# Зміст

**Вступ** ..... **5**
Знайомство з мовою Python ..... 6
Коротка історія та особливості мови Python ..... 9
Дещо про книгу ..... 16
Програмне забезпечення ..... 18
Робота з середовищем PyScripter ..... 30
Подяка ..... 37
Зворотний зв'язок ..... 38

**Розділ 1. Програма мовою Python ..... 39**
Розмірковуючи про програму ..... 41
Приклад простої програми ..... 44
Обговорюємо змінні ..... 50
Основні оператори ..... 56
Числові дані ..... 73
Підключення модулів ..... 80
Тернарний оператор ..... 83
Резюме ..... 86

**Розділ 2. Інструкції керування ..... 89**
Умовний оператор ..... 91
Оператор циклу **`while`** ..... 102
Оператор циклу **`for`** ..... 113
Обробка виняткових ситуацій ..... 125
Резюме ..... 137

**Розділ 3. Функції ..... 139**
Створення функції ..... 141
Функції для математичних обчислень ..... 145
Значення аргументів за замовчуванням ..... 149
Функція як аргумент ..... 154
Рекурсія ..... 165
Лямбда-функції ..... 170
Локальні та глобальні змінні ..... 176
Вкладені функції ..... 181
Функція як результат функції ..... 184
Резюме ..... 195

**Розділ 4. Робота зі списками і кортежами ...197**
Знайомство зі списками ....199
Основні операції зі списками ....208
Копіювання і присвоювання списків....219
Списки та функції ....227
Вкладені списки....234
Знайомство з кортежами....243
Резюме....247

**Розділ 5. Множини, словники і текст .....249**
Множини....251
Словники ....271
Текстові рядки....282
Резюме....298

**Розділ 6. Основи об'єктно-орієнтованого програмування.....301**
Класи, об'єкти й екземпляри класів....303
Конструктор і деструктор екземпляра класу ....314
Поле об'єкта класу ....320
Додавання й видалення полів ....329
Методи і функції....334
Копіювання екземплярів і конструктор створення копії...347
Резюме....359

**Розділ 7. Продовжуємо знайомство з ООП 361**
Наслідування ....363
Спеціальні методи і поля....381
Перевантаження операторів ....412
Резюме....432

**Розділ 8. Коротко про різне.....433**
Функції зі змінною кількістю аргументів....435
Декоратори функцій і класів....445
Документування й анотації у функціях ....456
Винятки як екземпляри класів....460
Ітератори і функції-генератори ....481
Резюме....495

**Післямова. Про що ми не поговорили.....497**
**Запитання і відповіді .....502**

# Вступ

# Знайомство з мовою Python

У всьому є своя мораль,
треба лише вміти її знайти!

*Л. Керролл. "Аліса в Країні Див"*[*]

У цій книзі йтиметься про те, як писати програми на мові програмування, яка називається Python (правильно читається як пайтон, але зазвичай назву мови читають як пітон, що теж цілком припустимо). Отже, розв'язувати будемо два завдання, одне з яких пріоритетне, а друге, хоч і допоміжне, проте досить важливе. Наше основне завдання — звичайно ж, вивчення синтаксису мови програмування Python. Паралельно ми будемо освоювати ази програмування, явно чи неявно беручи до уваги, що відповідні алгоритми передбачається реалізовувати мовою Python.

Навіть якщо у читача є досвід програмування іншими мовами, не варто ставитися поверхнево чи зверхньо до процесу побудови алгоритму програми. Правила хорошого тону в програмуванні передбачають, що написання програми починається задовго до набирання програмного коду. Непогано взяти аркуш паперу й накреслити загальну схему виконання програми. А для цього процедуру розв'язання великої та складної задачі варто розбити на послідовність простих дій. З одного боку, цей процес універсальний. З іншого — ті задачі, які ми назвали вище "простими", розв'язуються за допомогою базових команд або функцій мови програмування, якою збираються складати програму. Тому, обмірковуючи алгоритм, цілком абстрагуватися від конкретних можливостей мови програмування навряд чи вийде. Ураховуючи ж гнучкість й ефективність мови програмування Python, слід визнати, що алгоритми навіть "класичних" задач, реалізуючись на Python, стають простішими й зрозумілішими. Іншими словами, навіть якщо читач має досвід

[*] Тут і далі епіграфи подаються у перекладі автора.

складання алгоритмів, знайомство з мовою програмування Python дозволить йому побачити багато знайомих речей зовсім по-іншому.

Узагалі, мов програмування доволі багато. Більше того, час від часу з'являються нові. Тому природно виникає запитання: чому саме Python? Наша відповідь буде складатися з декількох пунктів.

- Мова програмування Python — мова високого рівня, досить "молода", проте дуже популярна, яка вже зараз широко використовується на практиці, і сфера застосування Python постійно розширюється.

Щодо мов програмування нерідко застосовують такі вирази, як мова ***високого рівня***, мова ***середнього рівня*** або мова ***низького рівня***. Ця класифікація досить умовна й базується на рівні абстракції мови. Адже мова, якою розмовляють люди, дещо відрізняється від тієї "мови", яку "розуміють" комп'ютери. Команда, написана простою людською мовою, буде зовсім неприйнятною для комп'ютера. Команда, готова до виконання комп'ютером (***машинний код***), буде незрозумілою для більшості простих смертних. Тому вибирати доводиться між Сциллою та Харибдою. Про мови, орієнтовані на програміста, говорять, що ці мови високого рівня. Про мови, орієнтовані на комп'ютер, говорять, що ці мови низького рівня. Проміжна група мов називається мовами середнього рівня. Хоча ще раз підкреслимо, що поділ цей досить умовний.

- Синтаксис мови Python мінімалістичний і гнучкий. Цією мовою можна складати прості й ефективні програми.
- Стандартна бібліотека для цієї мови містить багато корисних функцій, що значно полегшує процес створення програмних кодів.
- Мова Python підтримує декілька парадигм програмування, включаючи ***структурне***, ***об'єктно-орієнтоване*** й ***функціональне*** програмування. І це далеко не весь список.
- Мова Python — цілком вдалий вибір, як для першої мови в навчанні програмуванню.

Існують й інші причини, щоб вивчити мову програмування Python, можливо, навіть вагоміші за перераховані вище. Про деякі ми ще будемо говорити (у контексті особливостей мови програмування Python). У всякому разі, тут будемо виходити з того, що читач для себе ухвалив

рішення про вивчення мови Python або, принаймні, цікавиться цією мовою програмування.

Парадигма програмування — це найзагальніша концепція, яка визначає фундаментальні характеристики й базові методи реалізації програмних кодів. Наприклад, парадигма ***об'єктно-орієнтованого*** програмування (скорочено ООП) передбачає, що програму реалізують через набір взаємодіючих об'єктів, які, у свою чергу, зазвичай створюються на основі класів. У рамках ***структурного*** програмування програма є комбінацією даних і процедур (функцій) для їх обробки. Мова може підтримувати одразу декілька парадигм. Так, мови Java і C# — повністю об'єктно-орієнтовані, тому для написання найменшої програми цими мовами доведеться описати, як мінімум, один клас. У мові C підтримується парадигма структурного програмування, тому класів й об'єктів у мові C немає. Зате вони є в мові C++. Остання підтримує як парадигму об'єктно-орієнтованого програмування, так і парадигму структурного програмування. Як наслідок, під час роботи з мовою C++ класи й об'єкти можна використовувати, а можна й не використовувати залежно від потреб програміста й специфіки задачі, яку розв'язують. Це ж зауваження стосується й мови Python: із одного боку, під час написання програми мовою Python у нас є можливість удатися до потужного арсеналу об'єктно-орієнтованого програмування, а з іншого боку, часто бувають прийнятними й методи структурного програмування.

Існують й інші, більш витончені концепції програмування. Скажімо, парадигма функціонального програмування припускає, що результат функції в програмі визначається виключно значеннями аргументів, переданих функції, і не залежить від стану зовнішніх (стосовно функції) змінних. Відповідні функції прийнято називати ***чистими функціями***, і вони мають ряд корисних властивостей, що дозволяють істотно оптимізувати й прискорити обчислювальний процес. Ця концепція, як і ряд інших, знаходить реалізацію в мові Python.

Далі обговоримо деякі важливі моменти й "підводне каміння", яке може зустрітися на подекуди важкому, та все ж цікавому й захопливому шляху опанування новими вершинами у програмуванні.

# Коротка історія та особливості мови Python

> Серйозне ставлення до будь-чого в цьому світі є фатальною помилкою.
>
> *Л. Керролл. "Аліса в Країні Див"*

У мови Python є автор ***Гвідо ван Россум*** (Guido van Rossum). І хоча в розробці й популяризації мови на теперішній момент устигло взяти участь багато талановитих розробників, саме Гвідо ван Россум отримав заслужену славу творця цієї перспективної й популярної мови програмування. Взагалі ж робота над мовою почалася у 80-х роках минулого століття. Вважають, що перша версія мови з'явилася в 1991 році. Щодо назви мови програмування Python, то, формально, це назва рептилії. Відповідно, часто як логотип використовують милу (або не дуже) змійку типу "пітон". І хоча практично будь-який навчальний чи довідковий посібник із мови Python містить розповідь про те, що насправді Python це не "пітон", а назва гумористичної передачі "Літаючий цирк Монті Пайтона", для історії це вже не важливо.

Доречно згадати слова капітана Врунгеля з однойменної повісті Андрія Некрасова: "Як ви яхту назвете, так вона й попливе". У мови програмування Python досить агресивна назва, і, треба визнати, ця назва себе виправдовує. За аргумент до такого твердження може правити як гнучкість й ефективність самої мови, так і та швидкість, із якою вона завоювала собі "місце під сонцем" серед найпопулярніших мов програмування.

