파이써닉한 파이썬을 익히는 간결한 안내서
Python Distilled

Python Distilled
by David M. Beazley

파이써닉한 파이썬을 익히는 간결한 안내서

전자책 1쇄 발행 2022년 8월 25일 **지은이** 데이비드 M. 비즐리 **옮긴이** 송현제 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트 **편집** 신승준 **본문 디자인** 성은경 **등록번호** 제2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울특별시 마포구 연남로5길 19-5 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** http://blog.insightbook.co.kr **이메일** insight@insightbook.co.kr **ISBN** 978-89-6626-370-7

프로그래밍 인사이트

파이써닉한 파이썬을 익히는 간결한 안내서

데이비드 M. 비즐리 지음 | 송현제 옮김

인사이트

차 례

옮긴이의 글

《파이썬 완벽 가이드》(인사이트, 2012)를 번역하고 약 10년이 지났습니다. 10년 전과 달리 지금 파이썬은 컴퓨터 분야뿐 아니라 수학, 데이터 과학 등 거의 모든 도메인에서 널리 쓰이는 언어가 되었습니다. 다양한 파이썬 책들이 지금도 출판되고 있고, 온라인에서는 파이썬 공식 문서 및 개발자들이 정리한 자료들을 손쉽게 찾을 수 있습니다.

이러한 정보의 홍수 속에서 역자의 아쉬운 점은 파이썬의 핵심 내용을 다시 살펴보고자 할 때 마땅한 자료를 찾기가 힘들다는 점이었습니다. 마침 데이비드 비즐리가 새로운 책을 출판하였다는 소식을 들었고, 책의 제목과 목차를 본 후, 망설임 없이 책을 번역하기로 하였습니다. 역자와 같이 파이썬의 핵심 내용을 빠르게 파악하려는 독자들이 많을 것으로 생각했기 때문입니다.

《Python Distilled》라는 원제목에서도 유추할 수 있듯이, 저자 데이비드 비즐리는 파이썬이 다루고 있는 많은 내용을 이 책에 '정제'하듯 담으려고 하였습니다. 저자는 25년 이상 파이썬으로 코딩하고 강의하면서 그 누구보다도 파이썬을 가장 잘 파악하고 있는 사람 중의 하나입니다. 특유의 유머스러운 표현과 함께 최대한 그 내용을 간결하게 표현하려 노력하는 사람입니다. 따라서 이 책을 읽는 독자는 재밌게 읽으면서도 빠르게 파이썬의 핵심 내용을 파악하게 될 것입니다.

이 책은 다른 프로그래밍 언어를 능숙히 쓰는 독자가 파이썬을 배우고자 할 때 유용한 책입니다. 만약 코딩을 처음 배우는 독자라면 이 책의 핵심 내용을 파악하기가 만만치 않을 것입니다. 그래도 이 책으로 도전하려 한다면 처음부터 모든 내용을 하나도 빠짐없이 꼼꼼히 살펴보면서 읽기 바랍니다. 끝으로 역자는 파이썬을 어느 정도 접해본 경험이 있는 독자에게도 이 책을 권합니다. 잊고 있던 파이썬의 내용을 복습하는 것은 물론, 파이썬의 고급 기능에 대한 인사이트를 많이 얻게 될 것입니다. 궁극적으로 독자 모두 파이써닉한 코드를 작성하게 되리라 믿습니다.

이 책을 출판하기까지 많은 분이 도움을 주셨습니다. 가장 먼저, IT 분야의 좋은 책을 꾸준히 관심을 갖고 발간해 주시는 한기성 사장님께 감사의 말씀을 드립니다. 다음으로 독자 입장에서 초안을 검토해 주시고, 좋은 책으로 출판할 수 있게 편집해주신 신승준 님에게도 큰 감사를 드립니다. 또한, 이 책을 미리 검토해주신 리뷰어분들께도 감사드립니다. 마지막으로 번역 시작부터 끝까지 곁에서 도와준 아내, 김아영에게도 감사와 사랑의 말을 전합니다.

