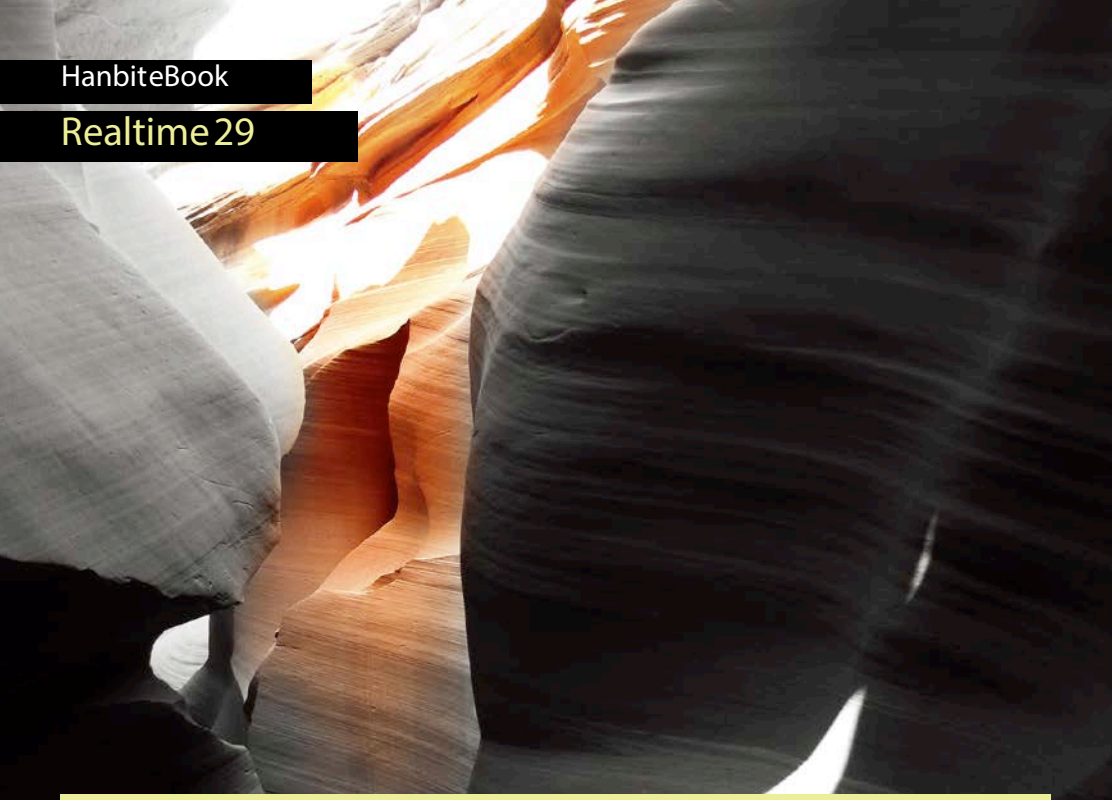


HanbiteBook

Realtime 29

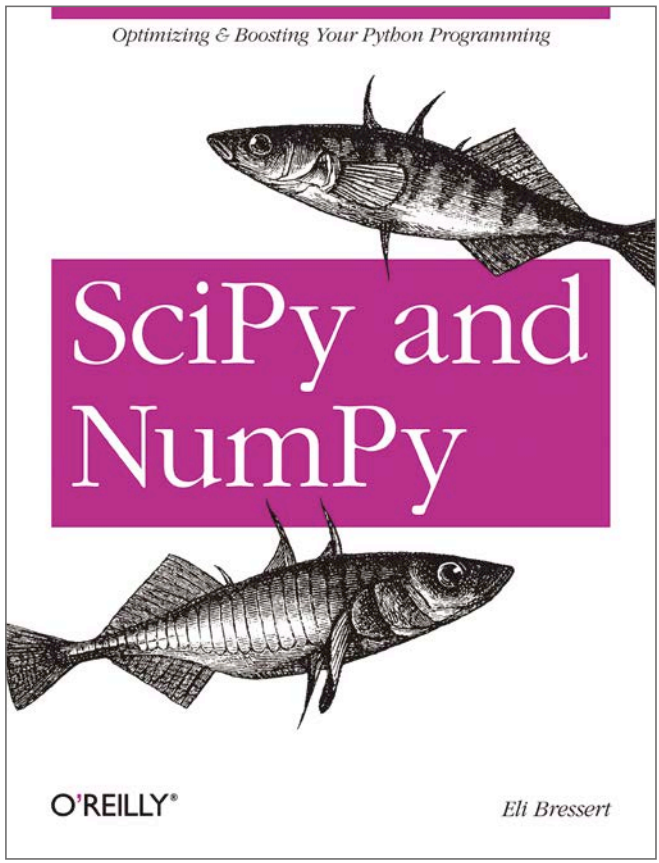
DATA SCIENCE

SciPy와 NumPy

데이터/수치 분석을 위한 파이썬 라이브러리

엘리 브레셜트 지음 / 이성주 옮김

O'REILLY® 한빛미디어 Hanbit Media, Inc.



이 도서는 O'REILLY의
SciPy and NumPy의
번역서입니다.