Мова Python бурхливо розвивається. Цьому сприяє не тільки досить вдала концепція мови, а також згуртоване співтовариство, сформоване

з розробників і популяризаторів мови. Важливий і той факт, що необхідне програмне забезпечення, включаючи середовища розробки, в основному безкоштовне. Усе це дає підстави розглядати Python як одну з найперспективніших мов програмування.

На сьогодні Python використовується при реалізації найрізноманітніших проектів, серед яких:

- розробка сценаріїв для роботи з Web та Internet-програмами;
- мережеве програмування;
- засоби підтримки технологій HTML і XML;
- програми для роботи з електронною поштою й підтримки Internet-протоколів;
- програми для обслуговування найрізноманітніших баз даних;
- програми для наукових розрахунків;
- програми з графічним інтерфейсом;
- створення ігор і комп'ютерної графіки та багато іншого.

Зрозуміло, охопити всі ці теми в одній книзі досить складно. Та ми й не ставимо це собі за мету. А втім, навіть на відносно невеликій кількості нескладних прикладів цілком можливо продемонструвати елегантність і виключну ефективність мови Python. Цим, власне, ми й займемося в основній частині книги — тобто трохи згодом. Зараз обговоримо особливості та деякі «технічні» моменти, які важливі для розуміння основ програмування мовою Python.

Python належить до мов програмування, ***що інтерпретуються***, і це має певні наслідки. Формально те, що мова програмування належить до інтерпретованих, означає, що програмний код виконується за допомогою спеціальної програми-***інтерпретатора***. Інтерпретатор виконує програмний код порядково (з попереднім аналізом виконуваного коду). Недолік такого підходу полягає в тому, що, по-перше, помилки виявляються фактично на етапі виконання програми і, по-друге, швидкість виконання програми відносно невисока. Тому нерідко застосовується більш складна схема: вихідний програмний код компілюється в проміжний код, а вже цей проміжний код виконується безпосередньо інтерпретатором. У цьому разі швидкість виконання програми збільшується, але разом із нею збільшується і потреба у системних ресурсах. Приблизно за такою схемою виконується програмний код, написаний мовою Python.

Нагадаємо, що окрім мов, які інтерпретуються, існують мови, програми на яких ***компілюються*** (мається на увазі компіляція в машинний код). У цьому випадку вихідний програмний код компілюється у виконавчий (машинний) код, який виконується (зазвичай) під керуванням операційної системи. Компільовані у виконавчий код програми характеризуються відносно високою швидкістю виконання.

Якщо ми хочемо написати програму на мові Python, то для цього, як мінімум, знадобиться набрати відповідний програмний код. Тут можливі варіанти, але, в принципі, код набирається в будь-якому текстовому редакторі, а відповідний файл зберігається з розширенням `py` чи `pyw` (для програм із графічним інтерфейсом). Під час першого запуску (після внесення змін у програмний код) створюється проміжний код, який записується у файл із розширенням `pyc`. Якщо після цього в програму зміни не вносилися, то під час виконання програми буде виконуватися відповідний `pyc`-файл. Після внесення змін у програму під час чергового запуску вона перекомпілюється в `pyc`-файл. Це загальна схема. Нас, насправді, вона цікавить виключно в плані загального розвитку, хоча на практиці іноді буває важливо враховувати особливості виконання програми, написаної мовою Python.

Як зазначалося вище, програмний код можна набирати й у текстовому редакторі. Та ось для виконання такого програмного коду знадобиться спеціальна програма, яка називається ***інтерпретатором***. Іншими словами, для роботи з Python на комп'ютер необхідно встановити програму-інтерпретатор. Ми окремо зупинимося на цьому питанні. Зараз же тільки зауважимо, що зазвичай використовується не просто інтерпретатор, а ***інтегроване середовище розробки***, яке, крім іншого, включає в себе як інтерпретатор, так і редактор програмних кодів.

Інтерпретатор виконує програму команда за командою. Тому, в принципі, якщо програма складається з декількох команд, її можна організувати у вигляді файлу з програмним кодом, а потім «відправити» цей файл на виконання. Ще один варіант — «передавати» інтерпретатору для виконання по одній команді. Обидва режими можливі й підтримуються інтерпретатором мови Python. Ми будемо складати і запам'ятовувати програмні коди у файлах — тобто використаємо «традиційний» підхід.

Про режим, за якого в командному вікні інтерпретатора команди вводяться і виконуються одна за одною, говорять, як про ***режим командного рядка,*** або ***режим калькулятора***.

Оскільки мова Python розвивається інтенсивно, й від версії до версії в синтаксис і структуру мови вносяться зміни, важливо враховувати, для якої версії мови Python складають (і передбачають використовувати) програмний код. Особливо це важливо, враховуючи ту обставину, що під час внесення змін ***принцип зворотної сумісності*** спрацьовує далеко не завжди: якщо програмний код коректно виконується в давніших версіях мови, то зовсім не факт, що він буде виконуватися під час роботи з новішими версіями. Однак панікувати з цього приводу не варто. Зазвичай проблема несумісності кодів для різних версій пов'язана з особливостями синтаксису деяких функцій або конструкцій мови і досить легко усувається.

На момент написання книги актуальною є версія Python 3.6. Саме вона використовувалася для тестування прикладів у книзі. У разі виходу нових версій або стандартів мови, для забезпечення сумісності програмних кодів є сенс проглянути той розділ довідкової системи, в якому описано нововведення. Зробити це можна, наприклад, на офіційному сайті підтримки мови Python `www.python.org/doc/` у розділі під назвою ***What's New In Python*** (у перекладі означає ***Що нового в Python***).

Але все це технічні деталі, які хоч і важливі, та все ж не першорядні. А першорядними для нас будуть синтаксис мови Python і його основні інструкції керування. І, власне, тут на нас чекає чимало приємних сюрпризів.

Особливо багато сюрпризів буде для читачів, які знайомі з такими мовами програмування, як Java чи C++, наприклад. Але це, до речі, зовсім не означає, що новачків у програмуванні мова Python залишить байдужими. Просто ті, хто вивчав основи ООП і програмує згаданими мовами, значно розширять свій кругозір щодо методів і прийомів програмування, а також, у певному сенсі, їм доведеться змінити своє уявлення про мови програмування.

Синтаксис мови Python більш ніж цікавий. По-перше, він простий, зрозумілий і наочний. Його навіть можна назвати по-спартанськи лаконічним.

Разом з цим програмні коди, написані на Python, зазвичай легко читаються й аналізуються, а обсяг програмного коду набагато менший, порівняно з аналогічними програмами, написаними іншими мовами програмування. Як ілюстрацію в лістингу В.1 наведено код програми мовою C++, у результаті виконання якого в консольне вікно виводиться повідомлення `Hello, world!`.

**Лістинг В.1. Програма мовою C++**

```
#include <iostream>
using namespace std;
int main(){
    cout<<"Hello, world!"<<endl;
    return 0;
}
```

Аналогічний програмний код, але вже мовою Java, представлено в лістингу В.2.

**Лістинг В.2. Програма мовою Java**

```
class MyClass{
    public static void main(String[] args){
        System.out.println("Hello, world!");
    }
}
```

Нарешті, у лістингу В.3 показано, як виглядатиме програма для виведення в консольне вікно текстового повідомлення, якщо для її написання скористатися мовою програмування Python.

**Лістинг В.3. Програма мовою Python**

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

Неважко помітити, що це всього одна команда, де вбудованій функції `print()` як аргумент передається текст, який необхідно надрукувати в консольному вікні. Зрозуміло, далеко не завжди у нас буде виходити писати такі «економні» коди, але приклад усе ж багато в чому показовий.

Про всяк випадок, коротко прокоментуємо наведені вище коди мовами C++ і Java — просто, щоб у читача, не знайомого з цими мовами, не виникло комплексу меншовартості. Почнімо з програми лістингу В.1, написаної мовою C++:

- Інструкція `#include <iostream>` підключає заголовковий файл бібліотеки вводу/виводу.
- Команда `using namespace std` означає використання стандартного простору імен.
- Функція з назвою `main()` називається головною функцією програми. Виконання програми в C++ — це виконання головної функції програми.
- Ідентифікатор `int` ліворуч від функції `main()` означає, що функція повертає цілочисловий результат.
- Пара фігурних дужок (`{` і `}`) виділяє тіло головної функції.
- Команда `cout<<"Hello, world!"<<endl` виводить у консоль текстове повідомлення `Hello, world!`, і відбувається перехід до нового рядка (інструкція `endl`). Оператор виводу `<<` виводить текст, зазначений праворуч від нього, на пристрій, визначений ідентифікатором `cout` (за замовчуванням — консоль).
- Інструкція `return 0` завершує виконання головної функції (тобто програми), а результатом функція повертає `0` (означає закінчення роботи програми «у штатному режимі» — тобто без помилок).

Програма в лістингу В.2, нагадаємо, написана мовою Java, і у відповідному програмному коді призначення інструкцій таке:

- Створюється клас із назвою `MyClass`: перед назвою класу зазначається ключове слово `class`, а тіло класу береться у фігурні дужки (зовнішня пара дужок `{` і `}`).
- У тілі класу описується головний метод із назвою `main()`, тіло методу береться в блок із фігурних дужок (внутрішня пара дужок `{` і `}`).
- Перед назвою головного методу зазначено такі ідентифікатори: `public` (відкритий метод — тобто доступний поза класом), `static` (статичний метод — для виклику методу немає необхідності створювати об'єкт класу), `void` (метод не повертає результат).
- Як параметр (аргумент) методу `main()` указано змінну `args`, яка є текстовим масивом (текст — це об'єкт класу `String`, а наявність порожніх квадратних дужок `[]` свідчить про те, що це текстовий масив — упорядкований набір текстових значень).
- Текст у консольне вікно виводиться за допомогою методу `println()`: як аргумент методу передається текст, що виводиться в консоль, а сам метод викликається з об'єкта потоку виводу `out`, який, у свою чергу, є статичним полем класу `System`.

