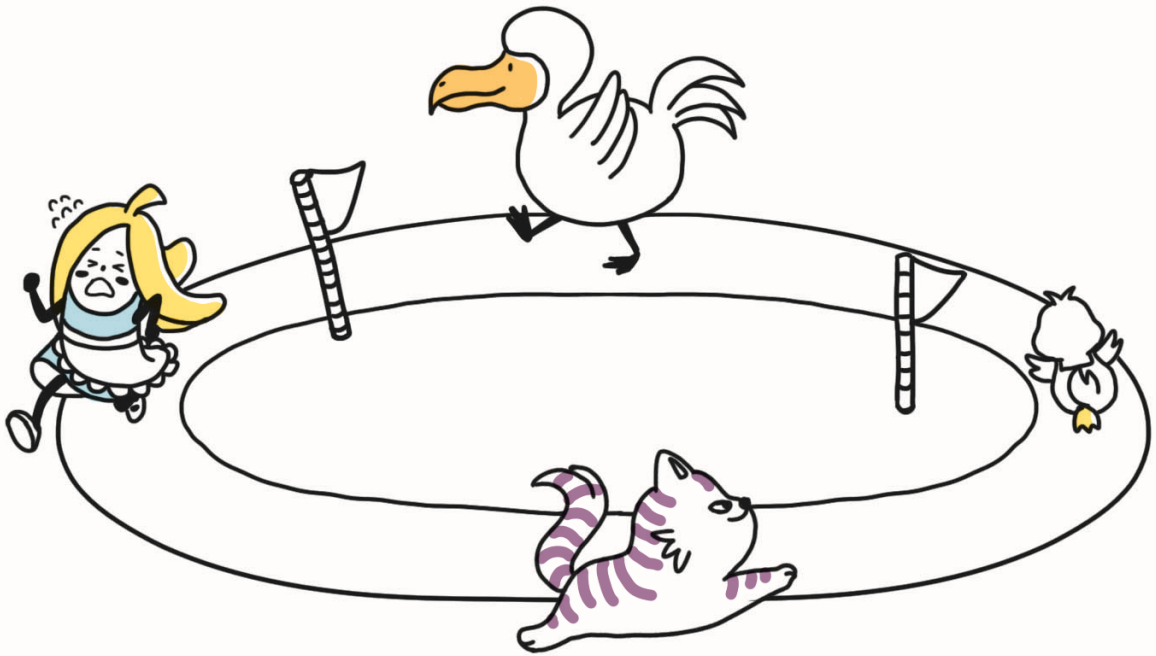
chapter 6

조건으로 반복하기

♠ 코커스 경주



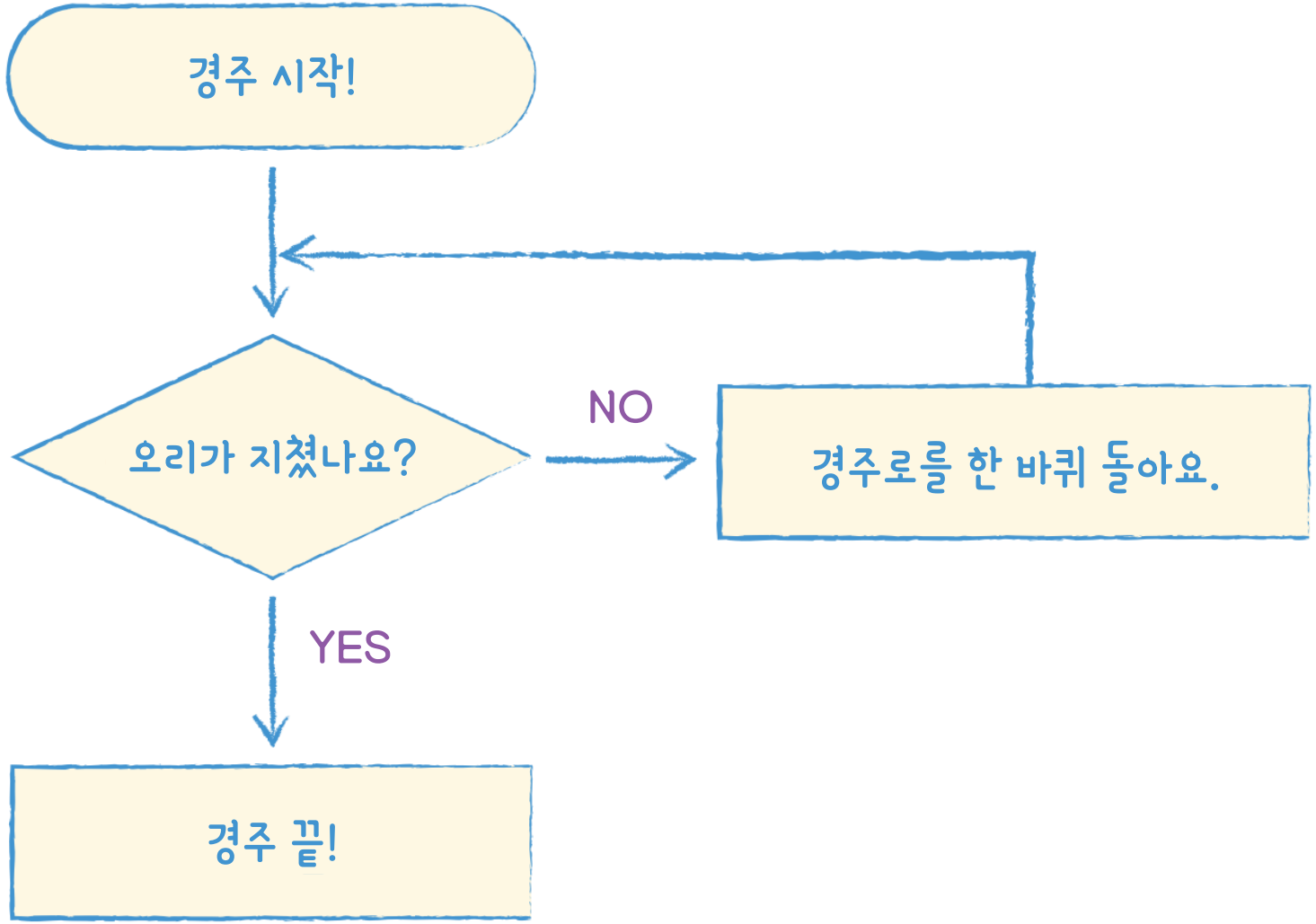
"코커스 경주가 뭐예요?"

코커스 경주를 하자는 도도새의 말을 듣고 있던 앨리스가 물었어요.

"뭐, 제일 좋은 방법은 직접 해보는 거지."

도도새는 먼저 둥그렇게 경주로를 그렸어요. 그런 다음
다들 선을 따라 여기저기 자리를 잡고 섰어요.
그리고 "하나, 둘, 셋, 출발!" 신호도 없이
자기 마음대로 달리기 시작했어요.

30분 정도 지났을까 가장 먼저 지쳐버린 오리가
달리기를 멈추자 도도새가 갑자기 소리쳤어요.
"경주 끝!"



while 기본 구조

```
while 조건:
    실행할_명령
```

들여쓰기

코드블록

코드 6-1 for를 사용해 거북이에게 인사하는 코드

```
01: for num in range(3):
02:     print('안녕 거북이', num)
```

↳ **실행화면**

코드 6-2 while을 사용해 거북이에게 인사하는 코드

```
01: num = 0
02: while num < 3:
03:     print('안녕 거북이', num)
04:     num = num + 1
```

↳ **실행화면**

6-1 이건 공평하지 않아!

"난 물갈퀴 때문에 빨리 달릴 수 없어! 이건 불공평해!"

화가 단단히 난 오리는 규칙을 바꾸지 않으면 경기에서 빠지겠다고 했어요. 다들 도도새 주변으로 몰려들더니 숨을 헐떡이며 물었어요.

"그럼 어떻게 하죠?"

그건 도도새로서도 쉽게 답할 수 있는 문제가 아니었어요. 그래서 도도새는 손가락 하나를 이마에 댄 채 한참을 앉아 있었어요. 마침내 도도새가 입을 열었어요.

"그럼 다섯 바퀴만 뛰는 거로 하지요."

- while의 코드블록을 다섯 번만 반복하게 하세요.
- 반복할 때마다 몇 번째 바퀴인지 출력하세요.
- 전부 다 돌면 '경주 끝!'을 외치세요.

