세상의 속도를
따라잡고 싶다면

Do it!

프로그래밍 분야 4년 연속
압도적 1위!
★★★★
전면
개정판

이미 200만 명이 '점프 투 파이썬'으로 프로그래밍을 시작했다!

점프 투 파이썬

키보드만 잡으면 중학생도, 문과생도 쉽게 배운다!

독자들과 소통하며 쌓은 13년 내공으로, 초보자 눈높이에 맞춘 입문서!

1일 평균 2만 뷰!
누적 방문자 200만 명!

위키독스 운영자
박응용 지음

파이썬 3 기준!
3.7 최신판 반영!

이지스퍼블리싱

"Life is too short,
You need Python!"

인생은 너무 짧으니,
파이썬이 필요해!

1일 평균 2만 뷰, 누적 방문자 수 200만 명!
베스트셀러 1위! '점프 투 파이썬'으로 프로그래밍을 시작하자!

프로그래밍을 처음 공부하려는 사람들이 배울 수 있는 프로그래밍 언어는 상당히 많다. 하지만 처음 배울 언어를 선택할 때는 약간의 주의가 필요하다. 너무 어려운 언어를 선택하거나 특정 기술에 특화된 언어를 선택할 경우 자칫 잘못하면 '우물 안 개구리'가 될 수도 있기 때문이다.
이 책을 들고 있는 여러분은 아마 파이썬의 명성에 대해 들어본 적이 있을 것이다. 많은 사람들이 초보자가 배우기 좋은 언어로 파이썬을 추천한다. 처음 프로그래밍의 세계에 발을 들인 사람들이 어려운 문법 때문에 포기하는 경우가 많은데, 파이썬은 프로그래밍의 핵심 개념을 아주 쉽게 배울 수 있기 때문이다. 파이썬은 이제 C, C++, 자바 등과 어깨를 나란히 할 만큼 유명한 언어가 되었지만 파이썬이 자바보다 더 오래된 언어라는 것을 아는 사람은 드물다. 파이썬은 혜성처럼 갑자기 등장해서 유명해진 스타 언어가 아닌, 그 역사가 매우 오래되어 숙성된 언어이기도 하다.

13년 이상 독자들과 함께 진화한 파이썬의 끝판왕, 전면 개정판으로 돌아왔다

이 책 역시 그 역사가 오래된 책이다. 이 책의 초판 격인 '점프 투 파이썬'이 처음 세상에 선을 보인 것이 2001년이니 무려 20살 가까이 되었다. 파이썬이 버전 업을 하며 진화해 가는 동안 이 책 또한 '위키독스(wikidocs.net)'라는 온라인 사이트에서 파이썬의 변화와 독자들의 요구에 발맞추어 진화를 거듭해 왔다. 그래서 이 책은 필자 한 사람에 의해 만들어진 책이 아니다. 위키독스의 '점프 투 파이썬'에 달린 무수한 댓글을 보면 알 수 있다. 댓글을 통해 독자들이 어려워하는 부분을 찾아 더욱 알기 쉽게 풀어 썼고, 이해하기 힘든 부분이 없는지 다시 한번 살펴볼 수 있었다. 2016년 《Do it! 점프 투 파이썬》 출간 이후부터는 오프라인 독자와의 소통도 더 해졌다. 이번 전면 개정판은 이렇게 더 넓어진 독자와의 소통으로 만들어 낸 진화의 결과물이다.

사례를 통해 개념을 이해했다면 예제는 직접 키보드를 치며 꼭 실습해 보자

'돈이 있으면 택시를 타고, 없으면 걸어가야겠다!'라는 생각은 누구나 한 번쯤 해봤을 것이다. 이 문장은 우리가 앞으로 배울 'if문'이라는 문법을 사용하여 프로그래밍할 수 있다. 이 책에서는 이렇게 실생활에서 쉽게 접할 수 있는 일들을 사례로 들어 독자들이 프로그래밍에 더 쉽게 접근할 수 있도록 설명하였다. 또 이번 전면 개정판에서는 독자들의 요구에 따라 실습 문제를 두 배로 늘렸다. '백문이 불여일견, 백견이 불여일타(打)'라 했다. 직접 키보드를 치며 따라 할수록 머릿속에 잘 남을 것이다.

감사의 말씀을 전하며…

'점프 투 파이썬'이 온라인상에서 계속 공개될 수 있도록 도움을 주신 이지스퍼블리싱 이지연 대표님과 책의 내용을 초보자의 입장에서 이해하기 쉽게 만들어 준 편집자 홍연의 씨, 한승우 씨에게 감사의 마음을 전하고 싶다.
이번 원고를 면밀히 검토하면서 자기도 모르게 파이썬 고수가 되어버린 나의 아내 김선정, 그리고 아빠가 책을 쓸 때면 조용히 옆에서 격려해 주었던 아들 박민규에게도 고마운 마음을 전하고 싶다. 마지막으로 오랜 시간 동안 이 책을 검토하고 읽어주신 '점프 투 파이썬'의 독자 여러분 모두에게 무한한 감사를 전한다.

