

파이썬 및 구글 코랩 사용법

Preview

- 인공지능은 진입장벽이 낮은 편이지만 상품화는 많은 노력과 인내 필요
 - 인공지능 개발에 필요한 강력한 소프트웨어가 모두 무료이고 클라우드 서비스도 있음
 - 시장에 나가려면 인간 수준의 성능 필요. 시장 경쟁이 치열
- 훌륭한 프로그래밍 환경이 제공됨
 - 입증된 논문은 구현되어 라이브러리 형태로 제공되며 꾸준히 업그레이드 중
 - 이 책은 이런 강점을 살려 독자가 인공지능 만들기에 쉽게 진입하도록 배려함
- 프로그래밍 경험 수준에 따른 공부 방법
 - 파이썬에 익숙 → 바로 시작
 - 다른 언어에 익숙하지만 파이썬은 처음 또는 프로그래밍이 처음이라면 → 파이썬 기초와 부록 A numpy와 부록 B matplotlib을 공부한 다음에 시작 또는 바로 시작하고 파이썬 공부를 병행(부록은 한빛 홈페이지에서 온라인으로 제공)

*다운로드 주소: https://www.hanbit.co.kr/store/books/look.php?p_code=B8091740296 - 부록/예제소스 - 예제소스1

2.1 프로그램 예제

■ 아주 간단한 프로그램

■ 연산자의 동작과 함수를 사용하는 방법 예시

프로그램 2-1

덧셈과 문자열 접합을 하고 난수 2개를 더하는 프로그램

```
01 # 연산자 오버로딩 예시
02 print(12+37)           # +는 두 정수를 더하는 연산자
03 print('python'+ ' is exciting.') # +는 두 문자열을 접합하는 연산자
04
05 # 라이브러리 불러오기
06 import random          # random이라는 라이브러리를 불러옴
07
08 # 정수 난수 생성
09 a=random.randint(10,20) # [10,20] 사이의 난수를 생성하고 변수 a에 대입
10 b=random.randint(10,20) # [10,20] 사이의 난수를 생성하고 변수 b에 대입
11
12 # 덧셈을 하고 결과를 출력
13 c=a+b                  # a와 b를 더하여 변수 c에 대입
14 print(a,b,c)           # 변수 a, b, c를 출력
```

```
49
python is exciting.
10 15 25
```

- #으로 시작하는 주석문
- 02~03행: + 연산자(덧셈 또는 문자열 접합). 연산자 오버로딩
- 06행: random 모듈을 불러옴
- 09~10행: 정수 난수를 생성하여 변수에 저장
- 13~14행: 두 변수를 더하고 출력

2.1 프로그램 예제

TIP import 명령어가 '모듈을 불러와 쓸 수 있게 해준다'라는 말의 구체적인 의미는 모듈에 정의되어 있는 변수, 함수, 클래스를 프로그램에 포함시켜 '모듈.변수', '모듈.함수', '모듈.클래스' 형태로 쓸 수 있게 만들어준다는 것이다. import 명령어의 사용법은 여러 가지가 있는데 『Do IT! 점프 투 파이썬(박응용, 이지스퍼블리싱)』의 5.2절을 참조한다. 또는 <https://docs.python.org/ko/3/tutorial/modules.html>을 참조한다.

NOTE 파이썬의 연산자 오버로딩

하나의 연산자가 피연산자에 따라 서로 다른 기능을 하는 특성을 연산자 오버로딩(operator overloading)이라 부른다. 예를 들어 1.5.3항에서 예시한 바와 같이, 피연산자가 벡터이면 벡터 덧셈, 행렬이면 행렬 덧셈을 해준다. 현대적 언어일수록 연산자 오버로딩이 풍부하며, 객체지향 언어에서는 더욱 풍부하다. 파이썬은 현대적 객체지향 언어로서 연산자 오버로딩이 매우 풍부하다. 파이썬의 객체지향 특성과 사용법은 2.7절에서 설명한다.

2.2 두 가지 프로그래밍 환경

- 클라우드 방식과 스탠드얼론 방식
 - 자신에게 적절한 환경을 사용하면 됨

2.2.1 클라우드 방식과 스탠드얼론 방식

■ 클라우드 방식

- 프로그램과 데이터가 서버에 저장되고 관리
 - 서버에 환경이 대부분 갖추어져 있어 로그인하면 바로 프로그래밍 가능
 - 인터넷 연결만 있으면 어느 곳에서나 개발 가능. 협업 가능
- 구글의 Colab, 아마존의 SageMaker, 마이크로소프트의 Azure
- 내 프로젝트에 최적의 환경을 갖추 수 없는 한계

