

Про автора Head First Python

Тієї час прогулянки Пол зупиняється, аби обговорити правильну вимову слова «tuple» (тобто «кортеж») зі своєю багатастраждальною дружиною.



Звичайна реакція Дейдри. ©

Пол Беррі живе і працює в Карлоу — маленькому містечку з населенням близько 35 тисяч осіб, розташованому за 80 км на південний захід від Дубліна.

Пол має ступінь бакалавра наук у галузі інформаційних систем і ступінь магістра в галузі обчислень. Він також закінчив аспірантуру й отримав свідоцтво на право викладання і навчання.

Пол працює в Технологічному інституті Карлоу з 1995 року. Перш ніж почати викладацьку діяльність, він десять років присвятив ІТ-індустрії, працював в Ірландії і Канаді, причому більша частина його роботи була пов'язана з медичними закладами. Пол має дружину Дейдру, у них троє дітей (двоє зараз вчаться в коледжі).

Мова програмування *Python* (і пов'язані з нею технології) стала невід'ємною частиною бакалаврського лекційного курсу Пола починаючи з 2007 року.

Він є автором (або співавтором) ще чотирьох книжок: дві з них про *Python*, а ще дві — про мову програмування *Perl*, а також цілої низки статей, опублікованих у «*Linux Journal Magazine*».

Пол виріс у Белфасті (Північна Ірландія), і це багато в чому пояснює деякі особливості його поглядів і кумедний акцент.

Ви можете зв'язатися з Полом Беррі у Твіттері (@barryp); також він має власну домашню сторінку за адресою <http://paulbarry.itcarlow.ie>.

Зміст (коротко)

Вступ	27
1. Підвалини. Швидке занурення	39
2. Списки даних. Робота з упорядкованими даними	85
3. Структуровані дані. Робота зі структурованими даними	133
4. Повторне використання. Функції та модулі	183
5. Створення вебзастосунку. Повернення в реальний світ	233
6. Зберігання та обробка даних. Де зберігаються дані	281
7. Застосування бази даних. Застосуємо DB-API у Python	319
8. Трохи про класи. Абстракція поведінки і стану	347
9. Протокол управління контекстом. Підключення до інструкції «with» мови Python	373
10. Декоратори функцій. Обгортання функцій	401
11. Обробка винятків. Що робити, коли щось іде не так	451
11 1/2. Трохи про багатопотоковість. Обробка очікування	499
12. Просунуті ітерації. Божевільні цикли	515

Додатки

A: Інсталяція. Інсталяція Python	559
B: PythonAnywhere. Розгортання вашого вебзастосунку	567
C: ТОП-10 тем, які ми не висвітлили. Завжди є чому навчитися	577
D: ТОП-10 проектів, що ми їх не розглянули. Ще більше інструментів, бібліотек і модулів	589
E: Додучайтесь! Спільнота Python	601

Зміст (докладно)

Вступ

Ваш мозок і Python. Коли ви пробуєте щось засвоїти, ваш мозок намагається надати вам послугу, переконуючи, що все це не варто уваги. Він думає: «Краще перейматися важливішими речами, наприклад, небезпечними дикими тваринами або тим, що катання голяка на сноуборді — погана ідея». Як же переконали свій мозок у тому, що від знання Python залежить ваше життя?

Для кого ця книжка?	28
Про що насправді думає ваш мозок	29
Ми знаємо, про що ви подумали	29
Метапізнання: мислення про мислення	31
Ось що зробили МИ	32
Прочитайте мені (1)	34
Подяки	37

Купити книгу на сайті kniga.biz.ua

1

Пігвалини

Швидке занурення

Починаємо програмувати на Python якомога швидше. У цьому розділі ми ознайомимося з основами програмування на Python і зробимо це в характерному для *Head First* стилі: із місця в кар'єр. Через кілька сторінок ви запустите свою першу програму. До кінця розділу ви зможете не лише запускати типові програми, але й розуміти їхній код (і це ще не все!) Водночас ви познайомитеся з деякими особливостями мови Python.

Розуміння вікон IDLE	42
Виконання коду, одна інструкція за раз	46
Функції + модулі = стандартна бібліотека	47
Структури даних надходять вбудованими	51
Виклик методу повертає результат	52
Ухвалення рішень про запуск блоків коду	53
Що за «else» можна отримати з «if»?	55
Набори коду можуть містити вбудовані набори	56
Повернення в командну оболонку Python	60
Експериментуємо в оболонці	61
Ітерація послідовності об'єктів	62
Повторюємо певну кількість разів	63
Застосуємо розв'язання завдання № 1 до нашого коду	64
Організація паузи виконання	66
Генерація випадкових чисел на Python	68
Створення серйозного бізнес-застосунок	76
Відступи вас дратують?	78
Попросимо інтерпретатор допомогти із функцією	79
Експерименти з діапазонами	80
Код із Розділу 1	84



2

Списки даних

Робота з упорядкованими даними

Усі програми обробляють дані, і програми на Python — не виняток. Насправді, досить розширитися — *дані скрізь*. Адже програмування — це, переважно, робота з даними: *отримання даних, обробка даних, інтерпретація даних*. І щоби працювати з даними ефективніше, потрібен якийсь контейнер, куди їх можна *складати*. Python надає зручні структури даних *широкого застосування*: **списки, словники, кортежі і множини**. У цьому розділі ми побіжно розглянемо всю цю четвірку, а тоді заглибимося у вивчення *списків* (інші три структури будуть докладніше розглянуті в наступному розділі). Ми вже торкалися цієї теми раніше, оскільки все, із чим нам доводиться стикатися при програмуванні на Python, так чи інакше стосується роботи з даними.