박응용 드림
pahkey@gmail.com

'점프 투 파이썬' 열렬한 애독자의 뜨거운 찬사!

"친구와 선배들이 입에서 입으로 추천하는 책"

다른 프로그래밍 언어를 공부하다 보면 거창한 용어들에 먼저 주눅이 들곤 했었는데 '점프 투 파이썬'은 이런 부담을 전혀 느끼지 않고 공부할 수 있었습니다. - 신의규 님

대단히 친절한 설명 감사합니다. 처음 배우는 사람들이 어떤 부분에서 헷갈려 하는지 잘 알고 계신 듯합니다. 프로그래머들의 유머 감각 또한 빠질 수 없죠. - wangchobo 님

저는 이번에 파이썬이라는 말도 처음 들어본, 완전 초짜인데요. 이렇게 좋은 책이 있다는 게 너무 감사합니다. - 김영국 님

누군가 파이썬 입문에 대해 물어보면 항상 이 책을 추천하고 있습니다. 그 어떤 입문서보다도 접근하기 쉽고 정리가 잘 되어 있기 때문입니다. - space 님

저는 비전공자인데다가 컴퓨터 언어라는 것을 공부한 지 이제 4~5일 되었어요. 프로그래밍이 이렇게 재미있다는 것을 느끼게 해준 고마운 책입니다. - Minsu 님

예제와 연습문제를 여러 번 풀다 보니 파이썬에 자신감이 생겼습니다. 이제 웹 크롤링과 데이터 수집, 가공에 대해서 고민해야 하는 단계까지 왔습니다. - vin님

가독성도 좋아 전공자가 마음만 먹으면 10일 이내로 끝낼 수 있는 책. 전 라즈베리파이 공부를 위해 파이썬을 빠르게 훑는 용도로 사용했습니다. 정말 좋은 책입니다. - ChapterF*** 님

이 책이 입문서로는 압도적 지지를 받는 거 같아서 구입했어요. - terrapin*** 님

학교 선배님으로부터 추천 받아서 미리 예습하고 있는데 이보다 더 좋은 책은 없을 것 같아요. - je**sais59 님

User friendly Interface랄까요. 책이 군더더기 없이 깔끔하고, 내용이 간결하여 이해가 쉬웠습니다. - gdc** 님

:: 독학 및 왕초보 30일 진도표

하루에 한 시간씩 한 달 공부하면 누구나 파이썬 프로그래밍 초보를 탈출할 수 있도록 구성했습니다. 진도표에 여러분이 공부할 날짜를 기록하며 계획을 세워 보세요.

1일 차 (월 일) 01장	2일 차 (월 일) 02-1~02-2	3일 차 (월 일) 02-3~02-4	4일 차 (월 일) 02-5~02-6	5일 차 (월 일) 02-7	6일 차 (월 일) 02-8	
7일 차 (월 일) 02장 연습문제	8일 차 (월 일) 03-1	9일 차 (월 일) 03-2	10일 차 (월 일) 03-3	11일 차 (월 일) 03장 연습문제		
12일 차 (월 일) 04-1	13일 차 (월 일) 04-2	14일 차 (월 일) 04-3	15일 차 (월 일) 04장 연습문제	16일 차 (월 일) 05-1		
17일 차 (월 일) 05-2	18일 차 (월 일) 05-3	19일 차 (월 일) 05-4	20일 차 (월 일) 05-5	21일 차 (월 일) 05-6		
22일 차 (월 일) 05장 연습문제	23일 차 (월 일) 첫째마당 복습	24일 차 (월 일) 06-1~06-3	25일 차 (월 일) 06-4~06-6	26일 차 (월 일) 07-1		
27일 차 (월 일) 07-2	28일 차 (월 일) 07-3	29일 차 (월 일) 둘째마당 복습	30일 차 (월 일) 코딩 면허 시험			

중요해요!

파이썬의 핵심!

:: 교재 및 중급자 16회 진도표

책 한 권으로 한 학기 수업 효과를! 프로그래밍 경험이 있는 사람이라면 16회 강의 코스를 16일 코스로도 빠르게 끝낼 수 있습니다.