■ 스탠드얼론 방식

- 자신에 최적의 환경 구축 가능. 프로그램과 데이터가 자신의 컴퓨터에 저장
- 소프트웨어를 설치하고 환경을 스스로 구축해야 함

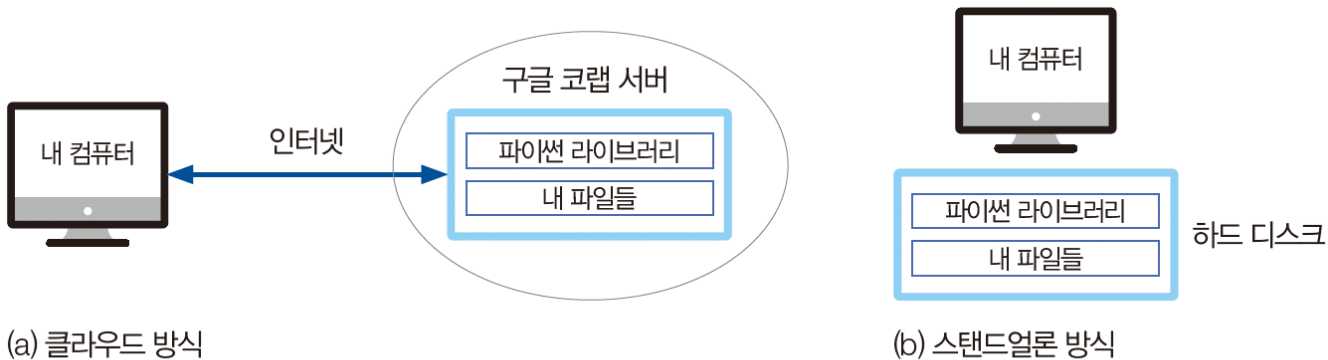


그림 2-1 파이썬 프로그래밍 환경

2.2.1 클라우드 방식과 스탠드얼론 방식

NOTE 프로그래밍 환경에 대한 좋은 태도

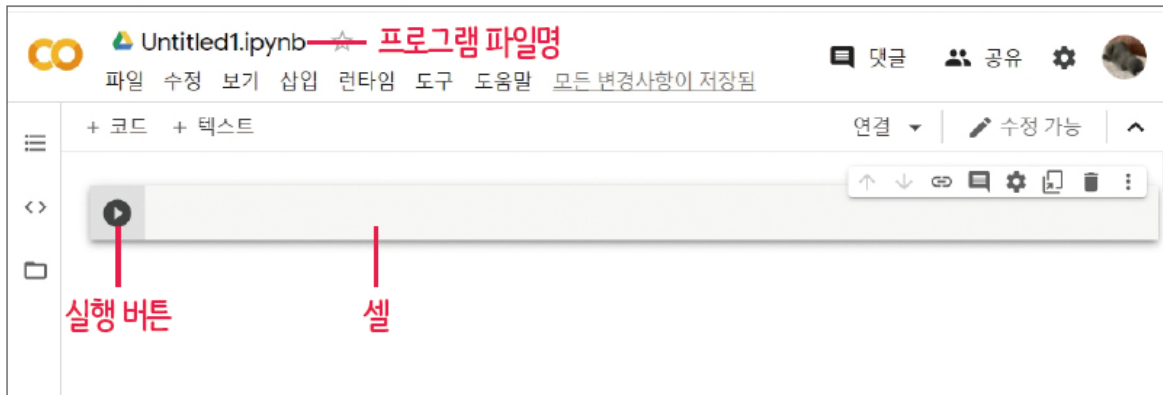
여러 가지 프로그래밍 환경이 주어진다는 사실은 인공지능 프로그래머 층이 두터워졌다는 뜻으로 해석할 수 있다. 시범적으로 프로그래밍을 해보려는 사람과 본격적으로 제품을 개발하는 사람의 요구사항은 다를 수밖에 없기 때문에 여러 계층의 프로그래밍 환경이 주어진다. 이 절에서 설명하는 두 가지 프로그래밍 환경 중에 반드시 하나를 선택해야 하는 것이 아니다. 두 환경 사이를 쉽게 전환할 수 있는 수준의 익숙함을 확보하는 것이 현명한 태도이다.

2.2.2 파이썬 시작하기: 클라우드 방식의 colab(코랩)

■ Colab을 사용하는 절차

1. <http://colab.research.google.com>에 접속한다.
2. 구글 계정으로 로그인한다(구글 계정이 없다면 계정을 만든 다음에 로그인한다).
3. 파이썬 프로그래밍을 한다.
4. 프로그래밍을 마치면 프로그램 파일을 구글 드라이브에 저장한다.

2.2.2 파이썬 시작하기: 클라우드 방식의 colab



(a) 로그인한 화면



(b) 파이썬 코드를 입력하고 실행한 결과 화면

그림 2-2 코랩 화면

2.2.2 파이썬 시작하기: 클라우드 방식의 colab

■ 프로그래밍 장면

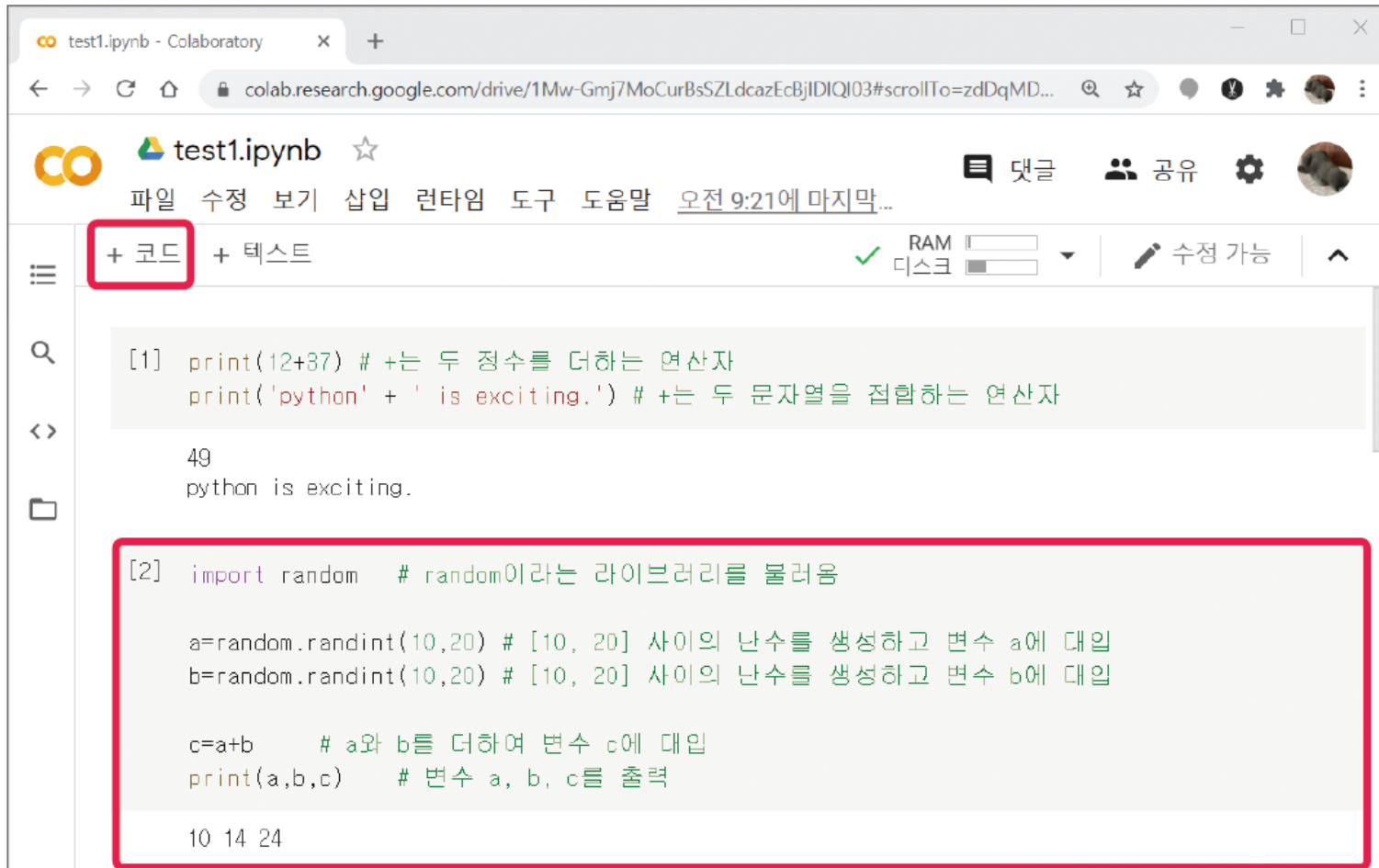


그림 2-3 코랩의 주피터 노트북에서 코드 추가

2.2.2 파이썬 시작하기: 클라우드 방식의 colab

NOTE 주피터 노트북

파이썬을 대화식으로 편리하게 실행할 수 있는 환경을 만드는 IPython 프로젝트를 수행하던 페르난도 페레스(Fernando Perez)는 2014년에 파이썬뿐 아니라 R과 Julia 언어도 지원하는 새로운 프로젝트인 주피터 노트북을 분리하여 진행한다. 나중에 주피터 노트북은 파이썬 프로그래밍을 위한 대화형 인터페이스의 표준으로 자리 잡는다. 갈릴레오가 목성(Jupiter) 주위를 도는 달의 운동을 기록한 노트북을 기리는 의미에서 주피터 노트북이라 이름 지었다고 전해진다.