코드 6-3 경주로를 다섯 바퀴만 돌게 하는 코드

```
01: count = 0
02:
03:
04:
05: print('경주 끝!')
```

↳ **실행화면**

```
1 번째 바퀴입니다.
2 번째 바퀴입니다.
3 번째 바퀴입니다.
4 번째 바퀴입니다.
5 번째 바퀴입니다.
경주 끝!
```

💘 다음 코드의 출력 결과를 적어보세요.

```
01: count = 0
02: while count < 3:
03:    print(count)
04:    count = count + 1
```

↳

💘 그림과 같은 실행화면을 출력하도록 빈칸을 채워보세요.

```
01: count = 1
02: while count < 4:
03:     count = count + 1
04:     print(count)
```

```
↳
```

💓 다음 코드의 출력 결과를 적어보세요.

```
01: count = 0
02: while count <= 5:
03:     if count % 2 != 0:
04:         print(count)
05:     count = count + 1
```

↳

💘 다음은 1부터 5까지의 총합을 구하는 코드입니다. 빈칸을 채워보세요.

```
01:  sum = 0
02:  count = 1
03:  while count <= 5:
04:
05:
06:  print('총합은', sum)
```

```
↳    총합은 15
```

💘 다음은 3부터 1까지 거꾸로 세는 코드입니다. 빈칸을 채워보세요.

```
01: count = 3
02:
03:
04:
```

```
↳   3
    2
    1
```

그림과 같은 실행화면을 출력하는 코드를 고르세요.

```
↳ 클로버1
  클로버2
  클로버3
```

```
01: clovers = ['클로버1', '클로버2', '클로버3']
02: cnt = 0
03: while cnt < 3:
04:     print(clovers[cnt])
05:     cnt = cnt + 1
```

```
01: clovers = ['클로버1', '클로버2', '클로버3']
02: cnt = 1
03: while cnt < 3:
04:     print(clovers[cnt])
05:     cnt = cnt + 1
```

```
01: clovers = ['클로버1', '클로버2', '클로버3']
02: for cnt in range(0, 2):
03:     print(clovers[cnt])
```

```
01: clovers = ['클로버1', '클로버2', '클로버3']
02: for cnt in range(1, 3):
03:     print(clovers[cnt])
```

💙 입력을 받는 방법

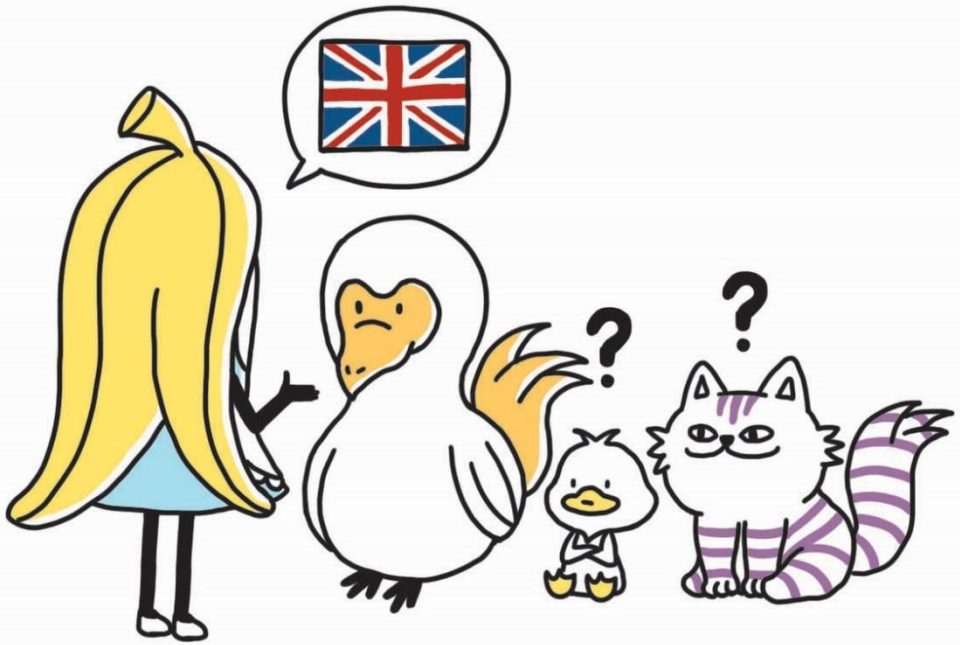
`input()`

코드 6-4 이름을 입력받는 코드

```
01:  name = input('이름이 뭔가요? ')
02:  print(name, '안녕!')
```

↳ **실행화면**

6-2 앨리스의 수수께끼



먼저 수수께끼 놀이를 하자고 한 건 체셔고양이였어요.
다들 입을 모아 물었어요.

