(/)

Topics

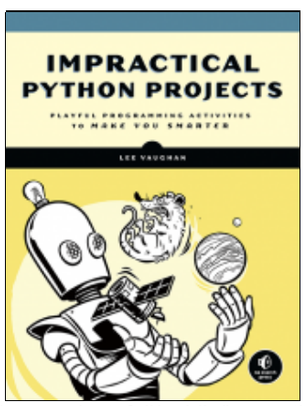
Free ebook edition with every print book purchased from nostarch.com! (/about_ebooks.htm)

(/cart)Shopping cart

0 Items **Total:** $0.00

User login

Log in (/user) Create account (/user/register)



(https://nostarch.com/sites/default/files/styles/uc_product_full/public/impracticalpython_cover03_front.png?itok=zz114Idw)

Impractical Python Projects

Playful Programming Activities to Make You Smarter
by Lee Vaughan
November 2018, 424 pp.
ISBN-13: 9781593278908

- (•) Print Book and FREE Ebook, $29.95
- ( ) Ebook (PDF, Mobi, and ePub), $23.95

✚ Add to cart

Contents | Reviews | Updates

"Python is a programming language, but it is also fun to play with. This book recognises that."
—Geek Tech Stuff (https://geektechstuff.com/2018/11/26/impractical-python-projects-book-review-python/)

"Rather than being an introductory text, Vaughan's book pushes you in interesting directions for solving a diverse set of problems. Most of these "impractical" projects, while themselves being not so useless after all, will have parallels to real life projects." **—Greg Laden, Greg Laden's Blog (http://gregladen.com/blog/2019/01/26/impractical-python-programming-for-fun/)**

Download Chapter 3: Solving Anagrams (/download/ImpracticalPython_Ch3.pdf)
Download the code files and solutions to the book's programs at its GitHub page (https://github.com/rlvaugh/Impractical_Python_Projects)

Impractical Python Projects picks up where the complete beginner books leave off, expanding on existing concepts and introducing new tools that you'll use every day. And to keep things interesting, each project includes a zany twist featuring historical incidents, pop culture references, and literary allusions.

You'll flex your problem-solving skills and employ Python's many useful libraries to do things like:

- ɔ James Bond crack a high-tech safe with a hill-climbing algorithm
- ɹe haiku poems using Markov Chain Analysis
- Ɔse genetic algorithms to breed a race of gigantic rats

- Crack the world's most successful military cipher using cryptanalysis
- Foil corporate security with invisible electronic ink
- Derive the anagram, "I am Lord Voldemort" using linguistical sieves
- Plan your parents' secure retirement with Monte Carlo simulation
- Save the sorceress Zatanna from a stabby death using palingrams
- Model the Milky Way and calculate our odds of detecting alien civilizations
- Help the world's smartest woman win the Monty Hall problem argument
- Reveal Jupiter's Great Red Spot using optical stacking
- Save the head of Mary, Queen of Scots with steganography

Simulate volcanoes, map Mars, and more, all while gaining valuable experience using free modules like Tkinter, matplotlib, Cprofile, Pylint, Pygame, Pillow, and Python-Docx.

Whether you're looking to pick up some new Python skills or just need a pick-me-up, you'll find endless educational, geeky fun with *Impractical Python Projects*.

Author Bio

Lee Vaughan is a geologist with over 30 years' experience in the petroleum industry. As the Senior Technical Professional for Geological Modeling at a major international oil company, he was involved in the construction and review of computer models, the development, testing, and commercialization of software, and the training of geoscientists and engineers. An advocate for nonprogrammers who must use programming in their careers, he wrote *Impractical Python Projects* to help self-learners hone their skills with the Python language.

Table of contents

Reviews

"Python is programming language, but it is also fun to play with and this book recognises that."
—Geek Tech Stuff (https://geektechstuff.com/2018/11/26/impractical-python-projects-book-review-python/)

"Rather than being an introductory text, Vaughan's book pushes you in interesting directions for solving a diverse set of problems. Most of these "impractical" projects, while themselves being not so useless after all, will have parallels to real life projects." **—Greg Laden, Greg Laden's Blog (http://gregladen.com/blog/2019/01/26/impractical-python-programming-for-fun/)**

"The book is not a Python tutorial or guide. Instead, it presents stimulating coding projects for non-programmers who want to use Python for doing experiments, test theories, or simulate natural phenomena."**—Paolo Amoroso, Moonshots Beyond the Cloud Blog (https://blog.paoloamoroso.com/2019/09/reading-impractical-python-projects.html?fbclid=IwAR374Kwd21DXhShowc-Jjj8h0W13LsVhF6nm8Q0nDotiZaE0ivz8a9GPvsk)**

"A must have second book for every developer that ever wants to learn Python as a language."
— Ian Mizer, Atlanta Python Programmers Group (https://pyatl.dev/blog/review-impractical-python-projects/18/)

Updates

Page 33: In the list of palingrams, change "dairy raids" to "welsh slew"

P In the first 2 lines of the 2nd code snippet, change:

```
  range(len(list_of_lists)):
st_of_lists[i])
```

To:

for nested_list in list_of_lists:
print(nested_list)

Page 79: code output at bottom of page is based on Figure 4-3, not Figure 4-2.