2.3.2 인터프리터 방식의 파이썬

■ 언어 번역기: 컴파일러 vs. 인터프리터 방식

■ 컴파일러 방식

- 프로그램 전체를 번역한 다음 한꺼번에 실행
- 실행이 빠른 장점
- C, C++ 등

■ 인터프리터 방식

- 한 라인 씩 번역하고 실행하는 일을 순차적으로 수행
- 일부 코드만 선택하여 실행하는 일이 가능한 장점 ← 이 장점을 잘 활용하길 권유함
(스파이더에서는 실행하고자 하는 코드를 마우스로 선택한 다음 [F9] 키를 누름)
- 파이썬 등

2.4 인공지능 프로그래밍 예제 1: 셈 지능

■ 인간의 셈 지능

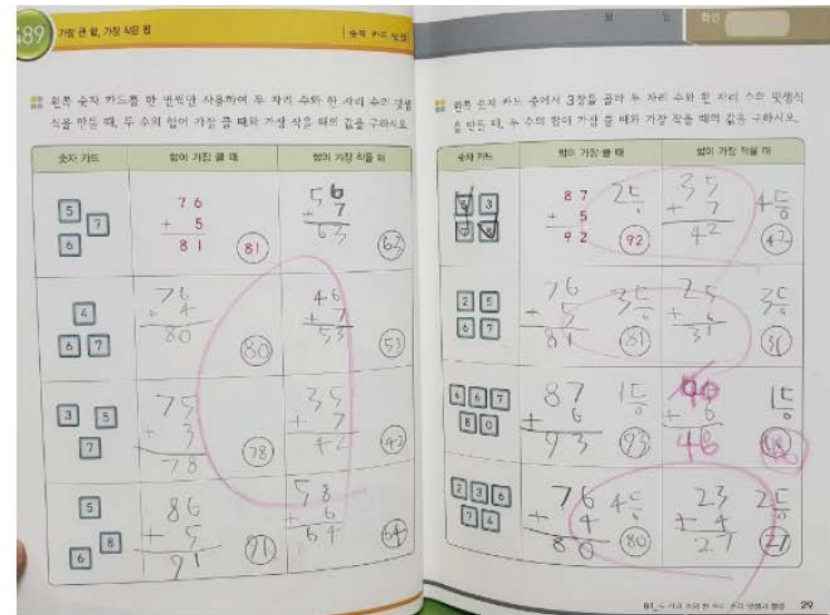
- 느리고 정확도에 한계
- 세계주산암산대회(2018) 우승자는 초당 1.3개의 사칙 연산 수행

$$\begin{array}{r} 48 \\ + 34 \\ \hline 82 \end{array}$$

- ① 일의 자리를 더한다. 결과 2를 기록하고 올림수 1을 왼쪽 자리로 보낸다.
- ② 십의 자리를 더한 결과에 올림수를 더한다. 결과 8을 기록한다.

(a) 덧셈 알고리즘의 적용 예

그림 2-12 인간의 셈 지능



(b) 불완전하고 느린 인간의 셈 지능

2.4 인공지능 프로그래밍 예제 1: 셈 지능

■ 컴퓨터의 셈 지능

- 무척 빠르고 정확함. 컴퓨터 하드웨어에 내장되어 있음
- [프로그램 2-2]는 컴퓨터의 셈 지능 속도를 측정
 - 10억 개 덧셈을 11.8초 만에 수행

프로그램 2-2

컴퓨터의 셈 지능 속도 측정

```
01 import time                # 시간 측정 라이브러리
02
03 start=time.time()          # 시작 직전 시각을 기록
04 sum=0
05 for i in range(1,100000001): # 1억 번 반복
06     sum=sum+i
07
08 end=time.time()             # 끝난 직후 시각을 기록
09
10 print('1+2+...+100000000=', sum)
11 print('소요 시간은 ', end-start, '초입니다.') # 시간 차이를 계산하여 출력
```

1+2+...+100000000=4999999950000000
소요 시간은 11.80145525932312초입니다.

2.5 인공지능 프로그래밍 예제 2: 인공지능 기자

■ 뉴스 기사를 자동으로 작성하는 인공지능

- <LA 타임스>의 퀘이커봇은 지진 전문 로봇 기자
- <연합뉴스>의 사커봇은 프리미어 리그 속보 로봇 기자
- 현재 높은 수준의 자연어 처리와 추론 등의 기술을 사용하여 인간 기사를 능가하는 수준
- 여기서는 원시적인 로봇 기자 프로그래밍

2.5 인공지능 프로그래밍 예제 2: 인공지능 기자

■ 손흥민의 경기 결과를 기사로 작성하는 '단순한' 인공지능 프로그램

- 04~10행: 경기 결과를 입력하는 곳
- 13~31행: 기사를 합성하는 곳
- 13행은 datetime 함수를 이용하여 속보 입력 시간을 자동으로 붙임

프로그램 2-3

손흥민 선수의 경기 속보를 전담하는 로봇 기자

```
01  from datetime import datetime
02
03  # 경기 결과 입력 받는 곳
04  place=input("경기가 열린 곳은? ")
05  time=input("경기가 열린 시간은? ")
06  opponent=input("상대 팀은? ")
07  goals=input("손흥민은 몇 골을 넣었나요? ")
08  aids=input("도움은 몇 개인가요? ")
09  score_me=input("손흥민 팀이 넣은 골 수는? ")
10  score_you=input("상대 팀이 넣은 골 수는? ")
11
12  # 기사 작성하는 곳
13  news="[프리미어 리그 속보("+str(datetime.now())+")]\n"
14  news=news+"손흥민 선수는 "+place+"에서 "+time+"에 열린 경기에 출전하였습니다. "
15  news=news+"상대 팀은 "+opponent+"입니다. "
```


2.5 인공지능 프로그래밍 예제 2: 인공지능 기자

■ (... 앞에서 계속)

- 17~22행: 두 팀의 골 수에 따라 경우를 나누어 기사 작성
- 24~31행: 손흥민의 골과 도움 수에 따라 변화를 주어 단조로움 해소

```
16
17 if score_me>score_you:
18     news=news+"손흥민 선수의 팀이 "+score_me+"골을 넣어 "+score_you+"골을 넣은 상대 팀
    을 이겼습니다. "
19 elif score_me<score_you:
20     news=news+"손흥민 선수의 팀이 "+score_me+"골을 넣어 "+score_you+"골을 넣은 상대 팀
    에게 졌습니다. "
21 else:
22     news=news+"두 팀은 "+score_me+"대"+score_you+"로 비겼습니다. "
23
24 if int(goals)>0 and int(aids)>0:
25     news=news+"손흥민 선수는 "+goals+"골에 도움 "+aids+"개로 승리를 크게 이끌었습니다. "
26 elif int(goals)>0 and int(aids)==0:
27     news=news+"손흥민 선수는 "+goals+"골로 승리를 이끌었습니다. "
28 elif int(goals)==0 and int(aids)>0:
29     news=news+"손흥민 선수는 골은 없지만 도움 "+aids+"개로 승리하는 데 공헌하였습니다. "
30 else:
31     news=news+"아쉽게도 이번 경기에서 손흥민의 발끝은 침묵을 지켰습니다. "
32
33 print(news)
```

2.5 인공지능 프로그래밍 예제 2: 인공지능 기자

■ 경기 결과 입력과 작성된 기사 예

경기가 열린 곳은? 런던 스타디움

경기가 열린 시간은? 8월 6일 오후 7시

상대 팀은? 맨체스터 유나이티드

손흥민은 몇 골을 넣었나요? 2

도움은 몇 개인가요? 1

손흥민 팀이 넣은 골 수는? 4

상대 팀이 넣은 골 수는? 2

[프리미어 리그 속보(2020-08-07 09:33:14.128724)]

손흥민 선수는 런던 스타디움에서 12월 12일 오후 7시에 열린 경기에 출전하였습니다. 상대 팀은 맨체스터 유나이티드입니다. 손흥민 선수의 팀이 4골을 넣어 2골을 넣은 상대 팀을 이겼습니다. 손흥민 선수는 2골에 도움 1개로 승리를 크게 이끌었습니다.