Числа, рядки — і об'єкти	86
Познайомтеся із чотирма вбудованими структурами даних	88
Невпорядкована структура даних: словник	90
Структура даних, що не дозволяє дублювати об'єкти: множина	91
Створення літеральних списків	93
Застосуйте редактор для роботи із фрагментом коду, що перевищує пару рядків	95
«Вирощування» списку під час виконання	96
Перевірка приналежності за допомогою «in»	97
Видалення об'єктів зі списку	100
Розширення списку за допомогою об'єктів	102
Додавання об'єкта у список	103
Як скопіювати структуру даних	111
Списки розширюють позначення у квадратних дужках	113
Зі списками можна використовувати діапазони	114
Початок і кінець діапазону у списках	116
Використання зрізів у списках	118
Цикл «for» у Python розуміє списки	124
Зрізи Марвіна в деталях	126
Коли не слід застосовувати списки	129
Код із Розділу 2	130

0	1	2	3	4	5	6	7	8	9	10	11
D	o	n	'	t		p	a	n	i	c	!
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Купити книгу на сайті kniga.biz.ua >>>

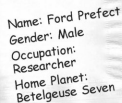
3

Структуровані дані

Робота зі структурованими даними

Списки в Python дуже зручні, але вони не є панацеєю. Коли є *дійсно* структуровані дані (для зберігання яких список виявляється не кращим вибором), то порятунок треба шукати у вбудованих **словниках** Python. Словники «із коробки» дозволяють зберігати й обробляти дані, що їх можна представити у вигляді пар *ключ / значення*. У цьому розділі ми докладно розглянемо словники, а також **множини** і **кортежі**. Разом зі списками (із ними ми знайомилися в Розділі 2) словники, множини і кортежі є вбудованими інструментами для роботи з даними, що дозволяють створити потужну комбінацію даних мовою Python.

Словники зберігають пари ключ/значення	134
Як визначити словники в коді	136
Порядок введення НЕ підтримується	137
Пошук значень за допомогою квадратних дужок	138
Робота зі словниками під час виконання програми	139
Оновлення лічильника частот	143
Ітерації по записях у словниках	145
Ітерації за ключами і значеннями	146
Ітерації за словниками з використанням items	148
Наскільки динамічні словники?	152
Уникнення KeyError під час виконання	154
Перевірка входження за допомогою «in»	155
Забезпечте ініціалізацію ключа перед застосуванням	156
Заміна «in» на «not in»	157
Застосування методу «setdefault» для роботи	158
Створюємо множини ефективно	162
Використання переваг методів множин	163
Підрунтя кортежів	170
Комбінування вбудованих структур даних	173
Доступ до даних, що зберігаються у складних структурах	179
Код із Розділу 3	181



Name: Ford Prefect
Gender: Male
Occupation:
Researcher
Home Planet:
Betelgeuse Seven

4

Повторне використання

Функції та модулі

Повторне використання коду — ключ до створення стабільних систем. У випадку з Python все повторне використання починається і закінчується **функціями**. Візьміть кілька рядків коду, дайте їм ім'я — й у вас уже готова **функція** (що її можна використовувати повторно). Візьміть колекцію функцій, запакуйте їх у файл — й у вас з'явиться готовий **модуль** (його теж можна використовувати повторно). Це правда, коли кажуть: *ділитися приємно*, тож наприкінці цього розділу ви вже зможете створювати код для **багаторазового** та **спільного** використання, завдяки усвідомленню того, як працюють функції та модулі Python.

Повторне використання коду за допомогою функцій	184
Знайомство із функціями	185
Виклик вашої функції	188
Функції здатні приймати аргументи	192
Повернення одного значення	196
Повернення більше одного значення	197
Згадуємо вбудовані структури даних	199
Створення універсальної корисної функції	203
Створення іншої функції	204
Указання значень за замовчуванням для аргументів	208
Позиційні та іменовані аргументи	209
Повторимо, що ми дізналися про функції	210
Запуск Python із командного рядка	213
Створення файлів, необхідних для установки	217
Створення файлу дистрибутиву	218
Установлення пакетів за допомогою «pip»	220
Демонстрація семантики виклику за значенням	223
Демонстрація семантики виклику за посиланням	224
Установка інструментів розробника для тестування	228
Чи сумісний наш код із PER 8?	229
Розгляд повідомлень про помилки	230
Код із Розділу 4	232



Модуль

Купити книгу на сайті kniga.biz.ua >>>

5

Створення вебзастосунку Повернення в реальний світ

На цьому етапі ви вже достатньо знаєте про Python, аби вважатися **небезпечними**. Ви вже засвоїли перші чотири розділи цієї книжки, тож тепер здатні продуктивно використовувати Python у багатьох галузях (хоча ще багато чого вам належить дізнатися). Замість того щоби звернутися до цих питань, у цьому та наступних розділах ми зосередимося на розробці вебзастосунків, що дозволить нам оцінити переваги мови Python. Водночас ви трохи більше дізнаєтесь про Python.

Python: що вам уже відомо	234
Чого ми вимагаємо від нашого вебзастосунку?	238
Давайте встановимо Flask	240
Як працює Flask?	241
Перший запуск вебзастосунку Flask	242
Створення об'єкта вебзастосунку Flask	244
Декорування функції URL	245
Запуск функцій вашого вебзастосунку	246
Розміщення функціональності у веб	247
Побудова HTML-форми	251
Шаблони, пов'язані з вебсторінками	254
Візуалізація шаблонів із Flask	255
Відтворення HTML-форми вебзастосунку	256
Підготовка до запуску коду з шаблонами	257
Коди стану HTTP	260
Обробка опублікованих даних	261
Оптимізація циклу редагування/зупинка/запуск/перевірка	262
Доступ до даних HTML-форми за допомогою Flask	264
Використання даних запити у вашому вебзастосунку	265
Виводимо результат у вигляді HTML	267
Підготовка вашого вебзастосунку до розгортання у хмарі	276
Код із розділу 5	279



6

Зберігання та обробка даних Де зберігаються дані

Рано чи пізно виникає необхідність забезпечити надійне зберігання даних. І коли справа доходить до **зберігання даних**, Python вам допоможе. У цьому розділі ви дізнаєтесь про зберігання й отримання даних із *текстових файлів*, що як технології зберігання можуть здатися надто простими, однак використовуються у багатьох проблемних галузях. Окрім збереження та отримання даних із файлів, ви засвоїте ще деякі премудрості, коли йтиметься про обробку даних. «Серйозний матеріал» (зберігання інформації в базах даних) припасений нами для наступного розділу, однак із файлами теж доведеться повозитися.