회	진행	계획한 날짜	완료 날짜
1회차	01장 파이썬이란 무엇인가?	(/)	(/)
2회차	02장 파이썬 프로그래밍의 기초, 자료형 - 1	(/)	(/)
3회차	02장 파이썬 프로그래밍의 기초, 자료형 - 2	(/)	(/)
4회차	03장 프로그램의 구조를 쌓는다! 제어문 - 1	(/)	(/)
5회차	03장 프로그램의 구조를 쌓는다! 제어문 - 2	(/)	(/)
6회차	04장 프로그램의 입력과 출력은 어떻게 해야 할까? - 1	(/)	(/)
7회차	04장 프로그램의 입력과 출력은 어떻게 해야 할까? - 2	(/)	(/)
8회차	중간 점검	배운 내용을 점검해 보세요	
9회차	05장 파이썬 날개 달기 - 1	(/)	(/)
10회차	05장 파이썬 날개 달기 - 2	(/)	(/)
11회차	05장 파이썬 날개 달기 - 3	(/)	(/)
12회차	06장 파이썬 프로그래밍, 어떻게 시작해야 할까? - 1	(/)	(/)
13회차	06장 파이썬 프로그래밍, 어떻게 시작해야 할까? - 2	(/)	(/)
14회차	07장 정규 표현식 - 1	(/)	(/)
15회차	07장 정규 표현식 - 2	(/)	(/)
16회차	마지막 점검	배운 내용을 점검해 보세요	

첫째 마당 파이썬 기본 문법 익히기

둘째 마당

파이썬 실력 키우기

이렇게 공부하세요

《Do it! 점프 투 파이썬》 실습 환경 안내!

책을 순서대로 보면 실습 방법은 자연스럽게 배울 수 있습니다. 책을 중간중간 보려는 분들은 아래 실습 환경을 먼저 읽고 시작하세요.

간단한 실습은 파이썬 대화형 인터프리터에서!

▶ 경로: [시작 → 프로그램 → Python 3.7 → Python 3.7(32-bit)]

변수에 문자 대입하고 출력하기

프롬프트(>>>)가 있으면 대화형 인터프리터에서 실습!

```
>>> a = "Python"
>>> print(a)
Python
```

프로그램 작성 및 저장은 파이썬 IDLE 에디터에서!

▶ 경로: [시작 → 프로그램 → Python 3.7 → IDLE]

우리는 이미 다음과 같은 프로그램을 C:\doit 디렉터리에 hello.py라는 이름으로 저장했다.

프롬프트(>>>)가 없으면 에디터에서 실습!

```
# hello.py
print("Hello world")
```

만든 프로그램 실행은 명령 프롬프트 창에서!

▶ 경로: [윈도우 키 + R → 실행 창에 'cmd' 입력]

실습 배경이 회색이면 명령 프롬프트 창 실행!

```
C:\doit> python hello.py
Hello World
```

책을 통해 스스로 발전하는 지적인 독자를 만나 보세요 — Do it! 스터디룸 카페

혼자 공부하면 질문할 곳이 마땅치 않아 공부 의욕이 떨어지기 쉽습니다. Do it! 스터디룸에서 같은 책을 공부하는 동료들을 만나 보세요! 서로 질문과 답변, 그리고 응원을 나누다 보면 공부가 더 즐거워질 것입니다.

https://cafe.naver.com/doitstudyroom

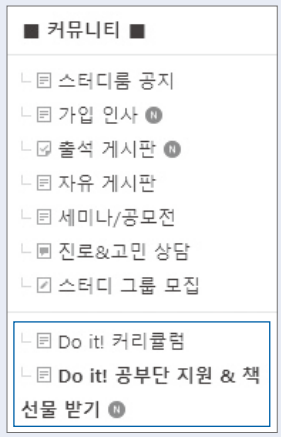
이 책으로 공부하고 책 선물도 받고! — Do it! 공부단 상시 모집 중

혼자 공부하면 계획을 세워도 잘 지켜지지 않죠? Do it! 스터디룸에서 운영하는 공부단에 지원해 보세요! 공부단에서는 자기 목표를 공유하고 매일 공부한 내용을 스터디 노트로 작성합니다. 꾸준히 공부하기 훨씬 수월하겠죠? 스터디 노트를 쓰며 책을 완독하면 원하는 이지스퍼블리싱 책 1권을 선물로 드립니다! 자세한 공부단 지원 및 진행 방법은 아래 설명을 참고해 주세요.

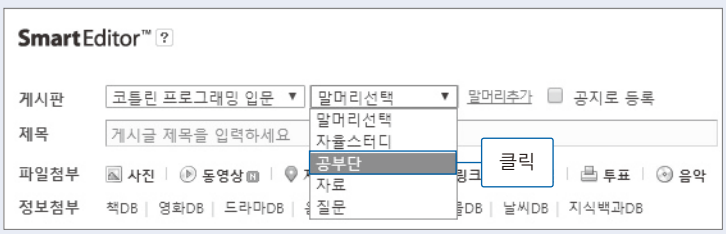
Do it! 스터디룸 카페 회원 가입 필수, 회원 등급 두잇 독자(게시글 1, 댓글 10, 출석 2회)부터 이용 가능