2.5 인공지능 프로그래밍 예제 2: 인공지능 기자

■ 확장

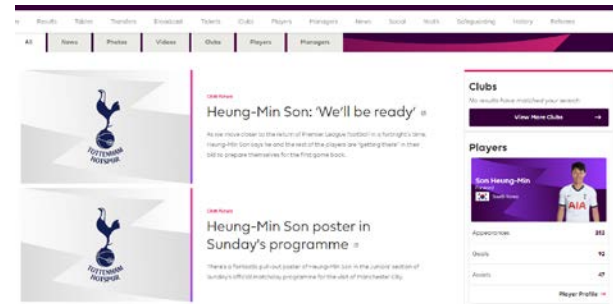
- 문장을 랜덤 선택하여 단조로움 해소

```
18 wording=["상대 팀을 이겼습니다.", "상대를 대상으로 통쾌한 승리를 거머쥐었습니다.", "상대 팀  
   을 꺾었습니다."]
19 news=news+"손흥민 선수의 팀이"+score_me+"골을 넣어"+score_you+"골을 넣은"+ random.  
   choice(wording)
```

- 자동으로 경기 결과를 크롤링하고 기사를 업로드
 - 프리미어 웹 사이트 파싱
 - 웹 프로그래밍(자바스크립트)

- 수준 높은 기사 작성

- 자연어 처리 기술과 word2vec(8.5.5항)과 같은 언어 모델 활용



2.6 인공지능 프로그래밍 예제 3: 더 똑똑한 인공지능 기자

- 언어 처리 라이브러리를 활용하면 더 똑똑한 인공지능 기자
 - 이 절은 가장 단순한 축에 속하는 TTS(text-to-speech) 기능 활용(gtts 라이브러리)

2.6.1 모듈 설치

■ 모듈 설치

```
[6] !pip install gtts playsound
```

NOTE 모듈 사용법에 익숙해지기

파이썬을 프로그래밍할 때 모듈 사용법을 제대로 익히는 일은 매우 중요하다. 파이썬 모듈을 모아둔 공식 사이트인 <http://pypi.org>에 접속하면 해당 모듈을 설명하는 공식 문서를 찾을 수 있다. 설명 문서는 모듈이 제공하는 함수 목록, 개별 함수의 API, 예제 프로그램 등을 제공한다. API(Application Program Interface)란 함수를 사용할 때 지정하는 매개변수(parameters)가 가질 수 있는 값의 범위, 매개변수를 생략했을 때 기본으로 사용하는 기본값 등을 뜻한다. 함수의 사용법이라고 생각하면 된다. 예를 들어 `random.randint` 함수의 API는 다음과 같다 (관련 웹 페이지: <https://docs.python.org/3/library/random.html>). 인공지능을 개발할 때 쓰는 함수 대부분은 `randint`보다 훨씬 복잡한 API를 가진다.

```
random.randint(a, b)
Return a random integer N such that a <= N <= b.
```

파이썬 모듈에 대한 좋은 문서를 구하는 또 다른 방법은 인터넷 검색이다. 예를 들어 `gtts` 모듈에 대해 공부하려는 경우 웹에서 '`gtts documentation`'을 검색하면 많은 문서와 예제 프로그램을 얻을 수 있다.

2.6.2 TTS 프로그래밍

■ [프로그램 2-4]: 소리를 들려줄 수 있게 [프로그램 2-3] 확장

프로그램 2-4

말하는 로봇 기자

```
01  ... } [프로그램 2-3]
33
34
35  # 음성으로 들려주는 곳
36  from gtts import gTTS
37  import playsound
38
39  tts=gTTS(text=news, lang='ko')           # 문자열 news를 위한 한국어 음성 합성
40  tts.save("news_Son.mp3")
41  playsound.playsound("news_Son.mp3", True)
```



TIP gTTS의 save 함수는 저장하려는 파일이 이미 있으면 오류를 발생시킨다. 따라서 [프로그램 2-4]를 재차 실행하면 오류가 난다. 파일을 삭제하고 실행하거나, import os를 한 후에 41행 뒤에 os.remove('news_Son.mp3') 명령어를 추가해 자동으로 파일을 삭제하면 된다.

2.7 인공지능 프로그래밍을 위한 파이썬 기초

■ 알아두면 유익한 것들

- 인공지능 프로그래밍에 주로 사용하는 라이브러리
- 파이썬의 객체지향 특성과 사용법

2.7.1 인공지능 개발에 많이 쓰는 라이브러리

■ 현대 프로그래밍 언어는 오픈소스로 공개

- 파이썬, R, Ruby, Perl, Julia, Swift 등
- 제3자 라이브러리 풍부한 장점
- 파이썬은 하루에도 수십 개의 새로운 라이브러리가 공개됨

2.7.1 인공지능 개발에 많이 쓰는 라이브러리

표 2-1 파이썬 언어와 라이브러리에 대한 정보

구분		공식 사이트	튜토리얼 문서
언어	파이썬 (Python)	https://www.python.org	• The Python Tutorial: https://docs.python.org/3/tutorial/
라이브러리 관리	파이파이 (Pypi)	https://pypi.org	• Installing packages: https://packaging.python.org/tutorials/installing-packages • Packaging Python projects: https://packaging.python.org/tutorials/packaging-projects
라이브러리	넘파이 (Numpy)	https://numpy.org	• Numpy Tutorial: https://numpy.org/devdocs/user/quickstart.html
	맷플롯립 (Matplotlib)	https://matplotlib.org	• User's guide: https://matplotlib.org/users/index.html
	사이킷 런 (Scikit-learn)	https://scikit-learn.org	• User guide: https://scikit-learn.org/stable/user_guide.html
	텐서플로 (TensorFlow)	https://www.tensorflow.org	• Tensorflow Tutorials: https://www.tensorflow.org/tutorials
	케라스 (Keras)	https://keras.io	• Keras documentation: https://keras.io
	파이토치 (PyTorch)	https://pytorch.org	• Welcome to Pytorch tutorials: https://pytorch.org/tutorials

2.7.1 인공지능 개발에 많이 쓰는 라이브러리

■ 기초 라이브러리

- 넘파이(Numpy): 다차원 배열 지원(부록 A)
- 맷플롯립(Matplotlib): 데이터 시각화(부록 B)

TIP 예제와 함께 PDF 파일로 제공되는 부록 A에서 간단하게 넘파이 사용법을 소개하니 넘파이에 익숙하지 않다면 3장으로 넘어가기 전에 꼭 공부하기 바란다.

TIP PDF 파일로 제공되는 부록 B에서 간단하게 맷플롯립 사용법을 소개한다. 맷플롯립에 익숙하지 않다면 3장으로 넘어가기 전에 꼭 공부하기 바란다.

■ 인공지능 라이브러리

- 사이킷런(Scikit-learn): 고전적인 기계 학습 지원
- 텐서플로(TensorFlow): 딥러닝 지원(이 책이 사용하는 라이브러리)
- 케라스(Keras): 텐서플로를 한 단계 추상화한 라이브러리(이 책이 사용하는 라이브러리)
- 파이토치(PyTorch): 딥러닝 라이브러리

2.7.2 객체지향 언어를 이용한 인공지능 프로그래밍

■ 객체지향 언어

- 서로 관련이 있는 변수와 함수를 하나로 묶어서 다룰 수 있는 클래스를 제공

■ 객체지향 프로그래밍

▪ 예제 코드

- 36행: gtts라는 모듈에서 gTTS 클래스를 불러옴
- 39행: gTTS 클래스의 인스턴스 변수(객체 변수 또는 객체) tts를 생성
- 40행: tts가 제공하는 save 메서드(함수) 호출

[코드 1]

```
36 from gtts import gTTS
37 import playsound
38
39 tts=gTTS(text=news,lang='ko')
40 tts.save("news_Son.mp3")
41 playsound.playsound("news_Son.mp3", True)
```

■ 이 책은

- 인스턴스 변수 대신 객체라는 용어 사용, 메서드 대신 함수라는 용어 사용

2.7.2 객체지향 언어를 이용한 인공지능 프로그래밍

■ 파이썬에서 코딩하는 전형적인 절차

1. 모듈에서 클래스를 불러온다
2. 클래스의 객체(인스턴스 변수)를 만든다.
3. 객체의 함수를 호출해 원하는 과업을 달성한다.