Робота з даними вашого вебзастосунку	282
Python дозволяє відкривати, обробляти і закривати	283
Читання даних з існуючого файлу	284
Краще «with», ніж «відкрити, обробити, закрити»	286
Перегляд журналу у вебзастосунку	292
Досліджуємо вихідні дані за допомогою вихідного коду сторінки	294
Настав час екранувати (ваші дані)	295
Перегляд усього журналу у вебзастосунку	296
Реєстрація окремих атрибутів вебзапиту	299
Реєстрація даних в один рядок з роздільником	300
Від вихідних даних до даних у читабельному форматі	303
Генеруємо читабельний вивід за допомогою HTML	312
Вбудовуємо логіку відтворення у шаблон	313
Створення читабельного виводу за допомогою Jinja2	314
Поточний стан коду нашого вебзастосунку	316
Стаavimo питання про дані	317
Код із Розділу 6	318

Form Data	Remote_addr	User_agent	Results
ImmutableMultiDict([('phrase', 'hitch-hiker'), ('letters', 'aeiou')])	127.0.0.1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36	('e', 'i')

Купити книгу на сайті kniga.biz.ua >>>

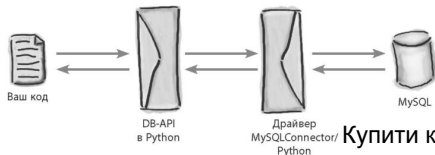
7

Застосування бази даних

Застосуємо DB-API у Python

Зберігати інформацію в реляційній базі даних дуже зручно. У цьому розділі ви дізнаєтеся, як організувати взаємодію з популярною базою даних (БД) **MySQL**, використовуючи універсальний прикладний програмний інтерфейс **DB-API**. Інтерфейс DB-API (входить до складу стандартної бібліотеки Python) дозволяє створювати код, що не залежить від конкретної бази даних... за умови, якщо база даних розуміє SQL. Хоча ми будемо використовувати MySQL, ніщо не завадить вам використовувати код DB-API з вашою улюбленою реляційною базою даних, хай би якою вона була. Давайте подивимося, як користуватися реляційною базою даних у Python. У цьому розділі не так багато нового з погляду вивчення Python, але застосування Python для спілкування з БД — це **дуже важливо**, тому варто докласти зусиль.

Включаємо підтримку баз даних у вебзастосунок.....	320
Завдання 1. Установка сервера MySQL.....	321
Вступ до Python DB-API.....	322
Завдання 2. Встановлення драйвера бази даних MySQL для Python.....	323
Установка MySQL-Connector/Python.....	324
Завдання 3. Створення бази даних і таблиць для вебзастосунків.....	325
Обираємо структуру журнальованих даних.....	326
Переконаймося, що таблиця готова до використання.....	327
Завдання 4. Програмування операцій із базою даних і таблицями нашого вебзастосунку.....	334
Збереження даних — лише половина справи.....	338
Як найкраще повторно використати код бази даних?.....	339
Подумайте, що ви збираєтеся використовувати повторно.....	340
А як щодо імпорту?.....	341
Ви бачили цей шаблон раніше.....	343
Прикрість не така вже прикра.....	344
Код із Розділу 7.....	345



Купити книгу на сайті kniga.biz.ua >>>

8

Трохи про класи

Абстракція поведінки і стану

Класи дозволяють зв'язати поведінку коду і його стан. У цьому розділі ми поки що залишимо у спокої вебзастосунок і почнемо вчитися створювати **класи** Python. Це вміння знадобиться вам при створенні диспетчера контексту. Класи є настільки корисною річчю, що вам у будь-якому разі варто ознайомитися з ними ближче, тому ми присвятили їм окремий розділ. Ми не будемо розглядати класи у всіх подробиць, а торкнемося лише тих аспектів, що стануть у пригоді при створенні диспетчера контексту, на який очікує наш вебзастосунок.

Підключаємося до інструкції «with».....	348
Об'єктно-орієнтований буквар.....	349
Створення об'єктів із класів.....	350
Об'єкти володіють загальною поведінкою, але не станом.....	351
Розширюємо можливості CountFromBy.....	352
Виклик методу: розуміння подробиць.....	354
Додавання методу у клас.....	356
Важливість «self».....	358
Межі видимості.....	359
Додавайте до імен атрибутів приставку «self».....	360
Ініціалізація значення (атрибута) перед застосуванням.....	361
Ініціалізація атрибутів в «init» із подвійними підкресленнями.....	362
Ініціалізація атрибутів в «_init_».....	363
Подання «CountFromBy».....	366
Визначення подання CountFromBy.....	367
Визначення доцільних замовчувань для CountFromBy.....	368
Класи: що ми про них знаємо.....	370
Код із Розділу 8.....	371

```

class CountFromBy:
    def __init__(self, v: int, i: int) -> None:
        self.val = v
        self.incr = i
    def increase(self) -> None:
        self.val += self.incr
  
```

9

Протокол управління контекстом

Підключення до інструкції «with» мови Python

Настав час застосувати усе вивчене на практиці. У Розділі 7 ми обговорили використання **реляційних баз даних** у Python, а в Розділі 8 ознайомилися з використанням **класів** у кодї Python. У цьому розділі ми об'єднаємо обидві методи і створимо **диспетчер контексту**, що дозволить розширити інструкцію **with** для роботи з системами реляційних баз даних. У цьому розділі ви підключитеся до інструкції **with**, створивши новий клас, що відповідає вимогам **протоколу управління контекстом** у мові Python.

Обираємо кращий спосіб спільного використання коду для роботи з базою даних	374
Управління контекстом за допомогою методів	376
Ви вже бачили, як діє диспетчер контексту	377
Створення нового класу диспетчера контексту	378
Ініціалізація класу параметрами з'єднання з базою даних	379
Виконуємо налаштування в «__enter__»	381
Виконання завершального коду за допомогою «__exit__»	383
Перегляд коду вебзастосунку	386
Виклик функції «log_request»	388
Зміна функції «log_request»	389
Згадаймо функцію «view_the_log»	390
Змінився не лише код	391
Зміна функції «view_the_log»	392
Відповіді на питання про дані	397
Код із Розділу 9	398

```
File Edit Window Help Checking out log
mysql -u vsearch -p vsearchlogns
Enter password:
Welcome to MySQL monitor ...

mysql> select * from log;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | ts      | phrase          | letters | ip      | browser_string | results |
+----+-----+-----+-----+-----+-----+-----+
| 1  | 2018-03-09 13:40:46 | life, the uni... | ything | 127.0.0.1 | firefox        | [' ', 'e', 'i', 'a'] |
| 2  | 2018-03-09 13:40:47 | hitch-hiker    |         | 127.0.0.1 | safari         | [' ', 'e'] |
| 3  | 2018-03-09 13:42:15 | galaxy         |         | 127.0.0.1 | chrome         | [' ', 'e'] |
| 4  | 2018-03-09 13:42:07 | hitch-hiker    |         | 127.0.0.1 | firefox        | set() |
+----+-----+-----+-----+-----+-----+-----+
< rows in set (0.0 sec)

mysql> quit
bye
```

10

Декоратори функцій

Обгортання функцій

Що стосується розширення вашого коду, то протокол управління контекстом із Розділу 9 — не єдина можливість. Python також дозволяє додавати код до існуючих функцій, технологію, за допомогою якої можна додати код до існуючих функцій, не змінюючи наявний код. Якщо ви побачите у цьому схожість із чорною магією, не переймайтеся: нічого подібного. Однак чимало програмістів, які працюють із Python, вважають створення декораторів одним із найскладніших процесів, тому користуються цією можливістю рідше, ніж мали б. У цьому розділі ми продемонструємо, що створення і використання декораторів — це не дуже складно, незважаючи на їхню «просунутість».