Зрозуміло, на фоні всього цього різноманіття програма (а точніше, одна єдина команда) мовою Python виглядає більш ніж ефектно. Але ще раз підкреслюємо, що, навіть якщо читач зрозумів не все з викладеного вище (щодо кодів C++ і Java), це не страшно. Нам, у нашій подальшій роботі, усе це не знадобиться. Ми просто хотіли проілюструвати масштаби, так би мовити, розбіжностей для різних мов.

Під час роботи з мовою Python ми не зустрінемо багатьох звичних для інших мов конструкцій. Наприклад, на відміну від мов C++ і Java, в яких командні блоки виділяються фігурними дужками `{` і `}`, і на відміну від мови Pascal, у якій блоки виділяються за допомогою інструкцій `begin` і `end`, у мові Python блок команд виділяється відступом (рекомендований відступ — чотири пробіли). У мові Python немає необхідності закінчувати кожну команду крапкою з комою. Існують й інші особливості мови Python. Ми будемо знайомитися з ними поступово і, перефразовуючи М.Є. Салтикова-Щедріна, російського письменника-сатирика, намагатимемося при цьому не застосовувати силу.

Мається на увазі цитата «Просвіту впроваджувати помірковано, по можливості уникаючи кровопролиття» із сатиричного роману «Історія одного міста» М.Є. Салтикова-Щедріна.

Тут просто важливо зрозуміти, що мова Python досить своєрідна, самобутня й у багатьох відношеннях не схожа на інші популярні сьогодні мови програмування.

# Дещо про книгу

Вона завжди давала собі хороші поради, хоча слідувала їм нечасто.

*Л. Керролл. «Аліса в Країні Див»*

Настав час сказати (прочитати, написати — кому як більше подобається) декілька слів безпосередньо про книгу: що вона собою являє, для кого написана, й, узагалі, що читачеві очікувати від прочитання.

Зрозуміло, книга писалася, насамперед, для тих, хто вирішив опанувати мову програмування Python. Тобто передбачається, що читач із мовою Python не знайомий узагалі. Більше того, ми неявно будемо виходити з того, що читач має, загалом, мінімальний досвід програмування. Останнє не завадить нам періодично посилатися до таких мов програмування, як C++ і Java. Звичайно, робитимемо ми це, передусім, розраховуючи на тих читачів, хто має хоча б мінімальну уяву про ці мови і/або ООП. Щоб компенсувати незручності, які могли б виникнути у читачів, не знайомих із C++ і Java, пояснення максимально адаптуються для сприйняття повністю непідготовленою аудиторією. Простіше кажучи, не має значення, знає читач інші мови програмування чи ні — у будь-якому разі він може розраховувати на успіх.

Матеріал книги охоплює всі основні теми, необхідні для успішної роботи з мовою Python, включаючи методи ООП. В основному, ми будемо розглядати конкретні задачі — тобто теорію буде наведено «на прикладах». Це прийом, який на практиці непогано себе зарекомендував. Особливо він ефективний, коли необхідно в стислі терміни з мінімальними затратами енергії і ресурсів опанувати на якісному рівні великий обсяг матеріалу. За такого підходу є й додатковий бонус: окрім особливостей мови читач

має можливість познайомитися з алгоритмами, які застосовують для розв'язання ряду прикладних задач. Щодо підбору прикладів і задач, то вони вибиралися так, щоб найяскравіше проілюструвати можливості й особливості мови Python.

# Програмне забезпечення

– А де я можу знайти кого-небудь нормального?
– Ніде, – відповів Кіт, – нормальних не буває. Адже всі такі різні і несхожі. І це, по-моєму, нормально.

*Л. Керролл. «Аліса в Країні Див»*

Перш ніж поринути в глибини світу під назвою Python і почати здобувати безцінні знання, доречно внести ясність у питання про програмне забезпечення, яке нам знадобиться для тестування прикладів із книги і написання власних оригінальних програм.

Як зазначалося вище, для виконання програмних кодів, написаних на Python, нам потрібна програма-інтерпретатор. Але найкраще скористатися будь-яким інтегрованим середовищем розробки (скорочено *IDE* від англійського *Integrated Development Environment*). Середовище розробки надає користувачу не тільки інтерпретатор, а й редактор кодів, так само як і ряд інших корисних утиліт. Інтегрованих середовищ розробки для роботи з Python існує досить багато, і в певному сенсі перед програмістом виникає непроста проблема вибору. Критерії для вибору середовища розробки можуть бути найрізноманітнішими. Але головні серед них, звичайно ж, — це зручність у використанні, набір вбудованих можливостей/функцій інтегрованого середовища розробки, а також його вартість (існують як комерційні продукти, так і у вільному доступі). Ми розглянемо декілька найпопулярніших і найдоступніших інтегрованих середовищ розробки для Python.

Якщо ми говоримо про програмне забезпечення, то, в першу чергу, є сенс вийти на офіційний сайт підтримки Python за адресою `www.python.org`. Вікно браузера, відкрите на відповідній сторінці, показано на рис. В.1.

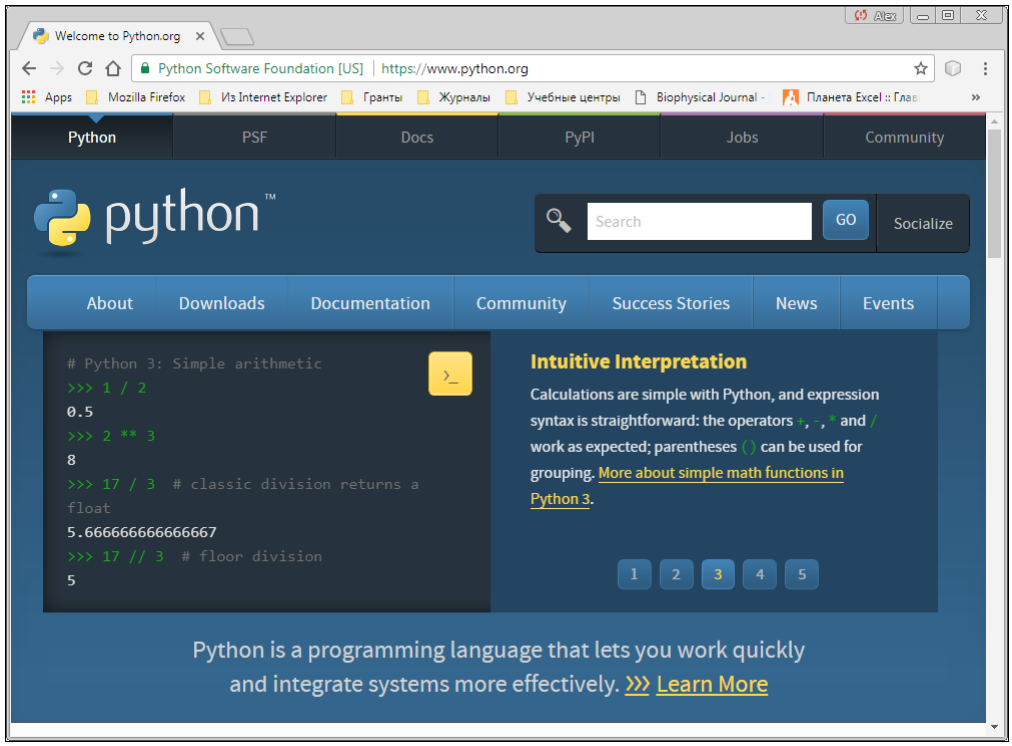


*Рис. В.1. Вікно браузера, відкрите на офіційній сторінці підтримки Python* `www.python.org`

Сайт містить чимало корисної інформації, включаючи всеосяжну довідку, і дозволяє завантажити необхідне програмне забезпечення — у тому числі, й середовище розробки, яке називається ***IDLE*** (скорочення від ***Integrated DeveLopment Environment***, що буквально означає ***інтегроване середовище розробки***). Для завантаження програмного забезпечення необхідно перейти до розділу Downloads (***завантаження***) — адреса `www.python.org/downloads`.

Сам процес завантаження й установки досить простий та інтуїтивно зрозумілий, тому зупинятися на ньому не будемо. Нас цікавить кінцевий результат. А в результаті ми отримуємо повноцінне середовище для роботи

з програмними кодами Python. Робоче вікно середовища IDLE представлено на рис. В.2.

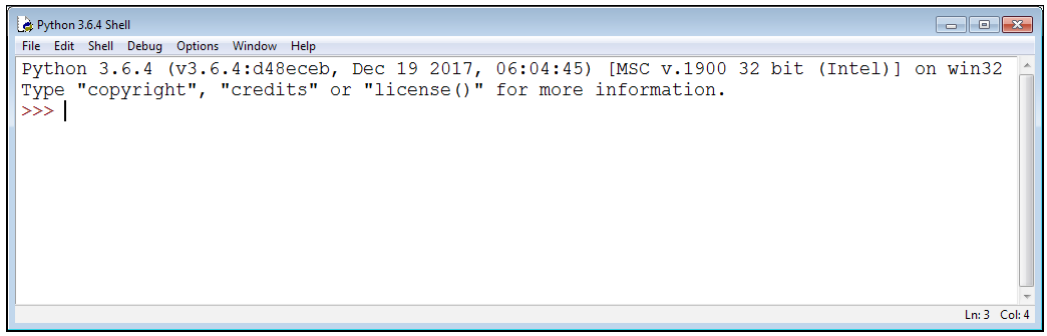


*Рис. В.2. Робоче вікно середовища розробки IDLE*

Перед нами командна оболонка інтерпретатора. Це вікно з декількома меню і великою робочою областю, в якій після символу потрійної стрілки >>> блимає курсор — це командний рядок. У цьому місці вводиться команда, яка виконується після натискання клавіші <Enter>. Наприклад, якщо ми хочемо виконати команду `print("Hello, world!")`, нам треба ввести цю команду в командний рядок (тобто там, де блимає курсор — після символу >>>) і натиснути клавішу <Enter>. Як наслідок, команду буде виконано, а результат її виконання відобразиться знизу, під командним рядком. Ситуацію проілюстровано на рис. В.3.