2.7.2 객체지향 언어를 이용한 인공지능 프로그래밍

■ 객체지향 프로그래밍 예

[코드 2]

```
36 from gtts import gTTS
37 import playsound
38
39 tts_korean=gTTS(text=news_ko, lang='ko')
40 tts_english=gTTS(text=news_en, lang='en')
41 tts_french=gTTS(text=news_fr, lang='fr')
42 tts_korean.save("news_Son_korean.mp3")
43 tts_english.save("news_Son_english.mp3")
44 tts_french.save("news_Son_french.mp3")
```

- 39~41행: gTTS 클래스로 세 개의 객체 tts_korean, tts_english, tts_french 생성(lang 매개 변수로 언어 설정)
- 42~44행: 언어에 따른 mp3 파일 저장

2.7.2 객체지향 언어를 이용한 인공지능 프로그래밍

■ 일상 생활의 비유

- 와플 메이커는 클래스, 와플 메이커로 찍어낸 와플은 객체



(a) 와플 메이커(클래스)



(b) 와플 메이커로 만든 와플(클래스로 생성한 객체)

그림 2-14 와플 메이커와 와플(클래스와 객체)

[코드 3]

```
01  from kitchenware import waffle_maker
02  mom_waffle=waffle_maker( )
03  dad_waffle=waffle_maker(bake_time=70)
04  my_waffle=waffle_maker( )
05  mom_waffle.put(fruit='strawberry')
06  dad_waffle.dip(drink='coffee')
07  ...
```

2.7.2 객체지향 언어를 이용한 인공지능 프로그래밍

■ 클래스의 API

▪ gTTS 클래스의 예

기본값(default value)

```
class gtts.tts.gTTS(text, tld='com', lang='en', slow=False, lang_
check=True, pre_processor_funcs=[<function tone_marks>, <function end_
of_line>, <function abbreviations>, <function word_sub>], tokenizer_
func=<bound method Tokenizer.run of re.compi
...
•lang (string, optional) – The language (IETF language tag) to read the
text in. Default is 'en'.
•slow (bool, optional) – Reads text more slowly. Defaults to False.
...
```

2.7.2 객체지향 언어를 이용한 인공지능 프로그래밍

■ 클래스의 멤버 변수와 멤버 함수 확인하기

- dir 명령어 사용

```
>>> dir(gTTS)
```

■ 객체가 어떤 클래스인지 확인하기

- type 명령어 사용

```
>>> type(my_waffle)
```

NOTE 객체지향 언어

1972년에 발표된 스몰토크(Smalltalk)는 세계 최초의 객체지향 언어이다. 프로그래밍 패러다임을 바꾼 스몰토크는 명성에 비해 널리 쓰이지는 않았다. 이후 절차적 언어인 C를 객체지향으로 변환한 언어인 C++가 1983년에 탄생한다. C++의 유행에 밀려 스몰토크는 객체지향의 원조 자리는 유지하지만 대표 자리는 C++에 내주고 만다. 이후에 나온 자바, 파이썬 등은 객체지향 언어이다. 객체지향 언어로 작성한 프로그램은 클래스 단위로 모듈화되어 있어 다른 프로젝트로 가져다 쓸 수 있는 재사용성이 좋아 소프트웨어 생산성을 높이는 데 효과적이다. 인공지능 분야도 객체지향의 좋은 특성을 잘 활용하고 있다.

Appendix

2.2.3 파이썬 시작하기: 스탠드얼론 방식

■ 파이썬 설치하기(세 가지 소프트웨어를 설치해야 함)

- 파이썬 컴파일러
- 통합 개발 환경(IDE, integrated development environment)
- 모듈(라이브러리)

NOTE 모듈과 라이브러리

모듈은 독립적으로 설치하고 import 명령어로 불러올 수 있는 최소 단위의 코드 모음으로, 실제로는 .py 파일로 구성되어 있다. 예를 들어 random 모듈은 파이썬이 설치되어 있는 폴더에 random.py 파일로 저장되어 있다. 다른 프로그래밍 언어에서는 모듈 대신 라이브러리라는 용어를 쓴다. 파이썬에서는 여러 모듈의 집합을 라이브러리라고 하는데 때때로 모듈 자체를 라이브러리라고 부르기도 한다. 이 책에서는 모듈과 라이브러리를 섞어 사용한다.

2.2.3 파이썬 시작하기: 스탠드얼론 방식

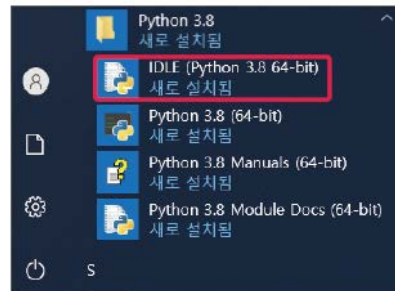
- 설치 장면(<http://www.python.org>에서 설치 파일 다운로드)



(a) 다운로드 화면



(b) 설치 시작 화면

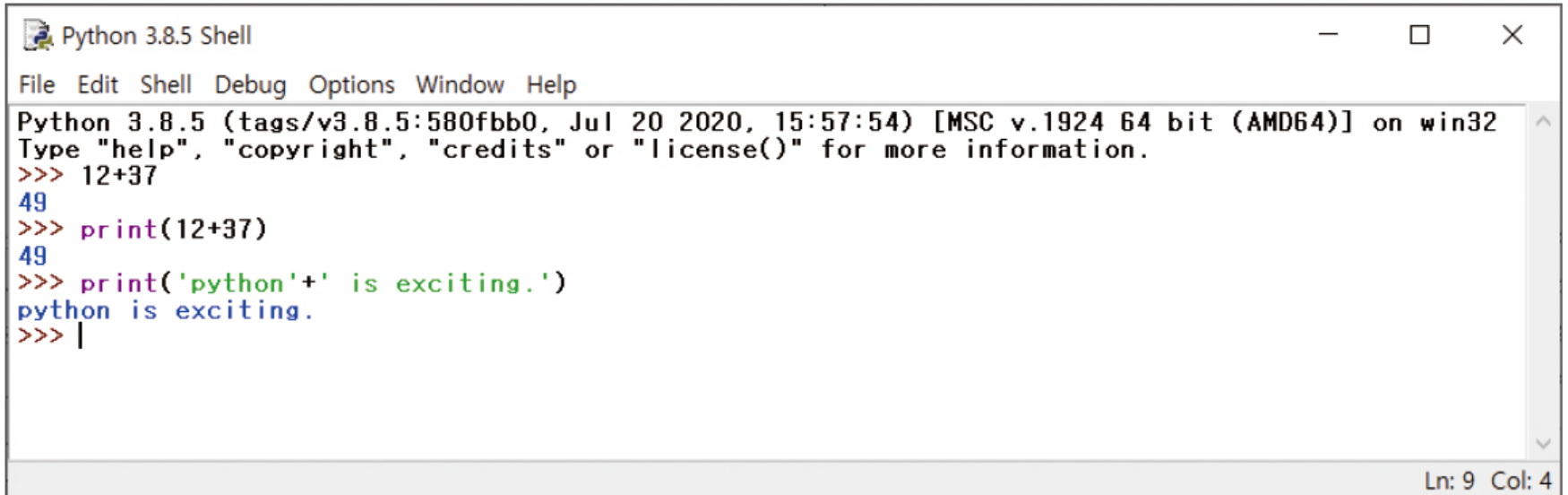


(c) 설치 확인

그림 2-4 파이썬 설치 과정

2.2.3 파이썬 시작하기: 스탠드얼론 방식

- 통합 개발 환경 IDLE로 시범 프로그래밍(셀 창에서 프로그래밍)