"하지만 누가 수수께끼를 내죠?"
"그야, 물론 저 애지요."

체셔고양이가 손가락으로 앨리스를 가리켰어요.
그러자 모두 순식간에 앨리스를 에워싸더니
정신없이 떠들어 댔어요.

"수수께끼! 수수께끼!"

앨리스는 어쩔 줄 몰랐지만 언니와 했던
수도 이름 맞추기 놀이를 간신히 기억해 냈어요.

- 영국의 수도를 묻고 입력을 받아 answer에 저장하세요.
- 정답인 '런던'을 맞힐 때까지 반복해서 질문하세요.
- 정답을 맞히면 반복을 멈추세요.

코드 6-5 영국의 수도를 묻는 코드

```
01: answer = ''
02:
03:
04:
```

↳ **실행화면**

```
영국의 수도는 어디일까요? 서울
영국의 수도는 어디일까요? 파리
영국의 수도는 어디일까요? 런던
정답입니다.
```

💓 그림과 같은 실행화면을 출력하도록 빈칸을 채워보세요.

```
01: price = 0
02: while price != -1:
03:     price = int(input('가격을 입력하세요 (종료:-1): '))
04:
05:
06:
07:
08:
09:
```

```
↳   가격을 입력하세요 (종료:-1): 3000
    정말 싸요.
    가격을 입력하세요 (종료:-1): 15000
    너무 비싸요
    가격을 입력하세요 (종료:-1): 7500
    괜찮은 가격이네요.
    가격을 입력하세요 (종료:-1): -1
```

넘어가기와 멈추기

`continue`

`break`

코드 6-6 count가 2일 때 넘어가는 코드

```
01: count = 0
02: while count < 3:
03:     count = count + 1
04:     if count == 2:
05:         continue
06:     print(count)
```

↳ **실행화면**

코드 6-7 count가 2일 때 멈추는 코드

```
01: count = 0
02: while count < 3:
03:     count = count + 1
04:     if count == 2:
05:         break
06:     print(count)
```

↳ **실행화면**

코드 6-8 count가 2일 때 멈추는 코드

```
01: for count in range(1, 3):
02:     if count == 2:
03:         break
04:     print(count)
```

↳ **실행화면**

💘 다음 중 출력 결과가 다른 코드를 골라보세요.

```
01: for i in range(3):
02:     print('안녕 거북이', i)
```

```
01: for j in range(0, 3):
02:     print('안녕 거북이', j)
```

```
01: k = 0
02: while k <= 3:
03:     print('안녕 거북이', k)
04:     k = k + 1
```

```
01: l = 0
02: while True:
03:     print('안녕 거북이', l)
04:     l = l + 1
05:     if l == 3:
06:         break
```

💓 다음 코드의 출력 결과를 적어보세요.

```
num = 1
while True:
    if num > 3:
        break
    print(num)
    num = num + 1
```

↳


무한 반복하기

```
while True:
    실행할_명령
```

코드 6-9 끝없이 반복해서 출력하는 코드

```
01: while True:
02:     print('Ctrl+C를 누르세요.')
```

↳ **실행화면**

6-3 수수께끼는 어려워

모두 끙끙대는 모습을 보니 앨리스는 왠지 자신이 똑똑해진 것 같아
어깨가 으쓱했어요.

“도저히 모르겠어. 어서 힌트를 줘!”

성질 급한 오리가 소리쳤어요. 그러자 다들 앨리스를
에워싸더니 또다시 정신없이 떠들어 대기 시작했어요.

“힌트 줘! 힌트 줘!”

모두의 아우성에 앨리스는 수수께끼의 힌트를 주기로 했어요.

- 보기와 함께 영국의 수도를 묻고 answer에 저장하세요.
- 틀린 답을 말하면 어느 나라의 수도인지 말해주세요.
- 보기에 없는 답을 말하면 보기에서 고르도록 안내하세요.
- 정답인 '**런던**'을 맞힐 때까지 반복해서 질문하세요.
- 정답을 맞히면 break를 사용해 반복을 멈추세요.

