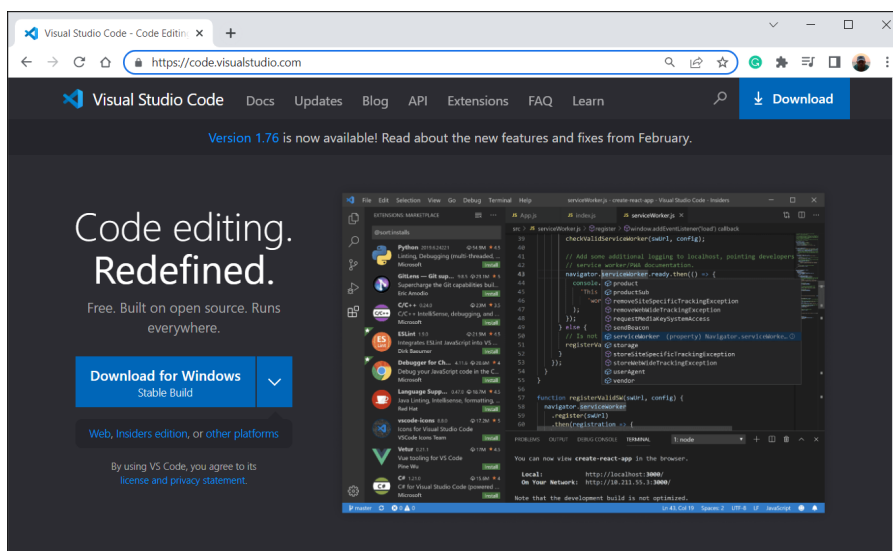


Рис. В.3. Сторінка <https://code.visualstudio.com> з посиланням для завантаження файлів для встановлення середовища розроблення Visual Studio Code

Середовище розроблення Visual Studio Code просте у використанні й містить усі основні утиліти, необхідні для ефективного програмування на Python.

Характеризуючи ситуацію в цілому, слід зазначити, що різні середовища розроблення надають користувачу практично однаковий "набір послуг". Принаймні на початковому етапі, коли читач тільки знайомитиметься з мовою програмування Python, немає принципової відмінності в тому, яке саме середовище розроблення використовувати. Так що тут скоріше питання естетики, а не ефективності.

### До уваги

Ситуація з середовищами розроблення "динамічна": якісь середовища стають популярними, а якісь відходять на другий план. Тому слід розуміти, що перелік доступних або пріоритетних середовищ, є досить умовним.

## Про автора

Автор книги, *Васильєв Олексій Миколайович*, доктор фізико-математичних наук, професор кафедри програмних систем і технологій факультету інформаційних технологій Київського національного університету імені Тараса Шевченка. Автор книг з математичного моделювання та програмування мовами C, C++, C#, Java, JavaScript, Python, PHP. Сфера наукових інтересів: програмування та інформаційні технології, теоретична фізика, біофізика, синергетика, математична економіка, моделювання соціально-політичних процесів, математична лінгвістика.



## Зворотній зв'язок

Висловити свої зауваження та пропозиції стосовно цієї або інших книг автора можна за електронною поштою [oleksii@vasyliiev.kyiv.ua](mailto:oleksii@vasyliiev.kyiv.ua). Автор заздалегідь вдячний своїм читачам за конструктивну критику. Інформацію про вже видані книги, а також деякі корисні матеріали, що стосуються цих книг (наприклад, програмні коди прикладів), можна знайти на сайті [www.vasyliiev.kyiv.ua](http://www.vasyliiev.kyiv.ua) та на YouTube-каналі [www.youtube.com/channel/UClyL2-UbbWXd7S1PHPefM0g](http://www.youtube.com/channel/UClyL2-UbbWXd7S1PHPefM0g).



## Подяки

Книги пишуться для того, щоб їх читали. Найкращий стимул – усвідомлення того, що твоя праця комусь потрібна. Користуючись нагодою, хочу висловити найщирішу подяку своїм читачам: за цікавість до книг, за критичні зауваження, за бажання ставати кращими й розумнішими.

## Розділ 1

# Знайомство з Python

*Найважливіша річ у мові програмування – назва. Мова не матиме успіху без гарного імені.*

Дональд Кнут

У цьому розділі відбудеться наше знайомство з мовою програмування Python. Ми створимо першу програму цією мовою й розглянемо (стисло) основні підходи та синтаксичні конструкції, характерні для Python. Зокрема, в цьому розділі ми дізнаємося, як створюється програма мовою Python, що таке змінні та як вони використовуються, познайомимося зі списками, навчимося розраховувати вирази. Також ми дізнаємося, що таке умовний оператор і як він використовується. Ми познайомимося з оператором циклу й навчимося створювати функції. Плани в нас великі, тому відразу перейдемо до справи. Почнемо зі створення дуже простої програми.

## Перша програма

Для написання першої програми нам необхідно знати, по-перше, які правила створення програм мовою Python і, по-друге, як і де набрати програмний код, та що з ним робити далі. Ми дамо відповіді на обидва цих запитання.