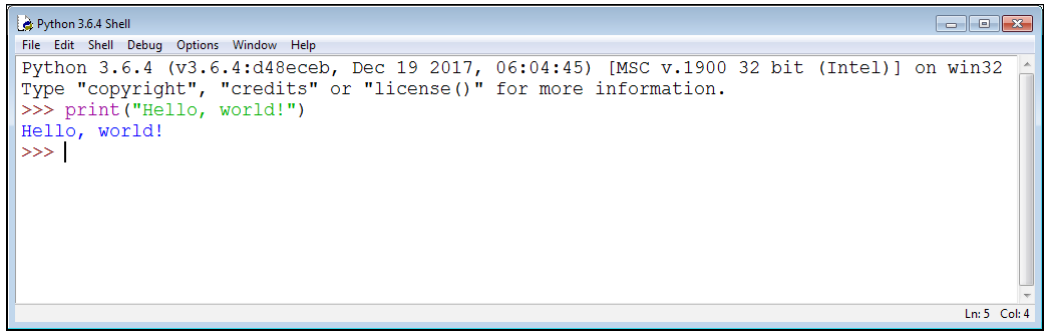


*Рис. В.3. Результат виконання команди в командній оболонці середовища розробки IDLE*

При цьому під результатом виконання команди знову з'являється потрійна стрілка >>>, і в цьому місці можна вводити нову команду. Наприклад,

можемо ввести який-небудь алгебраїчний вираз — скажімо, нехай це буде `5+3*4`, як показано на рис. В.4.

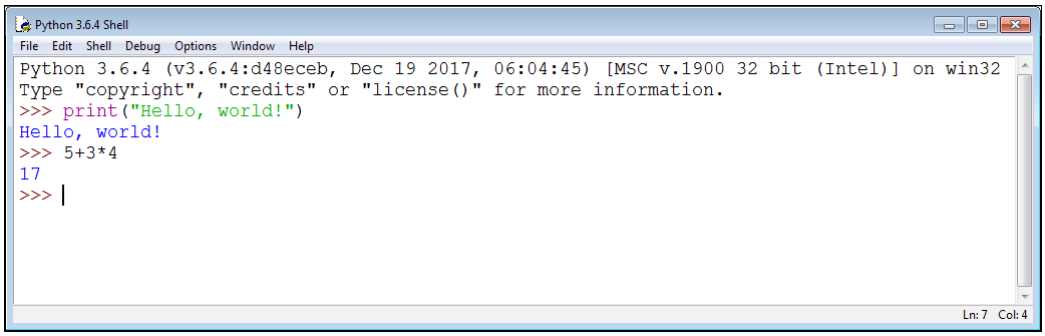


*Рис. В.4. Результат обчислення алгебраїчного виразу*

Зрозуміло, команди можуть бути й більш хитромудрі, так само як ніхто не забороняє нам команду за командою виконувати програмний код. Але це досить незручно. Зазвичай при написанні більш-менш серйозної програми її оформлюють у вигляді послідовності інструкцій і записують в окремий файл. Потім відповідна програма виконується.

Створити файл програми можна за допомогою все тієї ж оболонки середовища розробки. Якщо клацнути меню File, відкриється список команд і підменю, серед яких є й команда New File (рис. В.5).

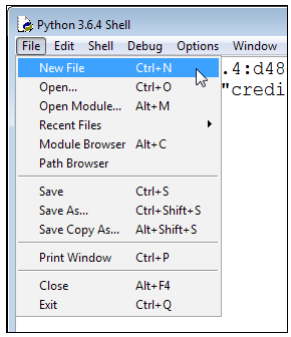


*Рис. В.5. Створення файла з програмою*

Одразу після вибору цієї команди відкривається редактор кодів, показаний на рис. В.6.

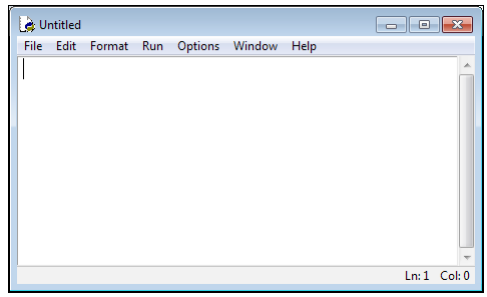


*Рис. В.6. Редактор кодів для створення файла з програмою*

У вікні редактора вводимо програмний код. У цьому разі наша програма складатиметься лише з декількох команд, наведених у лістингу В.4.

**Лістинг В.4. Декілька команд для запису у файл**

```
print("Починаємо обчислення!")
a=4
print("Значення змінної a = ",a)
b=12
print("Значення змінної b = ",b)
c=b/a
print("Результат ділення b/a = ",c)
print("Обчислення закінчено!")
```

Саме такий програмний код ми вводимо у вікні редактора кодів. Вікно редактора з програмним кодом показано на рис. В.7.

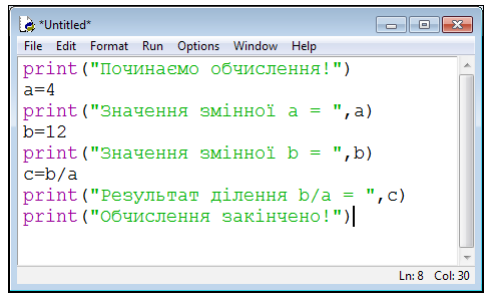


*Рис. В.7. Вікно редактора кодів із кодом програми*

Після того, як програмний код набрано, його можна одразу виконати. Для цього в меню Run вибираємо команду Run Module, як показано на рис. В.8.

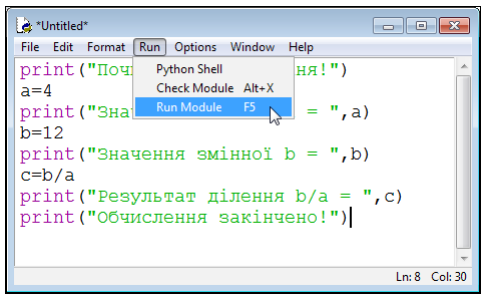


*Рис. В.8. Запуск програми на виконання*

Правда, попередньо все ж таки краще зберегти файл із програмою, для чого корисною буде команда Save із меню File (рис. В.9).

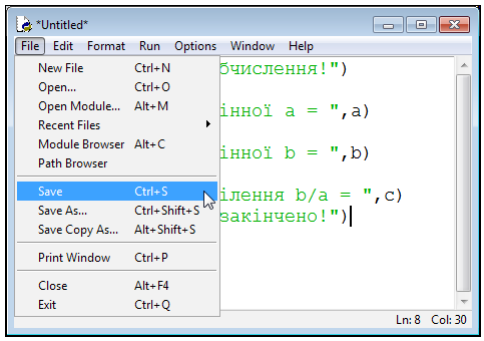


*Рис. В.9. Збереження файла з програмою*

Якщо перед запуском програми на виконання файл не зберегти, з'явиться діалогове вікно з пропозицією зберегти файл. Тому краще це зробити одразу.

Проте, хоч би там що було, в результаті виконання програми у вікні командної оболонки з'явиться результат, як на рис. В.10.

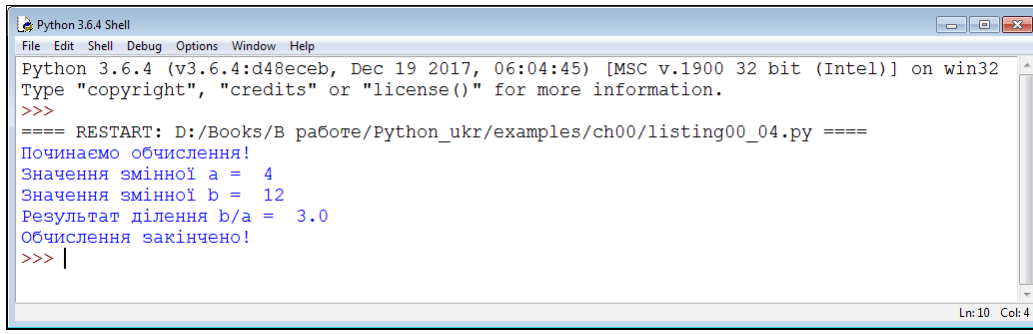


*Рис. В.10. Результат виконання програми*

Як бачимо, в області виводу результатів (під символом >>>) з'являється декілька повідомлень, які, очевидно, є наслідком виконання програми, наведеної в лістингу В.4. Результат програми подано нижче.

**Результат виконання програми (з лістингу В.4)**

```
Починаємо обчислення!
Значення змінної a =  4
Значення змінної b =  12
Результат ділення b/a =  3.0
Обчислення закінчено!
```

І хоча ми «офіційно» ще нібито не розпочали вивчення мови Python, є сенс прокоментувати відповідні команди і результат їхнього виконання.

Отже, команда `print("Починаємо обчислення!")` на початку виконання програми виводить текстове повідомлення `Починаємо обчислення!`. Аналогічно, команда `print("Обчислення закінчено!")` наприкінці програмного коду виводить текстове повідомлення `Обчислення закінчено!`, що свідчить про завершення виконання програми.

