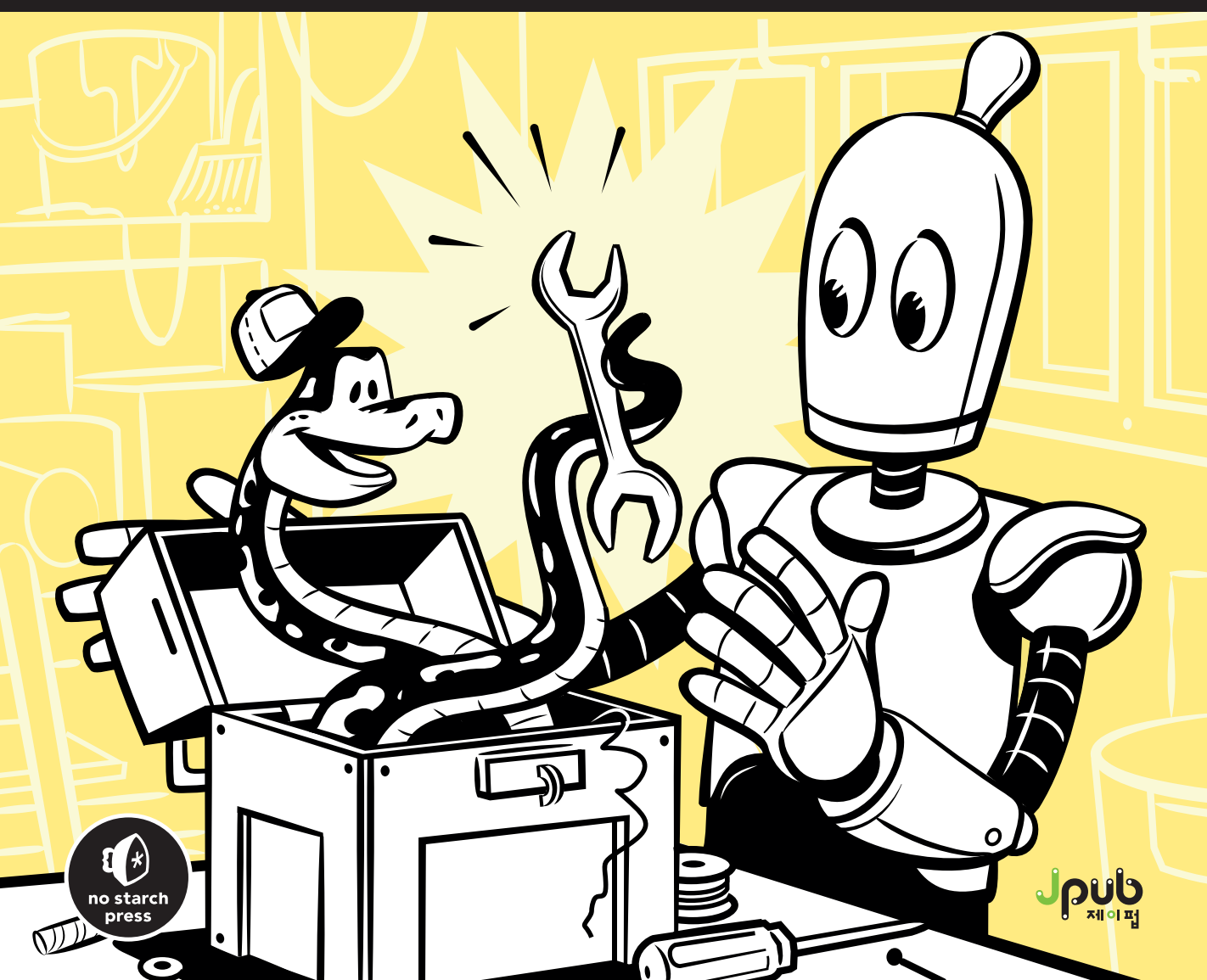
알 스웨이가트의 파이썬 프로젝트

81개의 실습 예제로 시작하는 파이썬 프로그래밍 입문

알 스웨이가트 지음 / 황반석 옮김



no starch press

Jpub 제이펍

The Big Book of Small Python Projects:
81 Easy Practice Programs

알 스웨이가트의 파이썬 프로젝트

1쇄 발행 2022년 2월 24일

지은이 알 스웨이가트
옮긴이 황반석
펴낸이 장성두
펴낸곳 주식회사 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 회동길 159 3층 / **전화** 070-8201-9010 / **팩스** 02-6280-0405
홈페이지 www.jpub.kr / **원고투고** submit@jpub.kr / **독자문의** help@jpub.kr / **교재문의** textbook@jpub.kr

편집부 김정준, 이민숙, 최병찬, 이주원, 송영화
소통기획부 이상복, 송찬수, 배인혜 / **소통지원부** 민지환, 김수연 / **총무부** 김유미

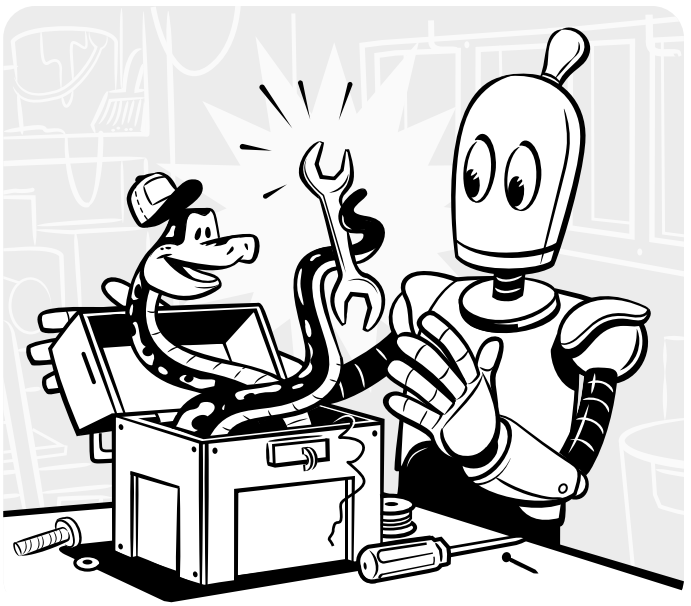
진행 및 교정 · 교열 이주원 / **내지디자인** 이민숙 / **내지편집** 북아이
용지 에스에이치페이퍼 / **인쇄** 한승문화사 / **제본** 일진제책사

ISBN 979-11-91600-66-7 (93000)
값 28,000원

알 스웨이가트의 파이썬 프로젝트

81개의 실습 예제로 시작하는 파이썬 프로그래밍 입문

알 스웨이가트 지음 / **황반석** 옮김

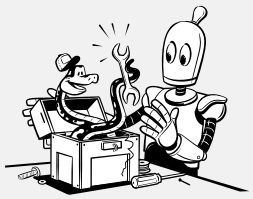


no starch press

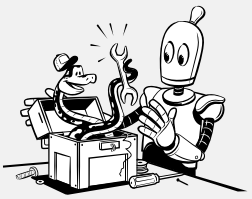
Jpub 제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저자 및 역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 등장하는 각 회사명, 제품명은 일반적으로 각 회사의 등록 상표 또는 상표입니다.
  본문 중에는 ™, ©, ® 마크 등이 표시되어 있지 않습니다.
- 이 책에서 소개한 URL 등은 시간이 지나면 변경될 수 있습니다.
- 이 책은 파이썬 3.10.0 버전을 기준으로 작성되었으며, 독자의 학습 시점이나 환경에 따라 책의 내용과 다를 수 있습니다.
- 깃헙(https://github.com/Jpub/PyProject)을 통해 제공하는 예제 코드는
  책의 코드 구성과 차이가 나지 않도록 가급적 다운을 받아 활용하시기 바랍니다.
- 책 내용과 관련된 문의사항은 옮긴이나 출판사로 연락해 주시기 바랍니다.
  - 옮긴이: naya.peter@gmail.com
  - 출판사: help@jpub.kr



차 례



옮긴이 머리말

"당신은 페이스북, 트위터, 인스타그램과 같은 SNS 애플리케이션을 만들 수 있나요? 혹은 메신저 또는 쇼핑몰 애플리케이션을 만들 수 있나요?"

이런 질문에 대해 자신 있게 '그렇다'고 답하지 못하는 개발자(또는 프로그래밍을 공부하는 분)는 어떤 이유 때문일까요? 여러 이유가 있겠지만, 적어도 프로그래밍 언어(문법)를 몰라서는 아닐 것입니다. 자신만의 멋진 프로그램을 만들고 싶어서 프로그래밍 언어와 문법 공부를 열심히 해서 끝냈는데, 본격적으로 무언가를 만들려 하면 막연해집니다. 아마 지금 막 프로그래밍에 입문하신 분이라면, 그리고 서툴렀던 초년 시절을 떠올리는 경험자라면, 어떤 느낌인지 이해하실 것입니다. 막연한 느낌이 드는 것은 열심히 공부한 프로그래밍 언어를 어떻게 사용해야 하는지 아직 모르기 때문입니다. 그래서 삽질을 자처하는 심정으로 여러 가지 개인 프로젝트를 만들어 보는 사람도 있고, 기술 세미나와 모임에 참가하는 사람도 있습니다. 그렇게 각자 자신만의 방법으로 프로그래밍 경험을 늘려갑니다. 하지만 안타깝게도 우리에겐 물리적인 한계가 있습니다. 몸은 하나뿐이고, 하루는 24시간이라는 시간제한이 있기 때문에 모든 것을 다 경험할 수 없습니다. 기술 블로그, 튜토리얼, 유튜브를 통해 접하는 방법도 있지만, 내가 원하는 것을 항상 찾을 수 있다는 보장도 없고, 더욱 난감한 것은 내가 정말 무엇을 원하는지 잘 모를 때도 있다는 것이죠.

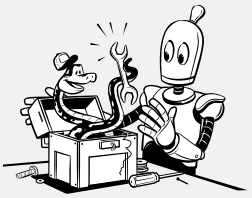
다시 돌아가서, 어떤 프로그램을 만들 수 있고 없고의 핵심은 '그 프로그램을 실행하는 내부 로직 또는 메커니즘을 알고 구현할 수 있는가?'입니다. 예를 들어, 수능에 출제된 수학 문제를 맞힐 수 없다는 것은 그 문제를 풀기 위한 수학 공식들을 어떻게 적용할지 모른다는 의미일 수 있습니다. 숫자를 읽지 못해서, 덧셈 뺄셈을 못 해서, 혹은 수학 공식을 외우지 못해서라기보다, 알고 있는 것들을 어떻게 적용해야 하는지 모르기 때문일 것입니다. 저는 프로그래밍도 마찬가지라고 생각합니다. 결국 다양한 문제를 접하고 그 해법에 대해 많은 경험을 쌓다 보면 비로소 문제(프로그램)가 보이는 것입니다.

이 책은 작지만 강력한 언어인 파이썬을 가지고 해결할 총 81개의 프로그래밍 문제와 코드를 담고 있습니다. 개인적으로 특히 맘에 드는 점은 프로그램 로직에 집중한다는 점이었습니다. 프로그램 실행 화면과 결과는 `print`문을 사용하여 콘솔 창에 표시하며, 사용자 인터페이스는 키보드를 사용합니다. 화면을 예쁘게(?) 꾸며야 할 때는 유니코드를 사용하여 모양을 만듭니다. 매우 클래식하다고(혹은 예스럽다고) 느껴질지 모르지만, 코드와 로직에 집중할 수 있어서 좋았습니다. 가끔 이런 종류의 책들은 시작과 다르게 '예쁜 UI 꾸미기'로 빠지곤 하거든요. 물론, 예쁜 UI가 중요하지 않다는 것은 절대 아닙니다. 그러나 프로그래밍과 로직을 설명한다는 책이 UI 꾸미기로 빠지면 책의 처음 의도가 변질된다고 생각합니다.

저자도 책에 설명했지만, 이 책은 몇 가지 원칙을 지키며 쓴 책입니다. 그 원칙 중에 언급하고 싶은 부분은 '심플함'입니다. 이 말에는 모두가 알다시피 '간결하다', '단순하다'는 뜻입니다. 소스 코드는 초보자도 쉽게 이해할 수 있으며, 경험이 있는 개발자라면 눈으로 코드를 따라가도 이해할 수 있을 것입니다. 코드가 직관적이라는 의미는 쉽게 이해할 수 있다는 의미이지만, 반대로 얘기하면 세련된 코드가 아니며, 고성능의 알고리즘이 적용되지도 않았고, 최적화도 되어 있지 않다는 뜻이기도 합니다. 다시 말해서, 이 책의 코드보다 더 좋은(?) 코드 또는 구조가 존재할 수 있다는 의미입니다(물론, 코드의 우열을 가릴 수 있는지 모르겠지만 말이죠). 경험이 조금 있는 독자라면 말 그대로 며칠 만에 다 읽고 '뭐, 쉽네~'라고 할지 모르겠지만, 역자로서는 직접 더 좋은 코드를 꼭 만들어 보기를 권합니다. '코드 리팩토링code refactoring'이란 거창한 용어까지는 아니더라도, '나라면 이 코드를 이렇게 고치겠어!' 정도의 마음으로 해본다면 본인에게 큰 도움이 될 것이라 믿습니다. 실제로 제 경험상, 회사에 입사하면 기존의 프로젝트 코드를 받아서 읽고 추가/변경/개선 작업을 하게 되는 경우가 훨씬 많았습니다.

저에게 번역 작업은 언제나 즐겁습니다만, 이번 책은 더욱더 그러했습니다. 번역하다 말고 문제 풀이에 몰두하기를 몇 번이고 반복했죠. 그만큼 재미있었고 개발자로서 풀어내고 싶은 승부욕(?) 같은 게 올라오는 문제들이었습니다. 번역 작업은 개인적으로 너무 즐거운 시간이지만, 아빠랑 놀 생각에 주말만 기다리던 단비에겐 참으로 미안한 기간입니다. 이제 저는 밀린 숙제를 하러 가봐야겠네요.

옮긴이 **황반석**



베타리더 후기

김진영(야놀자)

정성스러운 서문을 따라 준비를 마치면, 파이썬이라는 언어의 기본 컨셉 및 가벼운 문법 정도만 알고 있어도 본 책을 학습하는 데 크게 어려움은 없을 것입니다. 책의 전개는 프로젝트 하나하나를 소개하면서 진행되는데, 전체적으로 그 내용이 길지 않고 짧게 진행됩니다. 아울러 프로젝트를 통해 파이썬의 모듈과 기능을 자연스럽게 학습할 수 있는데, 스도쿠 같은 미니 게임을 즐기듯 부담 없이 재미있고 즐겁게 읽을 수 있는 책이었습니다. 문법을 중심으로 한 학습의 전개가 아닌, 실제로 만들 수 있는 프로젝트를 단위로 해서 책 내용이 진행되는 부분이 특히 재미있습니다. 프로젝트를 통해 어떤 모듈과 기능을 학습할 수 있는지 정리해서 말해 주는 점과 서문이 무척 정성스러웠다는 점이 인상 깊습니다. 코드에 주석이 무척 상세한 점도 좋았고요.

사지원(뉴빌리티)

파이썬의 기본 문법을 익혔다면 다음은 어떤 것을 해야 할지 다소 막막한 상황이 있을 수 있는데, 프로그래밍 실력 향상은 최대한 많은 코드를 직접 작성하며 문제를 풀어보는 것이 중요합니다. 이 책은 파이썬으로 해결할 수 있는 다양한 문제를 소개합니다. 기본적인 프로그래밍 실력을 키우고 싶다면 이 책에서 소개하는 문제를 풀어보는 걸 추천합니다. 재미있는 문제들이 많아 흥미롭습니다.