Ваш вебсервер (а не ваш комп'ютер) запускає ваш код	404
Підтримка сесій у Flask дозволяє зберігати стан	406
Пошук за словником дозволяє отримати стан	407
Організація входу до системи за допомогою сесій	412
Давайте зробимо вихід із системи і перевірку стану	415
Передаємо функцію функції	424
Викликаємо передану функцію	425
Приймання списку аргументів	428
Обробка списку аргументів	429
Приймання словника аргументів	430
Обробка словника аргументів	431
Приймаємо будь-яку кількість аргументів будь-якого типу	432
Створення декоратора функцій	433
Останній крок: обробка аргументів	439
Налаштування вашого декоратора на роботу	442
Назад до обмеження доступу до /viewlog	446
Код із Розділу 10	448

```
checker.py - /Users/baw/Desktop/NewBook/ch10checker.py (3.5.1)

from flask import session
from functools import wraps

def check_logged_in(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if 'logged_in' in session:
            return func(*args, **kwargs)
        return 'You are NOT logged in.'
    return wrapper
```

Купити книгу на сайті kniga.biz.ua >>>

11

Обробка винятків

Що робити, коли щось іде не так

Хай би яким гарним був ваш код, іноді все одно щось іде не так. Ви успішно виконали всі приклади із книжки та, ймовірно, переконалися, що наявний код працює. Однак чи означає це, що його можна вважати надійним? Найвірогідніше, ні. Створювати код і сподіватися, що нічого поганого ніколи не станеться, у кратшому разі наївно. А в гіршому — небезпечно, оскільки щось непередбачене часом усе ж відбувається (і буде відбуватися). При створенні коду краще бути обережним, ніж довірливим. І обережність потрібна, щоби ваш код робив саме те, що вам потрібно, і чітко реагував, якщо щось піде не так.

Бази даних не завжди доступні	456
Вебатаки — справжній біль	457
Вхід/вихід буває повільним (іноді)	458
Виклики функцій можуть завершуватися невдачею	459
Завжди використовуйте «try» для коду з підвищеним ризиком помилок	461
Одна інструкція «try», але декілька «except»	464
Оброблювач будь-яких винятків	466
Дізнаймося про винятки із «sys»	468
Універсальний обробник винятків (переглянутий)	469
Назад — до нашого вебзастосунок	471
Тиха обробка винятків	472
Обробка інших помилок у базі даних	478
Уникайте тісних зв'язків у коді	480
Модуль Dbcm, ще раз	481
Створення користувачького винятку	482
Що ще може піти не так із «Dbcm»?	486
Обробка «SQLException» відрізняється	489
Викликаємо «SQLException»	491
Швидкий огляд: додасмо надійності	493
Як бути з очікуванням? Це залежить від	494
Код із Розділу 11	495

```

...
Exception
+++ StopIteration
+++ StopAsyncIteration
+++ ArithmeticError
|   +++ FloatingPointError
|   +++ OverflowError
|   +++ ZeroDivisionError
+++ AssertionError
+++ AttributeError
+++ BufferError
+++ EOFError
...

```

11^{3/4}

Обробка очікування

Трохи про багатопотоковість

Іноді для виконання коду може знадобитися чимало часу. Іноколи це стає проблемою, іноколи — ні. Якщо код працює «за кулісами» і йому потрібно 30 секунд, аби зробити якісь свої справи, очікування не перетворюється на проблему. Однак якщо користувач протягом 30 секунд очікує відповіді від застосунку, це стає аж надто відчутним. Розв'язання проблеми залежить від того, які саме дії ви намагаєтеся виконати (і хто саме очікує). У цьому короткому розділі ми обговоримо деякі варіанти, а потім розглянемо одне з рішень наступної проблеми: як діяти, якщо щось вимагає занадто багато часу?

Очікування: що робити?	500
Як ви запитуйте свою базу даних?	501
Інструкції INSERT і SELECT бази даних різняться	502
Робимо кілька справ одночасно	503
Не засмучуємося: використовуємо потоки	504
Передусім: не панікувати	508
Не сумуємо: Flask нам допоможе	509
Чи надійний ваш вебзастосунок зараз?	512
Код із Розділу 11 ^{3/4}	513

Зачекайте!



Купити книгу на сайті kniga.biz.ua >>>

Просунуті ітерації

Божевільні цикли

Можна тільки дивуватися, скільки часу наші програми проводять у циклах! А втім, у цьому немає нічого дивного, бо більшість програм призначена для швидкого виконання однакових дій безліч разів поспіль. Говорячи про оптимізацію циклів, можна виділити два підходи: 1) поліпшення синтаксису (щоби спростити визначення циклів) і 2) поліпшення способу виконання (щоби прискорити роботу циклів). За часів Python 2 (тобто досить давно) творці мови додали єдину нову можливість, що реалізує обидва підходи і називається доволі дивно — **comprehension**, або **генераторний вираз**.



Читання CSV-даних у вигляді списків	517
Читання CSV-даних як словника	518
Видалення пробілів і розбивка вихідних даних	520
Будьте обережні при складанні ланцюжків із викликів методів	521
Перетворення даних на потрібний формат	522
Перетворення на словник списків	523
Виявлення патерна за допомогою списків	528
Перетворення патерна на генераторний вираз	529
Детальніше про генераторний вираз	530
Визначення генератора словників	532
Розширені генераторні вирази із фільтрами	533
Долаємо складності, як це заведено в Python	537
Генераторний вираз множини у дії	543
А як щодо «генераторних виразів кортежів»?	545
Круглі дужки навколо коду == Генератор	546
Застосування listcomp для обробки URL	547
Використання виразу-генератора для обробки URL	548
Визначимо, що має робити функція	550
Відчуєте міць функцій-генераторів	551
Досліджуємо функцію-генератор	552
Останнє питання	556
Код із Розділу 12	557

Додатки

A: Інсталяція. Інсталяція Python

Передусім давайте встановимо Python на вашому комп'ютері. Хай би якою ОС ви користувалися — Windows, Mac OS X або Linux,— Python для вас доступний. Послідовність установки на кожну із цих платформ залежить від організації процесу в кожній з операційних систем (ми знаємо, що це... трохи шокує, чи не так?), але розробники Python добре попрацювали, створивши майстер установки для кожної з популярних систем. У цьому додатку ви дізнаєтеся, як установити Python на свій комп'ютер.