데이터/수치 분석을 위한 파이썬 라이브러리

SciPy와 NumPy

데이터/수치 분석을 위한 파이썬 라이브러리 **SciPy와 NumPy**

초판발행 2013년 5월 29일

지은이 엘리 브레설트 / **옮긴이** 이성주 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-613-5 15000 / **정가** 9,900원

책임편집 배용석 / **기획** 김창수 **편집** 북누리
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 북누리
마케팅 박상용, 박주훈

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanb.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 소개

지은이_ **엘리 브레설트**Eli Bressert

엘리 브레설트는 미국 애리조나 주 투손 출신이다. 미 항공 우주국에서 찬드라 X-선 우주망원경Chandra X-ray Space Telescope의 과학 이미지 처리 연구원science imager으로 일하면서 책, 신문, 텔레비전을 비롯한 여러 매체의 과학 관련 이미지를 다루었다. 이후에는 엑서터대학교University of Exeter에서 천체물리학 박사학위를 취득하고 현재는 호주 시드니의 CSIRO[01] 천문연구원CSIRO Astronomy and Space Science에서 볼턴 펠로우Bolton Fellow로 재임하고 있다. 최근 6년간 엘리는 파이썬으로 프로그래밍을 하면서 하버드대학교, 유럽천체우주센터European Space Astronomy Centre, 그리고 유럽남방천문대European Southern Observatory 등에서 파이썬 관련 강의를 해왔다. 천체물리학에서 많이 사용하는 파이썬 패키지인 ATpy와 APLpy[02] 의 공동 개발자다.

---

01 **역자주_** CSIRO(Commonwealth Scientific and Industrial Research Organisation, 공립과학산업연구소)는 호주의 국립과학기관으로 세계에서 가장 규모가 크며 다양한 연구를 수행한다.

02 **역자주_** ATpy는 천체물리학 데이터를 테이블 형식으로 만들 수 있게 해 주는 패키지며, APLpy는 데이터를 그래프로 출력하는 데 사용되는 패키지다.

역자 소개

역자_ 이성주

연세대학교 컴퓨터과학과 박사과정 재학 중이다. 연세대학교에서 전기전자공학을 전공했으며 미국 캘리포니아주립대에서 3학기 동안 교환학생으로 있었다. 연구분야는 동영상의 얼굴인식과 그래프 이론Graph Theory 등이다. SciPy와 NumPy 패키지를 연구에 주로 사용하고 있으며, 영상처리를 위해 OpenCV, 파이썬, C++를 같이 사용하고 있다. 파이썬 관련 강의 요청을 기다리면서 C++와 자바를 강의하고 있다. 안드로이드 관련 컨설팅을 수행하기도 했으며, 2012년에는 중소기업청 산하기관에서 소프트웨어 관련 기술 컨설팅을 수행했다.

저자 서문

고수준 언어high-level language[03]인 파이썬은 읽기 쉬운 문법으로 되어 있으면서 매우 유연하여, 배우고 사용하기에 더 없이 좋은 언어다. 과학과 연구개발에서는 개발 프로세스를 간소화하고 가능한 적은 단계로 목표를 달성하고자 몇 가지 추가 패키지를 사용하는데, 그중 으뜸은 SciPy와 NumPy이다. 이 책에는 과학용 패키지를 자들이 연구에 즉시 적용할 수 있도록 패키지에 포함된 도구의 차이점을 개략적으로 살펴볼 것이다.

SciPy와 NumPy는 수치적 배열과 고급 데이터 분석에서 없어서는 안 될 파이썬 패키지다. 이 패키지에 있는 도구의 사용법은 프로그래머의 생활을 좀 더 즐겁게 한다. 이 책은 간단한 배열 생성부터 기계학습을 위한 도구 사용법을 다룰 것이다.

저자

엘리 브레설트

---

03 **역자주_** 어셈블리 언어 등 어떤 특정 컴퓨터의 하드웨어 구조에 의존하는 수준을 벗어나 프로그램을 표현하는 언어에 대해 인간이 이해하기 쉽고 인간의 사고에 적합한 개념과 구조를 갖는 프로그램 언어다. 이 책에서는 원저자의 뜻을 따라 고수준 언어로 번역하지만 언어의 등급이 아닌 프로그래밍 영역을 뜻한다.

역자 서문

대학원에서 매트랩으로 실험을 진행하면서 C/C++, 자바와 같은 컴파일 언어에 비해 스크립트 형으로 작성해 나가는 프로그래밍 환경이 수학적 구상이 근간을 이루는 과학 계산에 훨씬 적합하다는 것을 느끼게 되었다. 하지만 매트랩 만으로는 부족했다. 연구에서는 데이터를 수집, 가공, 정렬하는 데 대부분의 시간이 소요된다. 매트랩은 수집된 데이터가 행렬로 잘 표현되어 있으면 그런 데이터에 대해 연산을 수행할 수 있지만 데이터를 수집하고 가공하는 데는 부족한 부분이 있기도 했다. 매트랩과 유사한 수준의 생산성을 유지하면서도 실행 속도가 빠르고, 공학적 확장성이 있는 객체 지향적 환경의 SciPy와 NumPy를 발견했다. 그 후로 이런 환경이 있음을 감탄하고 감사하며 사용하고 있다. 처음 SciPy와 NumPy를 접했을 2008년에는 문헌이 많지 않아 온라인에 흩어진 문서로 더듬더듬 익혀갔다. 그때 이런 책이 있었다면 훨씬 더 수월하게 익힐 수 있었으리라 생각한다. 오늘도 연구실에서 데이터와 씨름하는 분들께 이 책이 도움이 되길 바란다.