Page 85: In Listing 4-9, the code at wingding 5 and the following line that read:

```
    row1 = (message[:row_1_len])
    row2 = (message[row_1_len:])
```

Should instead read:

```
    row1 = (message[:row_1_len]).lower()
    row2 = (message[row_1_len:]).lower()
```

Also in Listing 4-9, the code immediately below wingding 8 that reads:

```
        plaintext.append(r1.lower())
        plaintext.append(r2.lower())
```

Should instead read:

```
        plaintext.append(r1)
        plaintext.append(r2)
```

Page 100: The output printout near the bottom of page that reads:

```
Panel at east end of chapel slides
```

Should instead read:

```
Panelateastendofchapelslides
```

Page 103: The cipher example that reads:
T **h** e cold t **e** a didn't p **l** ease the o **l** d finicky w **o** man.
Should instead read:
So, t **h** e cold t **e** a didn't p **l** ease the o **l** d finicky w **o** man.

Page 141: Wingding 7 and the preceding line in Listing 7-10 that read:

```
lock_wheel = int(randrange(0, len(combo)))
next_try[lock_wheel] = randint(0, len(combo)-1)
```

Should instead read:

```
lock_wheel = randrange(0, len(combo))
next_try[lock_wheel] = randint(0, 9)
```

Page 205: The line and following equation that read:
The transformation to generate points over a *unit* disc is as follows:
$x = \sqrt{r} * cos$
Should instead read as:
The transformation to generate points *evenly* over a *unit* disc is:
$x = \sqrt{r} * cos\theta$

And the line that reads:
The equations yield (*x, y*) values between 0 and 1.
Should instead read as:
The equations yield (*x, y*) values between -1 and 1.

Page 218: Indentation for the listing on the page should be as follows:

```
>>> from random import randint
>>> trials = 100000
>>> success = 0
>>> for trial in range(trials):
            faces = set()
            for rolls in range(6):
                roll = randint(1, 6)
                faces.add(roll)
            if len(faces) == 6:
                success += 1
>>> print("probability of success = {}".format(success/trials))
probability of success = 0.01528
```

Page 250: Listing 12-1 should read as:

```
import sys
import random
i      matplotlib.pyplot as plt

c       _to_list(file_name):
    """Open a file of data in percent, convert to decimal & return a list."""
```

```
    with open(file_name) as in_file:
        lines = [float(line.strip()) for line in in_file]
        decimal = [round(line / 100, 5) for line in lines]
        return decimal

def default_input(prompt, default=None):
    """Allow use of default values in input"""
    prompt = '{} [{}]: '.format(prompt, default)
    response = input(prompt)
    if not response and default:
        return default
    else:
        return response
```

Page 252: The last line on page that reads:
Set the default to `'sbc_blend'`, since this is theoretically the most stable mix of the four choices.
Should instead read as:
Set the default to `'bonds'`, in order to see how this supposedly “safe” choice performs.

Page 259: The first line in last paragraph that reads:
Let’s work an example that assumes a starting value of $2,000,000, a “safe and secure” bond portfolio, a 4 percent withdrawal rate (equal to $80,000 per year), a 30-year retirement, and 50,000 cases.

Should instead read as:
Let’s work an example that assumes a starting value of $2,000,000, a “safe and secure” bond portfolio, a 4 percent withdrawal rate (equal to $80,000 per year), a 29-30-31 retirement range, and 50,000 cases.

Page 261: The last two lines in Listing 12-9 that read:

```
    investments -= withdraw_infl_adj
          investments = int(investments * (1 + i))
```

Should instead read as:

```
            investments -= withdraw_infl_adj
            investments = int(investments * (1 + i))
```

Page 305: In the last paragraph, `transform_rotate()` should be `transform.rotate()`

Page 356: Between wingdings 5 and 6, Listing 16-2 should read as:

```
    # check for missing digits
    keys = [str(digit) for digit in range(1, 10)]
    for key in keys:
        if key not in first_digits:
            first_digits[key] = 0
```

Page 368: The code for Dictionary Cleanup should read as:

```
"""Remove single-letter words from list if not 'a' or 'i'."""
word_list = ['a', 'nurses', 'i', 'stack', 'b', 'c', 'cat']
word_list_clean = []

permissible = ('a', 'i')

# remove single-letter words if not "a" or "I"
for word in word_list:
    if len(word) > 1:
        word_list_clean.append(word)
    elif len(word) == 1 and word in permissible:
        word_list_clean.append(word)
    else:
        continue

print("{}".
```

Page 369: In the "Finding Digrams" code listing, change line 17 which currently reads:

```
      digrams, sep='\n')

      sorted(digrams), sep='\n')
```

And change line 28 which currently reads:

```
for k in mapped:
```

To:

```
for k in sorted(mapped):
```



(http
/sha
/sha
%3A
%2F

(http
/inte
/twe
%20
%20
url=h
%3A
%2F
via=h

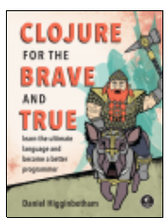
Navigation

My account (/user)

Want sweet deals?
Sign up for our newsletter. (/mailchimp/subscribe)

You might also like...

(/ghpython.htm)



(/clojure)

(/CodingWorkbook)

(/tlcl2)

(/ev3lab) **(/make-python-talk)**