A screenshot of the Python 3.8.5 Shell window. The window has a title bar with the text 'Python 3.8.5 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area contains the following text:

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 12+37
49
>>> print(12+37)
49
>>> print('python'+ ' is exciting.')
python is exciting.
>>> |
```

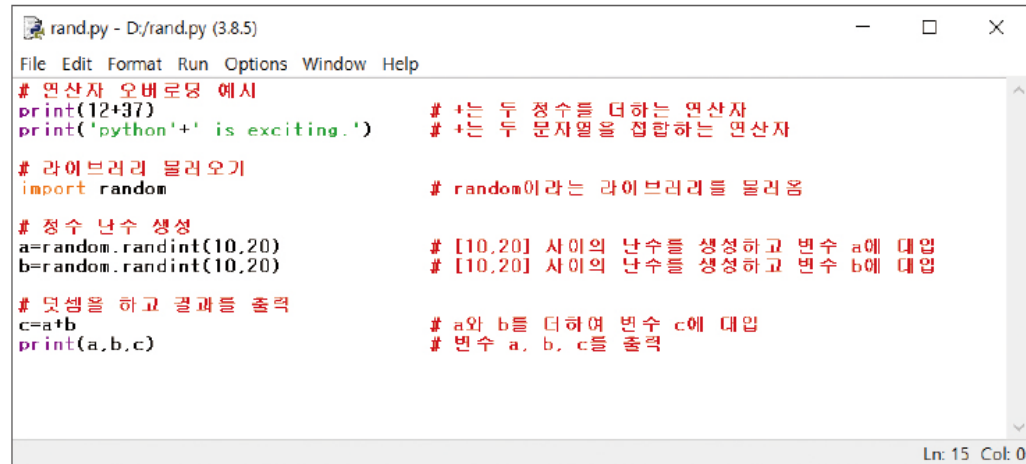
The status bar at the bottom right shows 'Ln: 9 Col: 4'.

그림 2-5 파이썬에서 간단한 명령어 수행(셀 창에서 작업)

2.2.3 파이썬 시작하기: 스탠드얼론 방식

■ 스크립트 창에서 프로그래밍

- 셸 창에서 [File]-[New File] 메뉴로 스크립트 창을 열 수 있음
- 스크립트 창에서는 프로그램을 입력하고 여러 번 실행 가능



```
# 연산자 오버로딩 예시
print(12+37)
print('python'+ ' is exciting.')

# 라이브러리 불러오기
import random

# 정수 난수 생성
a=random.randint(10,20)
b=random.randint(10,20)

# 덧셈을 하고 결과를 출력
c=a+b
print(a,b,c)


# +는 두 정수를 더하는 연산자
# +는 두 문자열을 접합하는 연산자

# random이라는 라이브러리를 불러옴

# [10,20] 사이의 난수를 생성하고 변수 a에 대입
# [10,20] 사이의 난수를 생성하고 변수 b에 대입

# a와 b를 더하여 변수 c에 대입
# 변수 a, b, c를 출력
```

(a) 스크립트 창에서 [프로그램 2-1]을 입력하고 실행



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 12+37
49
>>> print(12+37)
49
>>> print('python'+ ' is exciting.')
python is exciting.
>>>
===== RESTART: D:/rand.py =====
49
python is exciting.
12 17 29
>>>
```

(b) 셸 창에서 프로그램 실행 결과를 확인

그림 2-6 스크립트 창에서 간단하게 작성한 프로그램

2.2.4 클라우드 방식과 스탠드얼론 방식의 선택 기준

■ 간편한 실습에서는 colab

- 특히 단기 프로그래밍 과정에서는 설치 과정에 대한 시간 절약

■ 본격적인 개발에서는 스탠드얼론 방식

- 특히 인공지능 공학자로 성장하려 하는 경우에는 스스로 최적 환경을 구축할 수 있는 스탠드얼론 방식이 적절함

2.3 영리한 프로그래밍 환경: 아나콘다

■ 라이브러리 충돌 가능성

- 오픈 소스 특성으로 인해 라이브러리 충돌 가능성 발생(함수 API 변경 또는 환경 변수 값 충돌 등)
- 가상 환경을 사용하여 해결

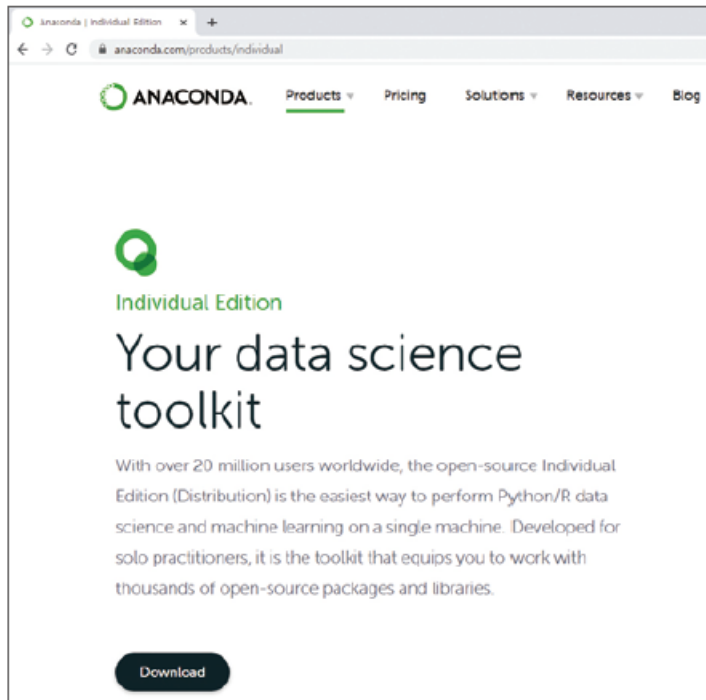
■ 가상 환경

- 프로젝트 별로 별도의 가상 환경을 만들어 일관된 환경 유지
- 아나콘다는 가상 환경을 지원

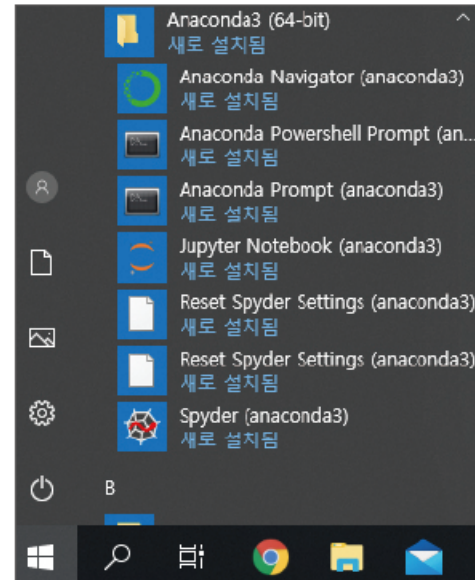
2.3.1 이 책에서 사용하는 프로그래밍 환경

■ 네 단계의 설치

1. 아나콘다를 설치한다. <http://anaconda.com>



(a) 아나콘다 다운로드 화면



(b) 윈도우 시작 버튼으로 설치 확인

그림 2-7 아나콘다 설치

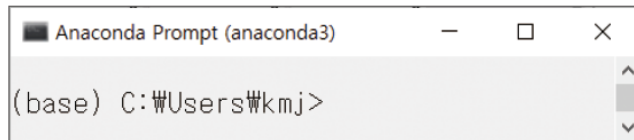
2.3.1 이 책에서 사용하는 프로그래밍 환경

■ 네 단계의 설치

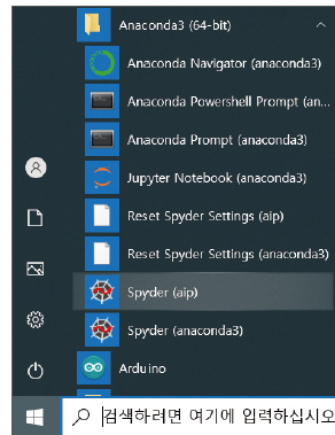
2. 가상 환경을 생성한다(아래 ① 번 명령어).