안선환

막 파이썬 문법을 끝낸 분에게 추천하고 싶습니다. 개념은 알았지만 활용하는 방법을 모를 때 이 책을 만난다면 많은 도움이 될 것이라고 생각합니다. 간단한 예제들을 통해서 파이썬의 여러 개념을 활용하는 법을 재미있게 익히도록 되어 있습니다. 다만, 순서가 수준별로 되어 있지 않아 초

보자분들은 책장을 빠르게 넘기며 짧은 코드부터 연습하시기를 추천합니다. 예제들도 오류 없이 모두 잘 실행되었습니다.

양성모(현대오토에버)

이 책에서는 각 단원별로 200줄 내외의 완성된 파이썬 프로그램을 작성합니다. 간단한 프로그램도 있고, 많이 고민해 보아야 하는 프로그램도 있습니다. 컴퓨터 프로그래밍을 처음 접하는 분만 아니라 파이썬 언어를 새로 익히는 개발자 분들에게도 언어의 사상을 이해하는 데 많은 도움이 될 것 같습니다. 저자의 말처럼 초보자를 위한 책이라고 하기에는 그렇게 친절한 책은 아닌 것 같습니다. 하지만 어느 정도 언어를 익히고 개인 프로젝트를 진행해 보고 싶어하거나, 프로그래밍 책의 예제 이상의 것을 만들어보고자 하는 입문자들에게는 매우 큰 도움이 될 것 같습니다.

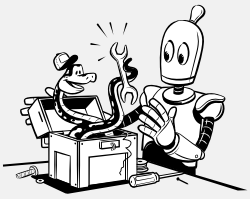
이요셉(지나가던 IT인)

요즘에는 많이 희미해진 감이 있지만, 사실 프로그래밍에서 필요로 하는 기본 능력은 논리력과 사고력입니다. 이 책은 파이썬을 통해 이러한 능력을 기르기 위한 책입니다. 코딩을 즐기는 저자가 신나게 만든 다양한 분야의 81개 연습 문제는 파이썬 문법에 어느 정도 익숙한 사람이라면 누구든지 동아리나 학교, 직장에서 친구와 함께 토론해가며 공부하기에 최적입니다. 이 책의 저자가 정말 코딩을 사랑한다는 느낌을 많이 받았네요. 전체적인 번역 질도 좋습니다.

정태일(지나가던 IT인)

다양한 주제의 파이썬 프로그램을 따라 만들어보면서 기본 파이썬 문법을 익힌 초보자가 원하는 기능과 프로그램을 개발하는 방법을 배울 수 있도록 돕습니다. 차근차근 코드를 따라 작성해 보고 실행한 뒤, 프로그램 살펴보기에서 제시되는 문제들을 고민하고 변경해서 결과를 확인하다 보면 더욱 기초가 탄탄해지는 느낌을 받게 되실 겁니다. 특히, 파이썬 책들 다수가 업무 자동화 프로그램이나 기본 문법 위주의 코드 스니펫 정도의 코드를 많이 다루는데, 이 책은 동작하는 다양한 주제의 프로그램을 가지고 파이썬을 익힐 수 있어서 유용했습니다.

제이펍은 책에 대한 애정과 기술에 대한 열정이 뜨거운 베타리더의 도움으로
출간되는 모든 IT 전문서에 사전 검증을 시행하고 있습니다.



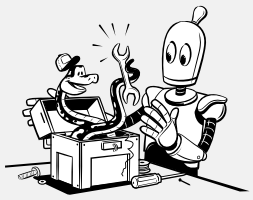
저자 및 기술 검수자 소개

저자에 대하여

알 스웨이가트Al Sweigart는 소프트웨어 개발자이자 작가이며, 파이썬 소프트웨어 재단의 펠로우Fellow다. 이전에는 캘리포니아주 오클랜드에 있는 비디오 게임 박물관인 'The Museum of Art and Digital Entertainment'에서 교육 책임자로 있었다. 그는 《뚝딱뚝딱 파이썬 자동화Automate the Boring Stuff with Python》(인사이트 2021)과 《나만의 Python Game 만들기Invent Your Own Computer Games with Python》(정보문화사, 2014)을 포함하여 여러 프로그래밍 책을 썼다. 그의 책은 그의 웹사이트인 https://inventwithpython.com에서 크리에이티브 커먼즈 라이선스Creative Commons License, CCL에 따라 무료로 제공된다. 그의 고양이Zophie는 김으로 만든 과자를 좋아한다.

기술 검수자에 대하여

사라 쿠친스키Sarah Kuchinsky는 기업 트레이너이자 컨설턴트다. 그녀는 건강 시스템 모델링, 게임 개발, 그리고 업무 자동화를 포함한 여러 애플리케이션에 파이썬을 사용하고 있다. 'North Bay Python' 콘퍼런스의 공동 창업자이자 PyCon US의 튜토리얼 위원장이며, PyLadies Silicon Valley의 수석 주최자다. 그녀는 경영과학공학Management Science & Engineering 학위와 수학 학위를 가지고 있다.



들어가며

그저 `print('Hello, world!')`와 같은 튜토리얼을 따라 할 때는 프로그래밍이 무척 쉬웠다. 아마 여러분은 초보자를 위해 잘 구성된 책이나 온라인 과정을 따라 연습을 했을 것이며, 전문 용어에 대해 웬만큼 이해했을 경우는 고개를 끄덕였을 것이다. 하지만 안락한 둥지를 떠나 혼자만의 힘으로 프로그램을 만들고자 했을 때는 어려움을 느꼈을 것이다. 나만의 파이썬 프로그램을 만들려면 어떻게 시작해야 좋을지 모른 채 비어 있는 에디터 화면만 멍하니 쳐다보는 자신을 발견하지는 않았는가?

튜토리얼을 따라 하는 것은 개념을 배우기에 좋지만, 그것이 프로그램 만드는 방법을 배우는 것과 반드시 일치하지는 않는다는 게 문제다. 이에 대한 일반적인 조언은 오픈 소스 소프트웨어의 소스 코드를 분석해 보거나 자신만의 프로젝트를 해보라는 것이겠지만, 대부분의 오픈 소스 프로젝트는 신규 사용자가 쉽게 접근할 수 있을 만큼 문서화가 잘 되어 있지는 않다. 그렇다면 자신만의 프로젝트를 진행하는 것에 관심이 가겠지만, 아무런 안내나 그 어떤 소프트웨어 구조조차 없는 상태로 홀로 남겨지게 된다.

이 책은 80개가 넘는 게임, 시뮬레이션, 그리고 디지털 아트 프로그램들과 함께, 프로그래밍 개념이 어떻게 적용되는지에 대한 실습 예제를 제공한다. 그것들은 단순한 코드 스니펫code snippet[1]이 아니다. 실행 가능한 전체 소스를 제공하는 파이썬 프로그램이다. 코드를 복사하여 동작 방식에 익숙해지고 여러분 마음대로 변경하여 실습한 후에 실제로 여러분의 프로그램에 그것들을 재현해 볼 수 있을 것이다. 그러다 보면 프로그램에 대한 개념을 제대로 이해하기 시작하고, 더 중요한 것은 프로그램을 어떻게 만들어야 하는지도 알 수 있게 된다는 점이다.

1 옮긴이 프로그램 개발 시 재사용 가능한 작은 단위의 프로그래밍 코드

이 책에 실린 프로그램에 대한 설계 원칙

프로그래밍은 수십억 달러 규모의 기술 회사를 만들고 놀라운 기술적 발전을 이끄는 강력한 방법임이 입증되었다. 자신이 만든 소프트웨어로 높은 목표를 달성하고 싶겠지만, 스스로가 할 수 있는 것보다 훨씬 큰 규모의 소프트웨어를 만들다 보면 미완성된 프로그램과 좌절감만 남게 될 수 있다. 그렇다고 해서 재미있고 창의적인 프로그램을 만들기 위해 여러분이 반드시 컴퓨터 천재가 되어야 할 필요는 없다.

이 책의 파이썬 프로그램은 프로그래밍을 처음 하는 개발자가 소스 코드를 이해할 수 있도록 몇 가지 설계 원칙을 따르고 있다.

- **코드 제한** — 이 책 대부분의 프로그램 코드는 256줄 이하로 제한한다. 이것은 독자가 코드를 조금 더 쉽게 이해할 수 있도록 제한한 것이다. 256은 저자가 임의로 고른 것이기도 하지만, 256은 2^8이며 자고로 2의 거듭제곱은 프로그래머에게 매우 친숙한 수이기도 하다.
- **텍스트 기반** — 텍스트는 그래픽보다 간단하다. 소스 코드와 프로그래밍 결과가 모두 텍스트일 경우, 예를 들어 `print('Thanks for playing!')`이라는 코드와 `Thanks for playing!`이라는 결과를 본다면 원인과 결과를 쉽게 따라갈 수 있을 것이다.
- **설치가 필요 없음** — 각 프로그램은 **tictactoe.py**처럼 파일 확장자가 **.py**인 단일 파이썬 소스 파일에 포함되어 있다. 따라서 어떠한 설치 프로그램도 실행할 필요가 없으며, 이들 프로그램을 온라인에 포스팅하여 다른 사람들에게 공유하기에도 좋을 것이다.
- **다수의 프로그램** — 이 책은 보드게임, 카드 게임, 디지털 아트워크, 시뮬레이션, 숫자 퍼즐, 미로 찾기, 유머 프로그램 등 81개의 프로그램을 담고 있다. 가짓수가 많은 만큼 이 중에 여러분이 좋아할 만한 것도 발견할 수 있을 것이다.
- **심플함** — 이 책의 프로그램들은 초보자도 쉽게 이해할 수 있도록 작성되었다. 저자는 이 책의 프로그램 코드를 작성할 때 정교하고 세련된 코드로 작성할 것인지, 고성능 알고리즘을 사용하여 작성할 것인지, 아니면 단순하고 간단한 코드로 작성할지가 고민될 때마다 항상 마지막을 선택했다.

텍스트 기반 프로그램이 구식으로 보일 수 있겠지만, 이러한 스타일의 프로그래밍은 이미지 다운로드하기와 추가적인 라이브러리 설치하기, 그리고 프로젝트 폴더 관리하기 등의 부가적인 작업 때문에 산만해지는 것을 미연에 방지해 줄 뿐만 아니라 코드에만 집중할 수 있도록 해준다.

누구를 위한 책인가?

이 책은 두 그룹을 대상으로 집필되었다. 첫 번째 그룹은 파이썬과 프로그래밍의 기초를 이미 배웠지만 혼자서 프로그램을 어떻게 작성해야 하는지 여전히 모르는 사람이다. 어쩌면 프로그래밍이 자신과는 맞지 않다고도 생각할 것이다. 이에 해당하는 분은 튜토리얼의 연습 문제는 풀 수 있지만, 완전한 전체 프로그램의 모습은 머릿속에 그려지지 않을 것이다. 처음에는 이 책의 프로그램을 복사해서 사용하다가 나중에 재작성해 본다면, 배웠던 프로그래밍 개념을 다양한 실제 프로그램에 어떻게 조합할지를 깨달을 수 있다.

두 번째 그룹은 프로그래밍은 처음이지만 약간의 모험심과 흥미를 느낀 사람이다. 책을 읽자마자 곧바로 뛰어들어 게임과 시뮬레이터, 그리고 숫자 계산 프로그램 등을 곧바로 만들어 보기를 원할 것이다. 이에 해당하는 분은 코드를 복사하거나 과정 안내에 따라 배우는 것에도 문제가 없다. 혹은 이미 다른 프로그래밍 언어로 프로그램을 만드는 방법을 알고 있지만, 파이썬이 처음일 뿐인 사람일 수도 있다. 이 책이 파이썬 입문 과정에 대한 완벽한 비전을 제시할 수는 없겠지만, 파이썬의 기초에 대한 간략한 소개와 프로그램이 실행될 때 내부에서 진행되는 코드를 디버거를 사용하여 검사하는 방법을 담고 있다.

마지막으로 경험이 많은 프로그래머도 이 책의 프로그램들을 재미있게 즐길 수 있겠지만, 이 책은 초보자를 위해 작성되었음을 알기 바란다.

이 책에 대하여

이 책의 대부분은 주요 프로그램에 할애하고 있지만, 일반 프로그래밍이나 파이썬 정보와 연관된 추가적인 내용도 담고 있다. 다음은 이 책에 포함된 내용이다.

- **프로젝트** — 81개의 프로젝트를 여기에 모두 나열하기에는 너무 많지만, 간략하게 요약하자면 각 프로젝트는 프로젝트 이름과 설명, 프로그램의 실행 결과 샘플, 그리고 소스 코드를 포함한 하나의 장chapter이다. 또한, 프로그램을 여러분에게 맞게 커스터마이징할 수 있도록 몇 가지 제안 사항도 담았다.
- **부록 A: 태그 인덱스** — 프로젝트 태그로 분류한 프로젝트 목록이다.
- **부록 B: 문자 맵** — 하트, 선, 화살표, 그리고 블록과 같이 프로그램에서 출력할 수 있는 기호에 대한 문자 코드 목록이다.

이 책의 프로그램을 통해 학습하는 방법

이 책은 파이썬 및 프로그래밍 개념을 전통적인 튜토리얼 방식으로 가르치지 않는다. 이 책은 체험 학습learn-by-doing 방식을 통해 책의 프로그램을 독자가 직접 입력하고, 가지고 놀다가 디버거로 실행하여 내부 작업을 살펴보도록 했다.

또한 이 책의 핵심은 프로그래밍 구문에 대해 자세하게 설명하는 것이 아니라 카드 게임, 애니메이션, 수학 퍼즐 등의 실제로 동작하는 확실한 프로그램 예제들을 보여 주는 것이다. 따라서 저자는 다음 순서대로 이 책을 활용하기를 권한다.

1. 프로그램 코드를 다운로드하고 실행하여 프로그램이 어떤 작업을 하는지 직접 확인한다.
2. 이 책에 있는 프로그램 코드를 빈 파일에 직접 타이핑하여 입력한다. 복사하여 붙여넣기는 하지 말자!
3. 프로그램을 다시 실행해 보고, 문제가 발생했다면 코드로 다시 돌아와서 오타 또는 버그를 수정하자.
4. 디버거로 프로그램을 실행하면 코드 한 줄 한 줄을 한 번에 하나씩 실행하여 그 코드가 무슨 작업을 하는지 이해할 수 있다.
5. (!)로 표시된 주석은 여러분이 수정할 수 있는 코드다. 그 부분을 찾아 수정하여 다시 실행했을 때 어떠한 영향을 미치는지 확인한다.
6. 마지막으로, 프로그램을 처음부터 다시 만들어 보자. 책의 코드와 완전히 똑같을 필요는 없다. 여러분만의 변경을 더해도 된다.

이 책의 코드를 타이핑할 때 # 기호 다음부터 그 줄 끝에 있는 텍스트인 주석까지 입력할 필요는 없다. 주석은 프로그래머를 위한 메모이기 때문에 파이썬은 이를 무시한다. 하지만 여러분의 파이썬 코드가 이 책의 프로그램의 코드와 동일한 줄(위치)에 있도록 작성한다면, 여러분의 코드와 책의 코드를 쉽게 비교할 수 있을 것이다. 만약에 여러분의 코드 어디에 오타가 있는지 찾기 어렵다면, 온라인 비교 툴(https://inventwithpython.com/bigbookpython/diff/)로 이 책의 코드와 여러분의 코드를 비교해 볼 것을 추천한다.