Отже, програма – це набір команд або інструкцій. Створення програми, яка відображає у вікні виведення певне повідомлення – простий приклад, з якого зазвичай починають вивчення будь-якої мови програмування. У мові програмування Python така програма буде до непристойності простою, оскільки складатиметься всього з однієї команди. Відповідний код наведено в лістингу 1.1.

### Лістинг 1.1. Перша програма

```
print("Починаємо вивчати мову Python.")
```

Команда, з якої складається програма, є інструкцією виклику вбудованої функції `print()`. Аргументом функції передається текстове значення (текстовий літерал) "Починаємо вивчати мову Python."



### До уваги

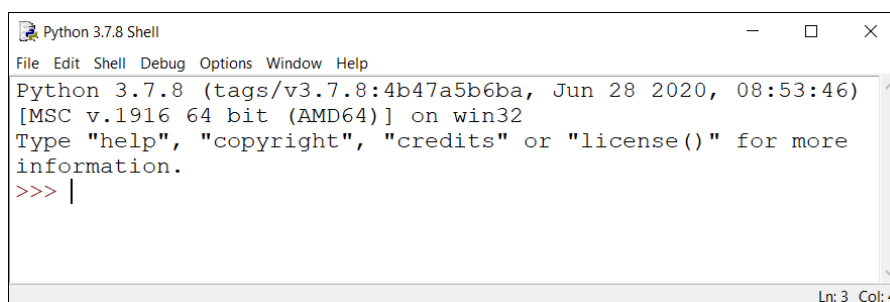
Текстові літерали в Python беруться в подвійні або одинарні лапки.

Текстовий аргумент, переданий у функцію `print()`, при виконанні відповідної команди відображається у вікні виведення інтерпретатора Python. Тому при виконанні програми з лістингу 1.1 ми маємо побачити таке повідомлення:

### Результат виконання програми (з лістингу 1.1)

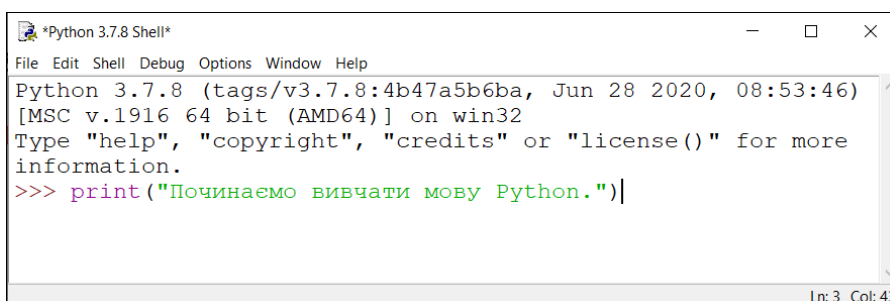
Починаємо вивчати мову Python.

Питання лише в тому, де саме ми побачимо це повідомлення. Існує декілька варіантів для виведення і виконання програмного коду. Багато залежить від середовища розроблення, яке використовується. Але в будь-якому випадку спочатку це середовище необхідно запусити. Якщо йдеться про середовище IDLE, то має відкритися вікно, представлене на рис. 1.1.



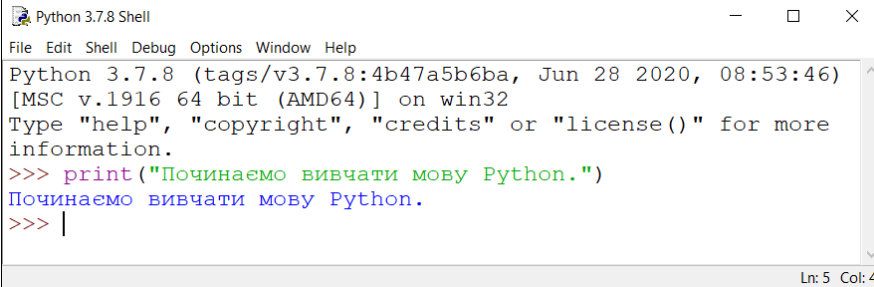
*Рис. 1.1. Вікно середовища розроблення IDLE*

Вікно, яке з'являється на екрані, є вікном оболонки інтерпретатора. Його характерна особливість – наявність потрійної стрілки `>>>`, яка є індикатором рядка введення команди. Справа від цього індикатора блимає курсор. У цьому місці можна вводити команди. Зокрема, ми можемо ввести туди команду `print("Починаємо вивчати мову Python.")`, як це показано на рис. 1.2.



*Рис. 1.2. Вікно оболонки інтерпретатора з уведеною командою*

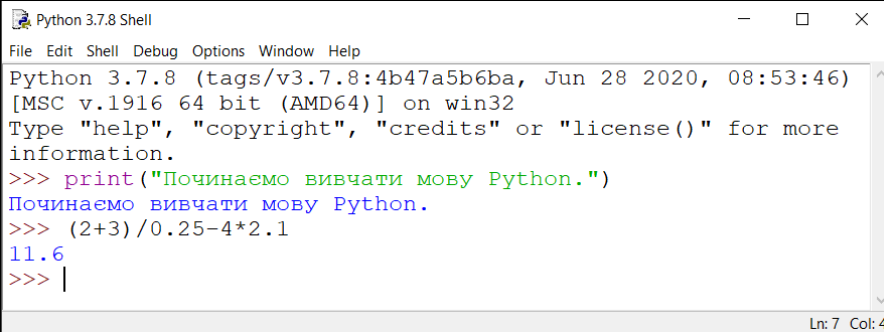
Для виконання команди слід натиснути клавішу `<Enter>`. Яким буде результат, показано на рис. 1.3.



```
Python 3.7.8 Shell
File Edit Shell Debug Options Window Help
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46)
[MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>> print("Починаємо вивчати мову Python.")
Починаємо вивчати мову Python.
>>> |
```