Інсталяція Python 3 для Windows	560
Перевіряємо правильність установки Python 3 для Windows	561
Розширюємо набір інструментів Python 3 у Windows	562
Інсталяція Python 3 для Mac OS X (macOS)	563
Перевірка та налаштування Python 3 для Mac OS X	564
Інсталяція Python 3 для Linux	565

B: PythonAnywhere.

Розгортання вашого вебзастосунок

Наприкінці Розділу 5 ми зазначили, що розгортання вебзастосунок у хмарі забере лише 10 хвилин. Настав час виконати обіцянку. У цьому додатку ми збираємося провести вас через весь процес розгортання вебзастосунок в PythonAnywhere за 10 хвилин від самого початку до повністю розгорнутого застосунок. PythonAnywhere — це служба, популярна у спільноті кодерів на Python, тому що вона працює в цілковитій відповідності з очікуваннями, має відмінну підтримку Python (а також Flask) і — що найважливіше — дозволяє почати з безкоштовного розміщення вебзастосунок.

Крок 0: Невеличка підготовка	568
Крок 1: Реєструємося в PythonAnywhere	569
Крок 2: Завантажуємо файли до хмари	570
Крок 3: Витягуємо і встановлюємо код	571
Крок 4: Створюємо початковий вебзастосунок (1)	572
Крок 5: Налаштовуємо вебзастосунок	574
Крок 6: Запускаємо хмарний вебзастосунок!	575

C: ТОП-10 тем, які ми не висвітлили.***Завжди є чому повчитися***

Ми й не прагнули розглянути всі питання та аспекти. Наша мета — познайомити вас із Python, аби ви якомога швидше могли почати працювати із ним. Ми могли б охопити більше тем, але не зробили цього. У цьому додатку ми обговоримо топ-10 тем, які ми обов'язково розглянули би докладніше, якби мали ще приблизно 600 вільних сторінок. Не всі із цих 10 пунктів будуть для вас цікавими або актуальними, але ви можете швидко проминути їх у пошуках більш важливого для вас матеріалу або відповіді на наболіле питання. Усі технології програмування, наведені у цьому додатку, готові до використання в Python і його інтерпретаторі.

1. А що там із Python 2?	578
2. Віртуальне програмне оточення	579
3. Детальніше про об'єкту орієнтованість	580
4. Форматування рядків — і таке інше	581
5. Сортування	582
6. Більше зі стандартної бібліотеки	583
7. Паралельне виконання коду	584
8. Графічний інтерфейс із використанням Tkinter (і веселощі з turtle)	585
9. Робота не завершена, поки не протестована	586
10. Налаштування, налагодження, налагодження	587

D: ТОП-10 проєктів, що ми їх не розглянули.***Ще більше інструментів, бібліотек і модулів***

Ми знаємо, про що ви подумали, прочитавши заголовок цього додатка.

А чому би їм не назвали попередній додаток: «Топ-20 тем, що не були висвітлені»? І навіщо нам ще 10 проєктів? Річ у тім, що в попередньому додатку ми обмежили обговорення інструментами, що поставляються разом із Python (ті самі «батареї в комплекті»). У цьому додатку ми рушимо далі й обговоримо низку технологій, що стали доступними саме тому, що існує Python. Тут перелічено багато корисного, але — як і у випадку з попереднім додатком, — якщо ви побіжно переглянете ті кілька сторінок, це неодмінно піде вам на користь.

1. Альтернативи командному рядку >>>	590
2. Альтернативи інтегрованому середовищу IDLE	591
3. Jupyter Notebook: IDE на основі веб	592
4. Наука роботи з даними	593
5. Технології веброзробки	594
6. Робота з вебданими	595
7. Ще більше джерел даних	596
8. Інструменти програмування	597
9. Ківу: наш вибір у номінації «Кращий проєкт усіх часів»	598
10. Альтернативні реалізації	599

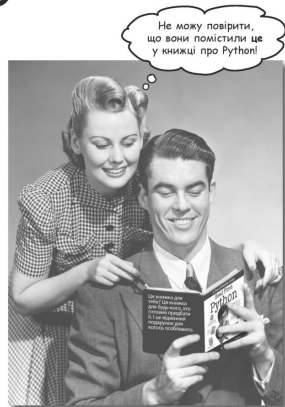
Е: Долучайтесь! Спільнота Python

Python — це набагато більше, ніж відмінна мова програмування. Це чудова компанія. Спільнота Python гостинна, різноманітна, відкрита, дружня, щедра і готова ділитися. Навіть дивно, що ніхто ще не надрукував таку вітальну листівку! Якщо серйозно, то Python — не стільки мова, скільки особлива наука програмування. Навколо Python сформувалася ціла екосистема: корисні книжки, блоги, вебсайти, конференції, зустрічі, групи користувачів і непересічні особистості. У цьому додатку ми познайомимося зі спільнотою Python і подивимося, що вона може запропонувати. Не пишіть код на самоті: **долучайтесь до спільноти!**

BDFL: Benevolent Dictator for Life — великодушний довічний диктатор	602
Толерантна спільнота: повага до різноманіття	603
Подкасти Python	604
Дзен Python	605
Яку книжку варто прочитати наступною?	606
Наші улюблені книжки про Python	607

Як користуватися цією книжкою

Вступ



У цьому розділі ми відповімо на актуальне питання: «ЧОМУ вони помістили це в книжку про Python?»

Купити книгу на сайті kniga.biz.ua >>>

Для кого ця книжка?

Якщо ви можете ствердно відповісти на всі ці питання:

- 1 Чи ви вже володієте якоюсь іншою мовою програмування?
- 2 Чи хочете ви мати покрокове ноу-хау із програмування мовою Python, додати його до свого переліку інструментів і змусити робити нові речі?
- 3 Чи надаєте ви перевагу активній розмові за вечерею перед снухою та нудною технічною лекцією?