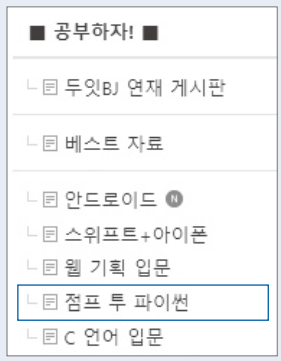
1. Do it! 스터디룸 카페에 방문하면 '■커뮤니티■' 메뉴에 'Do it! 공부단 지원 & 책 선물 받기' 게시판이 있습니다. 게시판에 입장하여 [글쓰기]를 누른 다음 공부단 지원 글 양식에 맞춰 공부단에 지원해 주세요. 자세한 방법은 아래 링크를 참고하세요.

https://cafe.naver.com/doitstudyroom/



2. 스터디 노트는 '■공부하자!■' 메뉴의 게시판에 업로드합니다. 이때 스터디 노트의 말머리를 반드시 [공부단]으로 설정해야 합니다. 꼭 기억하세요!





파이썬 초보 탈출
코딩 면허 시험 20제

파이썬은 웹, GUI, 네트워크, 딥러닝 등 상당히 많은 일을 할 수 있는 언어이다. 여러분이 지금까지 배운 내용을 충분히 숙지했다면 이제 이들을 향해 첫발을 내디딜 준비를 마친 것이다. 하지만 그전에 여기에 준비한 문제들을 풀어 보면서 여러분이 얼마나 파이썬에 익숙해졌는지 점검해 보도록 하자.

이곳에 준비한 문제들은 조금 어려울 수 있다. 하지만 파이썬과 함께라면 이 문제들을 해결하는 과정 역시 또 하나의 즐거움이라는 것을 분명 느끼게 될 것이다.

그럼 아무쪼록 즐거운 시간이 되기를 바란다. Happy Python!!

Q1 문자열 바꾸기 ★☆☆

다음과 같은 문자열이 있다.

```
a:b:c:d
```

문자열의 split와 join 함수를 사용하여 위 문자열을 다음과 같이 고치시오.

```
a#b#c#d
```

Q2 딕셔너리 값 추출하기 ★☆☆

다음은 딕셔너리의 a에서 'C'라는 key에 해당하는 value를 출력하는 프로그램이다.

```
>>> a = {'A':90, 'B':80}
>>> a['C']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'C'
```

a 딕셔너리에는 'C'라는 key가 없으므로 위와 같은 오류가 발생한다. 'C'에 해당하는 key 값이 없을 경우 오류 대신 70을 얻을 수 있도록 수정하시오.

Q3 리스트의 더하기와 extend 함수 ★★☆

다음과 같은 리스트 a가 있다.

```
>>> a = [1, 2, 3]
```

리스트 a에 [4, 5]를 + 기호를 사용하여 더한 결과는 다음과 같다.

```
>>> a = [1, 2, 3]
>>> a = a + [4,5]
>>> a
[1, 2, 3, 4, 5]
```

리스트 a에 [4,5]를 extend를 사용하여 더한 결과는 다음과 같다.

```
>>> a = [1, 2, 3]
>>> a.extend([4, 5])
>>> a
[1, 2, 3, 4, 5]
```

+ 기호를 사용하여 더한 것과 extend한 것의 차이점이 있을까? 있다면 그 차이점을 설명하시오.

Q4 리스트 총합 구하기 ★★☆

다음은 A학급 학생의 점수를 나타내는 리스트이다. 다음 리스트에서 50점 이상 점수의 총합을 구하시오.

```
A = [20, 55, 67, 82, 45, 33, 90, 87, 100, 25]
```

Q5 피보나치 함수 ★★★

첫 번째 항의 값이 0이고 두 번째 항의 값이 1일 때, 이후에 이어지는 항은 이전의 두 항을 더한 값으로 이루어지는 수열을 피보나치 수열이라고 한다.

```
0, 1, 1, 2, 3, 5, 8, 13 …
```

입력을 정수 n으로 받았을 때, n 이하까지의 피보나치 수열을 출력하는 함수를 작성해 보자.

Q6 숫자의 총합 구하기 ★★☆

사용자로부터 다음과 같은 숫자를 입력받아 입력받은 숫자의 총합을 구하는 프로그램을 작성하시오. (단 숫자는 콤마로 구분하여 입력한다.)

```
65,45,2,3,45,8
```


Q7 한 줄 구구단 ★★☆

사용자로부터 2~9의 숫자 중 하나를 입력받아 해당 숫자의 구구단을 한 줄로 출력하는 프로그램을 작성하시오.