[aip라는 새로운 가상 환경을 만들고 스파이더와 텐서플로를 설치하는 과정]

(base) C:/> conda create -n aip python=3.7	① aip 가상 환경 생성
(base) C:/> conda activate aip	② aip 가상 환경으로 이동
(aip) C:/> conda install spyder	③ aip 가상 환경에 스파이더 설치
(aip) C:/> conda install tensorflow	④ aip 가상 환경에 텐서플로 설치



(a) 아나콘다의 프롬프트 창



(b) 윈도우의 시작 화면

그림 2-8 아나콘다 프롬프트 화면에서 가상 환경 만들기

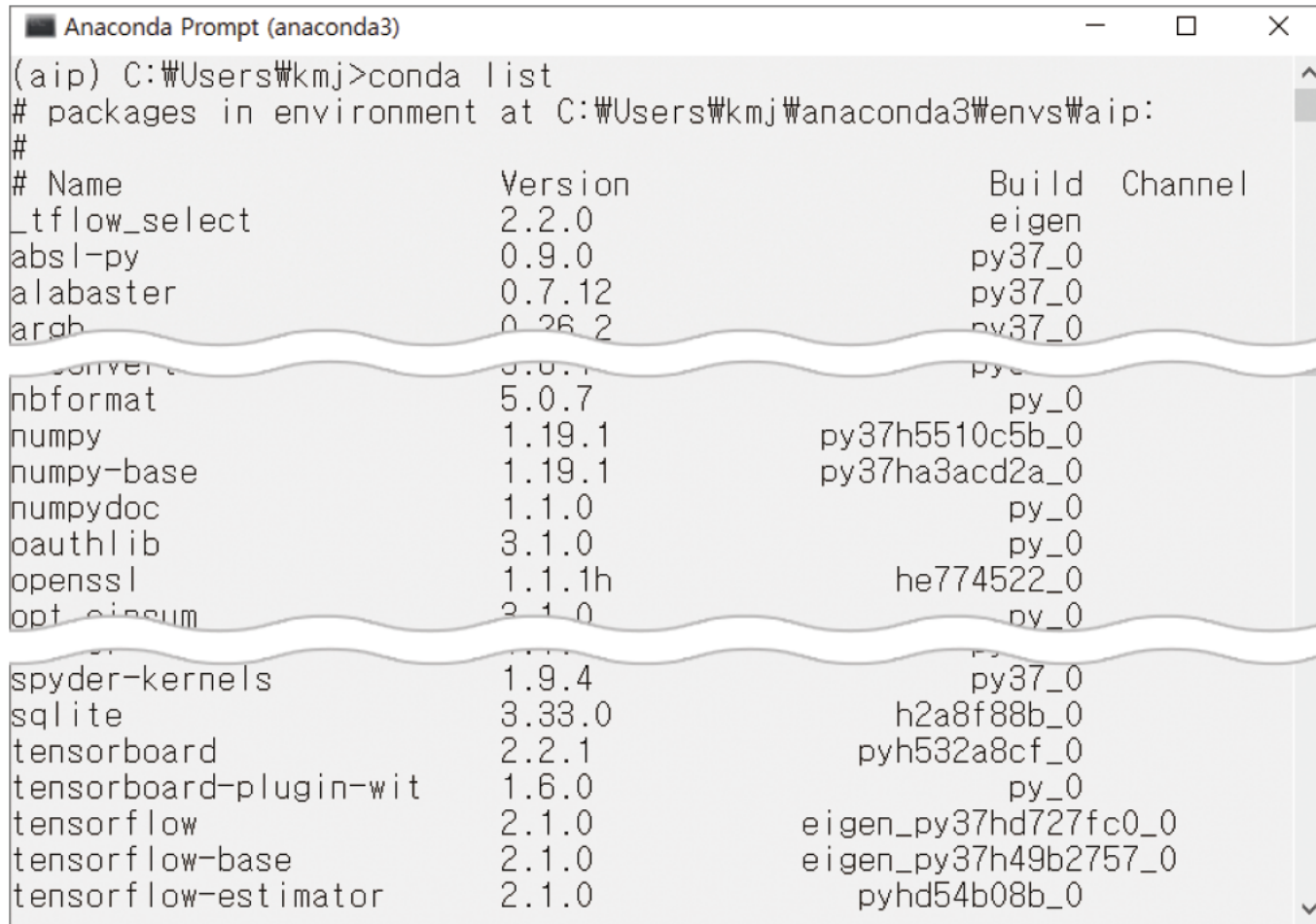
2.3.1 이 책에서 사용하는 프로그래밍 환경

■ 네 단계의 설치

3. 가상 환경에 Spyder와 텐서플로 모듈을 설치한다(2 3 4 번 명령어).
4. 스파이더에서 작업 디렉토리를 설정한다.
 - 바탕 화면에 aipSources 폴더 만들기
 - 스파이더에서 [Tools]-[Preferences]-[Current working directory]-[Startup]-[The following directory]를 선택하고 aipSources를 브라우징하여 설정

2.3.1 프로그래밍 환경 확인

- > conda list 명령어로 가상 환경에 설치된 모듈 목록 확인



```
(aip) C:\Users\Wkmj>conda list
# packages in environment at C:\Users\Wkmj\anaconda3\envs\aip:
#
# Name                                Version                                Build      Channel
_libflow_select                       2.2.0                                eigen
absl-py                              0.9.0                                py37_0
alabaster                             0.7.12                               py37_0
argb                                  0.26.2                               py37_0
nbformat                             5.0.7                                py_0
numpy                                 1.19.1                               py37h5510c5b_0
numpy-base                           1.19.1                               py37ha3acd2a_0
numpydoc                             1.1.0                                py_0
oauthlib                             3.1.0                                py_0
openssl                              1.1.1h                               he774522_0
opt_einsum                           2.1.0                                py_0
spyder-kernels                       1.9.4                                py37_0
sqlite                               3.33.0                               h2a8f88b_0
tensorboard                          2.2.1                               pyh532a8cf_0
tensorboard-plugin-wit               1.6.0                                py_0
tensorflow                           2.1.0                               eigen_py37hd727fc0_0
tensorflow-base                      2.1.0                               eigen_py37h49b2757_0
tensorflow-estimator                 2.1.0                               pyhd54b08b_0
```