2013년 5월 이성주

대상 독자

초급 | 초중급 | 중급 | 중고급 | 고급

이 책은 파이썬에 대한 기본 지식 또는 그 이상을 갖고 있는 사람들을 대상으로 한다. SciPy와 NumPy가 고급 기능을 제공하지만, 사용 자체는 간단해 초보 파이썬 프로그래머도 따라할 수 있다.

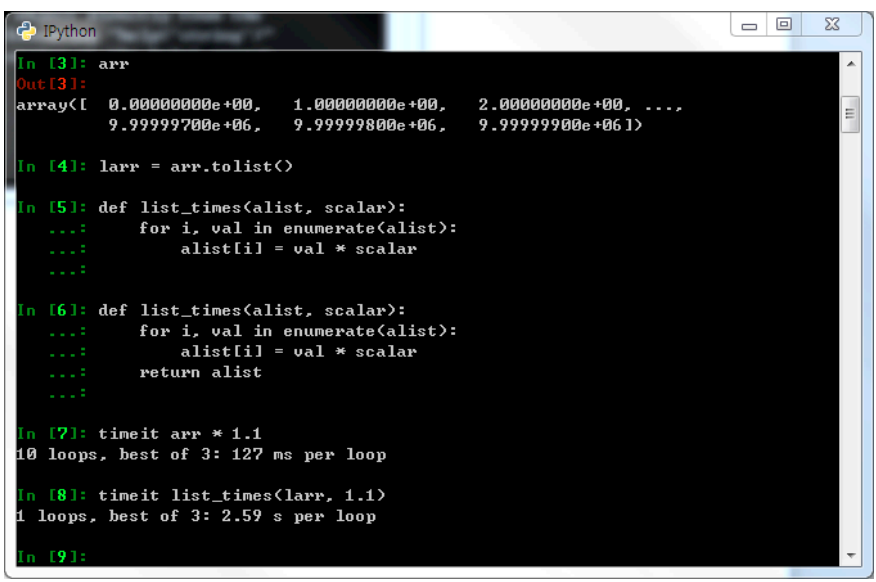
이 책은 SciPy와 NumPy의 기본적인 내용을 예제와 함께 다룬다. 첫 장은 SciPy와 NumPy 패키지가 무엇인지 살펴보고, 컴퓨터에 설치하는 법과 사용하는 법을 알아본다. 2장에서는 배열 생성부터 시작해 NumPy의 기본 내용을 다룬다. 3장은 책의 대부분을 차지하는데 여기서는 방대한 SciPy의 내용 중 일부를 다룬다(적분, 최적화, 보간법 등의 예제와 함께).예제 4장에서는 잘 알려진 scikit 패키지인 scikit-image와 scikit-learn 두 개를 살펴본다. 이 패키지들은 실제 문제들에 당장 적용할 수 있는 고급 기능을 제공한다. 5장 맺음말에는 책에서 다룬 내용 이상의 문제들을 다루기 위해 어떻게 해야 하는지에 대한 제언을 제공한다.

예제 테스트 환경

파이썬에서 기본적으로 제공하는 스크립트 셸Python Interactive Shell은 스크립트 환경에서 두 줄 이상의 파이썬 코드를 작성하기에 불편하다. 게다가 코드를 비롯한 필요한 파일들을 찾아 디렉터리를 이리저리 옮겨 다니기 마련인데, 이런 셸 환경은 그런 점에서 정말 불편하다.

존 헌터John D. Hunter 1968~2012는 기본 파이썬 셸의 한계를 극복하는 강력한 파이썬 셸 환경을 만들고자 했고, 그 결과물이 IPython이다. IPython은 초창기부터 많은 파이썬 사용자들에게 큰 지지를 받아오고 있어 SciPy와 NumPy 패키지를 사용하는 기본적인 환경인 경우가 많다.

IPython 웹 사이트: http://ipython.org/



IPython은 윈도우의 명령 프롬프트나 리눅스의 셸 환경처럼 파이썬 환경 내에 디렉터리를 변경할 수 있고, 탭으로 변수나 명령어 자동완성 기능도 제공한다. 그리고 무엇보다 단색의 무미건조한 기본 셸과는 달리 여러 색상을 표현해 좀 더 보기 좋고 가독성도 더 높다.

IPython 설치

윈도우 환경에서 IPython을 설치하려면, 아래 링크에서 해당되는 파일을 내려 받아 exe 파일을 실행한 뒤 설치 안내를 그대로 따라하면 간단하게 설치할 수 있다.

IPython 설치 파일 : https://github.com/ipython/ipython/downloads

설치는 간단한 IPython이지만 설치 파일을 얻을 수 있는 경로가 다양하고 복잡하다. 윈도우, 맥, 리눅스 환경을 모두 지원하는 파이썬 패키지 역시 여러 방식으로 설치가 가능하고 오픈 소스라는 점이 더해져 더욱 복잡하다.