실행 예)

```
구구단을 출력할 숫자를 입력하세요(2~9): 2
2 4 6 8 10 12 14 16 18
```

Q8 역순 저장 ★☆☆

다음과 같은 내용의 파일 abc.txt가 있다.

```
AAA
BBB
CCC
DDD
EEE
```

이 파일의 내용을 다음과 같이 역순으로 바꾸어 저장하시오.

```
EEE
DDD
CCC
BBB
AAA
```

Q9 평균값 구하기 ★★★

오른쪽과 같이 총 10줄로 이루어진 sample.txt 파일이 있다. sample.txt 파일의 숫자 값을 모두 읽어 총합과 평균 값을 구한 후 평균 값을 result.txt 파일에 쓰는 프로그램을 작성하시오.

```
70
60
55
75
95
90
80
80
85
100
```

Q10 사칙연산 계산기 ★★☆

다음과 같이 동작하는 클래스 Calculator를 작성하시오.

```
>>> cal1 = Calculator([1,2,3,4,5])
>>> cal1.sum() # 합계
15
>>> cal1.avg() # 평균
3.0
>>> cal2 = Calculator([6,7,8,9,10])
>>> cal2.sum() # 합계
40
>>> cal2.avg() # 평균
8.0
```

Q11 모듈 사용 방법 ★★★

C:\doit 디렉터리에 mymod.py 파이썬 모듈이 있다고 가정해 보자. 명령 프롬프트 창에서 파이썬 셸을 열어 이 모듈을 import해서 사용할 수 있는 방법을 모두 기술하시오.
(즉 다음과 같이 import mymod를 수행할 때 오류가 없어야 한다.)

```
>>> import mymod
>>>
```


Q12 오류와 예외 처리 ★☆☆

다음 코드의 실행 결과를 예측하고 그 이유에 대해 설명하시오.

```
result = 0
try:
    [1, 2, 3][3]
    "a"+1
    4 / 0
except TypeError:
    result += 1
except ZeroDivisionError:
    result += 2
except IndexError:
    result += 3
finally:
    result += 4

print(result)
```

Q13 DashInsert 함수 ★★☆

DashInsert 함수는 숫자로 구성된 문자열을 입력받은 뒤 문자열 안에서 홀수가 연속되면 두 수 사이에 - 를 추가하고, 짝수가 연속되면 * 를 추가하는 기능을 갖고 있다. DashInsert 함수를 완성하시오.

입력 예시: 4546793
출력 예시: 454*67-9-3

Q14 문자열 압축하기 ★★☆

문자열을 입력받아 같은 문자가 연속적으로 반복되는 경우에 그 반복 횟수를 표시해 문자열을 압축하여 표시하시오.

입력 예시: aaabbccccccca
출력 예시: a3b2c6a1

Q15 Duplicate Numbers ★★☆

0~9의 문자로 된 숫자를 입력받았을 때, 이 입력값이 0~9의 모든 숫자를 각각 한 번씩만 사용한 것인지 확인하는 함수를 작성하시오.

> 입력 예시: 0123456789 01234 01234567890 6789012345 012322456789
> 출력 예시: true false false true false

Q16 모스 부호 해독 ★★★

문자열 형식으로 입력받은 모스 부호(dot: . dash:-)를 해독하여 영어 문장으로 출력하는 프로그램을 작성하시오.

- 글자와 글자 사이는 공백 1개, 단어와 단어 사이는 공백 2개로 구분한다.
- 예를 들어 다음 모스 부호는 "HE SLEEPS EARLY"로 해석해야 한다.

```
.... .  ... .-.. . . .--. ...  . .- .-. .-.. -.--
```

모스 부호 규칙 표

문자	부호	문자	부호
A	.-	N	-.
B	-...	O	---
C	-.-.	P	.--.
D	-..	Q	--.-
E	.	R	.-.
F	..-.	S	...
G	--.	T	-
H	....	U	..-
I	..	V	...-
J	.---	W	.--
K	-.-	X	-..-
L	.-..	Y	-.--
M	--	Z	--..

Q17 기초 메타 문자 ★☆☆

다음 중 정규식 a[.]{3,}b과 매치되는 문자열은 무엇일까?

1. acccb
2. a....b
3. aaab
4. a.cccb

Q18 문자열 검색 ★★☆

다음 코드의 결괏값은 무엇일까?

```
>>> import re
>>> p = re.compile("[a-z]+")
>>> m = p.search("5 python")
>>> m.start() + m.end()
```

Q19 그루핑 ★★★

다음과 같은 문자열에서 휴대폰 번호 뒷자리인 숫자 4개를 ####로 바꾸는 프로그램을 정규식을 사용하여 작성하시오.

```
"""
park 010-9999-9988
kim 010-9909-7789
lee 010-8789-7768
"""
```

Q20 전방 탐색 ★★★

다음은 이메일 주소를 나타내는 정규식이다. 이 정규식은 park@naver.com, kim@daum.net, lee@myhome.co.kr 등과 매치된다. 긍정형 전방 탐색 기법을 사용하여 .com, .net이 아닌 이메일 주소는 제외시키는 정규식을 작성하시오.

```
.*[@].*[.].*$
```

