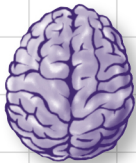


# Head First Python

Легкий для сприйняття довідник

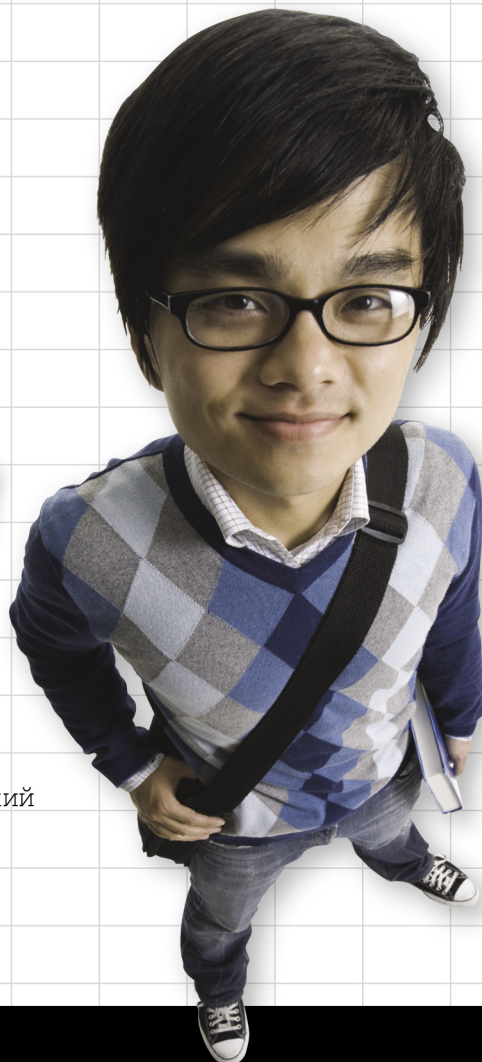


Завантажте  
найважливіші концепції  
Python просто  
у свій мозок

Не зависайте —  
натомість  
застосуйте  
DB-API



Створіть сучасний  
застосунок  
за допомогою  
Flask



Моделюйте дані  
у вигляді списків,  
кортежів, множин  
та словників

Об'єкти?  
Декоратори?  
Генератори?  
Усі вони тут



Поділіться кодом  
за допомогою модулів

Пол Беррі

Пол Беррі

Head First

# Python

Друге видання

Легкий для сприйняття довідник

ВИДАВНИЧИЙ ДІМ  
Ф А Б У Л А  
#PRO

Пол Беррі

## HEAD FIRST PYTHON, друге видання

Видавничий дім «Фабула»

2023

Оригінальна назва твору: HEAD FIRST PYTHON, SECOND EDITION

© 2016 Paul Barry

© Г. Якубовська, пер. з англ., 2021

© ВД «Фабула», 2023

ISBN 978-617-522-129-7 (epub)

*Усі права збережено. Жодна частина цієї книжки не може бути відтворена в будь-якій формі без письмового дозволу власників авторських прав.*

© 2021 Publishing House FABULA LLC

Authorized Ukrainian translation of the English edition of Head First Python, 2nd Edition

ISBN 9781491919538 © 2016 Paul Barry

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Даний переклад публікується та продається з дозволу O'Reilly Media, Inc, які володіють усіма правами або контролюють всі права на публікацію та продаж того ж самого перекладу.

Електронна версія створена за виданням:

### Беррі Пол

B25 Head First Python, друге видання / пер. з англ. Г. Якубовська. — Харків : ВД «Фабула», 2021. — 624 с.

ISBN 978-617-522-019-1

Мова Python сьогодні відома у всьому світі. Вона задіяна в пошукових системах Google, на ній майже цілком написані відеохостинг Youtube і соціальна мережа Reddit, активно використовують її Facebook та Instagram. Агентство національної безпеки США з її допомогою аналізує розвіддані, а NASA розв'язує найскладніші наукові проблеми. Інакше кажучи, Python під силу не тільки обчислювальні задачі, а й тестування, адміністрування і розробка програмних продуктів.

Дійсно хочете швидко оволодіти мовою Python, не докладаючи зайвих зусиль? У цьому допоможе чудовий посібник Пола Беррі *Head First Python*, створений на засадах останніх досягнень когнітивної науки. Ваш мозок буде змушений постійно працювати, а не губитися в нетрях занудливих текстів, що навіюють сон. Ви дізнаєтеся, як створити власний застосунок, як керувати базами даних, обробляти виключення і здійснювати первинну обробку даних. Якщо вас цікавить те, що можна робити з контекстними менеджерами, декораторами і генераторами, то ви точно потрапили за адресою. Захоплива, але не легковажна, інформативна, позбавлена зайвого пафосу книжка Пола Беррі містить більшу частину того, що вам доведеться використовувати чи не щодня у своїй подальшій роботі. Єдине застереження — бажано, щоби майбутній читач хоча би на початковому рівні володів однією з поширених мов програмування.