Рис. 1.3. Результат виконання команди у вікні оболонки інтерпретатора

Бачимо, що внизу під уведеною командою з'явилося повідомлення, строго у відповідності до значення, переданого аргументом функції `print()`. А ще рядком нижче з'являється новий індикатор уведення команди. Там може бути введена нова команда, і після натискання клавіші `<Enter>` команда буде виконана. На рис. 1.4 представлено результат розрахунку арифметичного виразу  $\frac{2+3}{0.25} - 4 \cdot 2.1$  (значення виразу дорівнює 11.6), якому відповідає команда  $(2+3)/0.25-4*2.1$ .



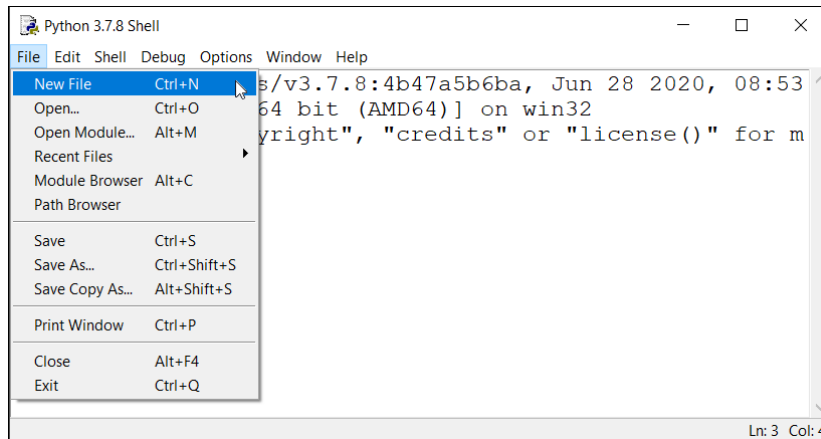
```
Python 3.7.8 Shell
File Edit Shell Debug Options Window Help
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46)
[MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>> print("Починаємо вивчати мову Python.")
Починаємо вивчати мову Python.
>>> (2+3)/0.25-4*2.1
11.6
>>> |
```

Рис. 1.4. Результат розрахунку арифметичного виразу

Фактично, ми можемо використовувати вікно оболонки інтерпретатора мови Python як калькулятор із розширеними можливостями. Але нам потрібно не це. Ми підемо іншими шляхом.

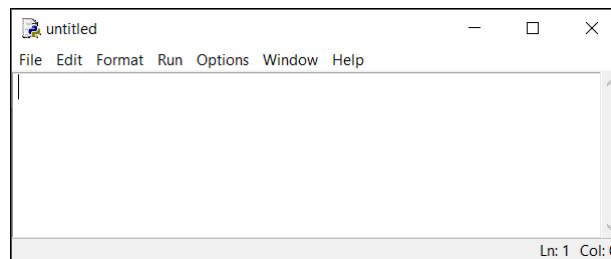
Оскільки в подальшому ми плануємо створювати програми, які складаються більше ніж з однієї команди, то кожен таку програму будемо записувати в окремий файл, а вже потім з файлу програма запускатиметься на виконання. Розглянемо весь процес, починаючи від створення файлу з програмою і запуску програми на виконання, на прикладі програми з лістингу 1.1. Учинимо так: у вікні оболонки інтерпретатора в меню **File** вибираємо команду **New File** (або натискаємо комбінацію клавіш `<Ctrl>+<N>`), як показано на рис. 1.5.





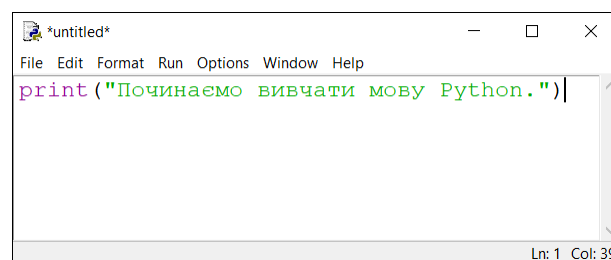
*Рис. 1.5. Створення нового файлу з кодом у вікні оболонки інтерпретатора*

Відкриється вікно редактора кодів. Пусте вікно (яке не містить команд) представлено на рис. 1.6.



*Рис. 1.6. Пусте вікно редактора кодів*

У вікні редактора вводимо команди програми. У нашому випадку програма складається з однієї команди `print("Починаємо вивчати мову Python.")`. Саме її вводимо у вікно, як показано на рис. 1.7.