Тоді ця книжка для вас.

Кому, ймовірно, краще триматися якнайдалі від цієї книжки?

Якщо ви можете відповісти «так» на будь-яке із цих питань:

- 1 Можливо, ви вже знаєте більшість того, що необхідно для програмування за допомогою Python?
- 2 Ви досвідчений програміст і шукаєте довідник із Python, що охоплює всі подробиці і наводить найдрібніші деталі?
- 3 Ви радше дозволите видерти ваші нігті 15 мавпам, аніж спробуєте вивчити щось нове? І заразом вважаєте, що книжка про Python має охоплювати геть усе, і якщо читачеві стає нудно аж до сліз, то тим краще?

У такому разі ця книжка **не** для вас.

[Примітка від маркетингового відділу: ця книжка призначена для всіх, хто має кредитну картку... ми також приймаємо чеки.]



*Це НЕ довідник,
і ми припускаємо,
що ви вже
програмували
раніше.*

Ми знаємо, про що ви подумали

«Як ця книжка із програмування мовою Python може бути серйозною?»

«Що це за ілюстрації?»

«Чи можу я насправді оволодіти матеріалом, що подається таким чином?»

І ми знаємо, чим переймається ваш мозок

Ваш мозок жадає новизни. Він завжди шукає, сканує, чекає на щось незвичне. Він побудований так, і саме це допомагає вам залишатися живими.

Нині ви малоймовірно станете перекусом для тигра. Але ваш мозок усе ще пильнує. Ви просто не помічаєте цього.

Отже, що робить ваш мозок зі всіма повсякденними, звичайними, нормальними речами, із якими ви стикаєтесь? Він робить усе можливе, аби вони не заважали йому запам'ятовувати дійсно важливу інформацію. Він не переймається запам'ятовуванням нудних речей; вони ніколи не встигають просочитися крізь фільтр «Це, вочевидь, не важливо».

Як саме ваш мозок дізнається, що є важливим? Припустімо, ви вийшли прогулятися і раптово перед вами з'являється тигр. Що відбувається у вашій голові?

Нейрони напружуються. Емоції зашкалюють. Активізуються хімічні процеси. Ось як ваш мозок працює...

Ваш мозок
вважає ЦЕ
важливим.



Це дуже важливо! Не забувайте цього!

А зараз уявіть, що ви вдома чи в бібліотеці. Це безпечне затишне місце, де немає жодного тигра. Ви щось вивчаєте. Готуетесь до іспиту. Або намагаєтесь засвоїти якусь важку технічну тему, опанувати яку, на думку вашого керівника, можна впродовж тижня, максимум за десять днів.

Однак є проблема. Ваш мозок намагається зробити вам велику послугу. Він намагається переконатися, що ця, вочевидь, не важлива інформація не споживає дефіцитні ресурси. Ресурси, що їх краще витратити на запам'ятовування дійсно важливих речей. На кшталт тигрів. Або безпеки раптової пожежі. Або настанови ніколи більше не кататися на сноуборді у шортах.

Не існує простого способу сказати своєму мозкові: «Агов, мозку, дуже дякую, але, хай якою нудною була б ця книжка, ти ж знаєш, що це дійсно важливо, тому що це циклікала, я дійсно хочу, аби ти читав це ці речі».

Ваш мозок
вважає, що ЦЕ
не варто уваги.



Чудово! Лише
450 нецікавих,
сухих, нудних
сторінок.

Купити книгу на сайті kniga.biz.ua >>>

Ми вважаємо читача «Head First» учнем

Як ми щось дізнаємося? Спочатку потрібно це «щось» зрозуміти, а потім не забути. Заштовхати в голову якнайбільше фактів недостатньо. Згідно з даними новітніх досліджень у галузі когнітивістики, нейробіології та психології навчання, для засвоєння матеріалу потрібно щось більше, ніж простий текст на сторінці. Ми знаємо, як змусити ваш мозок працювати.

Основні принципи серії «Head First»:

Робити все наочним. Графіка запам'ятовується значно краще за звичайний текст і значно підвищує ефективність сприйняття інформації (до 89 % підвищується пригадування та відтворення вивченого). До того ж, матеріал стає зрозумілішим. Якщо розташувати відповідний текст на малюнках або поряд з ними, а не під ними або на сусідній сторінці, учні отримають удвічі більше шансів зрозуміти зміст.

Використовувати розмовний та персоналізований стиль викладу. Нещодавні дослідження показали, що за особистісного розмовного стилю викладу матеріалу зміст формальних лекцій поліпшення результатів на підсумковому тестуванні складало до 40 %. Розповідайте історію замість того, щоби читати лекцію. Не ставтеся до себе занадто серйозно. Що ймовірніше приверне вашу увагу: цікава бесіда за вечерею чи лекція?

Змушувати учня мислити глибше. Поки ви не почнете ворухити звивинами, у вашій голові нічого не відбуватиметься. Читач повинен бути зацікавлений у результаті; він повинен натхненно вирішувати завдання, формувати висновки і генерувати нові знання. А для цього необхідні вправи і каверні питання, у пошуку відповідей на які задіані обидві півкулі мозку й різноманітні почуття.

Привертати увагу читача — і фіксувати її. Ситуація, знайома кожному: «Я дуже хочу вивчити це, але засинаю на першій сторінці». Мозок звертає увагу на цікаве, дивне, привабливе, несподіване. Вивчення складної технічної теми не має бути нудним. Ваш мозок осягне це набагато швидше, якщо це буде цікаво викладено.

Залучати емоції. Відомо, що наша здатність запам'ятовувати значною мірою залежить від емоційного співпереживання. Ми запам'ятовуємо те, що нам не байдуже. Ми запам'ятовуємо, коли щось відчуваємо. Ні, ми не маємо на увазі хвилюючі історії про хлопчика та його собаку. Мова йде про такі емоції, як здивування, цікавість, інтерес і відчуття власної крутизни при розв'язанні задачі, яку оточення вважає надто складною, або коли ви зрозуміли, що розбираєтеся в темі краще за всезнайку Боба з технічного відділу.

Метапізнання: мислення про мислення

Якщо ви дійсно хочете вчитися, і хочете дізнатися швидше і глибше, зверніть увагу на те, як ви звертаєте увагу. Подумайте, як ви мислите. Дізнайтеся, як ви вчитеся.