"행동의 가치는 그 행동을 끝까지 이루는 데 있다."
- 칭기즈 칸

시험 풀이
345~354쪽

풀이

Q1 문자열 바꾸기

```
>>> a = "a:b:c:d"
>>> b = a.split(":")
>>> b
['a', 'b', 'c', 'd']
>>> c = "#".join(b)
>>> c
'a#b#c#d'
```

Q2 딕셔너리 값 추출하기

딕셔너리의 get 함수를 사용하면 해당 key가 없을 경우에는 두 번째 매개변수로 전달된 default 값을 대신 돌려준다.

```
>>> a = {'A':90, 'B':80}
>>> a.get('C', 70)
70
```

위 예에서는 'C'에 해당되는 key가 없기 때문에 디폴트 값으로 전달된 70을 돌려주었다.

Q3 리스트의 더하기와 extend 함수

리스트 a에 + 기호를 사용하는 경우에 대해서 먼저 살펴보자.

```
>>> a = [1, 2, 3]
>>> id(a)
4302429640
```

id 함수는 입력으로 받은 리스트 a의 주소 값을 돌려 준다. 현재 a라는 리스트는 4302429640이라는 주소에 저장되어 있다.

```
>>> a = a + [4,5]
>>> a
[1, 2, 3, 4, 5]
```


리스트 a에 + 기호를 사용하여 [4, 5]라는 리스트를 더해 보았다. 그리고 다시 다음과 같이 리스트 a의 주소 값을 확인해 보자.

```
>>> id(a)
4302472072
```

이전에 리스트 a가 저장되어 있던 주소와 다른 값을 돌려주는 것을 확인할 수 있다. 주소 값이 다르기 때문에 +를 사용하면 리스트 a의 값이 변하는 것이 아니라 두 리스트가 더해진 새로운 리스트가 반환된다는 것을 확인할 수 있다.

이번에는 extend 함수를 사용해 보자.

```
>>> a = [1, 2, 3]
>>> id(a)
4302429640
```

리스트 a를 생성하고 그 주소 값을 출력해 보았다.

```
>>> a.extend([4, 5])
>>> a
[1, 2, 3, 4, 5]
```

그리고 리스트 a에 extend를 사용하여 [4, 5]라는 리스트를 더해 주었다. 그리고 다시 다음과 같이 리스트 a의 주소 값을 확인해 보도록 하자.

```
>>> id(a)
4302429640
```

+ 기호를 사용하여 더한 경우와는 달리 주소 값이 변하지 않고 그대로 유지되는 것을 확인할 수 있다.

Q4 리스트 총합 구하기

```
A = [20, 55, 67, 82, 45, 33, 90, 87, 100, 25]

result = 0
while A:    # A 리스트에 값이 있는 동안
    mark = A.pop()    # A리스트의 가장 마지막 항목을 하나씩 뽑아냄
    if mark >= 50:    # 50점 이상의 점수만 더함
        result += mark

print(result)    # 481 출력
```

Q5 피보나치 함수

피보나치 수열은 다음과 같은 순서로 결괏값을 반환한다.

① fib(0) → 0 반환
② fib(1) → 1 반환
③ fib(2) → fib(0) + fib(1) → 0 + 1 → 1 반환
④ fib(3) → fib(1) + fib(2) → 1 + 1 → 2 반환
⑤ fib(4) → fib(2) + fib(3) → 1 + 2 → 3 반환
⑥ …

n이 0일 때는 0을 반환, 1일 때는 1을 반환한다. n이 2 이상일 경우에는 이전의 두 값을 더하여 반환한다.

재귀 호출을 사용하면 피보나치 함수를 다음과 같이 간단하게 작성할 수 있다.

```
def fib(n):
    if n == 0 : return 0              # n이 0일 때는 0을 반환
    if n == 1 : return 1              # n이 1일 때는 1을 반환
    return fib(n-2) + fib(n-1)        # n이 2 이상일 때는 그 이전의 두 값을 더하여 반환


for i in range(10):
    print(fib(i))
```

0부터 9까지의 피보나치 수열의 결괏값을 출력하여 그 값을 확인해 보았다.