가장 편리한 방법은 EPD Free를 내려 받아 설치하는 방법이다. 그런데 EPD Free는 IPython뿐 아니라 NumPy/SciPy를 비롯한 다양한 패키지를 포함하고 있다. 현재 설치된 패키지와 충돌이 있을 수 있으니, 파이썬을 새로 설치하는 경우가 아니라면 IPython만 독립적으로 설치하는 것이 좋다.

IPython 설치 파일명은 'IPython 버전-파이썬2 또는 3-운영체제 환경'의 형식으로 되어 있다.

ipython-0.13.1.py2-win32.exe (윈도우 32비트 파이썬 2의 IPython 0.13.1 버전)

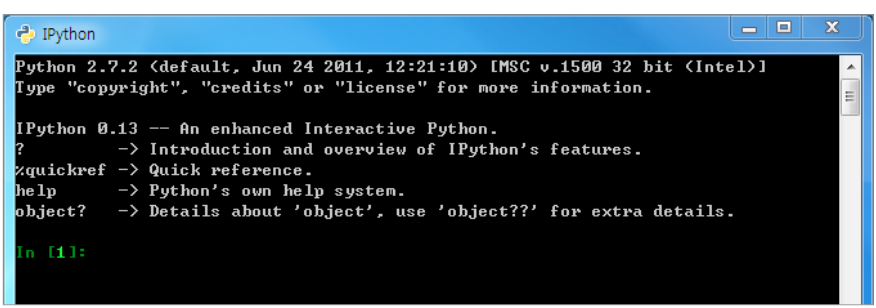
주의할 점은, 윈도우 64비트 환경라도 파이썬이 32비트라면 IPython 역시 32 비트를 설치해야 한다. 다른 모든 응용프로그램과 마찬가지로 윈도우 64비트 환경에서는 32비트 프로그램을 아무 문제없이 설치할 수 있지만, SciPy와 NumPy 패키지는 윈도우 64비트에 설치하기가 까다롭다(Enthought 사에서는 유료 서비스로 윈도우 64비트를 지원한다). 다른 많은 패키지들도 윈도우 64비트를 지원하지 않는 경우가 많다. 그러니 윈도우 환경에서는 파이썬 32비트 설치를 추천한다.

설치 후 해야 할 일

위의 설치 과정을 성공적으로 마치고 난 다음에 IPython을 실행시키면 시작이 안 되는 건 아닌데 오류 메시지가 발생 한다. IPython에서 색상을 표시하고 탭 자동완성 등의 기능을 사용하려면 'pyreadline'라는 패키지를 설치 해야 한다(단, 윈도우 환경에서만 이 패키지 설치가 필요하고, 리눅스 환경에선 설치 과정이 다르다).

```
〉pip install pyreadline
```

pyreadline 패키지를 설치한 다음 IPython 셸을 새로 실행시켜야 변경이 반영된다. 별다른 오류 메시지 없이, 다음과 같이 나오면 설치가 성공적으로 완료된 것이다.



예제 파일

이 책에서 사용된 예제 파일은 http://examples.oreilly.com/0636920020219/에서 내려 받을 수 있습니다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only: 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1 | 들어가며

파이썬은 이식성, 유연성, 간결한 문법과 스타일, 확장성이 뛰어난 강력한 프로그래밍 언어다. 파이썬은 귀도 반 로썸Guido van Rossum이 간결한 문법을 염두에 두고 만든 언어다. 함수의 정의와 반복문 등에서 중괄호('{}') 대신 들여쓰기를 사용한다. 그 결과 파이썬 코드는 주석의 도움이 없어도 프로그래머가 코드의 작동 방식과 의도를 빠르게 파악할 수 있다.

포트란이나 C와 같은 컴파일 언어는 보통 실행 속도가 파이썬보다 훨씬 빠르다.[01] 하지만 파이썬을 컴파일 언어와 함께 사용한다면 파이썬은 결코 느리지만은 않다. Cython[02]과 같은 패키지는 파이썬이 C 코드와 상호작용할 수 있도록 하여, C 프로그램과 파이썬이 메모리에서 정보를 주고받을 수 있도록 해준다. 이는 파이썬이 컴파일 언어와 호환되어 필요한 경우 실행 속도가 빠른 언어와 견줄만한 속도로 실행 가능하고, 레거시 코드도 활용할 수도 있다(FFTW 라이브러리는 레거시 코드를 사용하는 좋은 예다). 파이썬과 빠른 연산의 조합은 과학자를 비롯한 많은 사람들에게 매력적이다. 과학적 파이썬 프로그래밍의 대표적인 패키지는 SciPy와 NumPy이다. 이 두 패키지는 레거시 코드와 호환 및 연동을 용이하게 해준다.

1.1 왜 SciPy와 NumPy를 사용하는가?

과학적 프로그래밍에 사용되는 기본 연산에는 배열, 행렬, 적분, 미분방정식 연산, 통계 등이 있다. 파이썬은 행렬이나 배열이 아닌 일반적 변수에 대한 기초적 수학 연산을 제공하지만 기본적인 과학계산 기능은 내장되어 있지 않다. SciPy와 NumPy는 과학계산을 효율적으로 수행할 수 있는 강력한 파이썬 패키지다.