각 프로그램에는 **보드게임**, **시뮬레이션**, **예술**, **2인용** 등 프로그램을 설명하기 위한 태그들이 주어진다. 각 태그 및 태그와 관련된 프로젝트에 대한 설명은 부록 A에서 찾을 수 있으며, 프로젝트는 알파벳순으로 나열되어 있다.

파이썬 다운로드하고 설치하기

파이썬은 프로그래밍 언어의 이름이자 파이썬 코드를 실행하는 인터프리터 소프트웨어의 이름이기도 하다. 파이썬 소프트웨어는 무료로 다운로드하여 사용할 수 있다. 여러분이 이미 파이썬을 설치했는지는 커맨드 라인 윈도우에서 체크할 수 있다. 윈도우에서는 명령 프롬프트Command Prompt 프로그램을 열고 **py --version**이라고 입력한다. 만약에 다음과 같이 나온다면 파이썬이 설치되어 있는 것이다.

```
C:\Users\Al>py --version
Python 3.10.0
```

macOS와 리눅스에서는 터미널Terminal 프로그램을 열고 **python3 --version**이라고 입력한다. 만약에 다음과 같이 나온다면 파이썬이 설치되어 있는 것이다.

```
$ python3 --version
Python 3.10.0
```

이 책은 파이썬 버전 3을 사용한다. 파이썬 2와 3 사이에는 서로 호환되지 않는 것들이 있어서, 이 책의 프로그램은 적어도 2009년에 출시된 파이썬 버전 3.1.1 이상에서 실행해야 한다. 만약에 파이썬을 찾을 수 없다거나 파이썬 2 버전을 사용한다는 에러 메시지를 보게 된다면, https://python.org/에서 여러분의 운영체제에 맞는 최신 파이썬 설치 프로그램을 다운로드할 수 있다. 아울러 혹시 파이썬 설치에 문제가 있다면 https://installpython3.com/에서 자세한 안내를 받을 수 있다.

Mu 에디터 다운로드하고 설치하기

파이썬 소프트웨어로 여러분의 프로그램을 실행하게 된다면 텍스트 에디터나 통합 개발 환경Integrated Development Environment, 이하 IDE 애플리케이션에 여러분의 파이썬 코드를 입력하게 될 것이다. 만약 여러분이 초보자라면 IDE로 Mu 에디터를 사용할 것을 추천한다. Mu 에디터는 단순하며, 고급 기능이 많지 않아서 주의가 산만해지지 않도록 할 것이다.

브라우저에서 https://codewith.mu/를 열자. 여러분의 운영체제가 윈도우와 macOS라면 그에 맞는 설치 프로그램을 다운로드하고 더블 클릭하여 실행한다. 만약에 여러분이 macOS를 사용하고 있다면 설치 프로그램을 실행하면 설치를 계속해서 진행하기 위해 Mu 아이콘을 응용 프로그램

Applications 폴더 아이콘으로 드래그해야 하는 창이 열릴 것이다. 만일 여러분이 우분투Ubuntu를 사용하고 있다면 Mu를 파이썬 패키지로 설치해야 할 것이다. 이때 새로운 터미널 창을 열고 `pip3 install mu-editor` 명령어를 통해 실행하여 설치하고 `mu-editor`로 실행한다. 자세한 방법은 다운로드Download 페이지의 Python Package 섹션에 있는 **Instructions** 버튼을 클릭하자.

Mu 에디터 실행하기

Mu를 설치했다면 다음과 같이 실행해 보자.

- 윈도우 7 또는 이후 버전을 사용하고 있다면, 화면의 좌측 하단에 있는 **시작**Start 아이콘을 클릭하고 검색 상자에 `mu`를 입력하여 나타난 **Mu**를 선택한다.
- macOS라면, 파인더Finder 창을 열고 **응용 프로그램**Applications을 클릭하고 이어서 **mu-editor**를 클릭한다.
- 우분투라면, Ctrl+Alt+T를 눌러 터미널Terminal 창을 열고 `python3 -m mu`를 입력한다.

Mu를 처음 실행하면 BBC micro:bit, CircuitPython, Pygame Zero, 그리고 Python 3 등의 옵션을 표시하는 모드 선택Select Mode 창이 나타난다. 여기서 Python 3를 선택하자. 에디터 윈도우의 상단에 있는 **Mode** 버튼을 클릭하면 언제든지 변경할 수 있다.

Mu의 메인 윈도우에 코드를 입력한 다음, 상단에 있는 버튼으로 여러분의 파일을 저장하거나, 열거나, 실행할 수 있다.

IDLE과 다른 에디터 실행하기

파이썬 코드를 작성하기 위해 여러분이 사용할 수 있는 에디터는 매우 많다. 통합 개발 및 학습 환경Integrated Development and Learning Environment, 이하 IDLE 소프트웨어는 파이썬과 함께 설치되며, 어떠한 이유로 Mu를 설치하지 못하거나 작동이 안될 경우에 보조 에디터 역할을 할 수 있다. 이번에는 IDLE을 시작해 보자.

- 윈도우 7 또는 이후 버전을 사용하고 있다면, 화면의 좌측 하단에 있는 **시작**Start 아이콘을 클릭하고 검색 상자에 `idle`을 입력하여 나타난 **IDLE (Python GUI)**를 선택한다.
- macOS라면, 파인더Finder 창을 열고 **응용 프로그램 ▸ Python 3.10 ▸ IDLE**을 클릭한다.
- 우분투라면, **Applications ▸ Accessories ▸ Terminal**을 선택하고 `idle3`를 입력한다. 또는

화면 상단에 있는 **Applications**를 클릭하고 **Programming**을 선택한 다음 **IDLE 3**를 클릭할 수도 있다.

- 라즈베리 파이Raspberry Pi라면, 좌측 상단에 있는 라즈베리 파이Raspberry Pi 메뉴를 클릭하고 이어서 **Programming**을 클릭한 다음 마지막으로 **Python 3 (IDLE)**를 클릭한다. 또는 **Programming** 메뉴 아래에 있는 **Thonny Python IDE**를 선택할 수도 있다.

다음은 파이썬 코드를 입력하고 실행하는 데 사용할 수 있는 무료 에디터다.

- Thonny, 초보자를 위한 파이썬 IDE(https://thonny.org/)
- PyCharm Community Edition, 전문 개발자들이 사용하는 파이썬 IDE(https://www.jetbrains.com/pycharm/)

파이썬 모듈 설치하기

이 책에 있는 대부분의 프로그램은 파이썬을 설치할 때 자동으로 설치되는 파이썬 표준 라이브러리만 필요하다. 하지만 일부 프로그램은 `pyperclip`, `bext`, `playsound` 그리고 `pyttsx3`와 같은 서드-파티 모듈이 필요하다. 이들 모두는 `bigbookpython` 모듈을 설치하면 한 번에 설치할 수 있다.

Mu 에디터의 경우, 1.1.0-alpha 버전(또는 그 이상)을 설치해야 한다. 현재 https://codewith.mu/en/download의 다운로드 페이지에서 추천하는 버전은 1.1.0-beta-7이다. 설치를 했다면 Mu 에디터 화면의 우측 하단에 있는 톱니바퀴 아이콘을 클릭하여 Mu Administration 창을 연다. **Third Party Packages** 탭을 선택하고, 텍스트 필드에 `bigbookpython`을 입력한 뒤 **OK**를 클릭한다. 그러면 이 책의 프로그램들이 사용하는 모든 서드-파티 모듈을 설치하게 될 것이다.

비주얼 스튜디오 코드Visual Studio Code 또는 IDLE 에디터에서는 에디터를 열고 셸interactive shell에 다음의 파이썬 코드를 실행하자.

```
>>> import os, sys
>>> os.system(sys.executable + ' -m pip install --user bigbookpython')
0
```

모든 것이 문제없이 진행되었다면 두 번째 명령어 다음에 숫자 0이 나타날 것이다. 그렇지 않고 에러 메시지나 다른 숫자가 나타난다면 다음과 같이 `--user` 옵션 없이 해보자.

```
>>> import os, sys
>>> os.system(sys.executable + ' -m pip install bigbookpython')
0
```

어떠한 에디터를 사용하든, `import paperclip` 또는 `import bext`를 실행하면 설치가 잘 되었는지 확인할 수 있다. 만약에 이들 `import` 구문에 대한 에러 메시지가 없다면 이들 모듈은 올바르게 설치된 것이며, 이들 모듈을 사용하는 이 책의 프로젝트들을 실행할 수 있게 된 것이다.

이 책의 코드 복사하기

프로그래밍은 프로그래밍을 통해 향상되는 기술이다. 그저 이 책의 코드를 눈으로만 읽거나 복사 붙이기만 해서는 안 된다. 시간을 내서 에디터에 코드를 직접 입력하자. 편집기에서 새로운 파일을 열고 코드를 입력하자. 이 책의 코드 줄(번호)에 주의하여 실수로 줄을 건너 뛰지 않도록 하자. 입력한 코드에 오류가 발생하면, **온라인 비교 툴**(https://inventwithpython.com/bigbookpython/diff/)**을 사용하여 입력한 코드와 책의 코드 사이에 어떠한 차이가 있는지를 확인하자.** 프로그램에 대해 더 자세히 이해하고자 한다면 디버거를 사용하여 실행해 보자.

소스 코드를 입력하고 몇 번 실행한 후, 테스트를 위해 코드를 변경해 보자. (!) 표시가 있는 주석에는 독자가 변경해 볼 만한 제안이 포함되어 있으며, 각 프로젝트마다 더 큰 규모의 수정에 대해 제시하고 있다.

다음으로, 이 책의 소스 코드를 보지 않고 처음부터 프로그램을 다시 만들어 보자. 책의 소스와 똑같을 필요는 없다. 자신만의 버전을 만들어 볼 수도 있다.

이 책의 프로그램을 따라 하다 보면, 자신만의 프로그램을 만들고 싶을 것이다. 대부분의 현대 비디오 게임이나 소프트웨어 애플리케이션은 매우 복잡하기 때문에 프로그래머와 아티스트, 그리고 디자이너로 구성된 팀이 필요하다. 하지만 수많은 보드게임과 카드 게임, 그리고 종이와 연필로 하는 게임들은 프로그램으로 다시 만들 수 있을 만큼 단순하다. 이들 중 다수는 '추상 전략 게임abstract strategy games' 범주에 속한다. 이에 해당하는 게임들의 목록을 https://en.wikipedia.org/wiki/List_of_abstract_strategy_games에서 확인할 수 있다.

터미널에서 프로그램 실행하기

이 책의 프로그래밍 프로젝트는 출력을 위해 컬러풀한 텍스트를 가지고 있는 `bext` 모듈을 사용한다. 하지만 이러한 색상들은 Mu나 IDLE 또는 다른 에디터에서 실행하면 나타나지 않는다. 이

러한 프로그램은 **커맨드 라인**command line이라 불리는 **터미널**Terminal 화면에서 실행해야 한다. 윈도우라면 시작Start 메뉴에서 명령 프롬프트 프로그램을 실행하자. macOS라면, 스포트라이트Spotlight에서 터미널을 실행하자. 우분투 리눅스라면, 우분투 대시Ubuntu Dash에서 터미널을 실행하거나 Ctrl-Alt-T를 누르자.

터미널 창이 나타나면 cdchange directory 명령으로 현재 디렉터리를 **.py** 파일이 있는 폴더로 변경해야 한다(**디렉터리**Directory는 폴더의 또 다른 이름이다). 예를 들어, 윈도우를 사용하고 있고 **C:\Users\Al** 폴더에 파이썬 프로그램이 저장되어 있다면 다음과 같이 입력해야 한다.

```
C:\>cd C:\Users\Al

C:\Users\Al>
```

그런 다음, 파이썬 프로그램을 실행하기 위하여 윈도우에서는 python *yourProgram.py*를, macOS나 리눅스에서는 python3 *yourProgram.py*를 입력하자. 여기서 *yourProgram.py*는 여러분의 파이썬 프로그램의 이름으로 교체한다.

```
C:\Users\Al>python guess.py
Guess the Number, by Al Sweigart al@inventwithpython.com

I am thinking of a number between 1 and 100.
You have 10 guesses left. Take a guess.
--중략--
```

프로그램을 종료하기 위해서 터미널 윈도우 자체를 닫기보다는 Ctrl-C를 눌러 종료할 수 있다.

핸드폰 또는 태블릿에서 프로그램 실행하기

핸드폰이나 태블릿 키보드를 쳐서 코드를 작성하는 것은 지루할 수 있기 때문에 키보드가 있는 랩톱 또는 데스크톱 컴퓨터를 사용하는 게 (프로그래밍에) 이상적이다. 안드로이드 또는 iOS용 파이썬 인터프리터는 없지만, 웹 브라우저에서 사용할 수 있는 온라인 파이썬 인터랙티브 셸을 제공하는 웹사이트가 있다. 만약에 여러분이 강의실 컴퓨터에 새로운 소프트웨어를 설치할 수 있는 계정 권한이 없는 강사라면, 랩톱과 데스크톱에서도 사용할 수 있다.

https://repl.it/languages/Python3/와 https://www.pythonanywhere.com/에는 여러분의 웹 브라우저에서 무료로 사용할 수 있는 파이썬 인터프리터가 있다. 이 책에 있는 대부분의 프로젝트들은 이

들 웹사이트에서 동작할 것이다. 하지만 bext, pyperclip, pyttsx3, 그리고 playsound와 같은 서드-파티 모듈을 사용하는 프로그램은 동작하지 않을 것이다. 또한, open() 함수를 사용하여 파일을 읽거나 써야 하는 프로그램도 동작하지 않을 것이다. 프로그램의 코드 중에 이러한 용어들이 있다면 온라인 파이썬 인터프리터에서 동작하지 않을 것이다.

도움을 받는 방법

개인 교사를 고용한다거나 프로그래밍에 대해 물어볼 프로그래머인 친구가 없다면, 문제에 대한 답을 스스로 찾아야 할 것이다. 다행히도 여러분이 궁금해하는 것 대부분은 이미 다른 사람들이 가졌던 질문이기도 하다. 스스로 답을 찾는 것은 프로그래머가 배워야 할 중요한 덕목 중 하나다.

인터넷에서 프로그래밍 문제에 대한 답을 지속적으로 찾고 있는 자신을 발견한다고 해도 낙심하지 말자. 프로그래밍 기초부터 배우는 게 아니라 인터넷을 통해 확인하는 것이 마치 '꼼수'처럼 느껴질 수 있겠지만, 배움에 있어서 그런 것은 존재하지 않는다. 심지어 프로 개발자들도 매일 인터넷을 검색한다. 여러분은 이번 절에서 인터넷에서 질문을 잘하는 방법과 답변을 검색하는 방법을 배우게 될 것이다.

프로그램에서 유효하지 않은 명령을 수행하려고 하면, 트레이스백traceback이라 불리는 에러 메시지가 표시된다. 트레이스백은 어떤 종류의 에러가 발생했는지와 에러가 발생한 코드 위치를 알려 준다. 다음은 한 사람당 몇 개의 피자 조각을 가져야 하는지를 계산하는 중에 발생한 에러의 예다.

```
Traceback (most recent call last):
  File "pizza.py", line 5, in <module>
    print('Each person gets', (slices / people), ' slices of pizza.')
ZeroDivisionError: division by zero
```

이 트레이스백을 보고 people 변수가 0으로 설정되어 있기 때문에 slices/people 구문에서 0으로 나누는 문제가 있다는 것을 발견하지 못할 수 있다. 에러 메시지는 종종 너무 짧아서 완전한 문장이 아니곤 한다. 이러한 오류는 프로그래머들이 일반적으로 접하는 오류이기 때문에 전체적인 설명보다는 상기하려는 의도다. 만약에 이러한 오류를 처음 접하는 것이라면, 해당 부분을 복사해서 인터넷으로 검색하면 에러의 자세한 의미와 원인을 찾게 될 것이다.