**УДК 004.4:004.02**

*Шановний читачу!*

*Спасибі, що придбали цю книгу.*

Нагадуємо, що вона є об'єктом Закону України «Про авторське і суміжні право», порушення якого карається за статтею 176 Кримінального кодексу України «Порушення авторського права і суміжних прав» штрафом від ста до чотирьохсот неоподатковуваних мінімумів доходів громадян або виправними роботами на строк до двох років, з конфіскацією та знищенням всіх примірників творів, матеріальних носіїв комп'ютерних програм, баз даних, виконань, фонограм, програм мовлення та обладнання і матеріалів, призначених для їх виготовлення і відтворення. Повторне порушення карається штрафом від тисячі до двох тисяч неоподатковуваних мінімумів доходів громадян або виправними роботами на строк до двох років, або позбавленням волі на той самий строк, з конфіскацією та знищенням всіх примірників, матеріальних носіїв комп'ютерних програм, баз даних, виконань, фонограм, програм мовлення, аудіо- і відеокасет, дискет, інших носіїв інформації, обладнання та матеріалів, призначених для їх виготовлення і відтворення. Кримінальне переслідування також відбувається згідно з відповідними законами країн, де зафіксовано незаконне відтворення (поширення) творів.

Книга містить криптографічний захист, що дозволяє визначити, хто є джерелом незаконного розповсюдження (відтворення) творів.

Щиро сподіваємося, що Ви з повагою поставитеся до інтелектуальної праці інших і ще раз Вам вдячні!

ВИДАВНИЧИЙ ДІМ



# ЩО КАЖУТЬ ПРО ДРУГЕ ВИДАННЯ «HEAD FIRST PYTHON»

Книжка з мови *Python* має бути такою ж захопливою, як і сама мова. Із *Head First Python* майстер-викладач Пол Беррі пропонує стислий, дотепний, цікавий посібник, що дозволить вам добре підготуватися до створення реального коду мовою *Python*.

*Ерік Фрімен, учений-комп'ютерник, педагог з технологій,  
колишній технічний директор «Disney Online»*

*Head First Python* — відмінний вступ як у мову, так і в практичне застосування *Python* у реальних умовах. Він містить купу практичних порад стосовно кодування для мережі та баз даних, і не цурається таких складних речей, як колекції та незмінність. Якщо ви шукаєте чудовий вступ до *Python*, це саме місце для початку. Якщо ви шукаєте найкращий шлях до *Python*, то це той пункт, із якого варто почати.

*Девід Гріффітс, письменник, agile-коуч*

Це видання *Head First Python*, що зазнало істотних змін та оновлень, обов'язково стане найулюбленішим серед швидко зростаючого розмаїття посібників із мови *Python*. Контент побудовано так, аби забезпечити відчутний вплив на читача і захопити його увагу з метою підвищення ефективності засвоєння матеріалу. Усі необхідні теми висвітлюються вкрай чітко, а розважальна частина робить цю книжку вельми приємною і легкою для читання.

*Калєб Хаттінг, автор книжок «20 Python Libraries You Aren't Using (But Should)»  
і «Learning Cython»*

Ось зрозумілий і чіткий засіб долучитися до пулу *Python*. Без жодних операцій на черевній порожнині ви заглибитесь далі, ніж очікували!

*Білл Любановіч, автор книжки «Introducing Python»*

## Вігзуки на перше видання посібника

*Head First Python* — це чудовий вступ не просто в мову *Python*, а в той *Python*, що його застосовують у реальному світі. Книжка виходить за рамки синтаксису з метою навчити вас створювати застосунки для телефонів на *Android*, для *Google App Engine* та багато інших.

*Девід Гріффітс, письменник, agile-коуч*

Там, де інші автори починають із теорії і згодом переходять до прикладів, *Head First Python* одразу розпочинає з коду та пояснює теорію під час подальшого читання. Це значно ефективніше середовище навчання, бо воно захоплює увагу читача від самого початку. Справжнє задоволення — читати цю книжку. Вона захоплива, але не легковажна, інформативна, позбавлена пафосу. Широкий спектр прикладів та пояснень охоплює більшість того, що вам доведеться застосовувати у своїй роботі щодня. Я рекомендую цю книжку кожному, хто починає працювати з *Python*.

*Джеремі Джонс, співавтор книжки  
«Python for Unix and Linux System Administration»*

# Head First Python

Друге видання

Хіба не було би чудово,  
якби існувала така книжка  
з мови Python, що змусила би  
тебе зависати перед комп'юте-  
ром, працюючи над написанням  
коду? Хоча, здається, це чисті-  
сінька фантазія...