---

01 **역자주_** 파이썬은 프로그래머가 작성한 코드를 해석해 주는 해독기(interpreter)를 사용한다. 컴파일 언어보다 중간 단계가 많아 실행 속도 등에 영향을 미친다.

02 **역자주_** C 언어 패키지 호환성을 보다 강조한 다른 종류의 파이썬. http://www.cython.org/

NumPy는 다차원 ndarray를 사용한 수치 연산에 특화되어 있다. NumPy 행렬 ndarray는 브로드캐스팅broadcasting이라고 불리는 원소별 연산element-by-element이 가능하다. 필요한 경우, 브로드캐스팅을 사용하여 NumPy 배열을 특별히 조작하지 않고도 선형 대수적 연산[03]이 가능하며 배열의 크기가 동적으로 변한다. 이러한 특징은 다른 프로그래밍 언어로 하기 어려운 빠른 구현을 가능하게 해준다. 특정한 원소를 제거하고자 하는 경우에는 새 배열을 만들기 보다는 마스크를 적용할 수 있다.

SciPy는 NumPy 배열 프레임워크를 기반으로 만들어져 적분, 상미분방정식[04], 특수 함수, 최적화를 비롯한 다양한 고급 수학 함수들을 제공하여 과학적 프로그래밍을 완전히 다른 수준에서 수행할 수 있다. SciPy의 모든 함수를 나열하면 적어도 대여섯 쪽을 할애해야 한다. 수많은 SciPy의 함수들을 보면 때로는 어떤 함수를 써야 할지 결정하는 것조차 어려울 때가 있다. 이것이 이 책이 나온 이유다. 이 책을 통해 가장 기본적이고 널리 쓰이는 도구들을 살펴보면서 빠르게 결과를 얻을 수 있을 것이다. 또한, 실용적인 관점에서 SciPy와 NumPy 패키지를 탐구하여, 이 책의 범위를 넘어서는 문제에서도 적합한 도구를 선택하는 데에도 도움이 될 것이다.

1.2 SciPy와 NumPy 설치하기

이 책이 유용하리라 생각하고, "대단한데! 이 패키지를 어디서 구하고 어떻게 설치해야 하지?"라고 물을 것이다. 패키지를 설치하는 방법은 여러 가지다. OS X, 리눅스, 그리고 윈도우에서 설치하는 가장 쉬운 방법을 먼저 살펴보도록 한다.

---

03 **역자주_** 행렬의 조작 및 연산

04 **역자주_** 미지(未知) 함수가 독립변수를 한 개만 포함한 미분방정식이다. 함수가 도함수를 가지면 함수에 대해 상미분방정식이라 한다.

SciPy와 NumPy를 비롯해 다양한 기능을 탑재한 컴파일된 패키지로는 Enthought Python Distribution (EPD)[05]와 ActivePython (AP)[06] 등이 있다. 이 두 패키지는 OS X, 리눅스, 그리고 윈도우 운영체제 모두를 지원한다. 무료 버전을 설치하고 싶다면 'EPD Free' 또는 'AP Community Edition'을 내려 받아야 한다. 유료 버전으로 업그레이드를 하면 기술지원이 가능하다.

MacPorts3[07] 사용자라면 패키지 매니저를 통해 아래 MacPorts 명령어로 SciPy와 NumPy를 설치할 수 있는데, SciPy 패키지의 설치는 시간이 오래 걸린다.

```
sudo port install py27-NumPy py27-SciPy py27-ipython
```

MacPorts는 파이썬 2.6과 2.7 등 여러 버전을 지원한다. 위 명령어에는 'py27'이라고 했지만, 파이썬 2.6 버전으로 SciPy와 NumPy를 사용하고 싶다면 'py27'을 'py26'으로 바꾸면 된다.

우분투Ubuntu 또는 민트 리눅스Mint Linux와 같은 데비안Debian 기반 리눅스 배포판을 사용한다면 'apt-get'을 사용해 패키지를 설치할 수 있다.

```
sudo apt-get install python-NumPy python-SciPy
```

페도라, OpenSUSE와 같은 RPM 기반 시스템이라면 'yum'을 사용하여 파이썬 패키지를 설치할 수 있다.

```
sudo yum install NumPy SciPy
```

---

05 http://www.enthought.com/products/epd_free.php
06 http://www.activestate.com/activepython/downloads
07 http:// www.macports.com

윈도우 시스템에서 SciPy, NumPy 빌드와 설치는 유닉스 기반 시스템과 달리 코드 컴파일이 쉽지 않아 더 복잡하다. 다행히 'python(x,y)'[08]라는 좋은 설치 프로그램이 있는데 SciPy와 NumPy를 포함하고 있고 윈도우에 특화되어 있다.

SciPy와 NumPy를 소스 빌드 하고자 한다면 'www.SciPy.org/Download'에서 소스를 내려 받거나, 'SciPy.github.com'과 'NumPy.github.com'에서 코드 저장소repository를 클론clone할 수도 있다. 소스 코드에서 패키지를 빌드할 수 있는 전문가가 아니라면 컴파일 된 패키지로 설치하는 것을 추천한다.