코드 6-10 틀린 답을 말하면 힌트를 주는 코드

```
01:
02:
03:
04:
05:
06:
07:
08:
09:
10:
11:
```

↳ 실행화면

```
런던, 파리, 서울 중 영국의 수도는 어디일까요? 런던
정답입니다. 런던은 영국의 수도입니다.
```

💘 다음은 소수를 판정하는 코드입니다. 아래 보기 중 소수를 골라보세요.

- 소수(prime number) : 1과 자기 자신 외에 나누어 떨어지지 않는 수입니다.

```
01: while True:
02:     number = int(input('2 이상의 정수를 입력하세요 (종료:-1): '))
03:     if number == -1:
04:         break
05:     count = 2
06:     is_prime = True
07:     while count < number:
08:         if number % count == 0:
09:             is_prime = False
10:             break
11:         count = count + 1
12:     if is_prime:
13:         print('소수입니다.')
14:     else:
15:         print('소수가 아닙니다.')
```

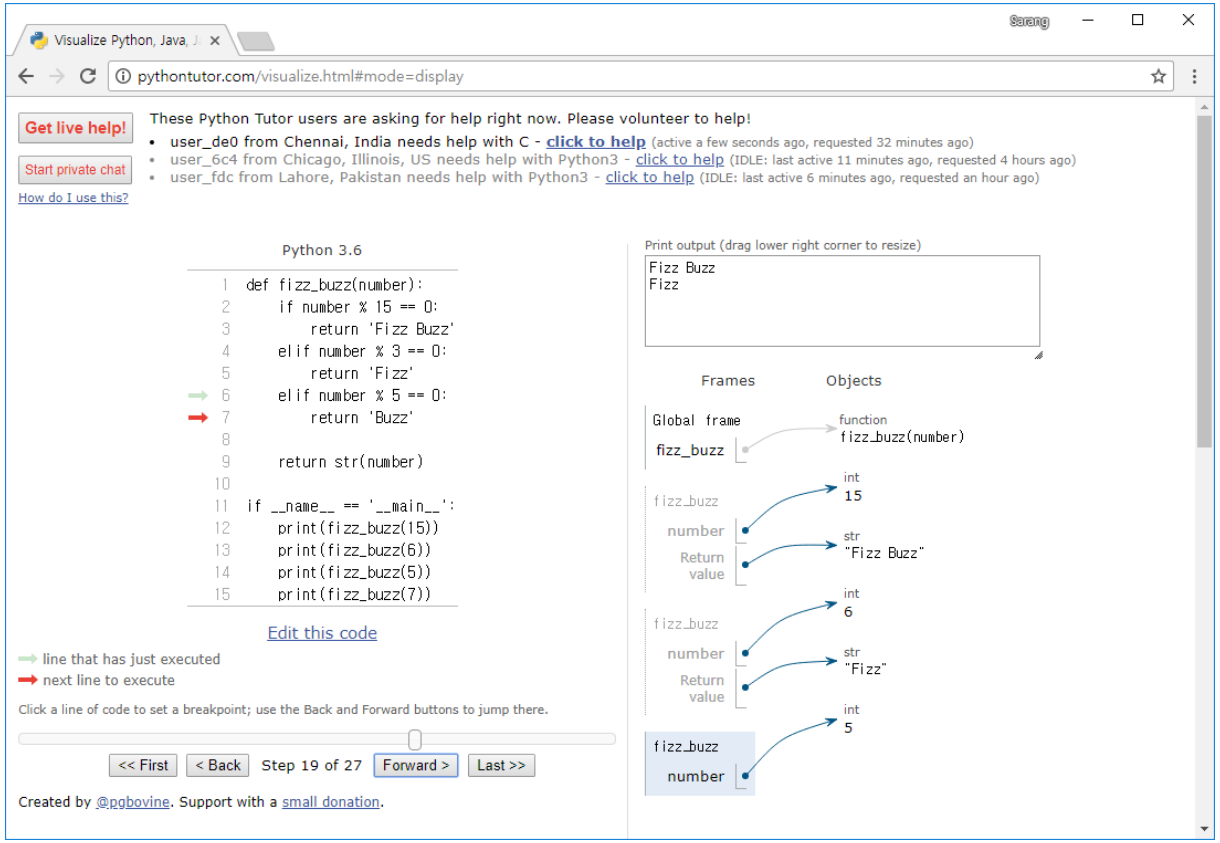
1) 2741

2) 2743

3) 2747

4) 2751

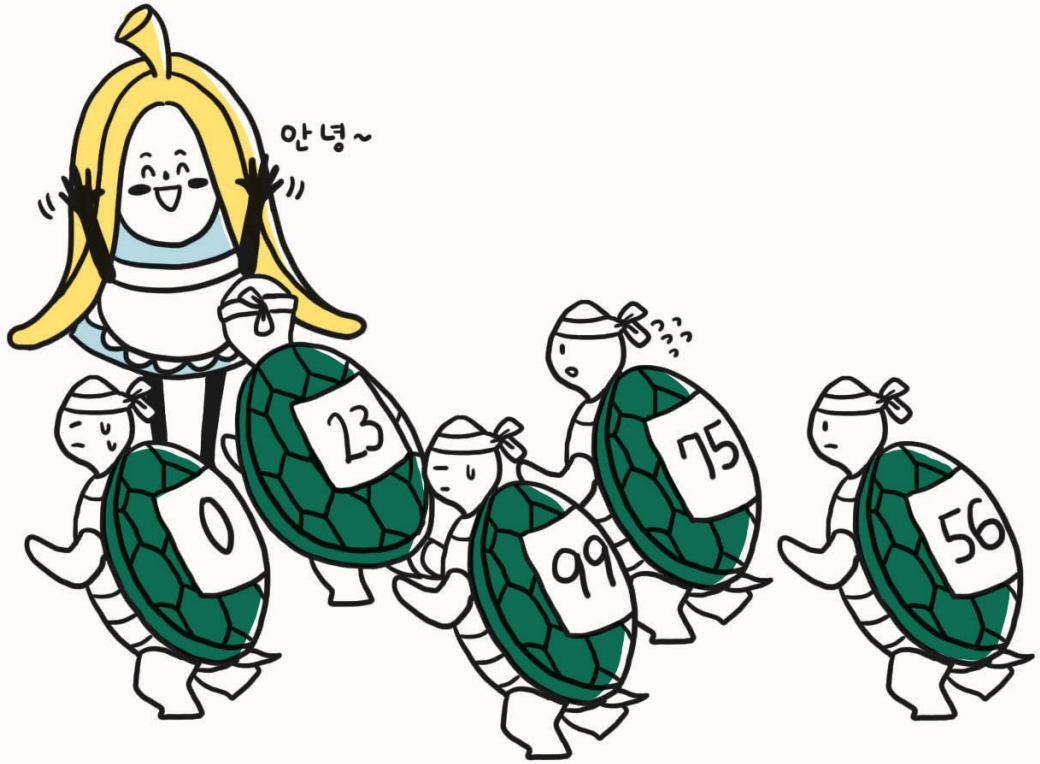
내 코드는 어떻게 돌고 있을까요?