송현제 드림

지은이의 글

필자가 《Python Essential Reference(1st Edition)》를 저술한 지 20년 이상의 시간이 흘렀다. 당시 파이썬은 엄청 작은 언어였고, 표준 라이브러리에는 유용한 것들이 함께 딸려 있었다. 당시 파이썬 표준 라이브러리는 대부분 우리 두뇌가 받아들이기에 적절한 분량이었다. 《Python Essential Reference(1st Edition)》는 그런 시대를 반영했다. 무인도나 자신만의 비밀 공간에서 파이썬 코드를 작성할 수 있도록 손쉽게 들고 다닐 수 있는 작은 책이었다. 세 번의 후속 개정을 거쳐 나온 《Python Essential Reference(4th Edition)》(파이썬 완벽 가이드, 인사이트, 2012)는 간결하면서도 완전한 참고서가 되겠다는 이런 비전을 고수했다. 휴가 중에 파이썬으로 코딩할 생각이라면, 왜 이것을 사용하지 않겠는가?

마지막 에디션, 《파이썬 완벽 가이드》를 출판한 지 10년이 넘은 지금, 파이썬 세계는 많이 달라졌다. 파이썬은 더 이상 마이너 언어가 아니며 세계에서 가장 인기 있는 프로그래밍 언어 중 하나가 되었다. 또한 파이썬 프로그래머는 고급 편집기, IDE(Integrated Development Environment, 통합개발환경), 노트북, 웹 페이지 등에서 클릭이나 키보드 조작으로 풍부한 정보를 얻을 수 있다. 사실 독자가 원하는 참고 자료는 몇 번 키보드를 누르면 손쉽게 얻을 수 있으므로 참고서를 살펴볼 일이 거의 없을 것이다.

오히려 정보 검색이 쉬워지고 파이썬 세계가 확장되면서 또 다른 형태의 도전이 생겼다. 이제 막 배우기 시작했거나 새로운 문제를 해결하려면, 어디서부터 시작해야 할지 막막할 수 있다. 또한 언어의 핵심 그 자체와 다양한 도구가 제공하는 기능을 구별하기 어려울 수 있다. 이러한 문제들을 해결하고자 이 책을 출간하게 되었다.

《파이써닉한 파이썬을 익히는 간결한 안내서》는 파이썬 프로그래밍에 관한 책이다. 이 책은 파이썬의 모든 것을 문서화하지 않는다. 이 책의 초점은 파이썬 언어의 현대적이면서도 엄선된(정제된) 핵심을 제공하는 데 있다. 필자는 과학자, 엔지니어, 소프트웨어 전문가에게 파이썬을 수년 동안 가르치면서 그 핵심에 도달하게 되었다. 그렇지만 이는 소프트웨어 라이브러리를 작성하고, 파이썬

의 한계를 알아보기 위해 시험해보면서 가장 유용한 방법이 무엇인지 찾아내려는 행위의 결과물이기도 하다.

이 책의 대부분은 파이썬 프로그래밍 자체에 초점을 맞추었다. 여기에는 추상화 기술, 프로그램 구조, 데이터, 함수, 객체, 모듈 등이 포함된다. 이 주제들은 프로젝트 규모와 상관없이 파이썬으로 작업하는 프로그래머에게 도움이 될 것이다. IDE에서 쉽게 얻을 수 있는 자료들(함수 목록, 명령어 이름, 인수 등)은 이 책에서 생략하였다. 또한 편집기, IDE, 배포와 같이 빠르게 변화하는 파이썬 도구에 대해서도 언급하지 않기로 하였다.

논란의 여지가 있겠지만, 필자는 대규모 소프트웨어 프로젝트 관리와 관련된 언어 기능에 초점을 맞추지 않았다. 파이썬은 때때로 수백만 줄의 코드로 구성된 대규모의 중요 작업에 사용되기도 한다. 이러한 응용 프로그램에서는 특별한 도구, 설계, 기능이 필요하다. 또한, 대규모 프로젝트에서는 위원회나 회의를 거쳐 중요한 문제를 결정하기도 한다. 이 모든 내용을 이 작은 책에서 다루기에는 너무 벅차다. 하지만 좀 더 정직하게 말하면 필자는 파이썬을 그런 대규모 응용 프로그램을 작성하는 데 사용하지 않는다. 그렇다고 취미로 사용한다는 뜻은 아니다.