Більшість із нас не вивчали теорію метапізнання або навчання під час навчання. Ми маємо вчитися, але нас рідко цьому навчають.

Проте, оскільки ви тримаєте в руках цю книжку, то, імовірно, дійсно хочете вивчати мову Python. І ви, мабуть, не хочете витратити на це забагато часу.

Аби отримати максимум користі від цієї книжки, або будь-якої книжки чи досвіду навчання, візьміть на себе відповідальність за роботу власного мозку. Вашого мозку у цьому контенті.

Хитрість полягає в тому, аби змусити ваш мозок сприймати новий матеріал, що ви вивчаєте, як дійсно важливого. Як те, що має вирішальне значення для вашого благополуччя. Як те, що не менш важливе, ніж тигр. Інакше вам доведеться постійно боротися, і ваш мозок зробить усе можливе, аби ухилитися від запам'ятовування нової інформації.

Отже, як переконати ваш мозок сприймати мову Java так само, як і зголоднілого тигра?

Існує спосіб повільний і нудний, а є швидкий та ефективний. Перший ґрунтується на безглуздому затвердженні. Усім відомо, що навіть найнуднішу інформацію можна запам'ятати, якщо повторювати її знов і знов. За достатньої кількості повторень ваш мозок розмірковує: «Здається, ніби це не є суттєвим для нього, але якщо те саме повторюється знов, і знов, і знов... гаразд, хай буде так».

Ефективніший спосіб ґрунтується на *стимульованій активності мозку*, переважно на поєднанні різних видів мозкової діяльності. Доведено, що всі фактори, перелічені на попередній сторінці, допомагають вашому мозку працювати на вас. Наприклад, дослідження засвідчують, що розміщення слів усередині малюнків (а не в підписах чи основному тексті) змушує мозок аналізувати зв'язок між текстом і графікою, а це призводить до активізації значно більшої кількості нейронів. Більше нейронів — вища ймовірність того, що інформація буде визнана важливою і вартою того, аби її запам'ятати.

Розмовний стиль теж важливий: зазвичай люди виявляють більше уваги, коли беруть участь у розмові, оскільки їм доводиться стежити за перебігом бесіди і висловлювати власну думку. Причому мозок абсолютно не обходиться те, що ви «розмовляєте» лише з книжкою! З іншого боку, якщо текст сухий і формальний, то мозок відключає те саме, що ви вивчаєте. Власні лекції в ролі пасивного слухача. Його хилить у сон. Проте малюнки і розмовний стиль — це лише початок.



Купити книгу на сайті kniga.biz.ua >>>

Ось що зробили МИ:

Ми використовували *малюнки*, бо мозок краще пристосований до сприйняття візуальних ефектів, аніж тексту. Із точки зору мозку найпростіший малюнок вартий 1024 слів. А коли текст комбінюється із графікою, ми вбудовуємо текст безпосередньо в малюнки, бо мозок при цьому працює ефективніше.

Ми задіяли *дублювання*: повторюємо те саме кілька разів, застосовуючи різні засоби передачі інформації, звертаємося до різних почуттів — і все це для підвищення ймовірності того, що матеріал буде закодований в кількох ділянках вашого мозку.

Ми використовуємо концепції і малюнки у дещо *несподіваний* спосіб, бо мозок краще сприймає нову інформацію, і використовуємо фотографії, оскільки вони зазвичай мають *емоційний зміст*, бо мозок звертає увагу на біохімію емоцій. Те, що змушує нас *відчувати*, краще запам'ятовується, навіть якщо це невеличкий *жарт, здивування* або *інтерес*.

Ми використовуємо персонафікований *розмовний стиль*, бо мозок краще сприймає інформацію, коли ви берете участь у розмові, а не пасивно слухаєте лекцію. Це відбувається і під час *читання*.

Книжка містить понад 80 *вправ*, бо мозок краще запам'ятовує, коли ви щось *робите*, а не просто читаєте. Ми постаралися зробити їх непростими, але цікавими, бо такому підходу надає перевагу більшість *читачів*.

Ми *поднали кілька стилів навчання*, бо одні читачі надають перевагу покроковим описам, у той час як інші прагнуть спочатку уявити «загальну картину», а третім вистачає фрагментів коду. Проте незалежно від ваших особистих уподобань завжди корисно мати справу із кількома варіантами подання того самого матеріалу.

Ми постаралися задіяти *обидві півкулі вашого мозку*, бо що більше ділянок мозку задіяно, то вища ймовірність засвоєння матеріалу і зосередженості на навчанні. Поки одна половина мозку працює, інша часто має можливість відпочити; це підвищує ефективність навчання протягом тривалого часу.

А ще книжка містить *історії* та *вправи*, що відтворюють *інші точки зору*, оскільки мозок яскравіше засвоює інформацію, коли йому доводиться оцінювати і робити припущення.

У книжці деінде зустрічаються *завдання* і *питання*, на які не завжди можна дати просту відповідь, бо мозок швидше вчиться і запам'ятовує, коли йому доводиться щось *робити*. Судіть самі, неможливо накачати м'язи, *спостерігаючи* за тим, як займаються інші. Однак ми подбали про те, щоби зусилля читачів були спрямовані у правильно-му напрямку. *Вам не доведеться ламати голову* над незрозумілими прикладами або розбиратися у складному, перенасиченому технічним жаргоном або занадто лаконічно-му тексті.

В історіях, прикладах, картинках «оживають» *люди*. Адже і ви людина. І ваш мозок більше уваги приділяє *людям*, а не *речам*.



Виріжте це та прилаштуйте на холодильник.

Ось що ВИ можете зробити, аби змусити свій мозок підкорятися

Отже, ми свою справу зробили. Решта за вами. Ці поради стануть відправною точкою; прислухайтесь до свого мозку й визначте, що вам підходить, а що не підходить. Спробуйте нове.

1 Не поспішайте. Що більше ви зрозумієте, то менше доведеться запам'ятовувати.

Просто *читати* недостатньо. Коли книжка ставить вам запитання, не переходьте одразу до відповіді. Уявіть, що хтось дійсно ставить вам запитання. Що глибше ваш мозок буде думати, то швидше ви зрозумієте і запам'ятаєте матеріал.

2 Виконуйте вправи. Робіть власні нотатки.

Ми включили вправи до книги, але виконувати їх за вас не збираємося. І не просто *розглядайте* вправи. *Беріть олівець* і пишіть. Фізична активність *під час* навчання підвищує його ефективність.