*Рис. 1.7. Вікно редактора кодів з уведеною командою програми*

Після того, як код програми введено, зберігаємо файл з програмою. Для цього в меню **File** вікна редактора кодів вибираємо команду **Save** (або використовуємо комбінацію клавіш `<Ctrl>+<S>`). Процес проілюстровано на рис. 1.8.

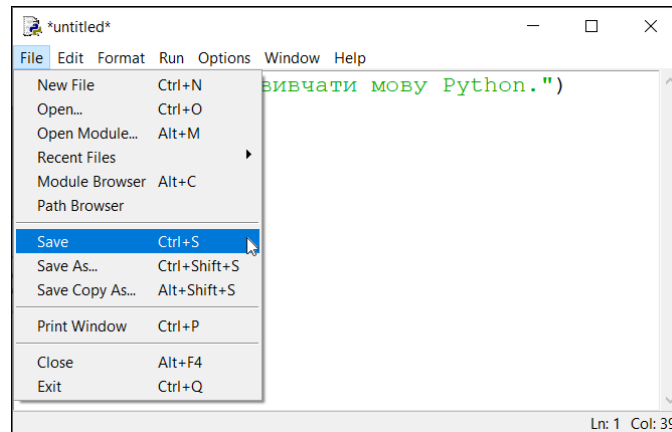


Рис. 1.8. Зберігаємо файл з програмою

Далі слід вибрати місце, де зберігатиметься файл з програмою, і назву для файлу. Для операційної системи Windows файли з програмними кодами, написаними мовою Python, зберігаються з розширенням `.py` (якщо не йдеться про застосунки з графічним інтерфейсом – для застосунків з графічним інтерфейсом використовується розширення `.pyw`).

Для запуску програми на виконання у вікні редактора кодів (з відкритим файлом програми) у меню **Run** вибираємо команду **Run Module** або натискаємо клавішу `<F5>`, як показано на рис. 1.9.

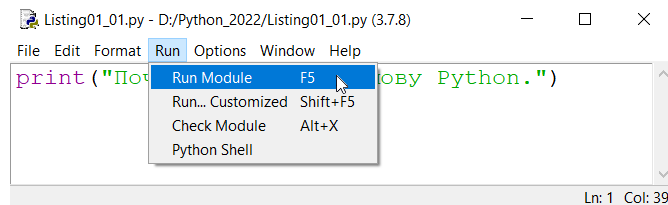


Рис. 1.9. Запуск програми на виконання

Як наслідок, у вікні оболонки інтерпретатора відображається результат виконання програми. Ситуацію ілюструє рис. 1.10.

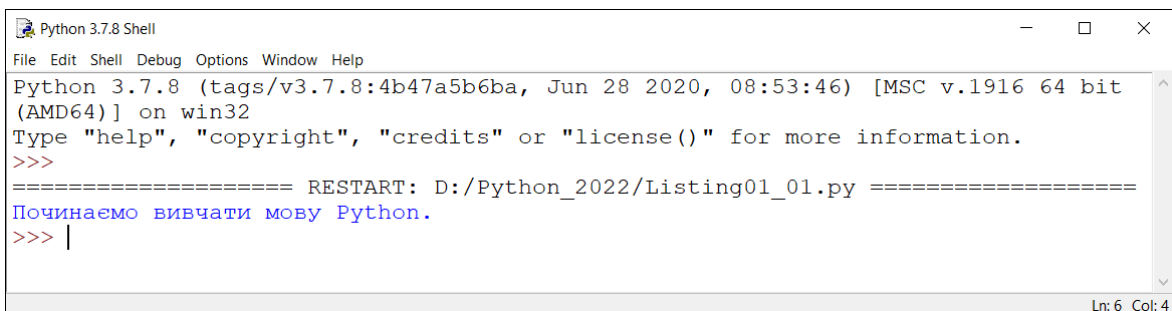


Рис. 1.10. Результат виконання програми

У загальних рисах це ті дії, які необхідно виконати для створення нескладної програми мовою Python у випадку, коли ми використовуємо середовище IDLE.

### До уваги

Якщо ми хочемо відкрити вже існуючий файл з програмою, то в меню **File** вікна оболонки інтерпретатора можна вибрати команду **Open** (або скористатися комбінацією клавіш <Ctrl>+<O>). Така сама команда є в меню **File** вікна редактора кодів. Взагалі, і вікно оболонки інтерпретатора, і вікно редактора кодів, мають досить просту систему налаштування зовнішнього вигляду й скромний, але зрозумілий, набір команд. Хочеться вірити, що читач, за потреби, без проблем зможе розібратися в них.

## Використання різних середовищ розроблення

Вище ми проілюстрували послідовність дій (при створенні програми) у тому випадку, коли використовується середовище IDLE. Але ми вже знаємо, що це далеко не єдина можливість. Є альтернативні варіанти. Далі ми стисло опишемо особливості таких середовищ розроблення, як PyCharm та Visual Studio Code.

### До уваги

Читачі, які не мають проблем з опануванням середовищ розроблення (які в багатьох моментах досить схожі), можуть пропустити цей розділ і відразу перейти до наступного розділу, присвяченого використанню змінних.

Якщо скористатися середовищем розроблення PyCharm, то при його запуску з'являється вікно, як показано на рис. 1.11.