인터넷 검색을 해도 문제에 대한 해결책을 찾을 수 없다면, 온라인 포럼에 질문을 올리거나 여러분이 알고 있는 파이썬 전문가에게 메일로 문의하는 방법이 있다. 그럴 때는 구체적이고 잘 표현된 질문을 하는 것이 가장 효과적이다. 질문 시 전체 소스 코드와 함께 에러 메시지를 구체적으

로 밝히고, 여러분이 시도했던 방법은 무엇인지, 그리고 여러분이 사용하는 운영체제와 파이썬 버전은 어떤 것이었는지 밝히도록 하자. 이렇게 하면 문제에 대한 해결 방법을 얻게 될 뿐만 아니라 동일한 문제를 겪는 다른 프로그래머도 여러분의 질문 덕에 도움을 받을 것이다.

코드 입력

프로그래머가 되기 위해서 빠른 타이핑 능력을 가질 필요는 없지만, 코드를 빨리 입력한다는 자체는 도움이 된다. 빠른 타이핑은 프로그래밍에서의 번거로운 일을 줄여준다. 여러분이 이 책의 프로그램을 가지고 작업할 때도 코드를 보면서 얼른 타이핑하고 싶어질 것이다.

https://typingclub.com/ 또는 https://www.typing.com/ 등의 무료 웹사이트에서 타이핑 방법을 배울 수 있다. 잘 만들어진 타이핑 프로그램은 키보드 모양과 가상의 손 모양을 컴퓨터 화면에 표시해 주기 때문에 입력할 키를 찾기 위해 키보드를 내려다보는 습관을 고칠 수 있다. 다른 모든 기술과 마찬가지로 타이핑은 연습의 문제이며, 코드를 작성하는 것은 수많은 타이핑 기회를 줄 것이다.

키보드의 단축키는 마우스 커서를 메뉴로 옮겨서 특정 작업을 선택하고, 수행하는 데 걸리는 시간을 단축시켜 줄 것이다. 단축키는 보통 'Ctrl-C'처럼 표현된다. 이것은 키보드의 Ctrl 키를 누른 상태에서 C 키를 누른다는 의미다. Ctrl 키를 한번 누른 다음에 C 키를 누르라는 게 아니다.

마우스를 사용하여 애플리케이션의 상단(윈도우 그리고 리눅스)이나 화면 상단(macOS)에 있는 메뉴바를 열면, 저장할 때 사용하는 Ctrl-S와 복사할 때 사용하는 Ctrl-C 등의 일반적인 단축키를 찾을 수 있다.[2] 시간을 투자해 이러한 키보드 단축키를 배워 두면 도움이 될 것이다.

예를 들어, 윈도우와 리눅스에서는 Alt-Tab, 그리고 macOS에서는 command-tab을 누르면 다른 애플리케이션 화면으로 전환시켜 준다. Alt 또는 command 키를 누른 상태에서 Tab을 계속 누르면 전환하고자 하는 특정 화면을 선택할 수 있다.

2 옮긴이 macOS에서는 command-S/command-C와 같이 command 키를 사용한다.

복사하기와 붙여넣기

클립보드clipboard는 붙여넣을 데이터를 임시로 저장할 수 있도록 해주는 운영체제의 기능이다. 여기서 데이터는 텍스트, 이미지, 파일, 또는 다른 종류의 정보가 될 수 있지만, 이번 절에서는 텍스트 데이터를 가지고 설명할 것이다. 텍스트를 복사하면 현재 선택된 텍스트의 복사본이 클립보드에 배치된다. 텍스트를 붙여넣으면 커서의 현재 위치에 직접 입력한 것처럼 클립보드에 있는 텍스트가 입력된다. 복사하여 붙여넣으면 그 텍스트 분량이 한 줄이든, 수백 페이지든 컴퓨터에 이미 있는 텍스트를 다시 입력할 필요가 없다.

텍스트를 복사하여 붙여넣으려면 복사할 텍스트를 먼저 선택(즉, 하이라이트)한다. 이 작업은 마우스의 기본 버튼(마우스를 오른손잡이용으로 설정한 경우는 왼쪽 버튼)을 누른 상태로 드래그하여 원하는 텍스트를 선택한다. 하지만 키보드를 이용하여 Shift 키를 누른 상태에서 커서 키를 움직이면 더 빠르고 정확하게 해낼 수 있다. 많은 애플리케이션에서는 단어를 더블 클릭하면 전체 단어가 즉시 선택된다. 또한 세 번 클릭하면 전체 줄 또는 전체 단락이 즉시 선택된다.

다음 단계는 윈도우에서는 Ctrl-C를, 그리고 macOS에서는 command-C를 눌러 선택한 텍스트를 클립보드에 복사한다. 클립보드는 하나의 텍스트만 저장할 수 있으므로 복사하면 이전에 있던 것은 사라진다.

마지막으로, 텍스트를 붙이고 싶은 위치로 커서를 옮기고 윈도우에서는 Ctrl-V를, macOS에서는 command-V를 눌러 붙여넣는다. 새로운 텍스트가 클립보드에 저장되기 전까지 원하는 만큼 붙여넣을 수 있다.

텍스트 찾기와 대체하기

구글의 검색 인류학자인 댄 러셀Dan Russell은 2011년 《**Atlantic**》의 기사(https://www.theatlantic.com/technology/archive/2011/08/crazy-90-percent-of-people-dont-know-how-to-use-ctrl-f/243840/)에서, 사람들의 컴퓨터 사용 습관을 연구했을 때 90퍼센트는 애플리케이션에서 단어를 찾고자 할 때 Ctrl-F(윈도우와 리눅스에서) 또는 command-F(macOS에서)를 눌러서 찾을 수 있었다는 것을 몰랐다고 했다. 이 기능은 코드 에디터에서뿐만 아니라 워드 프로세서, 웹 브라우저, 스프레드시트 애플리케이션을 포함하여 텍스트를 표시하는 거의 모든 종류의 프로그램에서 매우 유용한 기능이다. Ctrl-F를 누르면 프로그램 내에서 찾고자 하는 단어를 입력할 수 있는 찾기Find 창이 나타난다. 보통 F3 키는 그 다음에 위치한 단어를 찾기 위해 연속해서 검색해 준다. 이 기능은 어떤 단어를 찾기 위해 문서를 직접 스크롤하는 것에 비해 엄청난 시간을 절약해 준다.

에디터는 찾아서 고치는 기능도 가지고 있으며, 하나의 텍스트를 찾아 다른 텍스트로 바꿔 준다. 이 기능은 변수나 함수의 이름을 바꾸고 싶을 때 유용하다. 하지만 찾아서 고치는 기능은 주의해서 사용해야 한다. 왜냐하면 의도치 않게 찾고자 하는 기준과 일치하는 텍스트가 바뀔 수 있기 때문이다.

디버거

디버거는 한 번에 한 줄씩 실행하여 프로그램 변수의 현재 상태를 검사할 수 있게 하는 도구로, 버그를 추적하는 데 유용하다. 이번 절에서는 Mu 에디터의 디버거 기능에 대해 설명한다. 다른 디버거의 사용자 인터페이스가 이와 다르더라도 동일한 기능을 가지고 있으니 걱정하지 말자.

디버거로 프로그램을 실행하려면 IDE의 Run 메뉴 대신에 Debug 메뉴를 사용한다. 디버거는 프로그램의 첫 줄에서 일시 중지된 상태로 시작한다. 모든 디버거는 Continue, Step In, Step Over, Step Out, Stop 버튼을 가지고 있다.

Continue 버튼을 클릭하면 다음의 브레이크포인트에 도달할 때까지, 만약에 다음 브레이크 포인트가 없다면 프로그램 코드 끝까지 실행된다(브레이크포인트에 대해서는 이번 절 후반부에 설명한다). 디버깅을 끝내고 프로그램을 계속해서 실행하고자 한다면 Continue 버튼을 클릭한다.

Step In 버튼을 클릭하면 디버거는 다음 줄의 코드를 실행하고 다시 일시 중지된 상태가 된다. 만약에 다음 줄의 코드가 함수 호출이라면, 디버거는 그 함수로 '들어가서' 해당 함수의 첫 번째 줄로 이동한다.

Step Over 버튼을 클릭하면 Step In 버튼처럼 다음 줄의 코드를 실행한다. 하지만 다음 줄의 코드가 함수 호출이라면, 그 함수의 코드를 '건너뛸' 것이다. 그 함수의 코드는 최고의 속도로 실행되며, 디버거는 그 함수 호출이 반환되는 즉시 일시 중지된다. Step Over 버튼을 사용하는 것이 Step In 버튼을 사용하는 것보다 더 일반적이다.

Step Out 버튼을 클릭하면 디버거는 현재의 함수에서 빠져나오기 위해 최고의 속도로 코드를 실행한다. 만약에 Step In 버튼으로 함수 호출로 들어갔는데 다시 돌아가기 위해 함수의 코드를 그냥 실행하고자 한다면, Step Out 버튼을 클릭하여 현재의 함수 호출을 빠져나가면 된다.

디버깅을 완전히 멈추고 프로그램의 나머지 부분도 더 이상 실행하고 싶지 않다면, Stop 버튼을 클릭하자. Stop 버튼은 프로그램을 즉시 종료한다.

특정 프로그램 코드에 **브레이크포인트**breakpoint를 설정하고 그 브레이크포인트에 도착할 때까지 프로그램이 일반적인 속도로 실행되도록 할 수 있다. 그 지점에서 디버거가 일시 중지되므로 변수를 살펴볼 수 있으며, 각 줄의 코드를 단계적으로 실행할 수 있게 한다. 대부분의 IDE는 코드 창의 왼쪽에 있는 행 번호를 더블 클릭하면 브레이크포인트를 설정할 수 있다.

프로그램의 변수에 현재 저장된 값은 모든 디버거의 디버깅 창 안 어딘가에 표시된다. 프로그램을 디버깅하는 일반적인 방법들 중 하나는 **프린트 디버깅**print debugging으로, 변수의 값이 표시되도록 `print()` 호출을 추가하고 프로그램을 실행하는 것이다. 하지만 프로그램을 실행해 보면, 다른 변수의 값을 확인하기 위해 또 다른 `print()` 호출을 추가해야 한다는 것을 깨닫게 된다. 이것은 계속해서 프로그램을 다시 실행해야 한다는 의미이며, 이렇게 실행한다는 것은 또 다른 `print()` 호출을 추가하는 주기(사이클)가 돌아오게 될 것이라는 의미다. 또한, 추가했던 `print()` 호출 중 일부를 잊어버리기도 하므로 `print()` 호출을 삭제하는 시간도 필요하다. 프린트 디버깅은 단순한 버그를 잡는 데는 수월하겠지만, 실제 디버거를 사용하는 것이 장기적으로 볼 때 시간을 절약하는 방법이다.

요약

프로그래밍은 재미있고 창의적인 기술이다. 여러분이 파이썬 구문의 기초를 마스터했든, 아니면 실제 파이썬 프로그램을 만들고 싶어하는 것이든, 이 책의 프로젝트들은 몇 페이지의 코드만 해봐도 여러분에게 새로운 아이디어가 나도록 촉진제 역할을 할 것이다.

이 책의 프로그램을 가장 잘 활용하는 방법은 단순히 코드를 읽거나 복사 붙여넣기를 하는 게 아니다. 시간을 내서 이 책의 코드를 여러분의 에디터에 직접 입력하여 코드 작성의 머슬 메모리muscle memory[3]를 발달시키자. 또한, 코드를 단순히 눈으로만 훑어보지 말고 한 줄씩 천천히 검토하자. 이해가 되지 않는 코드는 인터넷 검색 엔진을 이용하여 찾아보거나 인터랙티브 셸에서 테스트를 해보자.

마지막으로, 프로그램을 처음부터 다시 만들어 보고 여러분만의 기능을 추가해 보자. 이러한 연습은 실제로 동작하는 프로그램을 만들기 위해 적용될 프로그래밍의 개념을 더욱 단단하게 만들어 줄 것이다. 그리고 무엇보다도 즐기는 마음을 잊지 말자!

3 옮긴이 직접 코드를 입력하면서 몸이 코드를 기억하도록 하는 방식

PROJECT

#1

베이글

단서를 바탕으로 세 자리 숫자를 알아내자

상수(constant)를 사용하는 연습을 한다

연역적 논리 게임인 베이글은 단서를 바탕으로 세 자리 숫자를 유추하는 게임이다. 이 게임은 여러분의 추측에 대한 응답으로 다음의 힌트 중 하나를 제공한다. 만약에 여러분이 추측한 숫자가 숫자는 맞지만 위치가 틀린 경우에는 'Pico', 숫자도 맞고 자리도 맞으면 'Fermi', 맞는 숫자가 없으면 'Bagels'라는 힌트를 제공한다. 숫자를 찾기 위한 기회는 10번이 주어진다.