Між цими командами виконуються невеликі обчислення:

- за допомогою команди `a=4` змінній `a` присвоюється числове значення `4`;
- за допомогою команди `print("Значення змінної a = ",a)` виводиться текст, а потім значення змінної `a`;

- за допомогою команди `b=12` змінній `b` присвоюється числове значення `12`;
- за допомогою команди `print("Значення змінної b = ",b)` виводяться текст і значення змінної `b`;
- за допомогою команди `c=b/a` змінній `c` як значення присвоюється результат ділення значення змінної `b` на значення змінної `a`;
- за допомогою команди `print("Результат ділення b/a = ",c)` виводяться текст і числове значення змінної `c`.

У справедливості цих тверджень читач може переконатися, ще раз поглянувши на рис. В.10.

Напевно, допитливий читач помітив, що змінні в програмному коді використано *без оголошення їхнього типу*. Іншими словами, ми ніде явно не зазначали тип змінних, які використовували в програмі. Це стандартна ситуація для програм, написаних на Python, — тип змінних не зазначається (він визначається автоматично за значенням, яке присвоюється змінній).

Також ми побачили, що функції `print()` можна передавати не тільки один, а декілька аргументів. У цьому випадку в область виводу (або консоль) послідовно, в один рядок, виводяться значення аргументів функції `print()`.

Про те, як правильно створювати програмні коди на Python, ми будемо говорити в основній частині книги. Тут нам важливо лише проілюструвати, що потім із цими програмними кодами робити. Також нам важливо дати читачеві найзагальніше уявлення про ті прикладні програми і середовища розробки, які дозволяють у зручному режимі створювати коди і запускати їх на виконання. Щодо коротко описаного вище середовища IDLE, то назвати його дуже вже вдалим навряд чи можна, хоча, звичайно, це суб'єктивна точка зору автора, і читач не зобов'язаний її поділяти.

Серед комерційних продуктів можна виділити інтегроване середовище розробки ***Komodo IDE*** (офіційний сайт `www.activestate.com`). Вікно середовища розробки з відкритим у ньому файлом програми, що розглядалася вище, показано на рис. В.11.

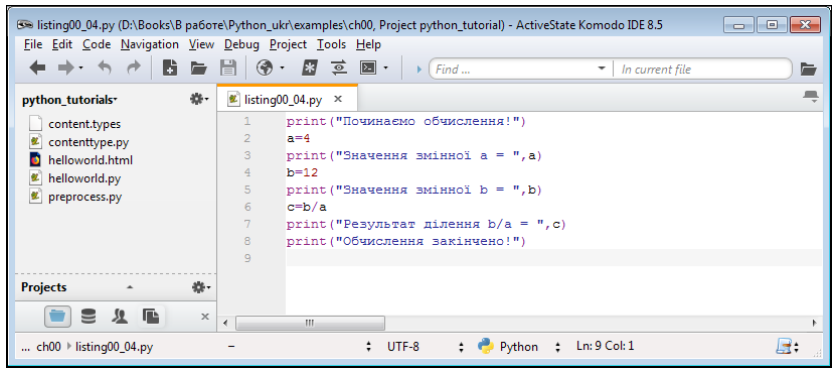


*Рис. В.11. Вікно інтегрованого середовища розробки Komodo IDE з програмним кодом*

Результат виконання програми в середовищі Komodo IDE показано на рис. В.12 (для запуску програми на виконання можна скористатися командою Run Without Debagging із меню Debug).

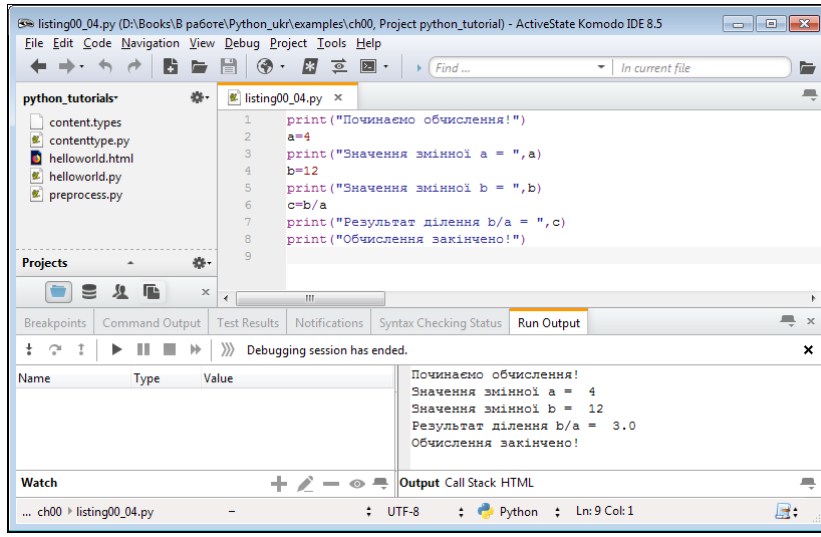


*Рис. В.12. Результат виконання програми в середовищі Komodo IDE*

Ми, однак, використовуватимемо для тестування прикладів із книги некомерційне, зручне і просте середовище розробки ***PyScripter***. Інсталяційні файли можна вільно завантажити у розділі `Downloads` на сторінці `https://sourceforge.net/projects/pyscripter`. Сторінку підтримки проекту PyScripter з відкритим вікном браузера показано на рис. В.13.

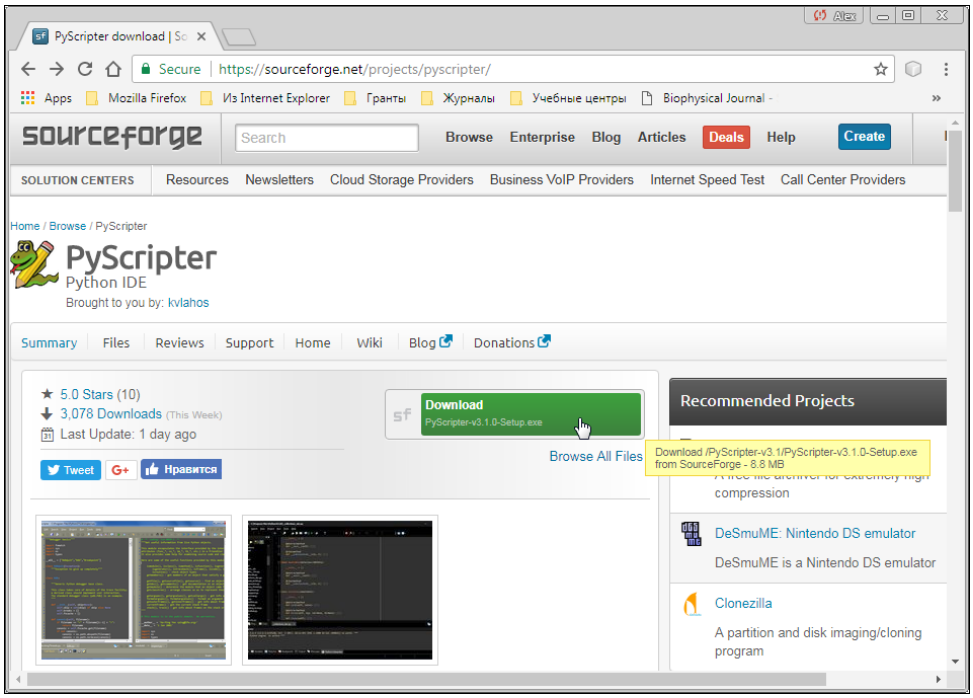


*Рис. В.13. Сторінка підтримки проекту PyScripter*

Вікно середовища розробки з програмним кодом показано на рис. В.14.

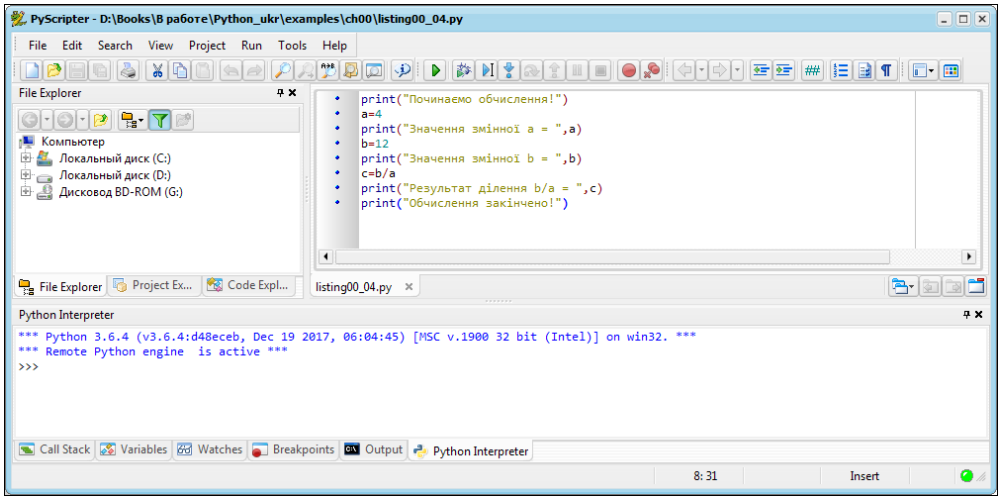


*Рис. В.14. Вікно середовища PyScripter із програмним кодом*

Щоб створити новий файл із програмою, вибираємо команду **New** в меню **File**, а щоб відкрити вже існуючий файл, вибираємо команду **Open** із того ж меню.

Для запуску програми на виконання вибираємо в меню Run команду Run (рис. В.15) або натискаємо кнопку з зеленою стрілкою на панелі інструментів.