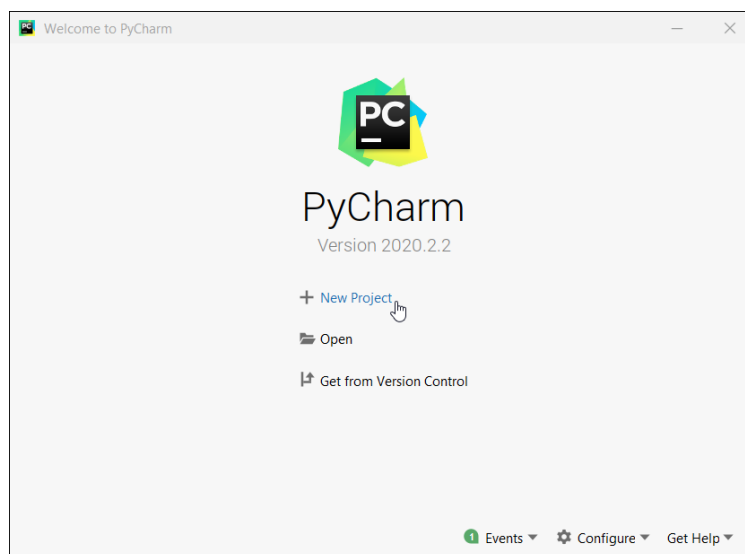


Рис. 1.11. Початкове вікно середовища розроблення PyCharm

Вікно досить просте й містить усього декілька команд. Якщо ми плануємо відкрити вже існуючий проект, то слід скористатися командою **Open**. Якщо ж до наших планів входить створення нового проекту, то слід скористатися командою **New Project**.



### До уваги

Якщо раніше вже створювалися проекти, то в лівій частині вікна буде список цих проектів. Відкрити один з них можна простим натисканням на назві проекту.

Після натискання на команді **New Project** відкривається діалогове вікно **New Project**, представлене на рис. 1.12.

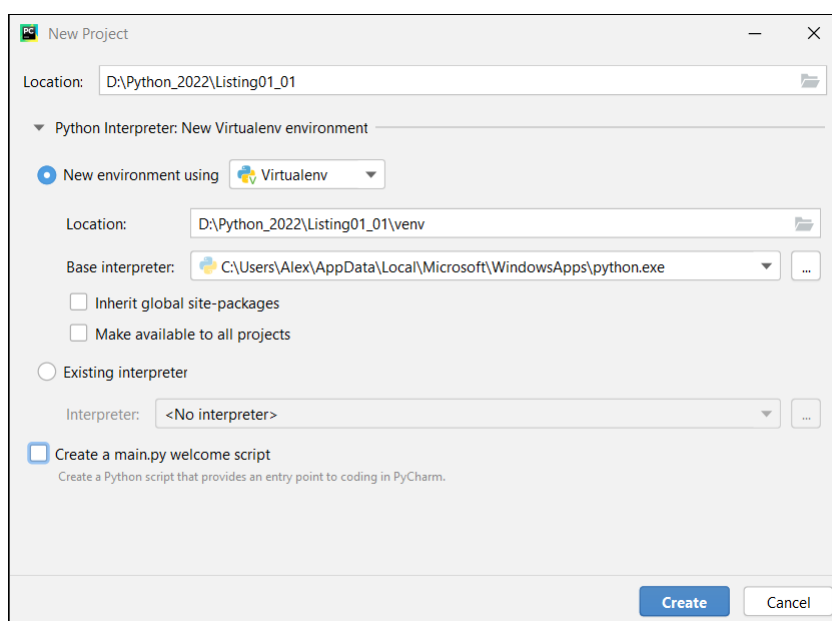


Рис. 1.12. У полі введення вікна **New Project** вибирається місце для розміщення файлів проекту

У полі **Location** слід вказати місце для записування файлів проекту й назву проекту. Піктограма з зображенням відкритої теки (або трьома крапочками, в залежності від версії середовища) справа від поля полегшує процес вибору місця для проекту: натискання піктограми приведе до того, що відкриється діалогове вікно для вибору місця зберігання проекту. За замовчуванням для проекту створюється окрема тека, назва якої співпадає з назвою проекту.

Після того, як місце зберігання й назва проекту вибрані, натискаємо кнопку **Create** у нижній частині діалогового вікна **New Project** (див. рис. 1.12). У результаті створюється пустий проект, а вікно середовища розроблення матиме вигляд як на рис. 1.13.