프로그램 실행

bagels.py를 실행하면 다음과 같다.

```
By Al Sweigart al@inventwithpython.com

I am thinking of a 3-digit number with no repeated digits.
Try to guess what it is. Here are some clues:
When I say:    That means:
  Pico         One digit is correct but in the wrong position.
  Fermi        One digit is correct and in the right position.
  Bagels       No digit is correct.

For example, if the secret number was 248 and your guess was 843, the
clues would be Fermi Pico.
I have thought up a number.
 You have 10 guesses to get it.
Guess #1:
> 123
Bagels
Guess #2:
> 456
Pico Pico
Guess #3:
> 645
Fermi Fermi
--중략--
Guess #6:
> 648
You got it!
Do you want to play again? (yes or no)
> no
Thanks for playing!
```

동작 원리

이번 프로그램은 정숫값이 아니라 숫자가 포함된 문자열 값을 사용한다는 점을 기억하자. 예를 들어, `'426'`과 `426`은 서로 다른 값이다. 우리는 수학적 연산이 아니라 비밀번호와의 문자열 비교를 해야 하기 때문에 문자열을 사용해야 한다. 그러므로 `'0'`이 첫 자리에 올 수 있음을 기억하자. 문자열 `'026'`과 `'26'`은 다르지만, 정수 `026`은 `26`과 같다.

```
1 """Bagels, by Al Sweigart al@inventwithpython.com
2 A deductive logic game where you must guess a number based on clues.
3 This code is available at https://nostarch.com/big-book-small-python-programming
```

```
A version of this game is featured in the book, "Invent Your Own
Computer Games with Python" https://nostarch.com/inventwithpython
Tags: short, game, puzzle"""

import random

NUM_DIGITS = 3  # (!) 이 값을 1 또는 10으로 설정해 보자.
MAX_GUESSES = 10  # (!) 이 값을 1 또는 100으로 설정해 보자.


def main():
    print('''Bagels, a deductive logic game.
By Al Sweigart al@inventwithpython.com

I am thinking of a {}-digit number with no repeated digits.
Try to guess what it is. Here are some clues:
When I say:    That means:
  Pico         One digit is correct but in the wrong position.
  Fermi        One digit is correct and in the right position.
  Bagels       No digit is correct.

For example, if the secret number was 248 and your guess was 843, the
clues would be Fermi Pico.'''.format(NUM_DIGITS))

    while True:  # 메인 게임 루프
        # 이것은 사용자가 예측해야 할 비밀번호를 저장한다:
        secretNum = getSecretNum()
        print('I have thought up a number.')
        print(' You have {} guesses to get it.'.format(MAX_GUESSES))

        numGuesses = 1
        while numGuesses <= MAX_GUESSES:
            guess = ''
            # 유효한 예측값을 입력할 때까지 루프를 계속 돈다:
            while len(guess) != NUM_DIGITS or not guess.isdecimal():
                print('Guess #{}: '.format(numGuesses))
                guess = input('> ')

            clues = getClues(guess, secretNum)
            print(clues)
            numGuesses += 1

            if guess == secretNum:
                break  # 숫자를 맞췄으니 이 루프에서 빠져나간다.
            if numGuesses > MAX_GUESSES:
                print('You ran out of guesses.')
                print('The answer was {}.'.format(secretNum))

        # 다시 게임을 하고 싶은지 묻는다.
        print('Do you want to play again? (yes or no)')
        if not input('> ').lower().startswith('y'):
            break
```

```
    print('Thanks for playing!')


def getSecretNum():
    """NUM_DIGITS개의 임의 숫자로 구성된 문자열을 반환한다."""
    numbers = list('0123456789')  # 0부터 9까지의 숫자 리스트를 생성한다.
    random.shuffle(numbers)  # 무작위 순서가 되도록 섞는다.

    # 비밀번호를 뽑기 위해 리스트의 처음부터 NUM_DIGITS자리까지의 수를 얻는다:
    secretNum = ''
    for i in range(NUM_DIGITS):
        secretNum += str(numbers[i])
    return secretNum


def getClues(guess, secretNum):
    """비밀번호에 대한 단서인 pico, fermi,
    bagels로 구성된 문자열을 반환한다."""
    if guess == secretNum:
        return 'You got it!'

    clues = []

    for i in range(len(guess)):
        if guess[i] == secretNum[i]:
            # 맞는 숫자이며 위치(자리)도 맞다.
            clues.append('Fermi')
        elif guess[i] in secretNum:
            # 숫자는 맞지만 잘못된 위치(자리)에 있다.
            clues.append('Pico')
    if len(clues) == 0:
        return 'Bagels'  # 일치하는 숫자가 전혀 없다.
    else:
        # 힌트를 알파벳순으로 정렬하여
        # 힌트의 순서가 또 다른 힌트가 되지 않도록 한다.
        clues.sort()
        # 문자열 힌트 리스트를 가지고 단일 문자열을 만든다.
        return ' '.join(clues)


# 이 프로그램이 다른 프로그램에 임포트(import)된 게 아니라면 게임이 실행된다:
if __name__ == '__main__':
    main()
```

소스 코드를 입력하고 여러 번 실행한 후, 실험을 위해 몇 가지를 변경해 보자. (!) 마크가 있는 주석은 여러분이 할 수 있는 간단한 변경에 대해 제안한 것이다. 다음 내용에 대해 스스로 방법을 찾아보자.

- NUM_DIGITS를 수정하여 비밀번호의 자릿수를 바꾸자.
- MAX_GUESSES를 수정하여 플레이어의 예측 횟수를 바꾸자.
- 비밀번호에 숫자뿐만 아니라 문자도 있는 버전을 만들어 보자.

프로그램 살펴보기

다음 질문에 대한 답을 찾아보자. 코드를 약간 수정하여 테스트하고, 변경 사항이 어떠한 영향을 미쳤는지 확인해 보자.

1. NUM_DIGITS 상수를 변경하면 어떻게 되는가?
2. MAX_GUESSES 상수를 변경하면 어떻게 되는가?
3. NUM_DIGITS에 10보다 큰 수를 설정하면 어떻게 되는가?
4. 30행에 있는 `secretNum = getSecretNum()`를 `secretNum = '123'`으로 바꾸면 어떻게 되는가?
5. 34행에 있는 `numGuesses = 1`을 삭제하거나 주석 처리하면 어떻게 되는가?
6. 62행에 있는 `random.shuffle(numbers)`를 삭제하거나 주석 처리하면 어떻게 되는가?
7. 74행에 있는 `if guess == secretNum:`과 75행에 있는 `return 'You got it!'`을 삭제하거나 주석 처리하면 어떻게 되는가?
8. 44행에 있는 `numGuesses += 1`을 주석 처리하면 어떻게 되는가?

PROJECT

#2

생일 역설

여러 규모의 그룹에서 두 사람의 생일이 동일할 확률을 측정

파이썬의 datetime 모듈을 사용한다

소규모 집단에서 두 사람의 생일이 서로 같을 확률이 놀라울 만큼 높은 것을 생일 문제Birthday Problem 또는 생일 역설Birthday Paradox이라고 한다. 70명이 있는 그룹에서 생일이 같은 두 사람이 나올 확률은 99.9퍼센트다. 심지어 23명만 있어도 50퍼센트가 된다. 이번 장의 프로그램은 다양한 규모의 그룹에 대해 어느 정도의 확률이 되는지 알아보는 확률 실험을 실시한다. 여러 번의 무작위 시도를 통하여 나타난 결과를 이해하는 방식으로, 이러한 유형의 실험을 몬테카를로 실험Monte Carlo experiment이라고 부른다.

생일 역설에 대한 자세한 내용은 위키백과(https://ko.wikipedia.org/wiki/생일_문제)를 참고하자.

프로그램 실행

birthdayparadox.py를 실행하면 다음과 같은 결과가 나올 것이다.

```
Birthday Paradox, by Al Sweigart al@inventwithpython.com
--중략--
How many birthdays shall I generate? (Max 100)
> 23

Here are 23 birthdays:
Dec 20, Jan 24, Sep 26, Feb 6, Nov 21, Jan 22, Jan 31, Sep 16, Apr 11, Nov 6, Dec 16, Mar
22, Dec 18, Jun 29, Sep 21, Jun 22, Oct 3, Nov 17, May 9, Oct 27, Mar 5, Feb 27, Oct 31

In this simulation, there are no matching birthdays.

Generating 23 random birthdays 100,000 times...
Press Enter to begin...
Let's run another 100,000 simulations.
0 simulations run...
10000 simulations run...
--중략--
90000 simulations run...
100,000 simulations run.
Out of 100,000 simulations of 23 people, there was a
matching birthday in that group 50808 times. This means
that 23 people have a 50.81 % chance of
having a matching birthday in their group.
That's probably more than you would think!
```

동작 원리

100,000번의 시뮬레이션을 실행하는 데 시간이 걸릴 수 있으니, 95행과 96행의 코드가 10,000번의 시뮬레이션이 끝날 때마다 알려 준다. 이 피드백을 통하여 프로그램이 멈춰버린 게 아니라는 것을 알 수 있다. 95행에 10_000 그리고 93행, 103행에 100_000이라고 정수에 밑줄이 들어가 있다. 이 밑줄은 특별한 의미가 없지만, 파이썬은 프로그래머가 정숫값을 더 쉽게 읽을 수 있도록 이를 허용한다. 다시 말해, 100000보단 100_000이 '십만one hundred thousand'이라 읽기가 더 수월하다.

```
1 """Birthday Paradox Simulation, by Al Sweigart al@inventwithpython.com
2 Explore the surprising probabilities of the "Birthday Paradox".
3 More info at https://en.wikipedia.org/wiki/Birthday_problem
4 This code is available at https://nostarch.com/big-book-small-python-programming
5 Tags: short, math, simulation"""
```

```
6
7 import datetime, random
8
9
10 def getBirthdays(numberOfBirthdays):
11     """생일에 대한 임의의 날짜 객체들의 리스트를 반환한다."""
12     birthdays = []
13     for i in range(numberOfBirthdays):
14         # 여기서 연도는 중요하지 않기 때문에
15         # 모든 생일을 같은 연도로 한다.
16         startOfYear = datetime.date(2001, 1, 1)
17
18         # 그 연도에서 임의의 날짜를 얻는다:
19         randomNumberOfDays = datetime.timedelta(random.randint(0, 364))
20         birthday = startOfYear + randomNumberOfDays
21         birthdays.append(birthday)
22     return birthdays
23
24
25 def getMatch(birthdays):
26     """생일 리스트에서 중복되는 생일 날짜인
27    객체를 반환한다."""
28     if len(birthdays) == len(set(birthdays)):
29         return None  # 모든 생일이 서로 다르다면 None을 반환한다.
30
31     # 모든 생일을 각각 다른 생일과 비교한다:
32     for a, birthdayA in enumerate(birthdays):
33         for b, birthdayB in enumerate(birthdays[a + 1 :]):
34             if birthdayA == birthdayB:
35                 return birthdayA  # 일치하는 생일을 반환한다.
36
37
38 # 인트로 출력:
39 print('''Birthday Paradox, by Al Sweigart al@inventwithpython.com
40
41 The birthday paradox shows us that in a group of N people, the odds
42 that two of them have matching birthdays is surprisingly large.
43 This program does a Monte Carlo simulation (that is, repeated random
44 simulations) to explore this concept.
45
46 (It's not actually a paradox, it's just a surprising result.)
47 ''')
48
49 # 월 이름이 순서대로 있는 튜플을 만든다:
50 MONTHS = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
51           'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')
52
53 while True:  # 사용자가 유효한 값을 입력할 때까지 계속 묻는다.
54     print('How many birthdays shall I generate? (Max 100)')
55     response = input('> ')
56     if response.isdecimal() and (0 < int(response) <= 100):
57         numBDays = int(response)
```

```
        break  # 사용자가 유효한 값을 입력
print()

# 생일을 생성하고 출력하기:
print('Here are', numBDays, 'birthdays:')
birthdays = getBirthdays(numBDays)
for i, birthday in enumerate(birthdays):
    if i != 0:
        # 첫 번째 생일 이후부터 각 생일마다 콤마를 표시한다.
        print(', ', end='')
    monthName = MONTHS[birthday.month - 1]
    dateText = '{} {}'.format(monthName, birthday.day)
    print(dateText, end='')
print()
print()

# 두 생일이 서로 일치하는지 판단한다.
match = getMatch(birthdays)

# 결과 출력하기:
print('In this simulation, ', end='')
if match != None:
    monthName = MONTHS[match.month - 1]
    dateText = '{} {}'.format(monthName, match.day)
    print('multiple people have a birthday on', dateText)
else:
    print('there are no matching birthdays.')
print()

# 100,000번의 시뮬레이션 실행하기:
print('Generating', numBDays, 'random birthdays 100,000 times...')
input('Press Enter to begin...')

print('Let\'s run another 100,000 simulations.')
simMatch = 0  # 생일이 일치하는 시뮬레이션 수
for i in range(100_000):
    # 10,000번의 시뮬레이션마다 진행 상황 출력하기:
    if i % 10_000 == 0:
        print(i, 'simulations run...')
    birthdays = getBirthdays(numBDays)
    if getMatch(birthdays) != None:
        simMatch = simMatch + 1
print('100,000 simulations run.')

# 시뮬레이션 결과 출력하기:
probability = round(simMatch / 100_000 * 100, 2)
print('Out of 100,000 simulations of', numBDays, 'people, there was a')
print('matching birthday in that group', simMatch, 'times. This means')
print('that', numBDays, 'people have a', probability, '% chance of')
print('having a matching birthday in their group.')
print('That\'s probably more than you would think!')
```

프로그램 살펴보기

다음 질문에 대한 답을 찾아보자. 코드를 약간 수정하여 테스트하고, 변경 사항이 어떠한 영향을 미쳤는지 확인해 보자.

1. 이 프로그램에서 생일은 어떻게 표현되는가?(힌트 16행)
2. 프로그램이 생성하는 최대 생일 수 100개라는 제한을 없애려면 어떻게 해야 하는가?
3. 57행의 `numBDays = int(response)`를 삭제하거나 주석 처리하면 어떤 에러 메시지가 표시되는가?
4. 월 이름을 `'Jan'`이 아니라 `'January'`라는 전체 이름으로 표시하려면 어떻게 해야 하는가?
5. `'X simulations run...'`이라는 메시지가 시뮬레이션 10,000번마다 나오는 게 아니라 1,000번마다 표시되도록 하려면 어떻게 해야 하는가?

PROJECT

#30

FOUR-IN-A-ROW

두 명의 플레이어가 4개의 타일을 연속으로 연결하는 보드게임

중력을 흉내내는 데이터 구조를 생성한다

두 명의 플레이어가 타일을 떨어뜨려서 플레이하는 고전적인 보드게임으로, 4개의 타일을 수평, 수직, 또는 대각선으로 연속되게 하는 동시에 상대가 그렇게 되는 것을 막는 게임이다. 이 게임은 커넥트 포Connect Four와 비슷하다.

프로그램 실행

fourinarow.py를 실행하면 다음과 같다.

```
Four in a Row, by Al Sweigart al@inventwithpython.com
--중략--
    1234567
    +-------+
    |.......|
    |.......|
    |.......|
    |.......|
    |.......|
    |.......|
    +-------+
Player X, enter a column or QUIT:
> 3

     1234567
    +-------+
    |.......|
    |.......|
    |.......|
    |.......|
    |.......|
    |..X....|
    +-------+
Player O, enter a column or QUIT:
> 5
--중략--
Player X, enter a column or QUIT:
> 2

     1234567
    +-------+
    |.......|
    |.......|
    |.X.....|
    |OXO.X..|
    |OXOXXO.|
    |OXXXOO.|
    +-------+
Player X has won!
```

동작 원리

이 책의 보드게임 프로젝트들은 유사한 프로그램 구조를 가지고 있다. 보드의 상태를 나타내기 위해 딕셔너리 또는 리스트를 주로 사용하며, 보드의 데이터 구조를 반환하는 `getNewBoard()` 함수, 화면에 보드 데이터 구조를 렌더링하는 `displayBoard()` 함수 등이 그러하다. 여러분만의 보드게임 프로그램을 만들고 싶다면, 이 책에서 **보드게임**board game으로 분류되는 프로젝트들을 가져다가 비교해 보자.

```
"""Four in a Row, by Al Sweigart al@inventwithpython.com
A tile-dropping game to get four in a row, similar to Connect Four.
This code is available at https://nostarch.com/big-book-small-python-programming
Tags: large, game, board game, two-player"""

import sys

# 보드를 표시하기 위해 사용되는 상수들:
EMPTY_SPACE = '.'  # 공백보다 점(period)이 카운팅에 용이하다.
PLAYER_X = 'X'
PLAYER_O = 'O'

# 참고: 만약에 BOARD_WIDTH를 바꾼다면 displayBoard()와 COLUMN_LABELS를 업데이트하자.
BOARD_WIDTH = 7
BOARD_HEIGHT = 6
COLUMN_LABELS = ('1', '2', '3', '4', '5', '6', '7')
assert len(COLUMN_LABELS) == BOARD_WIDTH


def main():
    print("""Four in a Row, by Al Sweigart al@inventwithpython.com

Two players take turns dropping tiles into one of seven columns, trying
to make four in a row horizontally, vertically, or diagonally.
""")

    # 새로운 게임 설정:
    gameBoard = getNewBoard()
    playerTurn = PLAYER_X

    while True:  # 플레이어의 차례를 실행한다.
        # 보드를 표시하고 플레이어의 움직임을 얻는다:
        displayBoard(gameBoard)
        playerMove = askForPlayerMove(playerTurn, gameBoard)
        gameBoard[playerMove] = playerTurn

        # 이겼는지 비겼는지 확인:
        if isWinner(playerTurn, gameBoard):
            displayBoard(gameBoard)  # 마지막으로 보드를 표시한다.
            print('Player ' + playerTurn + ' has won!')
```

```
            sys.exit()
        elif isFull(gameBoard):
            displayBoard(gameBoard)  # 마지막으로 보드를 표시한다.
            print('There is a tie!')
            sys.exit()

        # 다른 플레이어에게 차례를 넘김:
        if playerTurn == PLAYER_X:
            playerTurn = PLAYER_O
        elif playerTurn == PLAYER_O:
            playerTurn = PLAYER_X


def getNewBoard():
    """Four in a Row 보드를 나타내는 딕셔너리를 반환한다.

    키는 두 개의 정수인 (columnIndex, rowIndex) 튜플이며,
    값은 'X', 'O' 또는 '.'(빈 공간) 문자열 중 하나다."""
    board = {}
    for columnIndex in range(BOARD_WIDTH):
        for rowIndex in range(BOARD_HEIGHT):
            board[(columnIndex, rowIndex)] = EMPTY_SPACE
    return board


def displayBoard(board):
    """보드와 타일을 화면에 표시한다."""

    '''보드 템플릿의 format() 문자열 메서드에 전달할 리스트를 준비한다.
    리스트에는 왼쪽에서 오른쪽으로, 그리고
    위에서 아래로 이동하는 보드의 모든 타일(및 빈 공간)이 있다.'''
    tileChars = []
    for rowIndex in range(BOARD_HEIGHT):
        for columnIndex in range(BOARD_WIDTH):
            tileChars.append(board[(columnIndex, rowIndex)])

    # 보드 표시하기:
    print("""
     1234567
    +-------+
    |{}{}{}{}{}{}{}|
    |{}{}{}{}{}{}{}|
    |{}{}{}{}{}{}{}|
    |{}{}{}{}{}{}{}|
    |{}{}{}{}{}{}{}|
    |{}{}{}{}{}{}{}|
    +-------+""".format(*tileChars))


def askForPlayerMove(playerTile, board):
    """플레이어가 타일을 놓을 보드의 열을 선택하게 한다.