책을 쓸 때는 언어에서 계속 진화하는 기능을 담지 못하는 일이 늘 생기기 마련이다. 이 책을 쓸 당시 파이썬은 3.9 버전이었다. 따라서 구조적 패턴 매칭(structural pattern matching) 같은 후속 릴리스로 계획된 주요한 몇몇 추가 내용은 이 책에서 다루지 않는다. 이는 다음에 다른 기회에서 다룰 주제이다.

마지막으로, 필자가 꼭 하고 싶은 말은 프로그래밍은 재미있어야 한다는 것이다. 필자의 책을 읽고 독자들이 생산적인 파이썬 프로그래머가 되는 데 도움을 받을 뿐 아니라 사람들이 별을 탐험하고, 화성에서 헬리콥터를 조종하고, 뒤뜰에서 다람쥐에게 물대포를 쏜다든지 하는 데 파이썬을 사용해 보도록 영감을 얻는 마법이 일어나기를 바란다.

감사의 글

도움을 준 기술 감수자 Shawn Brown, Sophie Tabac, Pete Fein에게 감사의 말을 전한다. 과거 프로젝트부터 이번 프로젝트까지 오랜 기간 동안 편집해 준 Debra Williams Cauley에게도 감사의 말을 전한다. 필자의 수업을 들었던 많은 학생이 이 책에서 다루는 주제에 간접적으로 영향을 주었다. 마지막으로 격려와 사랑을 보내준 Paula, Thomas, Lewis에게 특별한 감사의 말을 전한다.

리뷰어의 글

김태윤 제약회사 연구원

파이썬 언어에 대한 정보는 인터넷에 모두 공개되어 있고 필요하면 언제든 찾아볼 수 있습니다. 하지만 파이썬 언어를 사용하는 데 진심이라면 이런 책 한 권 정도는 가지고 있는 편이 좋습니다. 이 책은 입문용이 아니라 현재 파이썬을 사용하고 있지만 파이썬의 모든 것을 기억하지는 못하는 대부분의 개발자가 읽기에 적합합니다. 우리는 모든 것을 다 기억하지도, 기억할 필요도 없습니다. 단지 코드를 작성하다가 필요할 때 꺼내서 읽어보고 파이썬다운 코드를 작성하면 됩니다. 그런 점에서 이 책에 대한 베타 리뷰는 제게 그동안 배운 개념들을 다시 한번 상기해보고 코딩 능력을 업그레이드하는 시간이었습니다. 파이썬을 이미 사용하지만, 더 깊게 파고들고 싶은 모든 분께 이 책을 추천합니다.

이요셉 솔루티스 실장

무인도에 단 한 권의 파이썬 책만 가져갈 수 있다면 반드시 가져갈 책입니다. 파이썬계에서 강의와 집필로 이름난 저자답게, 파이썬의 문법과 동작을 명쾌하게 정의해 줍니다. C 언어의 명저《The C Programming Language》와 서술 방식이 비슷하며, 감히 그에 못지않은 완성도의 책이라고 생각합니다. 오롯이 파이썬 언어 그 자체와 동작 구조를 알고 싶을 때 옆에 두고 다시 찾아볼 수 있는 핵심 레퍼런스로써, 많은 이에게 큰 도움이 되리라 생각됩니다. 파이썬 3.8에서 바뀐 내용까지 반영하고 있기 때문에, 자신의 실력이 초보, 중수, 고수 중 어디에 속하더라도 이 책을 통해 새로운 내용을 많이 배울 수 있을 것입니다.

장대혁 휴넷 인공지능교육연구소

파이썬에 대한 좋은 레퍼런스가 계속 늘어나고 있는 요즘, 이러한 중급 레벨 이상의 서적은 귀하고 그 가치가 높다고 생각합니다. 파이썬으로 개발하는 데이터 분석, AI, 백엔드 엔지니어 직무의 주니어 개발자들이 보면 좋을 내용을 담고 있습니다. 이 책은 본인이 개발하면서 막히는 부분을 다루고 있는 장을 직접 정주

행하면서 읽어보면 많은 도움이 될 것입니다. 파이썬으로 직접 모듈을 개발하거나, 코드 리팩토링을 하면서 고민하는 개발자에게 특히 도움이 되리라 예상합니다. 내용이 방대하기에 처음부터 다 보면 좋겠지만, 그것보다는 쿡북처럼 필요한 장을 그때그때 참조하면 좋을 것 같습니다.