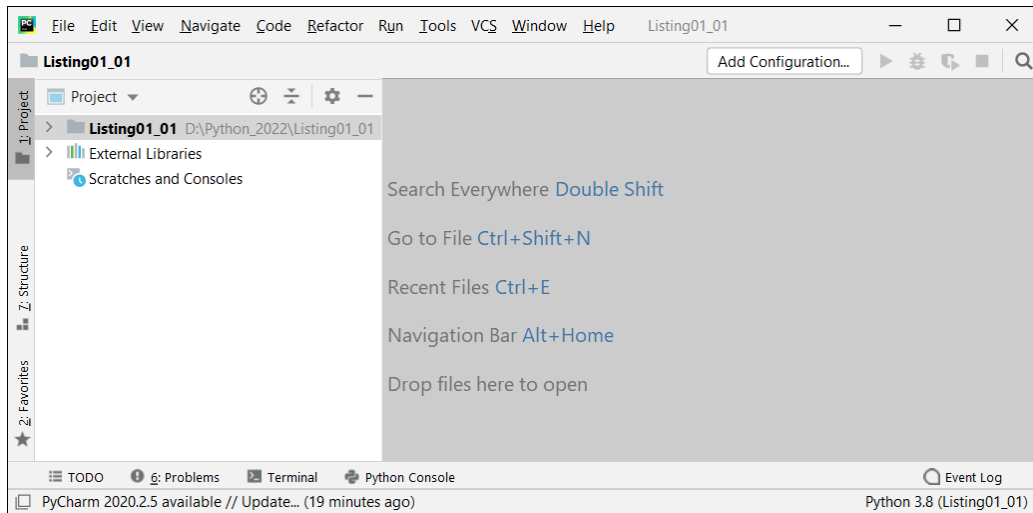


Рис. 1.13. Вікно середовища PyCharm відразу після створення нового проекту

У лівій частині вікна середовища розроблення є внутрішнє вікно проекту. У цьому внутрішньому вікні відображаються динамічні елементи, які відповідають власне проекту та допоміжним бібліотекам. Цікавість викликає елемент з файлами проекту: нам необхідно додати в проект новий файл, в який ми потім додамо програмний код. Для цього виділяємо елемент проекту (його назва співпадає з назвою проекту) і в контекстному меню вибираємо в підменю **New** команду **Python File**, як це показано на рис. 1.14.



### До уваги

Щоб додати файл у проект можна також скористатися командою **New** з головного меню **File**.

Для того щоб у вікні застосунку відображалася панель інструментів (на рис. 1.13 вона розташована під головним меню), необхідно встановити прапорець опції **Toolbar** у підменю **Appearance** меню **View**.

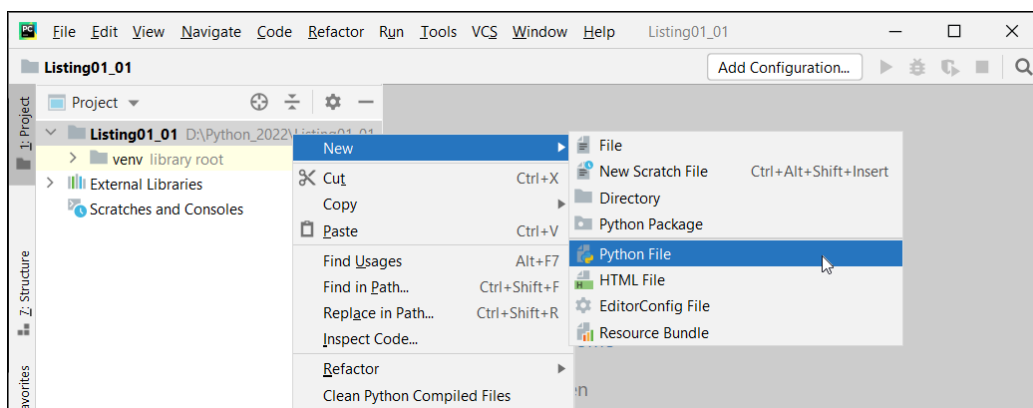
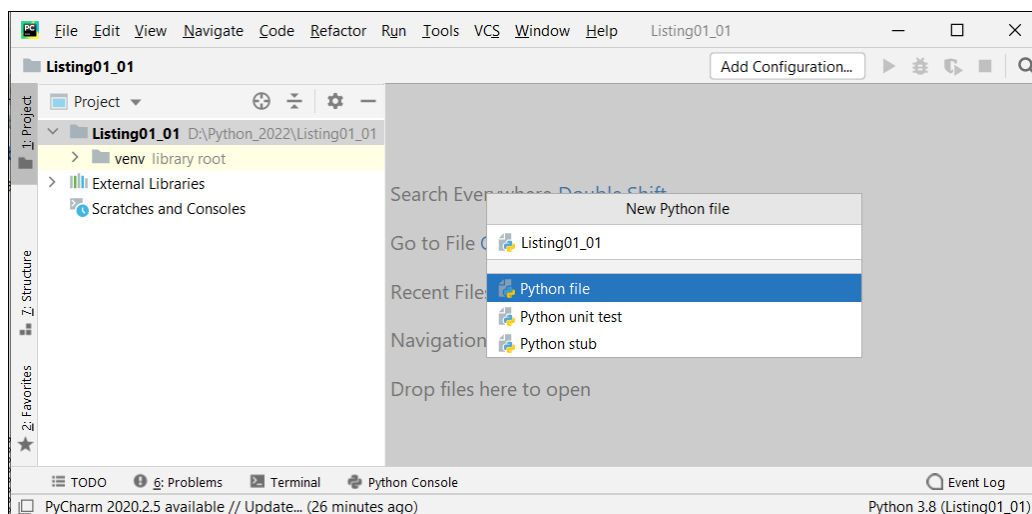