```

```
    타일이 놓이는 (열, 행) 튜플을 반환한다."""
    while True:  # 유효한 입력을 할 때까지 계속 물어 본다.
        print('Player {}, enter a column or QUIT:'.format(playerTile))
        response = input('> ').upper().strip()

        if response == 'QUIT':
            print('Thanks for playing!')
            sys.exit()

        if response not in COLUMN_LABELS:
            print('Enter a number from 1 to {}.'.format(BOARD_WIDTH))
            continue  # 플레이어에게 다시 물어 본다.

        columnIndex = int(response) - 1  # 0 기반의 인덱스를 위해 -1한다.

        # 모든 열이 가득 차면 다시 물어 본다:
        if board[(columnIndex, 0)] != EMPTY_SPACE:
            print('That column is full, select another one.')
            continue  # 플레이어에게 다시 물어 본다.

        # 바닥부터 시작하여 첫 번째 빈 공간을 찾는다.
        for rowIndex in range(BOARD_HEIGHT - 1, -1, -1):
            if board[(columnIndex, rowIndex)] == EMPTY_SPACE:
                return (columnIndex, rowIndex)


def isFull(board):
    """보드에 빈 공간이 없다면 True를 반환하고,
    그렇지 않으면 False를 반환한다."""
    for rowIndex in range(BOARD_HEIGHT):
        for columnIndex in range(BOARD_WIDTH):
            if board[(columnIndex, rowIndex)] == EMPTY_SPACE:
                return False  # 빈 공간을 찾았으므로 False를 반환한다.
    return True  # 모든 공간이 찼다.


def isWinner(playerTile, board):
    """playerTile'이 'board'에 연속으로 4개의 타일을 가지고 있으면 True를 반환하고,
    그렇지 않으면 False를 반환한다."""

    # 전체 보드에 대해 four-in-a-row를 검사한다:
    for columnIndex in range(BOARD_WIDTH - 3):
        for rowIndex in range(BOARD_HEIGHT):
            # 수평으로 four-in-a-row인지 확인한다:
            tile1 = board[(columnIndex, rowIndex)]
            tile2 = board[(columnIndex + 1, rowIndex)]
            tile3 = board[(columnIndex + 2, rowIndex)]
            tile4 = board[(columnIndex + 3, rowIndex)]
            if tile1 == tile2 == tile3 == tile4 == playerTile:
                return True

    for columnIndex in range(BOARD_WIDTH):
```

```
        for rowIndex in range(BOARD_HEIGHT - 3):
            # 수직으로 four-in-a-row인지 확인한다:
            tile1 = board[(columnIndex, rowIndex)]
            tile2 = board[(columnIndex, rowIndex + 1)]
            tile3 = board[(columnIndex, rowIndex + 2)]
            tile4 = board[(columnIndex, rowIndex + 3)]
            if tile1 == tile2 == tile3 == tile4 == playerTile:
                return True

    for columnIndex in range(BOARD_WIDTH - 3):
        for rowIndex in range(BOARD_HEIGHT - 3):
            # 오른쪽 아래 대각선으로 four-in-a-row인지 확인한다:
            tile1 = board[(columnIndex, rowIndex)]
            tile2 = board[(columnIndex + 1, rowIndex + 1)]
            tile3 = board[(columnIndex + 2, rowIndex + 2)]
            tile4 = board[(columnIndex + 3, rowIndex + 3)]
            if tile1 == tile2 == tile3 == tile4 == playerTile:
                return True

            # 왼쪽 아래 대각선으로 four-in-a-row인지 확인한다:
            tile1 = board[(columnIndex + 3, rowIndex)]
            tile2 = board[(columnIndex + 2, rowIndex + 1)]
            tile3 = board[(columnIndex + 1, rowIndex + 2)]
            tile4 = board[(columnIndex, rowIndex + 3)]
            if tile1 == tile2 == tile3 == tile4 == playerTile:
                return True
    return False


# 이 프로그램이 다른 프로그램에 임포트(import)된 게 아니라면 게임이 실행된다:
if __name__ == '__main__':
    main()
```

소스 코드를 입력하고 여러 번 실행한 후, 실험을 위해 몇 가지를 변경해 보자. (!) 마크가 있는 주석은 여러분이 할 수 있는 간단한 변경에 대해 제안한 것이다. 다음 내용에 대해 스스로 방법을 찾아보자.

- three-in-a-row 또는 five-in-a-row로 변형해 보자.
- 이 게임을 세 명이 플레이할 수 있도록 변형해 보자.
- 플레이어의 차례가 끝나면 무작위로 '와일드카드' 타일이 떨어지고 모든 플레이어가 사용할 수 있도록 만들어 보자.
- 다음 플레이어가 사용할 수 없는 '블록' 타일을 추가해 보자.

프로그램 살펴보기

다음 질문에 대한 답을 찾아보자. 코드를 약간 수정하여 테스트하고, 변경 사항이 어떠한 영향을 미쳤는지 확인해 보자.

1. 11행에 있는 `PLAYER_O = 'O'`를 `PLAYER_O = 'X'`로 변경하면 어떻게 되는가?
2. 116행에 있는 `return (columnIndex, rowIndex)`를 `return (columnIndex, 0)`으로 변경하면 어떻게 되는가?
3. 98행에 있는 `response == 'QUIT'`를 `response != 'QUIT'`로 변경하면 어떻게 되는가?
4. 72행에 있는 `tileChars = []`를 `tileChars = {}`로 바꾸면 어떤 에러 메시지가 나오는가?

PROJECT

#62

회전하는 큐브

회전하는 큐브 애니메이션

3차원 회전 및 라인 드로잉 알고리즘을 배운다

이번 프로젝트는 삼각 함수를 사용하여 회전하는 3차원 큐브 애니메이션을 구현한다. 여러분만의 애니메이션 프로그램을 만든다면 코드에 있는 3차원 포인트 회전 수식과 `line()` 함수를 조정하면 된다.

큐브를 그리는 데 사용되는 블록 텍스트 문자가 가는 직선으로 보이지 않겠지만, 물체 표현의 가장자리만 렌더링하기 때문에 이런 종류의 드로잉을 **와이어프레임 모델**wireframe model이라고 부른다. 그림 62-1은 큐브와 삼각형으로 이뤄진 거친 아이코스피어icosphere에 대한 와이어프레임 모델을 보여 준다.

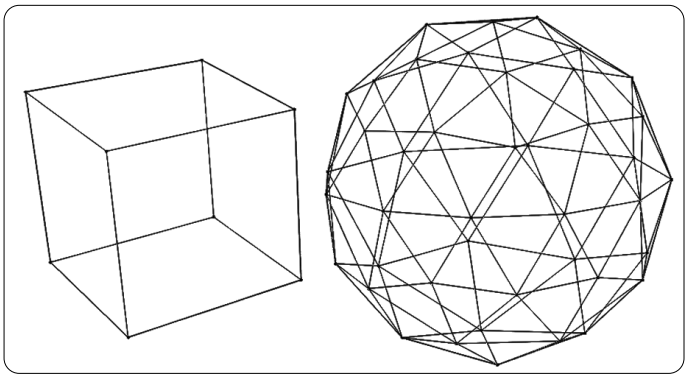


그림 62-1 큐브(왼쪽)와 아이코스피어(오른쪽)에 대한 와이어프레임 모델

프로그램 실행

그림 62-2는 **rotatingcube.py**를 실행한 결과다.