3 Обов'язково читайте рубрику «Жодного безглузлого питання».

Це означає: читайте все. Врізки — це частина основного матеріалу! Не пропускайте їх.

4 Не читайте інші книжки після цієї перед сном. Або нехай ця буде останньою складною річчю на сьогодні.

Частина навчання (особливо перенесення інформації в довгострокову пам'ять) відбувається *після* того, як ви відкладаєте книжку. Ваш мозок не одразу засвоює інформацію. Якщо під час обробки надійде нова інформація, частина того, що ви дізналися раніше, може бути втрачена.

5 Промовляйте прочитане. Уголос.

Мова активізує інші ділянки мозку. Якщо ви намагаетесь щось зрозуміти або краще запам'ятати, промовте вголос. А це краще спробуйте пояснити це комусь іншому. Ви будете швидше зрозуміти це, якщо будете вголос промовляти, відкриєте для себе щось нове.

6 Пийте воду. І якомога більше.

Мозок найкраще працює в умовах високої вологості. Дегідратація (яка може статися ще до того, як ви відчуєте спрагу) знижує когнітивні здатності.

7 Прислухайтесь до свого мозку.

Слідуйте за тим, коли ваш мозок починає втомлюватися. Якщо ви починаєте поверхнево сприймати матеріал або забуваєте щойно прочитане — настав час зробити перерву. Якщо ви цього не зробите, може настати мить, коли ви не зможете пришвидшитися у навчанні, намагаючись засвоїти більше, а напавки — лише зашкодите процесу.

8 Нехай це стане реальністю!

Ваш мозок повинен знати, що матеріал книжки дійсно *важливий*. Переймійтесь перебігом наших історій. Вигадайте власні підписи до фотографій. Покекувати над невідлним жартом *усе одно* краще за відсутність будь-яких почуттів.

9 Пишіть багато коду!

Оволодіти мовою Python можна лише в один спосіб: *писати багато коду на Python*. Саме цим вам і належить займатися. І це те, що ви маєте робити впродовж читання цієї книжки. Програмування — це навичка, і єдиний спосіб розвинути цю навичку — практика. Ми запропонуємо вам чимало практичних завдань: у кожному розділі є вправи, до яких треба знайти рішення. Не пропускайте їх: істотна частина навчання відбувається саме тоді, коли ви виконуєте вправи. Ми наводимо розв'язання до кожного завдання — *не бійтеся зазирнути туди*, якщо застрягли! (Дуже легко застрягти через якусь дрібницю.) Але все ж таки спробуйте розв'язати завдання самостійно і лише тоді звертайтеся до відповіді. І, звісно, перевірте його на практиці, перш ніж продовжите читати.

Купити книгу на сайті kniga.org.ua >>>

Прочитай мене (1)

Це навчальний практикум, посібник, а не довідник. Ми свідомо прибрали все, що може зашкодити оволодінню матеріалом. Приступаючи до навчання вперше, читайте все від самого початку, оскільки ми вже врахували все, що ви могли бачити і знати раніше.

Ця книжка створена для того, щоби ви якнайшвидше оволоділи матеріалом.

Ми навчимо вас лише тому, що необхідно знати. Тому ви не знайдете тут ані довгих переліків технічних подробиць, ані таблиць операторів Python, ані правил послідовності операторів. Ми не охопили *весь усе*, але наполегливо попрацювали, щоби охопити основний матеріал якнайкраще, і тепер ви зможете *швидко* «завантажити» Python у свій мозок і втримати його там. Єдине припущення, яке ми робимо,— це те, що ви вже знаєте принципи кодування якоюсь іншою мовою програмування.

Ця книжка орієнтована на Python 3.

У цій книжці ми використовуємо версію 3 мови програмування Python. Далі ми розглянемо, як отримати і встановити Python 3 (див. Додаток А). У цій книжці **не** використовується Python 2.

Ми одразу починаємо роботу з Python.

Починаючи з Розділу 1 ми навчатимемо вас корисним речам і надалі будемо на них спиратися. Немає сенсу ходити навколо, тому що ми хочемо, аби ви одразу почали продуктивно застосовувати Python.

Практичні завдання НЕ є факультативними — ви повинні виконувати їх.

Вправи та практичні завдання не є додатковими, це частина основного матеріалу книжки. Деякі з них сприяють запам'ятовуванню матеріалу, деякі важливі для розуміння, а деякі допоможуть вам застосовувати те, про що ви дізналися. *Не оминайте вправи!*

Надмірність у книжці є навмисною і важливою.

Важливою відмінною рисою книжки є наше щире прагнення до того, щоби ви *дійсно* засвоїли матеріал. І ми хочемо, щоби ви, прочитавши її, пам'ятали те, про що дізналися. Більшість довідників не ставлять за мету збереження та утримання інформації, але ця книжка призначена для навчання. І ви побачите, що деякі поняття зустрічаються набагато більше одного разу.

Приклади максимально спрощені.

Наші читачі зазначають, що їх розчаровує, коли доводиться продиратися крізь 200 рядків коду, щоби відшукати два рядки, потрібні для розуміння. Більшість прикладів у цій книжці надані в мінімально можливому контексті, тож та частина, що ви намагаєтеся вивчити, буде зрозумілою і простою. Не очікуйте, що всі приклади виявляться надійними або навіть повними — вони написані спеціально для навчання і не завжди є цілком функціональними (хоча ми намагалися максимально це забезпечити).

Прочитай мене (2)

Так, ще трохи...

Це друге видання, і воно НЕ таке саме, як перше.

Це оновлення першого видання Head First Python, що побачило світ наприкінці 2010 року. Незважаючи на те що автор той самий, тепер він став старшим і (сподіваюся) мудрішим, тому було вирішено цілковито переробити матеріал першого видання. Отже, тут усе нове: послідовність викладу відрізняється, зміст оновлено, приклади покращені, історії або зникли зовсім, або змінилися. Ми зберегли обкладинку з незначними виправленнями, бо вирішили не дуже розхитувати човен. Минуло чимало років... і ми сподіваємося, що вам сподобається те, що ми зробили.

Де код?

Ми розмістили приклади коду на вебсайтах, аби ви могли копіювати і вставляти їх за необхідності (хоча ми рекомендуємо вводити код вручну під час освоєння матеріалу). Ви можете знайти потрібний код за такими посиланнями:

<http://bit.ly/head-first-python-2e>

<http://python.itcarlow.ie>