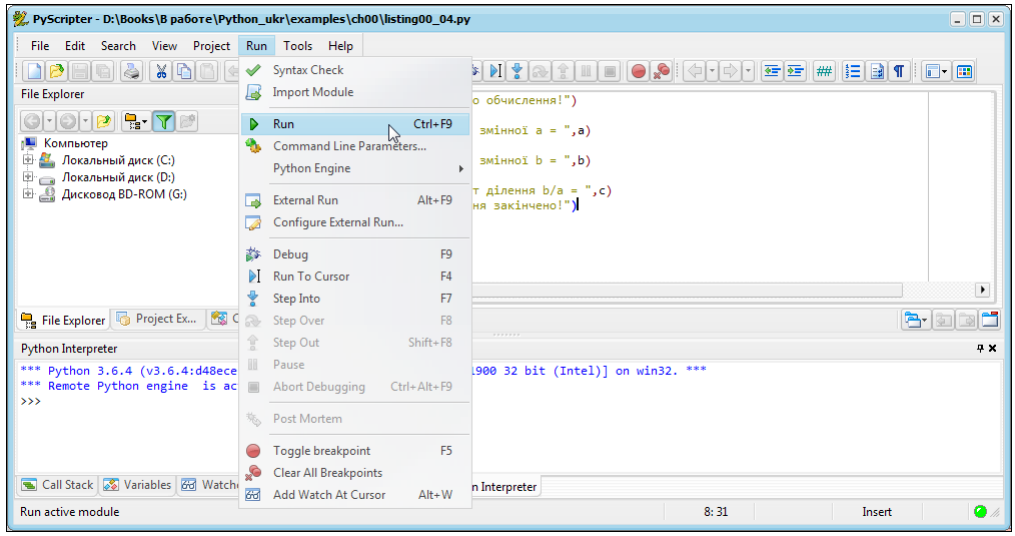


*Рис. В.15. Запуск програми на виконання в середовищі PyScripter*

На рис. В.16 показано результат виконання програми.

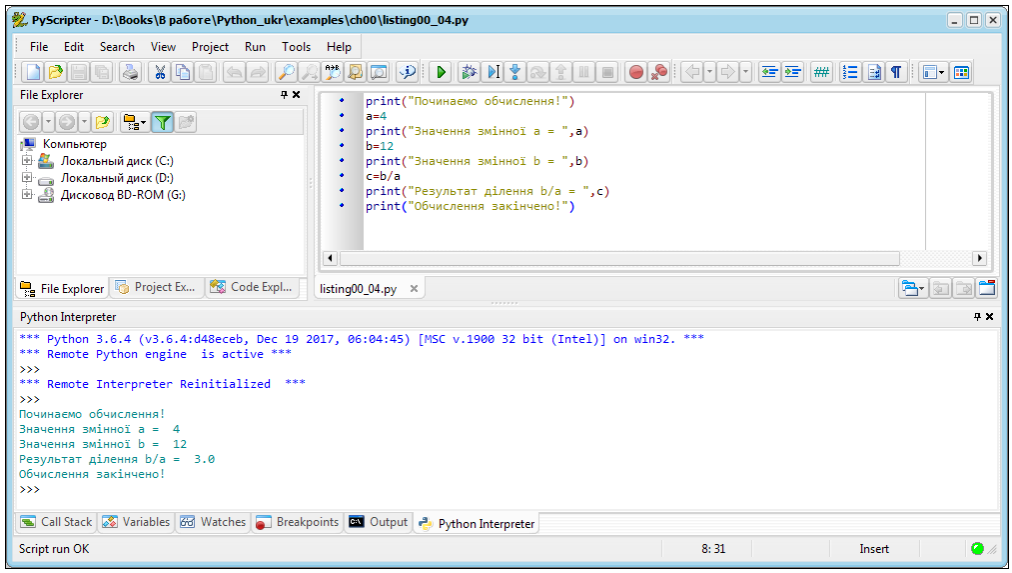


*Рис. В.16. Результат виконання програми в середовищі PyScripter*

Результат відображується у внутрішньому вікні Python Interpreter, і це доволі зручно.

За бажанням у внутрішньому вікні Python Interpreter можна виконувати окремі команди: інструкції для виконання вводяться після символу >>>.

Оскільки в наші плани входить широке використання середовища розробки PyScripter для роботи з програмними кодами, далі ми більш детально обговоримо деякі особливості цієї програми. Також зазначимо, що якщо читач унаслідок якихось об'єктивних або суб'єктивних причин надасть перевагу іншому середовищу розробки (у тому числі й одному з перерахованих вище) — немає жодних проблем. Правда, на сторінках книги відсутня можливість описати всі (або навіть деякі) найпопулярніші середовища розробки для Python: книга, все-таки, присвячена мові програмування, а не програмному забезпеченню. Та й більшість пропонованих утиліт для роботи з програмними кодами Python зазвичай прості у використанні, універсальні в плані методів і прийомів роботи з ними, а також інтуїтивно зрозумілі. Хочеться вірити, що читач у разі потреби сам зможе впоратися з опануванням необхідного програмного забезпечення.

# Робота з середовищем PyScripter

План, що й казати, був чудовий: простий і ясний, краще не придумати. Недолік у нього був тільки один: було зовсім невідомо, як його виконати.

*Л. Керролл. «Аліса в Країні Див»*

Одразу застерігаємо, що повністю описувати програму PyScripter ми не будемо: по-перше, можливості такої немає, а, по-друге, необхідності, якщо чесно, також. Тому ми зупинимося лише на тих налаштуваннях і режимах, які критичні й будуть (або можуть бути) корисними читачеві в процесі роботи над матеріалом книги (маються на увазі, насамперед, програмні коди, які розглядаються в книзі).

Передусім, варто звернути увагу, що інтерфейс середовища розробки PyScripter підтримує різні мови. На рис. В.17 показано вікно програми PyScripter із англомовним інтерфейсом.

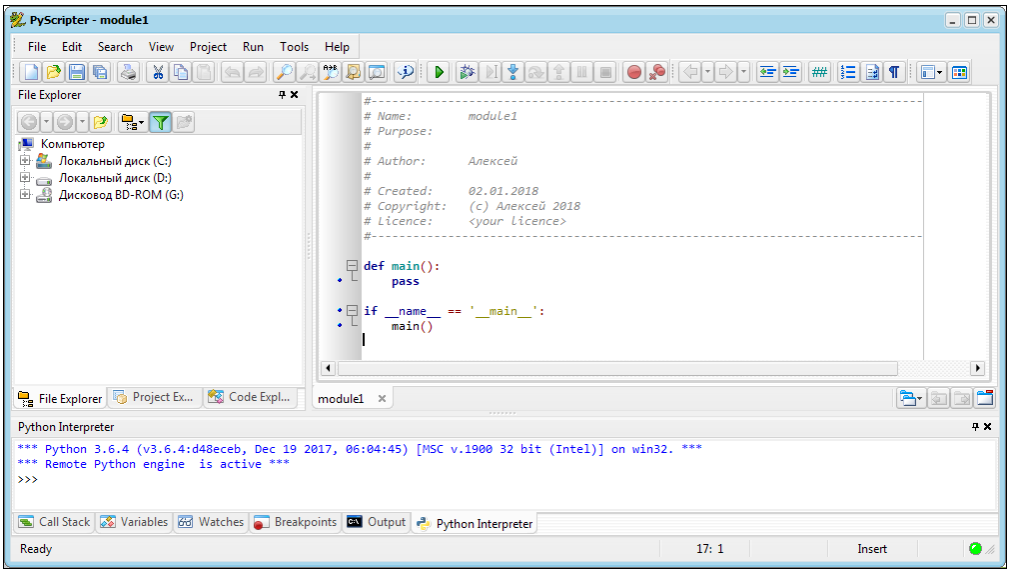


*Рис. В.17. Вікно програми PyScripter із англомовним інтерфейсом і шаблонним кодом у вікні редактора кодів*

За замовчуванням під час запуску програми PyScripter у внутрішньому вікні редактора кодів для нового, автоматично створеного (але ще не збереженого) файла пропонується шаблонний код, як це можна побачити на рис. В.17. Цей шаблонний код можна видалити й увести власний. Також користувач може змінити налаштування програми, в тому числі й уміст шаблонного коду.

Якщо ми хочемо використати іншу мову для інтерфейсу, то в меню View слід вибрати підменю Language, а в цьому підменю — команду з назвою мови, яку ми обираємо (наприклад, Russian), як показано на рис. В.18.

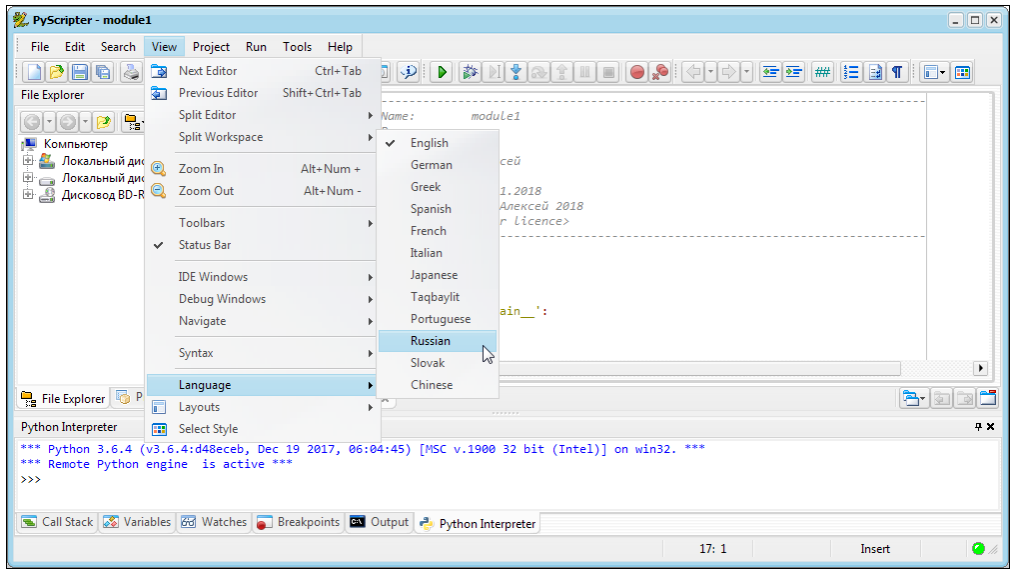


*Рис. В.18. Вибір мови для інтерфейсу програми PyScripter*

У результаті інтерфейс програми PyScripter стане (при виборі команди Russian) російськомовним. Щоб повернутися назад до англомовного інтерфейсу, в меню Вид у підменю Язык вибираємо команду Английский (рис. В.19).