1.3 SciPy와 NumPy 사용하기

파이썬 프로그램은 대화형과 스크립트, 두 가지 방식으로 작업할 수 있다. 어떤 프로그래머들은 모든 코드를 스크립트로 작성해야 코드가 필요할 때 여러 작업을 반복하는 일이 없다고 하거나, 대화형 프로그래밍으로 해야 기능을 상세하게 살펴볼 수 있다고 주장하지만, 필자는 개인적으로 두 가지 모두 좋다고 생각한다. 파이썬 터미널이 열려 있고 스크립트를 작성할 수 있는 텍스트 편집기가 있다면, 두 방식의 장점을 모두 취할 수 있다.

대화형으로 작업할 때는 IPython[09]을 강력하게 추천한다. 탭 자동완성 기능, 디렉터리 변경과 같은 배쉬Bash 환경의 강점이 파이썬 환경에 잘 결합되어 있다. 물론 여기서 얘기한 것보다 훨씬 더 많은 기능을 제공하지만, 이 책의 예제를 위해서는 이 정도만으로도 충분하다.

08 http://code.google.com/p/pythonxy/
09 http://ipython.org

버그 없는 코드란 없고, 피할 수도 없다. 버그를 찾아 쉽고 빠르게 고치는 것은 성공적인 프로그래밍의 중요한 요소다. IPython 실행 후 'debug'라는 명령을 입력하면 디버그 할 수 있다. 보다 자세한 내용은 'http://ipython.org/ipython-doc/stable/interactive/tutorial.html'에서 'Debugging' 섹션을 참조하기 바란다.

1.4 소스 코드

이 책의 소스 코드는 쉽게 사용하고 변경할 수 있게 텍스트 형태로 제공했다. 소스 코드에는 Matplotlib 패키지를 사용해 그래프를 출력하는 명령이 포함되어 있어 그래프를 다루는 기본 지식이 필요하다.

개정하면서 새로운 예시가 추가되어, SciPy와 NumPy의 API 변경도 반영하였다.

2 | NumPy

2.1 NumPy 배열

NumPy는 과학계산을 수행하는 파이썬 핵심 패키지다. 이 패키지는 N차원 배열, 원소별 연산(브로드캐스팅), 선형대수학과 같은 핵심 수학 연산, C/C++/포트란 코드 인터페이스를 제공한다. 이번 장에는 NumPy 배열이 무엇인지, 그리고 이 배열이 파이썬의 리스트, 딕셔너리와 비교해서 어떤 장점이 있는지 알아보면서 NumPy 패키지의 특징을 살펴보자.

파이썬은 데이터를 여러 방식으로 저장하지만 가장 많이 사용되는 방식은 리스트List와 딕셔너리Dictionary다. 파이썬 리스트 객체는 거의 모든 종류의 객체를 담을 수 있다. 그러나 리스트에 담긴 원소의 연산은 반복문 내에서만 수행되는데, 파이썬에서는 반복문 안에서의 연산이 비효율적이다. 하지만 NumPy 패키지는 'ndarray'라는 데이터 저장 객체를 제공하여 파이썬 리스트의 단점을 극복할 수 있다.

ndarray는 리스트와 비슷하지만, 한 행에 같은 형식의 원소를 저장한다는 점에서 여러 타입의 객체를 저장하는 유연성이 높은 리스트와는 다르다. 예를 들어 파이썬 리스트의 첫 번째 원소가 리스트일 때, 두 번째 원소는 리스트나 딕셔너리로 만들 수 있다. NumPy 배열에서는 부동소수, 정수, 또는 문자열 등과 같이 한 가지 타입만 저장할 수 있지만, ndarray의 연산 속도는 일반적 파이썬 리스트보다 훨씬 빠르다. IPython의 '%timeit' 명령어[10]를 사용하여 NumPy의 ndarray와 파이썬 리스트, 둘의 연산 속도를 비교해 보았다.

10 **역자주_** IPython에서 몇 가지 편리한 기능을 'magic command'라는 이름으로 제공한다. 이런 명령은 '%'로 시작한다.

```
import numpy as np

# 10^7개의 원소를 갖는 배열 생성
arr = np.arange(1e7)

# ndarray를 리스트로 변환
larr = arr.tolist()

# 리스트는 브로드캐스트가 불가능하다
# 그래서 ndarray의 브로드캐스트를 흉내 내는 함수를 제작

def list_times(alist, scalar):
  for i, val in enumerate(alist):
      alist[i] = val * scalar
  return alist

# IPython에서 timeit 명령 실행
timeit arr * 1.1
>>> 1 loops, best of 3: 76.9 ms per loop[11]

timeit list_times(larr, 1.1)
>>> 1 loops, best of 3: 2.03 s per loop[12]
```

예제에서 보다시피, ndarray 연산이 파이썬 반복문보다 25배 정도 빠르다. 이제는 NumPy의 ndarray를 사용해야겠다는 생각이 들지 않는가? 앞으로는 가급적 리스트가 아닌 ndarray를 사용하겠다.

11 **역자주_** 실행하는 환경에 따라 이 값이 다르게 나올 수 있다.

12 **역자주_** 실행하는 환경에 따라 이 값이 다르게 나올 수 있다.