그림 2-9 conda list 명령어로 모듈이 제대로 설치되었는지 확인

2.3.1 프로그래밍 환경 확인

■ 아나콘다 내비게이터 화면

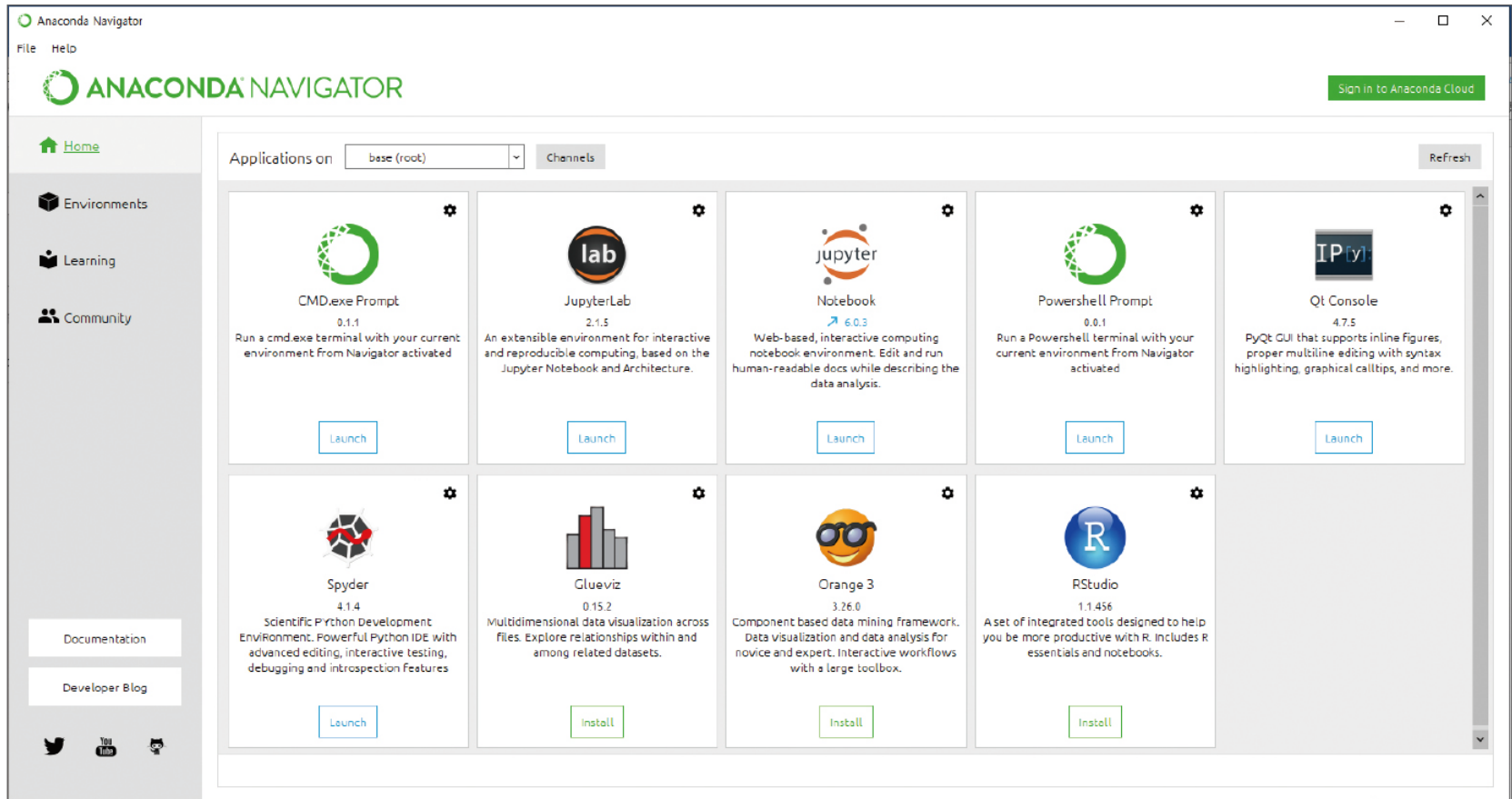


그림 2-10 아나콘다 내비게이터

2.3.2 스파이더로 프로그래밍 하기

- [Spyder (aip)] 아이콘 클릭
 - 아이콘을 윈도우의 작업 표시줄에 설정해 두면 편리
- 스파이더 화면

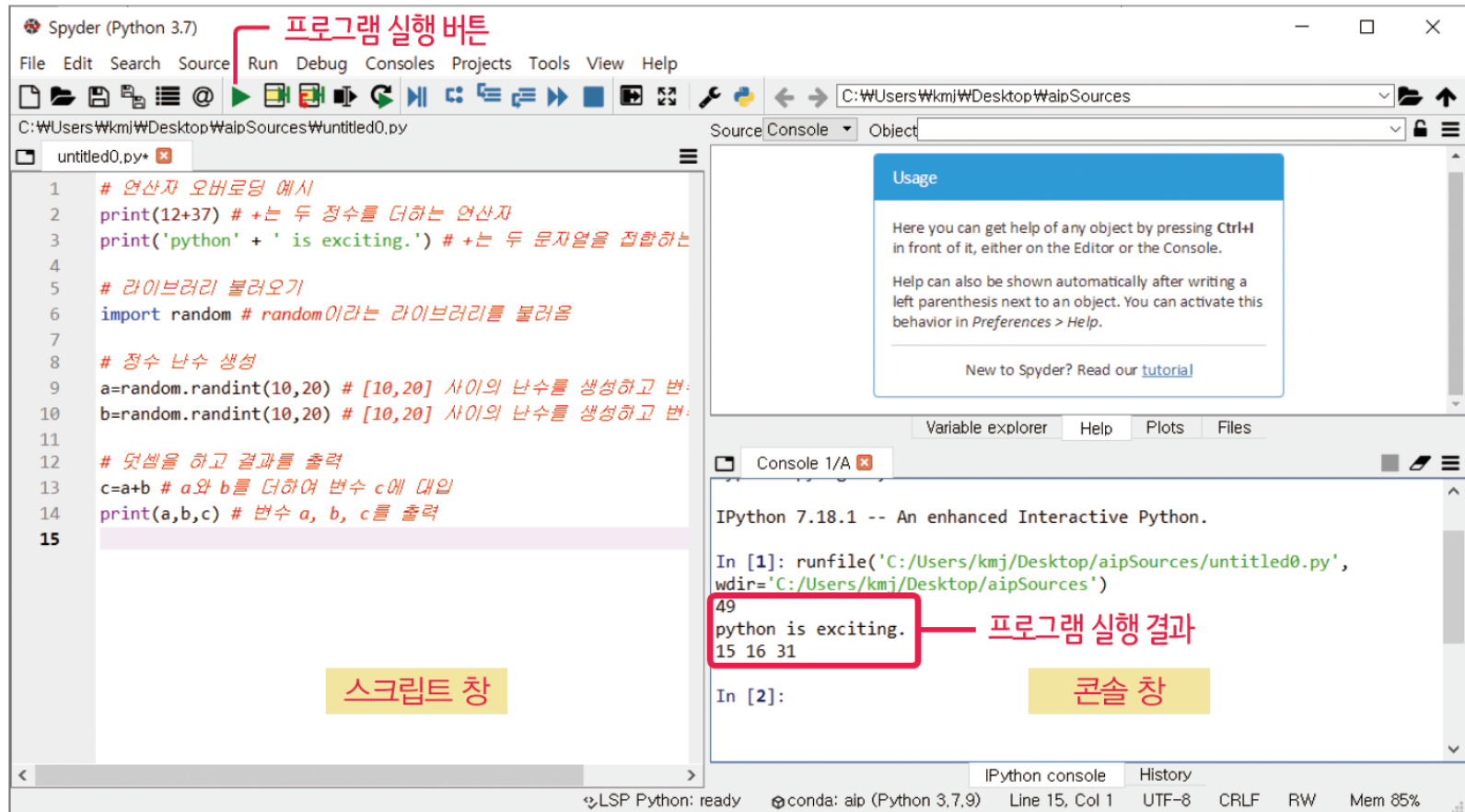


그림 2-11 스파이더의 통합 개발 환경