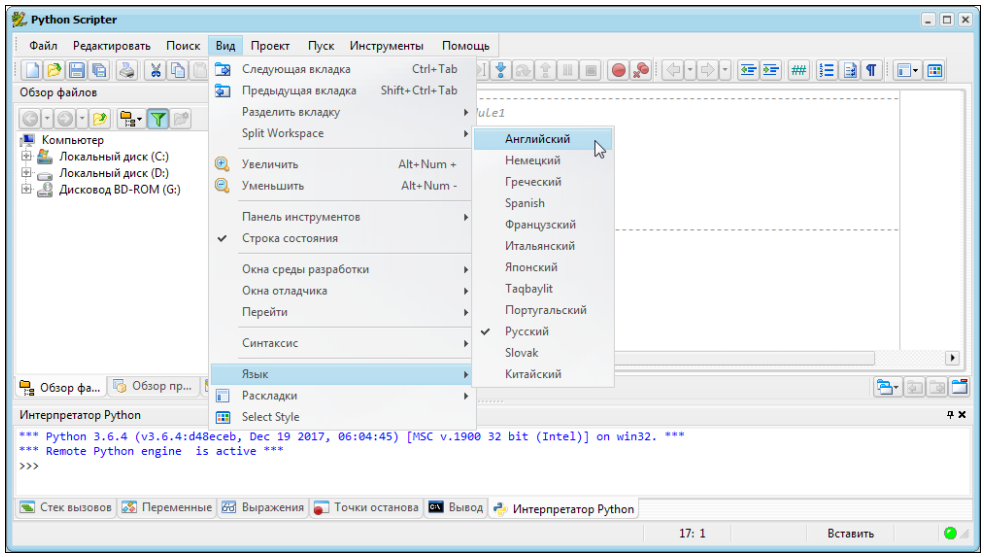


*Рис. В.19. Вікно програми PyScripter із російськомовним інтерфейсом*

Сподіваємося, що такі стандартні процедури, як створення, збереження, відкриття й закриття робочого документа (файла з програмою), у читача проблем не викличуть. Разом із тим звертаємо увагу на досить корисну утиліту: внутрішнє вікно File Explorer, безпосередньо в системі каталогів якого можна вибрати той документ, із яким збираємося працювати (рис. В.20).

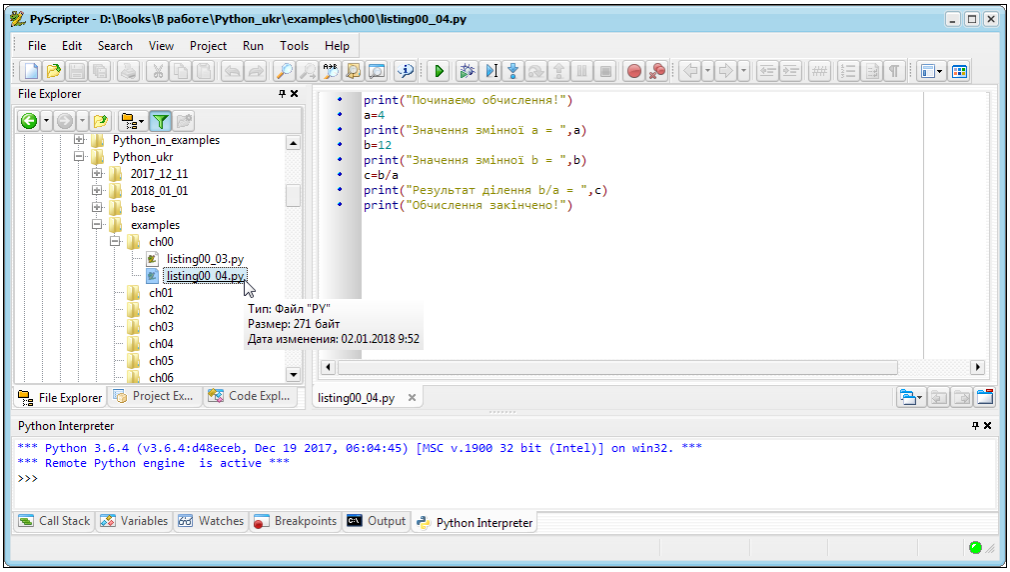


*Рис. В.20. Вибір потрібного файла з програмним кодом безпосередньо у вікні програми PyScripter*

Середовище розробки PyScripter доволі гнучке в плані налаштувань. Наприклад, щоб налаштувати параметри редактора кодів (такі, скажімо, як шрифт або режим виділення синтаксичних конструкцій) у меню Tools в підменю Options вибираємо команду Editor Options, як показано на рис. В.21.

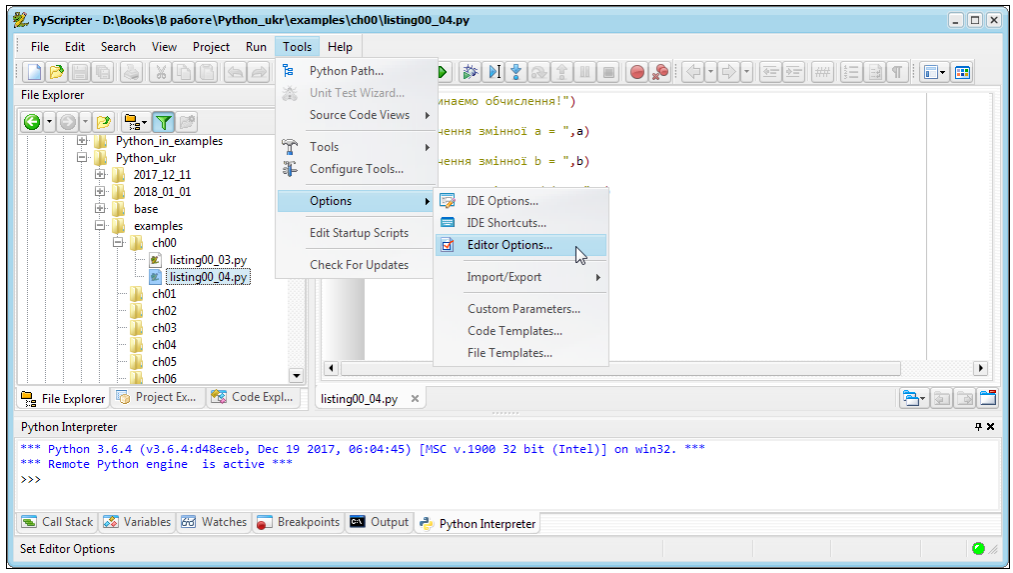


*Рис. В.21. Перехід у режим налаштування параметрів редактора програмних кодів*

У результаті відкриється діалогове вікно з назвою Editor Options, у якому, власне, і виконуються налаштування (рис. В.22).

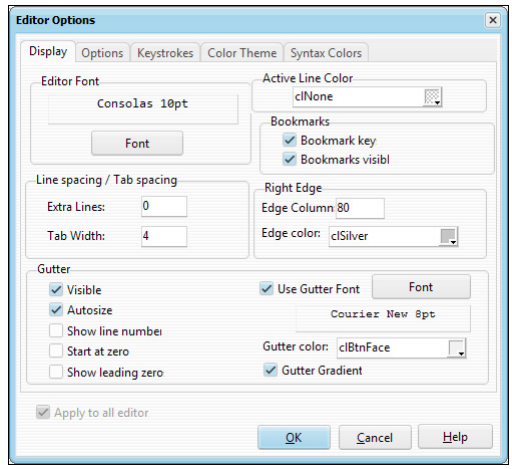


*Рис. В.22. Вікно налаштування параметрів редактора програмних кодів*

Одне досить важливе зауваження стосується програмних кодів, у яких використовується кириличний текст. Такі файли треба зберігати

у «правильному» кодуванні — інакше під час наступного відкриття в тому місці, де був кириличний текст, з'являться «карлючки». Зокрема, рекомендується перед збереженням файла в меню Edit у підменю File Format установити кодування UTF-8 (рис. В.23).