장진우 카카오 개발자

이 책은 파이썬을 새로운 언어로 배우려는 개발자 또는 파이썬을 서브 언어로 사용하려는 개발자 모두에게 도움이 되는 책입니다. 400여 페이지의 적지 않은 분량의 내용이기에 무조건 한 번에 완독하기보다는, 책장에 두고 필요할 때마다 꺼내본다는 생각으로 활용하면 좋을 듯합니다. 처음 파이썬을 접하는 개발자라면 전반부 1~5 장의 기본 내용 파악만으로도 충분히 파이썬을 활용할 수 있습니다. 좀 더 고급 기능을 알아보거나, 일반 파이썬 프로그램에서는 잘 사용하지 않는 클래스 내부의 스페셜 메서드와 내부 상속 메커니즘을 이해하거나, 각종 프레임워크 수준의 코드를 이해하고 싶다면, 그때 후반부를 참조해도 좋을 듯합니다.

책의 거의 절반이 7장 클래스와 8장 모듈에 할애되어 있는데, 개인적으로 이 부분이 매우 흥미롭고 도움이 많이 되었습니다. 이 파트만으로도 읽을 가치는 충분하다고 생각합니다. 추가로 좋았던 점은 내부 기능을 자세히 설명하면서도 파이썬다운 코드에 대한 설명을 놓치지 않는다는 점입니다. 복잡한 기능 설명에 독자가 답답해 할 때쯤 어떤 파이썬 기능이 주로 사용되고 사용되지 않는지 콕 짚어주기도 하고, 객체 지향 언어의 프로그래밍 패턴을 파이썬에 그대로 적용하는 실수를 범하지 않도록 친절히 안내하기도 합니다. 파이썬은 멀티패러다임 언어이기에 유연하면서도 간단한 파이써닉한 코드를 작성하는 게 매우 중요하기 때문이지요.

홍성민 GS 52g Studio 개발자

최신 파이썬(버전 3.8)의 기본 사용법부터 고급 기법까지 이 한 권은 파이썬과 관련된 거의 모든 내용을 담고 있습니다.

초보자라면 기본 사용법 위주로 살펴보고, 제너레이터와 비동기, 동적 클래스나 메타 클래스 같은 고급 주제는 기본 내용이 익숙해진 후에 참고하면 좋을 것 같습니다. 중급 이상의 개발자는 기본 내용을 전반적으로 빠르게 복습하면서 미처 알지 못했거나 간과했던 고급 주제를 좀 더 심도 있게 살펴보면, 파이썬으로

개발하거나 이미 개발한 제품을 업그레이드하는 새로운 아이디어를 얻는 데 도움을 받을 수 있겠습니다.

홍장유 엠포스 팀장

현재 파이썬 버전이 3.10.x까지 나왔지만, 주로 사용하는 3.8 이상 버전에서 새롭게 지원하는 파이써닉(Pythonic)한 기능들을 잘 설명하고 있어 최신 파이썬 코딩 기술을 익히려는 개발자에게는 좋은 가이드가 되겠다고 생각됩니다. 또한 파이썬을 새롭게 배우려는 초심자에게도 이 책을 모두 정독하고 습득한다면 중급 파이썬 개발자가 되기 위한 초석을 다질 수 있을 것입니다. 주제마다 예제가 풍부하여 따라 하다 보면 저절로 기능을 쉽게 이해할 수 있다는 게 이 책의 가장 큰 장점입니다.

1장

P Y T H O N D I S T I L L E D

파이썬 기초

이 장에서는 파이썬 언어의 핵심을 대략 살펴본다. 먼저 변수, 자료형, 표현식, 제어 흐름, 함수, 클래스, 입력/출력을 다룬 후, 모듈, 스크립트 작성, 패키지, 더 큰 프로그램 구성을 위해 필요한 몇 가지 팁으로 논의를 끝맺는다. 이 장에서는 모든 기능을 완벽히 다루지 않으며, 대규모 파이썬 프로젝트를 지원하는 데 필요한 도구 또한 설명하지 않는다. 그래도 경험이 많은 프로그래머는 여기에 실린 자료에서 더 나은 프로그램을 작성하기 위한 정보를 얻을 수 있어야 한다. 입문자는 터미널 창과 기본 텍스트 편집기 같은 간단한 환경에서 예제를 실행해 볼 것을 권한다.