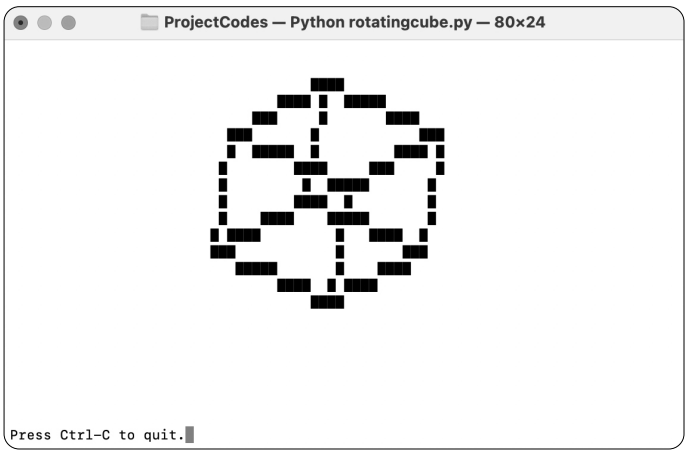


그림 62-2 프로그램이 화면에 그리는 와이어프레임 큐브

동작 원리

이번 프로그램의 알고리즘은 두 가지 주요 파트인 `line()` 함수와 `rotatePoint()` 함수로 구성된다. 큐브는 각 모서리마다 하나씩 총 8개의 꼭지점이 있다. 프로그램에서는 이들 꼭지점을 (x, y, z) 튜플로 `CUBE_CORNERS` 리스트에 저장한다. 또한, 이들 꼭지점은 큐브의 모서리 라인에 대한 연결을 정의한다. 모든 점들이 같은 방향을 동일한 양을 회전한다면, 큐브가 회전하는 것처럼 보이게 된다.

```
"""Rotating Cube, by Al Sweigart al@inventwithpython.com
A rotating cube animation. Press Ctrl-C to stop.
This code is available at https://nostarch.com/big-book-small-python-programming
Tags: large, artistic, math"""

# 이번 프로그램은 반드시 터미널 또는 명령 프롬프트 창에서 실행되어야 한다.

import math, time, sys, os

# 상수 설정하기:
PAUSE_AMOUNT = 0.1  # 일시 중지 시간은 10분의 1초다.
WIDTH, HEIGHT = 80, 24
SCALEX = (WIDTH - 4) // 8
SCALEY = (HEIGHT - 4) // 8
# 텍스트 셀의 길이는 폭의 두 배이므로, scaley를 설정한다:
SCALEY *= 2
TRANSLATEX = (WIDTH - 4) // 2
TRANSLATEY = (HEIGHT - 4) // 2

# (!) 이 값을 '#'이나 '*' 또는 다른 문자로 바꿔 보자:
LINE_CHAR = chr(9608)  # Character 9608은 꽉 찬 블록이다.

# (!) 단일 축에 따라 큐브를 회전하려면,
# 다음의 값들 중 2개를 0으로 설정하자:
X_ROTATE_SPEED = 0.03
Y_ROTATE_SPEED = 0.08
Z_ROTATE_SPEED = 0.13

# 이번 프로그램은 리스트에 XYZ 좌표를 저장한다.
# 인덱스 0에 X 좌표, 1에 Y 좌표, 2에 Z 좌표
# 이들 상수는 리스트의 좌표에 접근할 때 코드를 읽기 쉽게 해준다.
X = 0
Y = 1
Z = 2


def line(x1, y1, x2, y2):
    """주어진 점들 중, 라인에 있는 점들의 리스트를 반환한다.

    브레슨햄 라인 알고리즘을 사용한다.
    자세한 내용은 https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm을 참고하자."""
    points = []  # 라인의 위치를 담는다.
    # 'Steep'은 라인이 45도보다 크거나,
    # -45도보다 작다는 의미다:

    # 이 함수가 올바르게 처리하지 않는
    # 시작점과 끝점이 인접해 있는 특별한 경우에 대해 확인하여
    # 하드 코딩된 리스트를 반환한다:
    if (x1 == x2 and y1 == y2 + 1) or (y1 == y2 and x1 == x2 + 1):
        return [(x1, y1), (x2, y2)]

    isSteep = abs(y2 - y1) > abs(x2 - x1)
```

```
    if isSteep:
        # 이 알고리즘은 가파르지 않은 라인만 처리하므로,
        # 기울기를 비경사(non-steep)로 변경하고 나중에 다시 돌려놓는다.
        x1, y1 = y1, x1  # x1과 y1을 바꾼다.
        x2, y2 = y2, x2  # x2와 y2를 바꾼다.
    isReversed = x1 > x2  # 라인이 오른쪽에서 왼쪽으로 가는 경우 True

    if isReversed:  # 오른쪽에서 왼쪽으로 가는 라인의 점을 가져온다.
        x1, x2 = x2, x1  # x1과 x2를 바꾼다.
        y1, y2 = y2, y1  # y1과 y2를 바꾼다.

        deltax = x2 - x1
        deltay = abs(y2 - y1)
        extray = int(deltax / 2)
        currenty = y2
        if y1 < y2:
            ydirection = 1
        else:
            ydirection = -1
        # 이 라인의 모든 x에 대한 y를 계산한다:
        for currentx in range(x2, x1 - 1, -1):
            if isSteep:
                points.append((currenty, currentx))
            else:
                points.append((currentx, currenty))
            extray -= deltay
            if extray <= 0:  # extray <= 0인 경우에는 y만 변경한다.
                currenty -= ydirection
                extray += deltax
    else:  # 왼쪽에서 오른쪽으로 가는 라인의 점을 가져온다.
        deltax = x2 - x1
        deltay = abs(y2 - y1)
        extray = int(deltax / 2)
        currenty = y1
        if y1 < y2:
            ydirection = 1
        else:
            ydirection = -1
        # 이 라인의 모든 x에 대한 y를 계산한다:
        for currentx in range(x1, x2 + 1):
            if isSteep:
                points.append((currenty, currentx))
            else:
                points.append((currentx, currenty))
            extray -= deltay
            if extray < 0:  # extray < 0인 경우에는 y만 변경한다.
                currenty += ydirection
                extray += deltax
    return points


def rotatePoint(x, y, z, ax, ay, az):
```

```
    """회전된 x, y, z 인수의 (x, y, z) 튜플을 반환한다.

    회전은 ax, ay, az(라디언)에 의해
    0, 0, 0 원점을 중심으로 회전한다.
        각 축의 방향:
          -y
           |
           +-- +x
          /
        +z
    """

    # x 축을 중심으로 회전한다:
    rotatedX = x
    rotatedY = (y * math.cos(ax)) - (z * math.sin(ax))
    rotatedZ = (y * math.sin(ax)) + (z * math.cos(ax))
    x, y, z = rotatedX, rotatedY, rotatedZ

    # y 축을 중심으로 회전한다:
    rotatedX = (z * math.sin(ay)) + (x * math.cos(ay))
    rotatedY = y
    rotatedZ = (z * math.cos(ay)) - (x * math.sin(ay))
    x, y, z = rotatedX, rotatedY, rotatedZ

    # z 축을 중심으로 회전한다:
    rotatedX = (x * math.cos(az)) - (y * math.sin(az))
    rotatedY = (x * math.sin(az)) + (y * math.cos(az))
    rotatedZ = z

    return (rotatedX, rotatedY, rotatedZ)


def adjustPoint(point):
    """화면에 표시하기 위해 3차원 XYZ 포인트를 2차원 포인트로 조정한다.
    2차원 포인트의 크기를 SCALEX와 SCALEY 만큼 조정한 다음,
    TRANSLATEX와 TRANSLATEY 만큼 포인트를 이동한다."""
    return (int(point[X] * SCALEX + TRANSLATEX),
            int(point[Y] * SCALEY + TRANSLATEY))


"""CUBE_CORNERS는 정육면체 모서리의 XYZ 좌표를 저장한다.
CUBE_CORNERS의 각 모서리에 대한 인덱스는 다음 다이어그램에 표시된 것과 같다:
      0---1
     /|  /|
    2---3 |
    | 4-|-5
    |/  |/
    6---7"""
CUBE_CORNERS = [[-1, -1, -1], # 포인트 0
                [ 1, -1, -1], # 포인트 1
                [-1, -1,  1], # 포인트 2
                [ 1, -1,  1], # 포인트 3
```

```
                [-1,  1, -1], # 포인트 4
                [ 1,  1, -1], # 포인트 5
                [-1,  1,  1], # 포인트 6
                [ 1,  1,  1]] # 포인트 7
# rx, ry, rz만큼 회전한 다음,
# CUBE_CORNERS의 XYZ 좌표를 rotatedCorners에 저장한다:
rotatedCorners = [None, None, None, None, None, None, None, None]
# 각 축의 회전량:
xRotation = 0.0
yRotation = 0.0
zRotation = 0.0

try:
    while True:  # 메인 프로그램 루프
        # 각 양만큼 각 축에 따라 회전한다:
        xRotation += X_ROTATE_SPEED
        yRotation += Y_ROTATE_SPEED
        zRotation += Z_ROTATE_SPEED
        for i in range(len(CUBE_CORNERS)):
            x = CUBE_CORNERS[i][X]
            y = CUBE_CORNERS[i][Y]
            z = CUBE_CORNERS[i][Z]
            rotatedCorners[i] = rotatePoint(x, y, z, xRotation,
                yRotation, zRotation)

        # 큐브 라인의 점들을 얻는다:
        cubePoints = []
        for fromCornerIndex, toCornerIndex in ((0, 1), (1, 3), (3, 2), (2, 0),
        (0, 4), (1, 5), (2, 6), (3, 7), (4, 5), (5, 7), (7, 6), (6, 4)):
            fromX, fromY = adjustPoint(rotatedCorners[fromCornerIndex])
            toX, toY = adjustPoint(rotatedCorners[toCornerIndex])
            pointsOnLine = line(fromX, fromY, toX, toY)
            cubePoints.extend(pointsOnLine)

        # 중복된 점 제거하기:
        cubePoints = tuple(frozenset(cubePoints))

        # 화면에 큐브 표시하기:
        for y in range(HEIGHT):
            for x in range(WIDTH):
                if (x, y) in cubePoints:
                    # 전체 블록 표시하기:
                    print(LINE_CHAR, end='', flush=False)
                else:
                    # 빈 공간 표시하기:
                    print(' ', end='', flush=False)
            print(flush=False)
        print('Press Ctrl-C to quit.', end='', flush=True)

        time.sleep(PAUSE_AMOUNT)  # 잠깐 멈춤

        # 화면 정리하기:
```

```
208         if sys.platform == 'win32':
209             os.system('cls')  # 윈도우는 cls 명령어를 사용한다.
210         else:
211             os.system('clear')  # macOS와 리눅스는 clear 명령어를 사용한다.
212
213 except KeyboardInterrupt:
214     print('Rotating Cube, by Al Sweigart al@inventwithpython.com')
215     sys.exit()  # Ctrl-C를 누르면 프로그램을 종료한다.
```

소스 코드를 입력하고 여러 번 실행한 후, 실험을 위해 몇 가지를 변경해 보자. (!) 마크가 있는 주석은 여러분이 할 수 있는 간단한 변경에 대해 제안한 것이다. 다음 내용에 대해 스스로 방법을 찾아보자.

- CUBE_CORNERS와 184행에 있는 튜플을 수정하여 피라미드 또는 평면 육각형 등의 다양한 와이어프레임 모델을 생성하자.
- 큐브가 자신의 중심을 기준으로 회전하지 않고 화면의 중심을 기준으로 회전하도록 CUBE_CORNERS의 좌표를 1.5만큼 증가시키자.

프로그램 살펴보기

다음 질문에 대한 답을 찾아보자. 코드를 약간 수정하여 테스트하고, 변경 사항이 어떠한 영향을 미쳤는지 확인해 보자.

1. 208~211행의 코드를 삭제하거나 주석 처리하면 어떻게 되는가?
2. 184행에 있는 튜플을 <((0, 1), (1, 3), (3, 2), (2, 0), (0,4), (4, 5), (5, 1))>로 변경하면 어떻게 되는가?

PROJECT

#74

텍스트 음성 변환

여러분의 컴퓨터가 여러분에게 말하도록 만들자!

여러분의 운영 체제의 텍스트 음성 변환 엔진을 사용한다

이번 프로그램은 서드-파티 모듈인 `pyttsx3`를 사용하는 방법을 보여 준다. 여러분이 입력한 메시지를 운영 체제의 텍스트 음성 변환text-to-speech, TTS 기능에 의해 음성으로 전달된다. 컴퓨터가 만든 음성은 컴퓨터 과학 분야 중 매우 복잡한 것이지만, `pyttsx3` 모듈은 이에 대한 쉬운 인터페이스를 제공하여 이번의 작은 프로그램이 초보자에게도 적합할 수 있도록 만든다. 이 모듈을 사용하는 방법을 배운다면, 여러분의 프로그램에도 생성한 음성을 추가할 수 있게 될 것이다.

`pyttsx3` 모듈에 대한 자세한 내용은 https://pypi.org/project/pyttsx3/을 참고하자.

프로그램 실행

texttospeechtalker.py를 실행하면 다음과 같다.

```
Text To Speech Talker, by Al Sweigart al@inventwithpython.com
Text-to-speech using the pyttsx3 module, which in turn uses
the NSSpeechSynthesizer (on macOS), SAPI5 (on Windows), or
eSpeak (on Linux) speech engines.

Enter the text to speak, or QUIT to quit.
> Hello. My name is Guido van Robot.
<컴퓨터가 텍스트를 읽어 준다>
> quit
Thanks for playing!
```

동작 원리

이번 프로그램은 짧다. 왜냐하면 pyttsx3 모듈이 텍스트 음성 변환에 대한 모든 것을 처리해 주기 때문이다. 이 모듈을 사용하려면 이 책의 서문에 안내한 내용을 따라 설치하도록 하자. 이렇게 했다면 파이썬 스크립트는 `import pyttsx3`로 가져올 수 있으며, `pyttsx3.init()` 함수를 호출할 수 있다. 이 함수는 텍스트 음성 변환 엔진을 가리키는 Engine 객체를 반환한다. 이 객체는 `runAndWait()` 메서드를 실행할 때 컴퓨터가 말할 텍스트 문자열을 전달할 수 있는 `say()` 메서드를 가지고 있다.

```
"""Text To Speech Talker, by Al Sweigart al@inventwithpython.com
An example program using the text-to-speech features of the pyttsx3
module.
View this code at https://nostarch.com/big-book-small-python-projects
Tags: tiny, beginner"""

import sys

try:
    import pyttsx3
except ImportError:
    print('The pyttsx3 module needs to be installed to run this')
    print('program. On Windows, open a Command Prompt and run:')
    print('pip install pyttsx3')
    print('On macOS and Linux, open a Terminal and run:')
    print('pip3 install pyttsx3')
    sys.exit()

tts = pyttsx3.init()  # TTS 엔진을 초기화한다.
```

```
20
21 print('Text To Speech Talker, by Al Sweigart al@inventwithpython.com')
22 print('Text-to-speech using the pyttsx3 module, which in turn uses')
23 print('the NSSpeechSynthesizer (on macOS), SAPI5 (on Windows), or')
24 print('eSpeak (on Linux) speech engines.')
25 print()
26 print('Enter the text to speak, or QUIT to quit.')
27 while True:
28     text = input('> ')
29
30     if text.upper() == 'QUIT':
31         print('Thanks for playing!')
32         sys.exit()
33
34     tts.say(text)   # TTS 엔진이 말할 텍스트를 추가한다.
35     tts.runAndWait()   # TTS 엔진이 말하도록 한다.
```

프로그램 살펴보기

이것은 기본 프로그램이기 때문에 커스터마이징하는 옵션이 많지 않다. 그 대신, 여러분이 만든 다른 프로그램이 텍스트 음성 변환을 통해 어떤 이점을 얻을 수 있는지 생각해 보자.

PROJECT

#77

하노이 타워

고전적인 디스크 쌓기 퍼즐

퍼즐 상태를 시뮬레이션하기 위해 스택 데이터 구조를 사용한다

하노이 타워는 다양한 크기의 디스크를 쌓을 수 있는 3개의 기둥이 있는 스택 이동 퍼즐이다. 이 게임의 목표는 한쪽 타워에 있는 디스크를 다른 쪽 기둥으로 이동하는 것이다. 하지만 한 번에 단 하나의 디스크만 이동할 수 있으며, 작은 크기의 디스크 위에 그보다 큰 디스크를 올려 놓을 수 없다. 특정 패턴을 파악하면 이 퍼즐을 푸는 데 도움이 될 것이다. 그 패턴을 발견할 수 있겠는가?(힌트 더 쉬운 버전으로 먼저 풀어 보려면 `TOTAL_DISKS` 변수에 `3` 또는 `4`를 설정한다)

프로그램 실행

towerofhanoi.py를 실행하면 다음과 같다.

```
The Tower of Hanoi, by Al Sweigart al@inventwithpython.com

Move the tower of disks, one disk at a time, to another tower. Larger
disks cannot rest on top of a smaller disk.

More info at https://en.wikipedia.org/wiki/Tower_of_Hanoi

     ||          ||          ||
    @_1@         ||          ||
   @@_2@@        ||          ||
  @@@_3@@@       ||          ||
 @@@@_4@@@@      ||          ||
@@@@@_5@@@@@     ||          ||
      A           B           C

Enter the letters of "from" and "to" towers, or QUIT.
(e.g. AB to moves a disk from tower A to tower B.)
> ab
     ||          ||          ||
     ||          ||          ||
   @@_2@@        ||          ||
  @@@_3@@@       ||          ||
 @@@@_4@@@@      ||          ||
@@@@@_5@@@@@    @_1@         ||
      A           B           C

Enter the letters of "from" and "to" towers, or QUIT.
(e.g. AB to moves a disk from tower A to tower B.)
--중략--
```

동작 원리

타워를 나타내는 데이터 구조는 정수 리스트다. 각 정수는 디스크의 크기다. 이 리스트의 첫 번째 정수는 맨 아래 디스크를 나타내며, 마지막 정수는 맨 위의 디스크를 나타낸다. 예를 들어, [5, 4, 2]는 다음의 타워를 나타낸다.

```
     ||
     ||
   @@_2@@
 @@@@_4@@@@
@@@@@_5@@@@@
```

파이썬의 append()와 pop() 리스트 메서드는 리스트의 끝에 값을 추가하거나 제거할 수 있다. someList[0]과 someList[1]은 리스트의 첫 번째와 두 번째 값에 접근할 수 있는 것처럼, 파이썬은 음수 인덱스를 사용하여 리스트의 끝에 있는 값을 접근할 수 있게 해준다. 예를 들어, someList[-1]과 someList[-2]는 리스트의 마지막 값과 마지막에서 두 번째 값을 접근한다. 이것은 현재 상태의 타워에 맨 위에 있는 디스크를 찾는 데 유용하다.

```
"""The Tower of Hanoi, by Al Sweigart al@inventwithpython.com
A stack-moving puzzle game.
This code is available at https://nostarch.com/big-book-small-python-programming
Tags: short, game, puzzle"""

import copy
import sys

TOTAL_DISKS = 5  # 더 많은 디스크는 퍼즐의 난이도가 높아진다는 의미다.

# 모든 디스크가 타워 A에 있는 것으로 시작한다:
COMPLETE_TOWER = list(range(TOTAL_DISKS, 0, -1))


def main():
    print("""The Tower of Hanoi, by Al Sweigart al@inventwithpython.com

Move the tower of disks, one disk at a time, to another tower. Larger
disks cannot rest on top of a smaller disk.

More info at https://en.wikipedia.org/wiki/Tower_of_Hanoi
"""
    )

    # 타워 설정하기. 리스트의 끝은 타워의 맨 위다.
    towers = {'A': copy.copy(COMPLETE_TOWER), 'B': [], 'C': []}

    while True:  # 한 턴을 실행한다.
        # 타워와 디스크를 표시한다:
        displayTowers(towers)

        # 사용자에게 움직임 요청하기:
        fromTower, toTower = askForPlayerMove(towers)

        # fromTower에서 toTower로 맨 위의 디스크를 옮긴다:
        disk = towers[fromTower].pop()
        towers[toTower].append(disk)

        # 사용자가 퍼즐을 풀었는지 확인한다:
        if COMPLETE_TOWER in (towers['B'], towers['C']):
            displayTowers(towers)  # 마지막으로 타워를 표시한다.
            print('You have solved the puzzle! Well done!')
            sys.exit()
```

```
44
45
46 def askForPlayerMove(towers):
47     """플레이어에게 이동을 요청하고 (fromTower, toTower)를 반환한다."""
48
49     while True:  # 유효한 움직임을 입력할 때까지 플레이어에게 계속 요청한다.
50         print('Enter the letters of "from" and "to" towers, or QUIT.')
51         print('(e.g. AB to moves a disk from tower A to tower B.)')
52         response = input('> ').upper().strip()
53
54         if response == 'QUIT':
55             print('Thanks for playing!')
56             sys.exit()
57
58         # 사용자가 유효한 타워 문자를 입력했는지 확인한다:
59         if response not in ('AB', 'AC', 'BA', 'BC', 'CA', 'CB'):
60             print('Enter one of AB, AC, BA, BC, CA, or CB.')
61             continue  # 플레이어에게 다시 이동을 요청한다.
62
63         # 신택틱 슈거(Syntactic sugar) - 더 짧은 변수명 사용:
64         fromTower, toTower = response[0], response[1]
65
66         if len(towers[fromTower]) == 0:
67             # 'from' 타워는 빈 타워일 순 없다:
68             print('You selected a tower with no disks.')
69             continue  # 플레이어에게 다시 이동을 요청한다.
70         elif len(towers[toTower]) == 0:
71             # 어떤 디스크도 비어 있는 'to' 타워로 이동할 수 없다:
72             return fromTower, toTower
73         elif towers[toTower][-1] < towers[fromTower][-1]:
74             print('Can\'t put larger disks on top of smaller ones.')
75             continue  # 플레이어에게 다시 이동을 요청한다.
76         else:
77             # 유효한 이동이므로 선택한 타워를 반환한다:
78             return fromTower, toTower
79
80
81 def displayTowers(towers):
82     """현재 상태를 표시한다."""
83
84     # 3개의 타워를 표시한다:
85     for level in range(TOTAL_DISKS, -1, -1):
86         for tower in (towers['A'], towers['B'], towers['C']):
87             if level >= len(tower):
88                 displayDisk(0)  # 디스크가 없는 기둥 표시한다.
89             else:
90                 displayDisk(tower[level])  # 디스크를 표시한다.
91         print()
92
93     # 타워 레이블 A, B, C를 표시한다.
94     emptySpace = ' ' * (TOTAL_DISKS)
95     print('{0} A{0}{0} B{0}{0} C\n'.format(emptySpace))
```

```
 96
 97
 98 def displayDisk(width):
 99     """주어진 너비의 디스크를 표시한다. 너비가 0이면 디스크가 없다는 의미다."""
100     emptySpace = ' ' * (TOTAL_DISKS - width)
101
102     if width == 0:
103         # 디스크 없는 기둥을 표시한다.
104         print(emptySpace + '||' + emptySpace, end='')
105     else:
106         # 디스크를 표시한다:
107         disk = '@' * width
108         numLabel = str(width).rjust(2, '_')
109         print(emptySpace + disk + numLabel + disk + emptySpace, end='')
110
111
112 # 이 프로그램이 다른 프로그램에 임포트(import)된 게 아니라면 게임이 실행된다:
113 if __name__ == '__main__':
114     main()
```