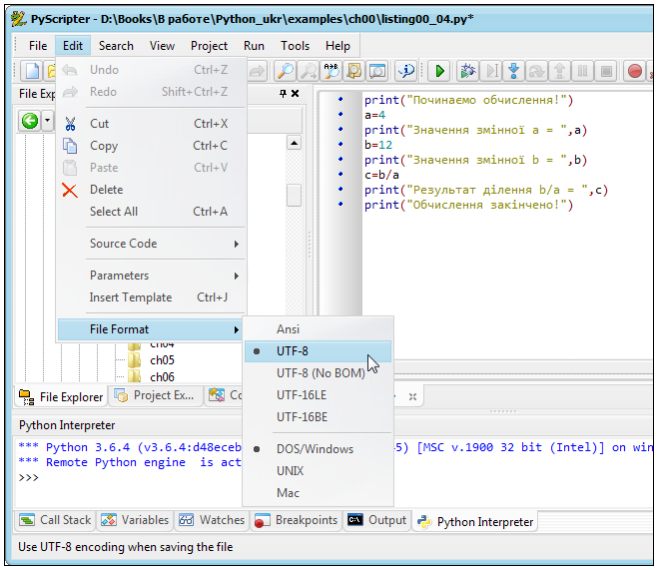


*Рис. В.23. Збереження файлів із кириличним текстом у «правильному» кодуванні для подальшої коректної роботи*

Також раніше ми відзначали, що використовувані за замовчуванням шаблони програмних кодів, що пропонуються користувачеві в різних ситуаціях (при створенні нового документа), можна змінити (відредагувати). Корисними в таких випадках будуть команди підменю Options в меню Tools. На рис. В.24 показано діалогове вікно File Templates, за допомогою якого редагуються шаблони.

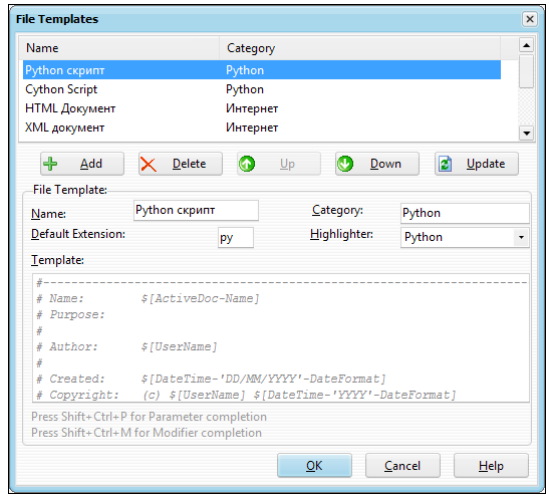


*Рис. В.24. Вікно налаштування шаблонів*

Характеризуючи ситуацію в цілому, варто сказати, що середовище PyScripter налаштовувати доволі просто, тому на практиці проблем із налаштуванням зазвичай не виникає навіть у непідготовлених користувачів. У разі нагальної потреби, зрозуміло, завжди можна звернутися до довідкової системи середовища розробки PyScripter.

У книзі ми будемо розглядати вихідні програмні коди на мові Python і результат виконання цих кодів — у текстовому форматі. Тому роботу із середовищем розробки окремо обговорювати не будемо. Фактично, задача набору й виконання програмного коду повністю лягає на плечі читача. Добре, що завдання це не надто складне.

Якщо все ж таки виникнуть непередбачені проблеми з програмним забезпеченням, нагадаємо, що в найгіршому разі програмний код можна набрати у звичайному текстовому редакторі й зберегти його у відповідний файл із розширенням `py`. Потім цей файл виконується за допомогою програми-інтерпретатора. Тобто програму-інтерпретатор все ж таки доведеться встановити.

# Подяка

> Якби кожна людина займалася своєю справою, Земля крутилася б швидше.
>
> *Л. Керролл. «Аліса в Країні Див»*

Автор висловлює щиру вдячність ***Ілоні Васильєвій***, без напруженої і плідної праці та всебічної підтримки якої ця книга не вийшла б українською мовою.

Приклади й задачі з книг із програмування звичайно проходять апробацію на терплячій, але вимогливій аудиторії — студентах і слухачах курсів. Їм — окрема подяка. Адже безпосереднє спілкування з тими, хто навчається програмуванню, дає безцінний досвід для викладача, і це, звісно, знаходить своє відображення в книгах.

Величезної подяки заслуговують читачі, які надсилали і надсилають листи зі своїми критичними зауваженнями й пропозиціями. Завдяки їхньому небайдужому ставленню втілено в життя багато чудових ідей. Від шановного читача цілком залежить, чи триватиме цей процес.

# Зворотний зв'язок

> Якщо в світі все безглуздо, – сказала Аліса, – що заважає вигадати якийсь сенс?
>
> *Л. Керролл. «Аліса в Країні Див»*

Свої пропозиції і зауваження читачі можуть направляти електронною поштою `alex@vasilev.kiev.ua` або `vasilev@univ.kiev.ua`. Книги, в першу чергу, пишуться для читачів. Тому вкрай важливо знати, що в книзі вдалося, а що — ні. Конструктивна критика й зворотний зв'язок із читачами — дуже дієві інструменти, які дозволяють рухатися у правильному напрямку.

Деяку корисну інформацію щодо цієї та інших книг автора представлено на сайті `www.vasilev.kiev.ua`. Якщо ж потрібної читачам інформації там немає, але її розміщення технічно можливе, є сенс написати листа з пропозицією додати на сторінку відсутні відомості — адреси електронної пошти наведено вище.

РОЗДІЛ 1

# Програма мовою Python

*Усе, що сказано три рази,*
*стає істиною.*

*Л. Керролл. «Аліса в Країні Див»*

Ми розпочинаємо вивчення мови Python. У цьому розділі познайомимося з базовими конструкціями, які дозволять нам створювати нескладні програмні коди. Деякі моменти спочатку пояснюватимуться на дещо поверхневому рівні, аби полегшити для новачків процес переходу від теоретизування до практичного використання Python для написання програм.

Нерідко програми, написані мовою Python, називають ***сценаріями***. Ми трохи відійдемо від традиції й термін *сценарій* уживати не будемо.

Почнемо ми з того, що обговоримо загальні принципи організації програмного коду, написаного на мові Python.

# Розмірковуючи про програму

– Скажіть, будь ласка, куди мені звідси йти?
– А куди ти хочеш потрапити? – відповів Кіт.
– Мені все одно – сказала Аліса.
– Тоді все одно, куди і йти – зауважив Кіт.

*Л. Керролл. «Аліса в Країні Див»*

Програма в Python — це послідовність команд. Жодних спеціальних інструкцій для формального позначення початку чи кінця коду програми використовувати не треба. Таким чином, під час написання програми ми розміщуємо команди одна за одною — у тому порядку, як вони повинні виконуватися. Команда розташовується на початку рядка. Відступів робити не потрібно. В кожному рядку зазвичай по одній команді — тобто кожна команда в новому рядку.

У цьому разі йдеться про «прості» команди, на зразок команди виводу в консольне вікно текстового повідомлення. Як зазначалося вище, ці команди розташовані на початку рядка, відступ (пробіл) перед ними не ставиться, і в кінці команди теж нічого ставити не треба. Трохи згодом ми познайомимося з інструкціями керування — такими, як оператор циклу й умовний оператор. У цих синтаксичних конструкціях за допомогою пробілів виділяються складові частини виразу для оператора (блок команд тіла оператора). Але про це поговоримо пізніше. На даний момент важливо запам'ятати, що пробіл у мові Python — важливий елемент із точки зору синтаксису й використовувати його слід надзвичайно обережно. Зайвий пробіл може зумовлювати помилку.

У принципі, це все, що нам поки що необхідно знати про структуру програми для того, щоб розпочати написання програмного коду. З часом ми познайомимося з різними синтаксичними конструкціями, які вносять у програмний код велику інтригу й роблять процес програмування цікавим і часом непередбачуваним. Але це буде пізніше.

Що важливо зараз? Зараз важливо розуміти, що команди в програмі повинні бути коректними, як мінімум, з точки зору синтаксису мови Python. Проте навіть формально правильні команди не гарантують правильності виконання програми. Зазвичай програми пишуть для розв'язання тієї чи іншої задачі. Задача розв'язується відповідно до певного алгоритму, який розроблює й реалізує переважно програміст. Тобто в програмі реалізується деякий алгоритм. Якщо цей алгоритм неправильний, то і програма видасть не той результат, якого від неї очікують. Тому написання програми починається з розробки алгоритму, а вже потім цей алгоритм реалізується у вигляді команд програми. Ми для простоти виходимо з припущення, що алгоритм є, він — правильний, і його лише необхідно «перекласти» на мову інструкцій Python.

Практично будь-яка програма оперує деякими даними. Це може бути як реально велика база даних, так і одне-єдине значення — принципової різниці тут немає. Важливо те, що дані в програмі повинні якось зберігатися. Ми поки що будемо «зберігати» дані за допомогою ***змінних***. Щодо змінних є два важливих моменти:

- у змінної є ім'я (або назва);
- змінна ***посилається*** на деяке значення.

У принципі, значення, на яке посилається змінна, ми можемо:

- прочитати;
- змінити.

І в тому, і в іншому випадках ми використовуємо у контексті команди ім'я змінної. Оскільки в Python тип змінної явно не зазначають (він визначається за значенням, на яке посилається змінна), то попередньо оголошувати змінні не потрібно. Просто за першого використання змінної їй одразу присвоюється значення.

Зазвичай, коли пояснюють призначення змінних, порівнюють їх, наприклад, із кошиком або банківською коміркою. Значення змінної за такої аналогії — це те, що знаходиться в кошику/комірці. Поки що ми можемо думати про змінну саме так. Проте у Python справи зі змінними виглядають трохи інакше. Технічно змінна містить адресу в області пам'яті, де зберігається значення, яке ототожнюється зі значенням змінної. Втім, дуже часто (у простих випадках) зовнішній ефект такий, неначе б змінна реально містила своє значення — як кошик чи банківська комірка. На даному етапі механізм зберігання значень змінних не важливий. Трохи згодом, коли це питання стане актуальним, ми розставимо всі крапки над «і».

Спочатку ми розглянемо невелику програму. У ній, крім декількох простих команд, будуть використані коментарі. ***Коментар*** — це текст, призначений для програміста. Інтерпретатором коментар ігнорується. Для створення коментаря використовують символ #. Все, що праворуч від символу #, є коментарем.