1.1 파이썬 실행

파이썬 프로그램은 인터프리터에서 실행된다. 파이썬 인터프리터는 IDE(통합 개발환경), 브라우저, 터미널 등 다양한 환경에서 실행할 수 있지만, 무엇보다도 인터프리터의 핵심은 bash와 같은 명령 셸(shell)에서 `python`을 입력해 시작하는 텍스트 기반 응용 프로그램이다. 파이썬 2와 파이썬 3 둘 다 동일한 컴퓨터에 설치할 수 있으며, `python2` 또는 `python3`과 같이 버전을 선택할 수 있다. 이 책에서는 파이썬 3.8 버전을 사용한다.

인터프리터가 시작되면, '읽기-평가-출력(REPL) 루프'에 프로그램을 입력할 수 있는 프롬프트가 나타난다. 예를 들어 다음 출력 결과는 인터프리터에서 저작권 메시지를 출력한 다음, 친숙한 `Hello World` 명령을 `>>>` 프롬프트에 입력한 모습을 보여주고 있다.

```
Python 3.8.0 (default, Feb  3 2019, 05:53:21)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World')
Hello World
>>>
```

어떤 환경에서는 다른 형태의 프롬프트가 표시될 수 있다. 다음 출력은 ipython (파이썬을 위한 다른 셸)에서 얻어진 것이다.

```
Python 3.8.0 (default, Feb 4, 2019, 07:39:16)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.5.0 -- An enhanced Interactive Python. Type '?' for help.
In [1]: print('Hello World')
Hello World
In [2]:
```

이 코드에서 표시되는 출력 형태와는 상관없이 기본 원칙은 같다. 명령어를 입력하면 실행되고 즉시 결과를 알 수 있다.

파이썬의 대화식 모드는 유효한 문장들을 입력하고 그 결과를 바로 확인할 수 있는 매우 유용한 기능 가운데 하나다. 이것은 디버깅과 실험을 할 때도 유용하다. 필자를 포함한 많은 이는, 예를 들면 다음과 같이 파이썬의 대화식 모드를 탁상용 계산기처럼 사용한다.

```
>>> 6000 + 4523.50 + 134.25
10657.75
>>> _ + 8192.75
18850.5
>>>
```

파이썬을 대화식 모드로 사용하고 있을 때, 변수 _(언더스코어)는 최종 연산 결과를 담고 있다. 이 변수는 이어지는 문장에서 해당 연산 결과를 사용할 때 유용하게 쓰인다. 단, 이 변수는 대화식 모드에서만 정의되므로, 저장할 프로그램에서는 사용하면 안 된다.

quit() 또는 EOF(파일의 끝)를 입력하여 대화식 인터프리터를 종료할 수 있다. 유닉스에서 EOF는 Ctrl+D이며, 윈도우에서 EOF는 Ctrl+Z이다.

1.2 파이썬 프로그램

반복해서 실행할 프로그램을 만들고 싶다면, 텍스트 파일에 문장을 작성한다.

```
# hello.py
print('Hello World')
```

파이썬 소스 파일은 UTF-8로 인코딩된 텍스트 파일이며, 일반적으로 파일 확장자는 .py이다. # 문자는 줄(line) 끝까지 이어지는 주석을 의미한다. UTF-8 인코딩을 사용하는 한, 국제 문자(유니코드)는 소스 코드에서 자유롭게 사용할 수 있다. UTF-8 인코딩은 대부분 편집기에서 기본으로 설정되어 있지만, 확실하지 않다면 편집기 설정을 확인할 필요가 있다.

파일 hello.py를 실행하려면, 다음과 같이 파일 이름을 입력해 인터프리터를 실행하면 된다.

```
shell % python3 hello.py
Hello World
shell %
```

다음과 같이 #!를 사용하는 것이 일반적인데, 프로그램 첫째 줄에서 인터프리터를 지정하면 된다.

```
#!/usr/bin/env python3
print('Hello World')
```

유닉스에서는 파일에 실행 가능 권한(예: `chmod +x hello.py`)을 주어야만, 셸에서 `hello.py`를 입력했을 때 프로그램을 실행할 수 있다.