프로그램 살펴보기

다음 질문에 대한 답을 찾아보자. 코드를 약간 수정하여 테스트하고, 변경 사항이 어떠한 영향을 미쳤는지 확인해 보자.

1. 73, 74, 75행의 코드를 삭제하거나 주석 처리하면 어떻게 되는가?
2. 100행에 있는 `emptySpace = ' ' * (TOTAL_DISKS - width)`를 `emptySpace = ' '`로 바꾸면 어떻게 되는가?
3. 102행에 있는 `width == 0`을 `width != 0`으로 바꾸면 어떻게 되는가?

APPENDIX

A

태그 색인

이 책의 프로젝트는 프로그램의 유형을 설명하는 태그들로 표시되어 있다. 첫 번째 태그는 크기를 나타낸다. 아주 작음(tiny, 1~63행), 짧음(short, 64~127행), 큼(large, 128~255행), 매우 큼(extra-large, 256행 이상)이다. 사이즈 태그는 다음과 같다.

- **아주 작음(tiny)**: #3 비트맵 메시지, #7 카이사르 해커, #12 콜라츠 추측, #14 카운트다운, #15 깊은 동굴, #16 다이아몬드, #19 디지털 시계, #20 디지털 스트림, #24 인수 파인더, #25 패스트 드로우, #31 숫자 맞추기, #32 속이기, #35 헥사 그리드, #40 리트 스피크, #42 매직 포춘 볼, #46 백만 번 주사위 굴림에 대한 통계 시뮬레이터, #49 곱셈표, #50 Ninety-Nine Bottles, #52 진법 카운터, #56 소수, #57 프로그레스 바, #58 무지개, #60 가위 바위 보 (항상 이기는 버전), #61 ROT13 암호, #65 빛나는 카펫, #67 사인 메시지, #72 스펀지 표기법, #74 텍스트 음성 변환
- **짧음(short)**: #1 베이글, #2 생일 역설, #5 돌아다니는 DVD 로고, #6 카이사르 암호, #8 캘린더 메이커, #10 쵸우한, #13 콘웨이의 라이프 게임, #18 주사위 굴리기, #21 DNA 시각화, #26 피보나치, #29 산불 시뮬레이션, #51 niNety nniinE BoOttels, #53 원소 주기율표, #54 피그 라틴, #55 파워볼 복권, #59 가위 바위 보, #64 7 세그먼트 디스플레이 모듈, #66 간단한 치환 암호, #69 달팽이 경주, #71 사운드 흉내, #76 틱-택-토, #77 하노이 타워, #80 비즈네르 암호
- **큼(large)**: #4 블랙잭, #9 상자 속 당근, #11 낚시성 기사 제목 생성기, #17 주사위 계산, #22 오리, #23 에칭 그림판, #28 플로더, #30 FOUR-IN-A-ROW, #33 해킹 미니 게임, #34 행맨과 기요틴, #36 모래시계, #37 굶주린 로봇, #39 랭턴의 개미, #41 럭키 스타, #43 만 칼라, #44 메이즈 러너 2D, #47 몬드리안 아트 생성기, #48 몬티 홀 문제, #62 회전하는 큐브, #63 우르의 로열 게임, #68 슬라이딩 타일 퍼즐, #70 소로반, 일본 주판, #73 스도쿠 퍼즐, #75 3-카드 몬테, #78 함정이 있는 질문, #79 2048, #81 물통 퍼즐
- **매우 큼(extra-large)**: #27 수족관, #38 J'ACCUSE!, #45 메이즈 러너 3D

나머지 태그는 프로그램의 기능을 가리킨다.

- **예술적**: #3 비트맵 메시지, #5 돌아다니는 DVD 로고, #13 콘웨이의 라이프 게임, #14 카운트다운, #15 깊은 동굴, #16 다이아몬드, #17 주사위 계산, #19 디지털 시계, #20 디지털 스트림, #21 DNA 시각화, #22 오리, #23 에칭 그림판, #27 수족관, #33 해킹 미니 게임, #35 헥사 그리드, #36 모래시계, #39 랭턴의 개미, #45 메이즈 러너 3D, #47 몬드리안 아트 생성기, #58 무지개, #62 회전하는 큐브, #65 빛나는 카펫, #67 사인 메시지, #69 달팽이 경주, #70 소로반, 일본 주판
- **초보자**: #3 비트맵 메시지, #6 카이사르 암호, #7 카이사르 해커, #9 상자 속 당근, #10 쵸우한, #11 낚시성 기사 제목 생성기, #12 콜라츠 추측, #15 깊은 동굴, #16 다이아몬드, #20 디지털 스트림, #24 인수 파인더, #25 패스트 드로우, #31 숫자 맞추기, #32 속이기, #35

헥사 그리드, #40 리트 스피크, #42 매직 포춘 볼, #46 백만 번 주사위 굴림에 대한 통계 시뮬레이터, #49 곱셈표, #50 Ninety-Nine Bottles, #58 무지개, #65 빛나는 카펫, #69 달팽이 경주, #71 사운드 흉내, #72 스펀지 표기법, #74 텍스트 음성 변환

- **bext**: #5 돌아다니는 DVD 로고, #27 수족관, #28 플로더, #29 산불 시뮬레이션, #36 모래시계, #39 랭턴의 개미, #47 몬드리안 아트 생성기, #58 무지개
- **보드 게임**: #30 FOUR-IN-A-ROW, #43 만칼라, #63 우르의 로열 게임, #76 틱-택-토
- **카드 게임**: #4 블랙잭, #75 3-카드 몬테
- **암호화**: #6 카이사르 암호, #7 카이사르 해커, #61 ROT13 암호, #66 간단한 치환 암호, #80 비즈네르 암호
- **게임**: #1 베이글, #4 블랙잭, #9 상자 속 당근, #10 쵸우한, #17 주사위 계산, #25 패스트 드로우, #28 플로더, #30 FOUR-IN-A-ROW, #31 숫자 맞추기, #33 해킹 미니 게임, #34 행맨과 기요틴, #37 굶주린 로봇, #38 J'ACCUSE!, #41 럭키 스타, #43 만칼라, #44 메이즈 러너 2D, #45 메이즈 러너 3D, #48 몬티 홀 문제, #59 가위 바위 보, #60 가위 바위 보 (항상 이기는 버전), #63 우르의 로열 게임, #68 슬라이딩 타일 퍼즐, #69 달팽이 경주, #71 사운드 흉내, #73 스도쿠 퍼즐, #75 3-카드 몬테, #76 틱-택-토, #77 하노이 타워, #79 2048, #81 물통 퍼즐
- **유머**: #11 낚시성 기사 제목 생성기, #32 속이기, #38 J'ACCUSE!, #42 매직 포춘 볼, #55 파워볼 복권, #60 가위 바위 보 (항상 이기는 버전), #78 함정이 있는 질문
- **수학**: #2 생일 역설, #6 카이사르 암호, #7 카이사르 해커, #12 콜라츠 추측, #17 주사위 계산, #24 인수 파인더, #26 피보나치, #46 백만 번 주사위 굴림에 대한 통계 시뮬레이터, #48 몬티 홀 문제, #49 곱셈표, #52 진법 카운터, #56 소수, #62 회전하는 큐브, #66 간단한 치환 암호, #70 소로반, 일본 주판, #80 비즈네르 암호, #81 물통 퍼즐
- **미로**: #44 메이즈 러너 2D, #45 메이즈 러너 3D
- **모듈**: #57 프로그레스 바, #64 7 세그먼트 디스플레이 모듈
- **멀티플레이어**: #41 럭키 스타, #69 달팽이 경주
- **객체지향**: #22 오리, #73 스도쿠 퍼즐
- **퍼즐**: #1 베이글, #33 해킹 미니 게임, #34 행맨과 기요틴, #38 J'ACCUSE!, #68 슬라이딩 타일 퍼즐, #73 스도쿠 퍼즐, #77 하노이 타워, #79 2048, #81 물통 퍼즐
- **과학**: #21 DNA 시각화, #53 원소 주기율표

- **과학**: #21 DNA 시각화, #53 원소 주기율표
- **스크롤**: #15 깊은 동굴, #20 디지털 스트림, #21 DNA 시각화, #22 오리, #50 Ninety-Nine Bottles, #51 niNety nniinE BoOttels, #56 소수, #58 무지개
- **시뮬레이션**: #2 생일 역설, #13 콘웨이의 라이프 게임, #18 주사위 굴리기, #29 산불 시뮬레이션, #36 모래시계, #39 랭턴의 개미, #46 백만 번 주사위 굴림에 대한 통계 시뮬레이터, #48 몬티 홀 문제, #55 파워볼 복권, #70 소로반, 일본 주판
- **2인용**: #9 상자 속 당근, #30 FOUR-IN-A-ROW, #43 만칼라, #63 우르의 로열 게임, #76 틱-택-토
- **단어**: #11 낚시성 기사 제목 생성기, #34 행맨과 기요틴, #40 리트 스피크, #51 niNety nniinE BoOttels, #54 피그 라틴, #72 스펀지 표기법

APPENDIX

B

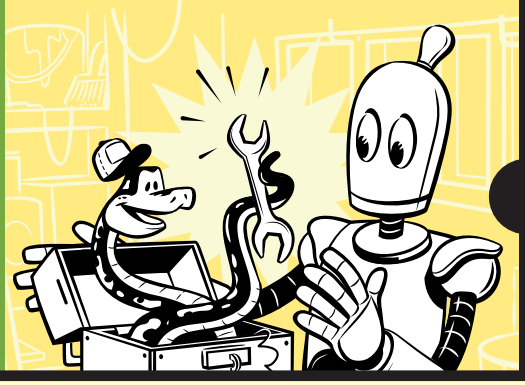
문자 맵

print() 함수는 키보드로 입력할 수 있는 모든 문자를 화면에 쉽게 표시해 준다. 하지만 키보드의 문자 외에도 표시하고 싶은 다른 문자들이 많을 것이다(하트, 다이아몬드, 클로버, 스페이드 카드 모양, 선, 음영 처리된 상자, 화살표, 음악 음표 등). 유니코드 코드 포인트라고 불리는 숫자 코드를 chr() 함수에 전달하면 이러한 문자의 문자열 값을 얻을 수 있다. 텍스트는 컴퓨터에 일련의 숫자로 저장되며, 각 문자는 서로 다른 숫자로 표시된다. 이번 부록에는 이러한 코드 포인트의 목록이 포함되어 있다.

chr()와 ord() 함수 사용하기

파이썬의 내장 함수인 chr()는 정수 인수를 받아서 그 숫자의 문자에 대한 문자열을 반환하다. ord() 함수는 반대로 단일 문자의 문자열 인수를 받아서 그 문자의 숫자를 반환한다. 이 숫자는

파이썬
3.10+ 기준



단계적 튜토리얼에 질린
코딩 초보자를 위한
창의적 프로젝트

창의력을 최대로 끌어올려 80개 이상의 재미있는 프로그램을 가능한 한 적은 코드로 손쉽게 구현하는 방법을 배울 수 있습니다. 게임, 시뮬레이션, 그리고 디지털 아트 등의 다양한 실습 예제를 통해 프로그래밍 개념이 어떻게 적용되는지를 직접 확인해 보세요.

파이썬 기본 문법을 익히고 프로그램을 작성할 준비가 되었다면 이 책을 통해 깨달음과 재미를 동시에 느낄 수 있습니다. 디지털 아트, 게임, 애니메이션, 계수 프로그램 등 81개의 파이썬 프로그램을 만들면서 코드가 어떻게 작동하는지를 알고 나면, 여러분의 아이디어를 반영해 프로그램을 변경하는 새로운 시도를 하게 됩니다.

이 단순한 텍스트 기반 프로그램들은 256줄 이하의 코드로 구성되어 있으며, 빈티지 화면 보호기, 달팽이 경주 게임, 클릭비트 헤드라인 생성기, DNA 애니메이션 등의 각 프로젝트는 온라인에서 쉽게 공유할 수 있도록 설계되어 있습니다. 또한, 이것들은 단순한 코드 스니펫이 아니라 실행 가능한 전체 소스가 제공되는 완전한 파이썬 프로그램입니다.

코드 동작 방식에 익숙해지고 나면 여러분의 생각을 반영하여 실제 프로그램에서 구현해 보세요. 그러다 보면 프로그램에 대한 개념을 제대로 이해하기 시작하고, 더 중요한 것은 프로그램을 어떻게 만들어야 하는지도 알 수 있을 것입니다.

주요 내용

- 행맨, 블랙잭 및 기타 게임으로 친구나 컴퓨터와의 대결
- 산불 모의실험, 주사위 백만 개 굴리기, 일본 주판 시뮬레이션
- 가상 어항, 회전 큐브 및 돌아다니는 DVD 로고 화면 보호기와 같은 애니메이션
- 1인칭 3D 미로 게임
- ROT13 및 비즈네르와 같은 암호를 사용하여 텍스트를 숨기는 암호화 프로그램

no starch press

난이도

분 야 프로그래밍 / 파이썬

정가 28,000 원

93000

9 791191 600667

ISBN 979-11-91600-66-7

Jpub 제이펍 "○과 ▮이 꿈꾸는 세상" www.jpub.kr

아름다운재단 The Beautiful Foundation 출판 수익 및 역자 번역료의 일부는 아름다운재단의 사회영역기금에 기부됩니다.