선형대수 연산이 필요한 경우 matrix[행렬] 객체를 사용할 수 있다. matrix 객체는 ndarray의 브로드캐스트 연산을 사용하지 않는다. 예를 들면 동일한 크기의 ndarray A와 B를 곱할 때 A의 원소는 B의 원소하고만 곱해지며, 일반적인 행렬 곱하기 연산이 수행된다.

ndarray 객체와는 달리 matrix 객체는 2차원만 가능하다. 따라서 3차원 이상의 행렬을 만드는 것은 불가능하다. 다음의 예제를 살펴보자.

```
import numpy as np

# 3차원 Numpy 배열 생성
arr = np.zeros((3,3,3))

# 행렬로 변환을 시도하면 오류가 발생 한다.
mat = np.matrix(arr)

# "ValueError: shape too large to be a matrix."
```

행렬을 사용할 때 이러한 점을 염두에 두어야 한다.

2.1.1 배열 생성과 데이터 형식

NumPy 배열을 생성하는 방법은 여러 가지가 있지만 여기서는 가장 유용한 방법만 논의한다.

NumPy_211_ex1.py

```
# 먼저 리스트를 하나 만든다.
# 생성한 리스트를 np.array() 함수에 넘겨준다.
```

```
import numpy as np

alist = [1, 2, 3]
arr = np.array(alist)

# 0으로 초기화된 5개의 원소를 갖는 배열을 생성
arr = np.zeros(5)

# 0부터 99까지를 원소로 하는 배열을 만들려면 어떻게 해야 할까?
arr = np.arange(100)

# 혹은 10부터 99는?
arr = np.arange(10,100)

# 0부터 1까지 100단계로 만들고 싶다면...
arr = np.linspace(0, 1, 100)

# 로그 스케일로 1부터 10까지 100단계로
# 배열을 만들고 싶다면...
arr = np.logspace(0, 1, 100, base=10.0)

# 5x5 형식의, 0으로 채워진 배열(이미지) 만들기
image = np.zeros((5,5))

# 5x5x5 형식의, 1로 채워진 배열을 만든다.
# astype() 메서드는 원소들을 정수로 설정한다.
cube = np.zeros((5,5,5)).astype(int) + 1

# 16비트 부동소수점으로 더 간단하게 할 수 있다.
cube = np.ones((5, 5, 5)).astype(np.float16)
```

배열을 생성할 때 NumPy는 파이썬 환경에 따라 비트 깊이bit depth를 기본적으로 사용한다. 만일 64비트 파이썬을 사용하면 배열의 원소는 64비트로 지정된다.[13] 비트 깊이에 따라 메모리 사용량이 달라진다. 비트 깊이를 설정해서 NumPy 배열에 소요되는 메모리 사용량을 조절할 수 있다. 배열을 생성할 때 데이터 형식 파라미터 dtype을 int, NumPy.float16, NumPy.float32 또는 NumPy.float64로 지정할 수 있다. 다음 예제는 비트 깊이를 지정하는 방법이다.

NumPy_211_ex1.py

```
# 정수 0으로 된 배열
arr = np.zeros(2, dtype=int)

# 부동소수 0으로 된 배열
arr = np.zeros(2, dtype=np.float32)
```

배열을 생성한 후, 이 배열을 여러 형태로 변환할 수 있다. 만일 25개의 원소가 있으면, 이 배열을 5x5로 만들거나, 3차원 배열로도 만들 수 있다.

NumPy_211_ex1.py

```
# 0부터 999를 원소로 갖는 배열 생성
arr1d = np.arange(1000)

# 생성된 배열을 3차원의 10x10x10 배열로 변형
arr3d = arr1d.reshape((10,10,10))
```

13 **역자주_** 32비트 파이썬에서와 같이 별도의 옵션 없이 배열을 생성하면 64비트로 설정된다.
〉〉〉 a = np.array([1.0])
〉〉〉 a.dtype
d.type('float64')

```
# reshape은 다음과 같이 호출할 수 있다.
arr3d = np.reshape(arr1d, (10, 10, 10))

# 거꾸로 행렬을 길게 늘여 1차원으로 만들 수도 있다.
arr4d = np.zeros((10, 10, 10, 10))
arr1d = arr4d.ravel()

print arr1d.shape
 (1000,)
```

배열 재구성 방법은 아주 다양하면서 구성이 쉽다는 장점이 있다.

> 위에서 재구성된 데이터는 메모리에서 동일한 데이터에 대한 다른 관점일 뿐이다. 그래서 위의 배열 중 하나를 수정하면 다른 배열도 바뀐다. 예를 들어 위 예제의 arr1d의 첫 번째 원소를 1로 설정하면, arr3d의 첫 번째 원소 또한 1이 된다. 이런 현상을 방지하려면 NumPy.copy 함수를 사용해 배열들이 메모리 공간memory-wise에서 분리되도록 해야 한다.

실행 속도와 ndarray의 구조가 관련이 있을까? ndarray가 브로드캐스팅을 수행하기 때문에 배열의 형태는 연산 속도에 큰 영향이 없다. 예를 들어 1000 * 1000의 ndarray가 있고, 똑같은 원소를 가지고 있지만 평평한 1,000,000 * 1의 배열이 있다면 어떤 형태의 브로드캐스트 연산을 수행해도 실행 속도는 거의 동일하다. 배열의 형태는 선형 대수적 연산을 수행할 때만 중요하다.