Q6 숫자의 총합 구하기

```
user_input = input("숫자를 입력하세요: ")
numbers = user_input.split(",")
total = 0
for n in numbers:
    total += int(n)    # 입력은 문자열이므로 숫자로 변환해야 한다.
print(total)
```

수행결과

```
숫자를 입력하세요: 65,45,2,3,45,8
168
```


Q7 한 줄 구구단

```
user_input = input("구구단을 출력할 숫자를 입력하세요(2~9):")
dan = int(user_input)        # 입력 문자열을 숫자로 변환
for i in range(1, 10):
    print(i*dan, end= ' ')   # 한 줄로 출력하기 위해 줄 바꿈 문자 대신 공백 문자를 마지막에
                               출력한다.
```

Q8 역순 저장

파일 객체의 readlines를 사용하여 모든 라인을 읽은 후에 reversed를 사용하여 역순으로 정렬한 다음 다시 파일에 저장한다.

```
f = open('abc.txt', 'r')
lines = f.readlines()      # 모든 라인을 읽음
f.close()

lines.reverse()            # 읽은 라인을 역순으로 정렬

f = open('abc.txt', 'w')
for line in lines:
    line = line.strip()    # 포함되어 있는 줄 바꿈 문자 제거
    f.write(line)
    f.write('\n')          # 줄 바꿈 문자 삽입
f.close()
```

Q9 평균 값 구하기

```
f = open("sample.txt")
lines = f.readlines( )     # sample.txt를 줄 단위로 모두 읽는다.
f.close( )

total = 0
for line in lines:
    score = int(line)      # 줄에 적힌 점수를 숫자형으로 변환한다.
    total += score
average = total / len(lines)
```

```
f = open("result.txt", "w")
f.write(str(average))
f.close()
```

sample.txt의 점수를 모두 읽기 위해 파일을 열고 readlines를 사용하여 각 줄의 점수 값을 모두 읽어 들여 총 점수를 구한다. 총 점수를 sample.txt 파일의 라인(Line) 수로 나누어 평균 값을 구한 후 그 결과를 result.txt 파일에 쓴다. 숫자 값은 result.txt 파일에 바로 쓸 수 없으므로 str 함수를 사용하여 문자열로 변경한 후 파일에 쓴다.

Q10 사칙연산 계산기

```
class Calculator:
    def __init__(self, numberList):
        self.numberList = numberList

    def add(self):
        result = 0
        for num in self.numberList:
            result += num
        return result

    def avg(self):
        total = self.add()
        return total / len(self.numberList)


cal1 = Calculator([1,2,3,4,5])
print (cal1.add())
print (cal1.avg())

cal2 = Calculator([6,7,8,9,10])
print (cal2.add())
print (cal2.avg())
```


Q11 모듈 사용 방법

파이썬 셸에서 mymod.py 모듈을 인식하기 위해서는 다음과 같은 3가지 방법이 있을 수 있다.

1) sys 모듈 사용하기
다음과 같이 sys.path 에 C:₩doit 이라는 디렉터리를 추가하면 C:₩doit 이라는 디렉터리에 있는 mymod 모듈을 사용할 수 있게 된다.

```
>>> import sys
>>> sys.path.append("c:/doit")
>>> import mymod
```

2) PYTHONPATH 환경 변수 사용하기
다음처럼 PYTHONPATH 환경 변수에 C:₩doit 디렉터리를 지정하면 C:₩doit 디렉터리에 있는 mymod 모듈을 사용할 수 있게 된다.

```
C:₩Users₩home>set PYTHONPATH=c:₩doit
C:₩Users₩home>python
>>> import mymod
```

3) 현재 디렉터리 사용하기
파이썬 셸을 mymod.py가 있는 위치로 이동하여 실행해도 mymod 모듈을 사용할 수 있게 된다. 왜냐하면 sys.path 에는 현재 디렉터리인 . 이 항상 포함되어 있기 때문이다.

```
C:₩Users₩home>cd c:₩doit
C:₩doit>python
>>> import mymod
```

Q12 오류와 예외 처리

7이 출력된다.

1. result의 초깃값은 0이다.
2. try문 안의 [1, 2, 3][3] 이라는 문장 수행 시 IndexError가 발생하여 except IndexError: 구문으로 이동하게 되어 result에 3이 더해져 3이 된다.
3. 최종적으로 finally 구문이 실행되어 result에 4가 더해져 7이 된다.
4. print(result)가 수행되어 result의 최종 값인 7이 출력된다.

Q13 DashInsert 함수

다음 프로그램의 주석문을 참고하자.

```
data = "4546793"

numbers = list(map(int, data))    # 숫자 문자열을 숫자 리스트로 변경
result = []

for i, num in enumerate(numbers):
    result.append(str(num))
    if i < len(numbers)-1:                    # 다음 수가 있다면
        is_odd = num % 2 == 1                 # 현재 수가 홀수
        is_next_odd = numbers[i+1] % 2 == 1   # 다음 수가 홀수
        if is_odd and is_next_odd:            # 연속 홀수
            result.append("-")
        elif not is_odd and not is_next_odd:  # 연속 짝수
            result.append("*")

print("".join(result))
```