윈도우에서는 .py 파일을 더블 클릭하거나 윈도우 시작 메뉴의 [실행] 명령에서 프로그램 이름을 입력해 시작할 수 있다. #! 행이 있다면, 인터프리터 버전(파이썬 2 vs 파이썬 3)을 선택하는 데 사용된다. 프로그램은 프로그램이 완료되면 바로 사라지는 콘솔 창에서 실행되는데, 어떤 때는 출력 결과를 다 읽기도 전에 사라진다. 디버깅을 생각한다면 파이썬 개발 환경에서 프로그램을 실행하는 게 좋은 방법이다.

인터프리터는 입력 파일의 끝에 도달할 때까지 문장을 순서대로 실행한다. 그 시점에서 프로그램이 종료되고 파이썬도 종료된다.

1.3 기본 자료형, 변수 그리고 표현식

파이썬은 정수, 실수, 문자열과 같은 기본 타입(primitive types, 기본 자료형)[1]을 제공한다.

```
42              # 정수
4.2             # 실수
'forty-two'     # 문자열
True            # 불리언
```

변수(variable)는 값을 참조하고 있는 이름이다. 값은 특정 타입 객체를 나타낸다.

```
x = 42
```

때때로 다음과 같이 타입이 명시적으로 이름에 붙어 있는 것을 볼 수 있다.

```
x: int = 42
```

이때의 타입은 단지 코드의 가독성(code readability)을 높이려고 사용하는 힌트일 뿐이다. 서드파티(third-party) 코드 검사 도구에서 사용하며, 그 외는 무시된다. 즉, 나중에 다른 타입의 값을 할당해도 상관없다.

표현식은 값을 생성하기 위한 기본 타입, 이름, 연산자의 조합이다.

```
2 + 3 * 4       # -> 14
```

다음 프로그램은 변수와 표현식을 사용하여 복리를 계산한다.

```
# interest.py

principal = 1000          # 초기 금액
rate = 0.05               # 이자율
num_years = 5             # 햇수
year = 1
while year <= num_years:
    principal = principal * (1 + rate)
    print(year, principal)
    year += 1
```

1 (옮긴이) 타입은 데이터의 종류 또는 유형을 뜻한다. 자료형이라고도 한다. 이 책에서는 영어 낱말을 발음 그대로 표기한 타입을 같은 의미로 사용한다.

이 프로그램을 실행하면 다음 결과를 출력한다.

```
1 1050.0
2 1102.5
3 1157.625
4 1215.5062500000001
5 1276.2815625000003
```

while 문은 바로 이어 나오는 조건식을 평가한다. 평가한 조건식이 참이면 while 문의 본문이 실행된다. 그러고 나서 조건식이 다시 평가되고, 그 조건식이 거짓이 될 때까지 본문이 반복 실행된다. 들여쓰기로 루프의 본문을 표현하므로, interest.py에서 while 문에 이어지는 세 개의 문장은 반복할 때마다 실행된다. 파이썬은 블록 안에서 일관성만 있다면 들여쓰기를 얼마나 해야 하는지 규정하지 않는다. 각각의 들여쓰기 수준은 네 개의 공백을 사용하는 게 일반적이다.

interest.py 프로그램에서 한 가지 문제점은 출력이 예쁘지 않다는 점이다. 열을 오른쪽으로 정렬하고, 원금(principal)의 정밀도를 소수 두 자릿수로 제한하면 좀 더 보기가 좋을 듯하다. 이를 위해 f-문자열(f-string)을 사용하여 print() 함수를 다음과 같이 변경하자.

```
print(f'{year:>3d} {principal:0.2f}')
```

f-문자열에서 변수 이름과 표현식은 중괄호로 감싸 평가할 수 있다. 선택적으로 구성 요소마다 포매팅 지정자(formatting specifier)[2]를 첨부할 수 있다. '>3d'는 세 자리 십진수를 오른쪽 정렬하는 것을 의미한다. '0.2f'는 소수점 이하 두 자리의 정밀도를 가지는 부동 소수점 수를 의미한다. 포맷 코드와 관련한 자세한 내용은 9장에서 찾아볼 수 있다.