2.1.2 레코드 배열Record Arrays

일반적으로 배열은 정수 또는 부동소수의 모음이지만 때로는 각 열의 데이터 형식이 다른 복잡한 데이터 구조를 저장할 때 유용하기도 하다. 연구 논문에서 표로 정리할 때는 종종 어떤 열은 식별을 위한 문자열로, 다른 열은 수리 량을 위한 실수 값인 경우가 많다. 이런 형식으로 정보를 저장하면 여러 장점이 있다. NumPy에서는 NumPy.recarry가

있다. recarray를 처음 만드는 것은 조금 어려울 수 있다. 그래서 아래 기본적인 내용을 다루어 보도록 한다. 첫 번째 예제는 NumPy 참조 문서의 레코드 배열record arrays 설명에서 발췌했다.

NumPy_212_ex1.py

```
# 0으로 된 배열을 생성하고 열의 데이터 형식을 정의
import numpy as np

recarr = np.zeros((2,), dtype=('i4,f4,a10'))
toadd = [(1,2.,'Hello'),(2,3.,"World")]
recarr[:] = toadd
```

선택적 매개변수인 dtype은 첫 번째부터 세 번째 열의 데이터 형식을 정의한다. i4는 32비트 정수, f4는 32비트 실수 그리고 a10은 문자 10개로 된 문자열을 의미한다. 다양한 데이터 형식 정의 방법은 NumPy 문서[14]에서 찾을 수 있다. 이 예제는 recarray가 어떻게 생겼는지를 보여주고 있지만, 이런 배열을 생성하는 것은 좀 어려워 보인다. 다행히 파이썬에 내장된 zip이라는 함수가 위의 toadd 객체처럼 튜플tuple로 된 리스트를 생성해준다. zip을 사용해 위와 같은 recarray를 생성하는 법을 살펴보자.

NumPy_212_ex1.py

```
# 0으로 된 배열을 생성하고 열의 데이터 형식을 정의
recarr = np.zeros((2,), dtype=('i4,f4,a10'))

# recarray에 넣을 열을 생성
col1 = np.arange(2) + 1
col2 = np.arange(2, dtype=np.float32)
```

14 http://docs.SciPy.org/doc/NumPy/user/basics.rec.html

```
col3 = ['Hello', 'World']

# 위의 toadd 리스트와 동일한  튜플 리스트를 생성한다.
toadd = zip(col1, col2, col3)

# recarr에 값을 할당
recarr[:] = toadd

# 기본적으로 'f0', 'f1', 'f2'로 설정된 열에 명칭을 설정한다.
recarr.dtype.names = ('Integers', 'Floats', 'Strings')

# 열을 명칭으로 접근하고 싶다면 다음과 같이 할 수 있다.
Print recarr['Integers']
#[1, 2]
```

recarray 구조는 좀 번잡스러워 보일 수 있다. 하지만 '읽기와 쓰기'에서 다룰 복잡한 데이터를 읽어 들이는 내용에서는 이 구조가 보다 중요해 진다.

> 천문학이나 천체물리학을 연구하다 보면 데이터 테이블을 많이 다루게 되는데 ATpy2라는 고수준 패키지high-level package에 관심을 가질만 하다. 이 패키지는 FITS, ASCII, HDF5, SQL 형식으로 데이터 테이블을 읽고 쓰거나 변환할 수 있도록 해준다.

2.1.3 인덱싱Indexing과 슬라이싱Slicing

파이썬의 인덱스 리스트는 0부터 시작하고 NumPy 역시 그 방식을 따른다. 파이썬에서 리스트를 인덱싱할 때는 2 x 2 객체에 대해 다음과 같이 할 수 있다.

NumPy_213_ex1.py

```
alist=[[1, 2], [3, 4]]
```

```
# (0,1) 원소를 반환하려면 다음처럼 인덱스를 사용해야 한다.
alist[0][1]
```

오른쪽 열('2'와 '4')을 반환할 때, 파이썬 리스트로는 간단하게 처리하기 어렵다. NumPy에서는 좀 더 편한 문법을 사용하여 인덱싱할 수 있다.

NumPy_213_ex1.py

```
# 위에서 정의된 리스트를 배열로 변환
arr = np.array(alist)

# (0,1) 원소를 반환하려면
arr[0, 1]

# 맨 오른쪽 열에 접근하려면, 다음처럼
arr[:, 1]

# 열을 접근하는 것과 마찬가지로, 마지막 행에 접근할 수 있다.
arr[1, :]
```

때로는 조건적 인덱싱처럼 보다 복잡한 인덱싱이 필요할 때도 있다. 가장 많이 사용되는 것은 NumPy.where()이다. 이 함수를 사용하면 배열의 차원과 관계없이 특정한 조건에 따라 원하는 배열 인덱스를 반환할 수 있다.

NumPy_213_ex1.py

```
# 배열 생성
arr = np.arange(5)

# 인덱스 배열 생성
index = np.where(arr > 2)
```