Q14 문자열 압축하기

먼저 입력 문자열의 문자를 확인하여 동일한 문자가 들어올 경우에는 해당 문자의 숫자 값을 증가시킨다. 만약 다른 문자가 들어올 경우에는 해당 문자의 숫자 값을 1로 초기화하는 방법을 사용하여 작성한 코드이다.

```
def compress_string(s):
    _c = ""
    cnt = 0
    result = ""
    for c in s:
        if c!=_c:
            _c = c
            if cnt: result += str(cnt)
            result += c
            cnt = 1
        else:
            cnt +=1
    if cnt: result += str(cnt)
    return result

print (compress_string("aaabbccccccca"))     # a3b2c6a1 출력
```


Q15 Duplicate Numbers

```
def chkDupNum(s):
    result = []
    for num in s:
        if num not in result:
            result.append(num)
        else:
            return False
    return len(result) == 10
print(chkDupNum("0123456789"))              # True 리턴
print(chkDupNum("01234"))                   # False 리턴
print(chkDupNum("01234567890"))             # False 리턴
print(chkDupNum("6789012345"))              # True 리턴
print(chkDupNum("012322456789"))            # False 리턴
```

리스트 자료형을 사용하여 중복된 값이 있는지 먼저 조사한다. 중복된 값이 있을 경우는 False를 리턴한다. 최종적으로 중복된 값이 없을 경우 0~9까지의 숫자가 모두 사용되었는지 판단하기 위해 입력 문자열의 숫자 값을 저장한 리스트 자료형의 총 개수가 10인지를 조사하여 10일 경우는 True, 아닐 경우는 False를 리턴한다.

Q16 모스 부호 해독

```
dic = {
    '.-':'A','-...':'B','-.-.':'C','-..':'D','.':'E','..-.':'F',
    '--.':'G','....':'H','..':'I','.---':'J','-.-':'K','.-..':'L',
    '--':'M','-.':'N','---':'O','.--.':'P','--.-':'Q','.-.':'R',
    '...':'S','-':'T','..-':'U','...-':'V','.--':'W','-..-':'X',
    '-.--':'Y','--..':'Z'
}

def morse(src):
    result = []
    for word in src.split("  "):
        for char in word.split(" "):
            result.append(dic[char])
        result.append(" ")
    return "".join(result)


print (morse('.... .  ... .-.. . . .--. ...  . .- .-. .-.. -.--'))
```

모스 부호 규칙 표를 딕셔너리로 작성한 후 입력에 해당되는 모스 부호 문자열을 먼저 단어(공백 문자 2개)로 구분한다. 그 후 단어(공백 문자 1개)를 문자로 구분하여 해당 모스 부호 값을 딕셔너리에서 찾아서 그 결괏값을 구한다.

Q17 기초 메타 문자

보기 중 이 조건에 해당되는 것은 B이다.
다음은 위 문제의 정규식 매치 결과를 확인해 보는 파이썬 코드이다.

```
import re

p = re.compile("a[.]{3,}b")

print (p.match("acccb"))        # None
print (p.match("a....b"))       # 매치 객체 출력
print (p.match("aaab"))         # None
print (p.match("a.cccb"))       # None
```

Q18 문자열 검색

정규식 '[a-z]+'은 소문자로 이루어진 단어를 뜻하므로 '5 python' 문자열에서 'python'과 매치될 것이다. 따라서 'python' 문자열의 시작 인덱스(m.start())는 2이고 마지막 인덱스(m.end())는 8이므로 10이 출력된다.

```
import re

p = re.compile('[a-z]+')
m = p.search("5 python")
print(m.start() + m.end())   # 10 출력
```

Q19 그루핑

전화번호 패턴은 다음과 같이 작성할 수 있다.

```
pat = re.compile("\d{3}[-]\d{4}[-]\d{4}")
```

이 전화번호 패턴 중 뒤의 숫자 4개를 변경할 것이므로 필요한 앞부분을 다음과 같이 그루핑한다.

```
pat = re.compile("(\d{3}[-]\d{4})[-]\d{4}")
```

컴파일된 객체 pat에 sub 함수를 사용하여 다음과 같이 문자열을 변경한다.

```
import re

s = """
park 010-9999-9988
kim 010-9909-7789
lee 010-8789-7768
"""

pat = re.compile("(\d{3}[-]\d{4})[-]\d{4}")
result = pat.sub("\g<1>-####", s)

print(result)
```

Q20 전방 탐색

.com과 .net에 해당되는 이메일 주소만을 매치하기 위해서 이메일 주소의 도메인 부분에 다음과 같은 긍정형 전방탐색 패턴을 적용한다.

```
pat = re.compile(".*[@].*[.](?=com$|net$).*$")
```

다음은 위 패턴을 적용한 파이썬 코드이다.

```
import re

pat = re.compile(".*[@].*[.](?=com$|net$).*$")

print(pat.match("pahkey@gmail.com"))
print(pat.match("kim@daum.net"))
print(pat.match("lee@myhome.co.kr"))
```