프로그램의 출력 결과는 다음과 같이 달라진다.

```
1 1050.00
2 1102.50
3 1157.62
4 1215.51
5 1276.28
```

2 (옮긴이) 포맷(format)을 우리말로는 서식, 형식으로 번역할 수 있다. 하지만 개발 현장에서는 포맷이라는 용어를 일상적으로 사용하고 있어 이 책에서는 포맷으로 통일한다.

1.4 산술 연산자

파이썬에는 표 1.1에서 보듯이 표준 수학 연산자가 있다. 이 연산자는 대부분 다른 프로그래밍 언어에서 동작하는 것과 의미가 같다.

표 1.1 산술 연산자

연산	설명
`x + y`	더하기
`x - y`	빼기
`x * y`	곱하기
`x / y`	나누기
`x // y`	끝수를 버리는 나누기
`x ** y`	제곱(x^y)
`x % y`	나머지(`x mod y`)
`-x`	단항 마이너스
`+x`	단항 플러스

나누기 연산자(`/`)는 정수에 적용될 때, 부동 소수점 수를 만든다. 말하자면, `7/4`는 `1.75`다. 끝수를 버리는 나누기(`//`, 바닥 나누기(floor division)라고 부른다) 연산자는 결과의 끝수를 버림으로써 정수를 만드는데, 모든 부동 소수점 수와 정수에서 동작한다. 나머지 연산자(`%`)는 `x // y` 연산 결과의 나머지를 반환한다. 예를 들어, `7 % 4`는 `3`이 된다. 부동 소수점 수의 경우, 나머지 연산자는 `x // y` 계산 결과의 나머지인 부동 소수점 수를 반환하는데, 그 값은 `x - (x // y) * y`와 같다.

표 1.2는 수치 연산에서 공통으로 사용하는 내장 함수의 일부이다.

표 1.2 일반 수학 함수

함수	설명
`abs(x)`	절댓값
`divmod(x, y)`	`(x // y, x % y)` 반환
`pow(x, y [, modulo])`	`(x ** y) % modulo` 반환
`round(x, [n])`	`10`의 `-n`승의 가장 가까운 수로 반올림

round() 함수는 '은행원식 반올림(banker's rounding)'을 수행한다. 은행원식 반올림이란 정수를 2로 나누어 소수점 수가 생기면, 가까운 짝수로 반올림하는 방법이다. 예를 들어, 0.5의 경우 0.0으로 반올림되며, 1.5는 2.0으로 반올림된다.

파이썬에서는 표 1.3과 같이 정숫값에 대해 비트 조작을 수행하는 내장 연산자들이 있다.

표 1.3 비트 조작 연산자

연산	설명
x << y	왼쪽 이동
x >> y	오른쪽 이동
x & y	비트 and
x | y	비트 or
x ^ y	비트 xor(exclusive or)
~x	비트 negation

비트 조작 연산자는 주로 이진수와 함께 사용한다. 다음은 그 예이다.

```
a = 0b11001001
mask = 0b11110000
x = (a & mask) >> 4   # x = 0b1100(12)
```

예제에서 0b11001001은 정숫값을 이진수로 쓰는 방법이다. 십진수 201 또는 16진수 0xc9로도 작성할 수 있지만, 비트를 조작할 때는 이진수를 사용하는 게 작업을 시각화하기 더 쉽다.

비트 연산자 동작 방식은 정수를 2의 보수 이진 표현(complement binary representation)으로 나타내고, 부호 비트가 왼쪽으로 무한히 확장된다고 가정한다. 하드웨어의 기본 정수에 매핑하기 위한 원시(raw) 비트 패턴으로 작업하는 경우, 약간의 주의가 필요하다. 파이썬은 비트를 버리거나 값의 오버플로(overflow)를 허용하지 않기 때문이다. 그 대신, 파이썬에서 결괏값은 시스템의 메모리가 허용하는 만큼 커질 수 있다. 결과의 크기를 확인하거나 필요에 따라 자르는 것은 개발자의 몫이다.

숫자를 비교하기 위해서는 표 1.4와 같이 비교 연산자를 사용하면 된다.