Пол Беррі

**O'REILLY®**

Пекін • Бостон • Фарнем • Себастопол (Каліфорнія) • Токіо

*Я присвячую цю книжку  
безкорисливим членам спільноти Python,  
які допомагають цій мові рухатися в ногу із часом.*

*І всім тим, хто зробив вивчення Python  
і пов'язаних із ним технологій настільки складними,  
що виникла необхідність створити цю книжку,  
аби впоратися із цим.*

## Про автора Head First Python

Під час прогулянки Пол зупиняється, аби обговорити правильну вимову слова «tuple» (тобто «кортеж») зі своєю багатостраждальною 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 (@barrypj); 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. Застосовуємо 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 <sup>3/4</sup> . 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 висвітили. Завжди є чому повчитися .....	577
D: ТОП-10 проєктів, що ми їх не розглянули. Ще більше інструментів, бібліотек і модулів .....	589
E: Долучайтеся! 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 переконати 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
Прочитай мене (1) .....	34
Подяки .....	37

## 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-застосунку. ....	76
Відступи вас драгують? ....	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 роззирнутися — *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
Застосовуйте 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
Додання 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	

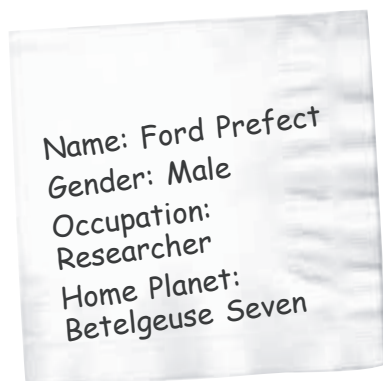
## 3

## 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 записих 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
Підґрунтя кортежів .....	170
Комбінування вбудованих структур даних .....	173
Доступ до даних, що зберігаються у складних структурах ...	179
Код із Розділу 3 .....	181



# 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 PEP 8? .....	229
Розгляд повідомлень про помилки.....	230
Код із Розділу 4.....	232



Модуль

## 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 запиту 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 дізнаєтеся 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 вебзастосу .....	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'}

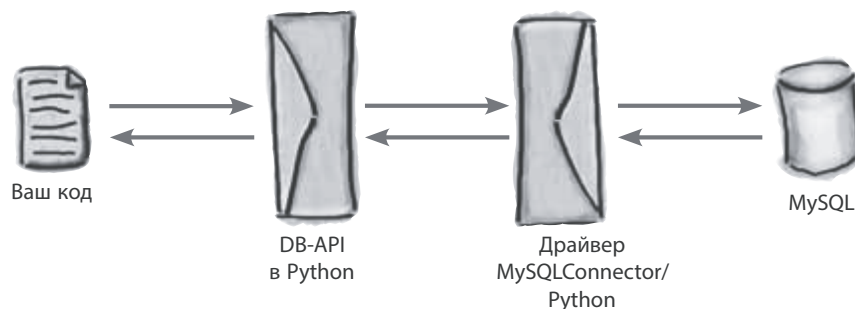
## 7

## Застосування бази даних

## Застосовуємо 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 вебзастосунку.....	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





# 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 подробицях, 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
Додання 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
  
```

Ln: 2 Col: 0

# 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 коді Python. У цьому розділі ми об'єднаємо обидві методики 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 our log DB
$ mysql -u vsearch -p vsearchlogDB
Enter password:
Welcome to MySQL monitor...

mysql> select * from log;
+----+-----+-----+-----+-----+-----+-----+
| id | ts          | phrase                | letters | ip          | browser_string | results                |
+----+-----+-----+-----+-----+-----+-----+
| 1  | 2016-03-09 13:40:46 | life, the uni ... ything | aeiou   | 127.0.0.1   | firefox        | {'l', 'e', 'i', 'a'} |
| 2  | 2016-03-09 13:42:07 | hitch-hiker           | aeiou   | 127.0.0.1   | safari         | {'i', 'e'}           |
| 3  | 2016-03-09 13:42:15 | galaxy                | xyz     | 127.0.0.1   | chrome         | {'y', 'x'}           |
| 4  | 2016-03-09 13:43:07 | hitch-hiker           | xyz     | 127.0.0.1   | firefox        | set()                 |
+----+-----+-----+-----+-----+-----+-----+
4 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 застосовувати **декоратори** функцій — технологію, за допомогою якої можна додавати 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 функції . . . . .	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/paul/Desktop/_NewBook/ch10/checker.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
  
```

Ln: 13 Col: 0

## 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 кращому 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 вебзастосунку.....	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 користувацького винятку.....	482
Що ще може піти не так із «DBcm»?.....	486
Обробка «SQLException» відрізняється.....	489
Викликаємо «SQLException».....	491
Швидкий огляд: додаємо 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
...

```

Кінець безкоштовного уривку. Щоби читати далі,  
придбайте, будь ласка, повну версію книги.