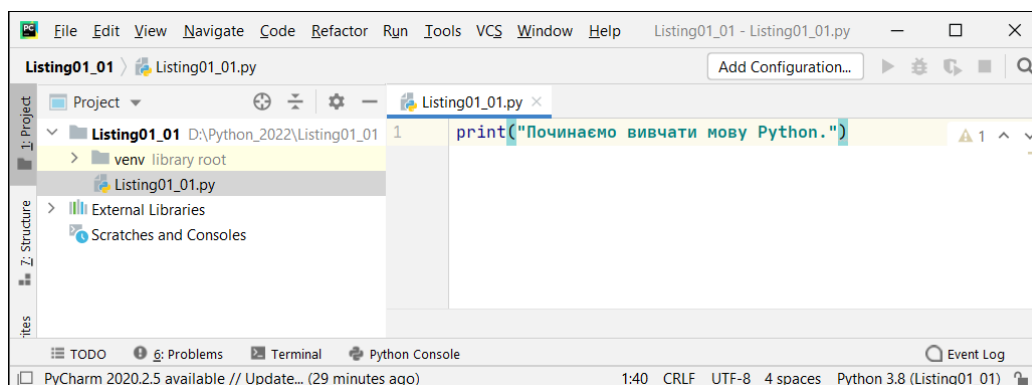
Рис. 1.14. Для додавання файлу в проект у контекстному меню проекту в підменю **New** вибираємо команду **Python File**

Відкриється діалогове вікно **New Python File**, у полі введення якого слід вказати назву для файлу, що додається в проект. Процес уведення назви файлу проілюстровано на рис. 1.15.



*Рис. 1.15. У полі введення діалогового вікна **New Python File** вказується назва файлу*

Якщо все пройшло успішно, то для елемента проекту з'явиться позиція з назвою доданого в проект файлу, як показано на рис. 1.16.



*Рис. 1.16. У внутрішньому вікні редактора кодів середовища PyCharm уведений код програми (одна команда)*

Якщо виділити позицію з назвою файлу, то в правій частині вікна середовища розроблення відображається внутрішнє вікно редактора кодів. Вікно містить код файлу. Саме туди нам слід увести код нашої програми. Для запуску програми на виконання можна скористатися командою **Run** з однойменного головного меню (рис. 1.17).

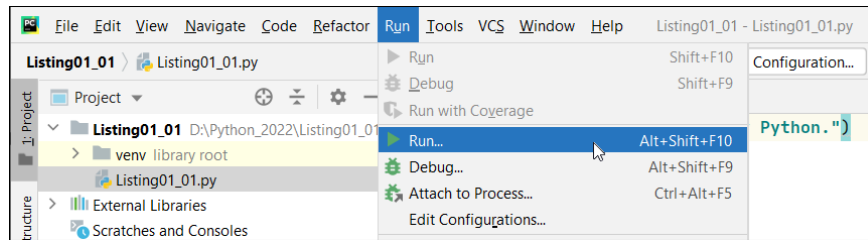


Рис. 1.17. Для запуску програми на виконання слід вибрати в меню **Run** однойменну команду

З'явиться внутрішнє діалогове вікно **Run**, у якому слід вибрати назву проекту, що запускається на виконання (рис. 1.18).

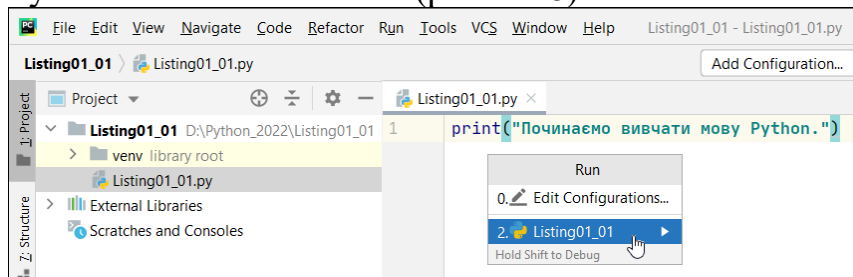


Рис. 1.18. У внутрішньому діалоговому вікні **Run** слід вибрати файл для запуску на виконання

Результат виконання програми відображається у внутрішньому вікні виведення, розташованому в нижній частині вікна середовища розроблення (рис. 1.19).