표 1.4 비교 연산자

연산	설명
`x == y`	~와 같은
`x != y`	~와 다른
`x < y`	~보다 작은
`x > y`	~보다 큰
`x >= y`	~보다 크거나 같은
`x <= y`	~보다 작거나 같은

비교 결과는 불리언(Boolean) 값으로, `True` 또는 `False`이다.

`and`, `or`, `not` 연산자(앞서 비트 조작 연산자와 혼동하지 말 것)는 보다 복잡한 불리언 표현식을 구성할 수 있다. 이 연산자의 기능은 표 1.5에서 볼 수 있다.

표 1.5 논리 연산자

연산	설명
`x or y`	x가 거짓이면 y를 반환. 그렇지 않으면 x를 반환
`x and y`	x가 거짓이면 x를 반환. 그렇지 않으면 y를 반환
`not x`	x가 거짓이면 `True`를 반환. 그렇지 않으면 `False`를 반환

`False`, `None`, 숫자 `0`, 빈 문자열은 거짓으로 간주한다. 그 외는 참으로 간주한다.

다음과 같이 값을 업데이트하는 표현식이 흔히 사용된다.

```
x = x + 1
y = y * n
```

이 표현식은 다음과 같이 단축 연산자(shorten operation)로 작성할 수 있다.

```
x += 1
y *= n
```

이 축약 형태의 업데이트는 `+`, `-`, `*`, `**`, `/`, `//`, `%`, `&`, `|`, `^`, `<<`, `>>` 연산자와 함께 사용할 수 있다. 파이썬은 일부 다른 언어에서 볼 수 있는 증가(`++`), 감소(`--`) 연산자가 없다.

1.5 조건식과 제어 흐름

while, if, else 문은 반복과 조건식 코드의 실행을 위해 사용된다. 다음은 그 예이다.

```
if a < b:
    print('Computer says Yes')
else:
    print('Computer says No')
```

if와 else 절의 본문은 들여쓰기로 표기하고, else 절은 생략할 수 있다. 해당 절에서 실행할 문장이 없다면 다음과 같이 pass 문을 사용한다.

```
if a < b:
    pass      # 아무 일도 안 함
else:
    print('Computer says No')
```

여러 개의 조건을 검사할 때는 다음과 같이 elif 문을 사용하면 된다.

```
if suffix == '.htm':
    content = 'text/html'
elif suffix == '.jpg':
    content = 'image/jpeg'
elif suffix == '.png':
    content = 'image/png'
else:
    raise RuntimeError(f'Unknown content type {suffix!r}')
```

조건 검사와 더불어 값을 할당할 경우, 조건부 표현식(conditional expression)을 사용한다.

```
maxval = a if a > b else b
```

이 코드는 다음과 같이 길게 쓴 코드와 동일하다.

```
if a > b:
    maxval = a
else:
    maxval = b
```

때때로 := 연산자를 사용하여 변수 대입과 조건부를 결합한 코드를 볼 수 있다. 이는 대입 표현식(assignment expression)이라 한다. 흔히 '바다코끼리 연산자

(walrus operator)'로 부르는데, 이는 := 연산자가 바다코끼리가 죽은 척하며 옆으로 넘어져 있는 것처럼 보이기 때문이다. 다음은 그 예이다.

```
x = 0
while (x := x + 1) < 10:  # 1, 2, 3, ..., 9를 출력
    print(x)
```

대입 표현식에서는 표현식을 감싸는 괄호가 필요하다.

break 문은 루프를 빠져나올 때 사용하며, 빠져나오는 것은 가장 안쪽 루프에만 적용된다. 다음은 그 예이다.

```
x = 0
while x < 10:
    if x == 5:
        break     # 루프가 중단되고, 루프를 빠져나간다.
    print(x)
    x += 1

print('Done')
```

continue 문은 루프 본문의 나머지를 건너뛰고, 루프의 맨 앞으로 돌아간다. 다음은 그 예이다.

```
x = 0
while x < 10:
    x += 1
    if x == 5:
        continue # print(x)를 건너뛴다. 루프의 시작점으로 돌아간다.
    print(x)

print('Done')
```

1.6 문자열

문자열 리터럴을 정의할 때는 다음과 같이 작은따옴표나 큰따옴표 또는 삼중따옴표로 둘러싼다.

```
a = 'Hello World'
b = "Python is groovy"
c = '''Computer says no.'''
d = """Computer still says no."""
```