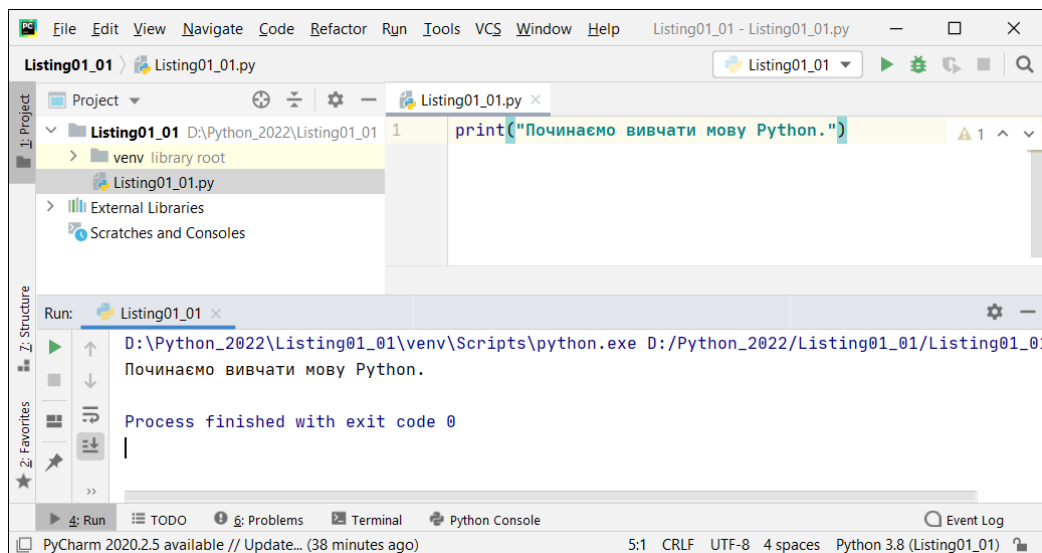


Рис. 1.19. Результат виконання програми відображається у внутрішньому вікні виведення (в нижній частині) середовища розроблення PyCharm

Так виглядає процес створення й запуску на виконання програми у середовищі PyCharm. Існують інші важливі операції, котрі доведеться виконувати в процесі роботи з програмними кодами. Для більшості таких операцій існують відповідні команди в головному меню застосунку. Наприклад, щоб закрити проект, можна скористатися командою **Close Project** з головного меню **File**. За допомогою команди **Open** цього меню можна відкрити вже існуючий проект, а команди **Save as** та **Save All** використовують для того, щоб зберегти внесені зміни. Є величезна кількість інших команд, враховуючи й пов'язані з налаштуванням зовнішнього вигляду застосунку PyCharm. Однак їх описання не входить до наших планів. Хочеться вірити, що читач зможе розібратися з цими питаннями самостійно або за допомогою довідки щодо застосунку.

Ще одним досить популярним середовищем розроблення є Visual Studio Code. При його запуску з'являється діалогове вікно, показане на рис. 1.20.

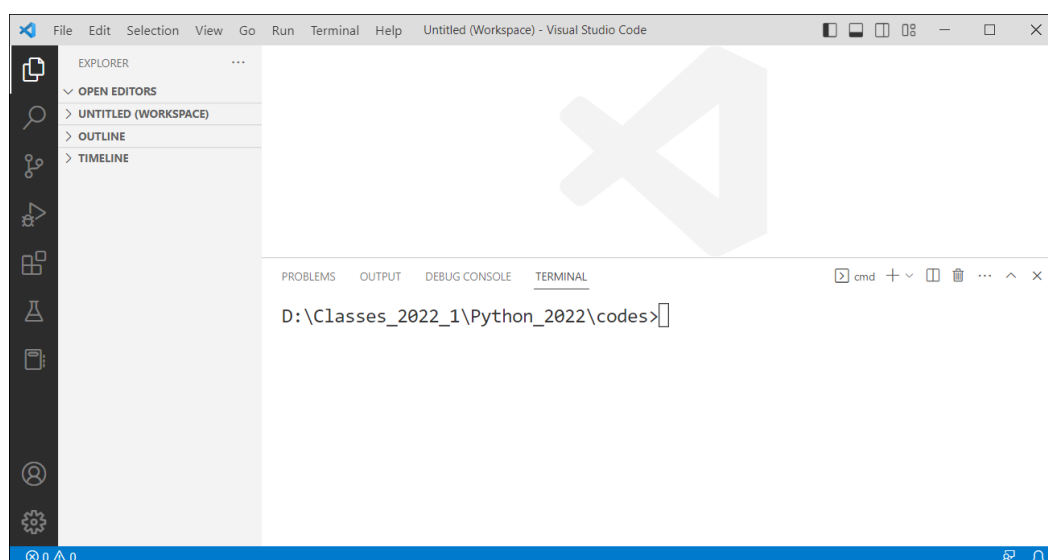


Рис. 1.20. Вікно середовища розроблення Visual Studio Code

Для створення нового проекту можна скористатися командою **New File** з меню **File**. У результаті у внутрішньому вікні редактора з'явиться вкладка для введення програмного коду, як показано на рис. 1.21.

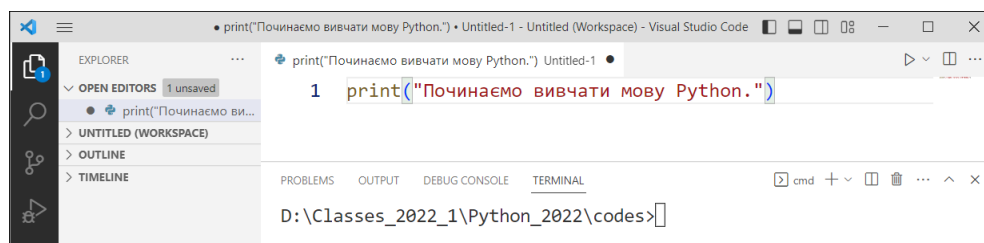


Рис. 1.21. У вікні редактора введено код програми