프로젝트 : 가위바위보 2단계

5장의 가위바위보는 게임이라고 하기엔 어딘가 부족했습니다.
게임은 한 판으로 끝나버리고 사용자와 컴퓨터의 선택을
바꾸려면 직접 코드를 수정해야 했습니다. 게다가
대결 결과를 판단하는 부분도 너무 복잡했죠.

이번에는 5장의 가위바위보 게임을 개선해보겠습니다.



- 사용자에게 '가위', '바위', '보'를 입력받으세요.
- 컴퓨터는 무작위로 '가위', '바위', '보'를 선택합니다.
- '끝'을 입력할 때까지 게임을 반복하세요.
- and를 사용하지 않고 대결 결과를 판단하세요.

코드 project2 while로 반복하는 가위바위보 게임

```
01: import random
02:
03: rps = ['가위', '바위', '보']
04:
05: while True:
06:
07:     computer = random.choice(rps)
08:
09:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27:
28:
29:
30:
```

- 7_1_b.py
- 7_2_b.py
- 7_3_b.py
- 7_4_b.py
- 7_5_b.py
- 7_6_b.py
- 7_8_a.py
- 7_9_a.py

chapter 7

재료 모으기. 둘

튜플 기본 구조

(값1, 값2, 값3, …)

값1, 값2, 값3, …

코드 7-1 튜플을 만드는 코드

```
>>> my_tuple1 = ()
>>> print(my_tuple1)

>>> my_tuple2 = (1, -2, 3.14)
>>> print(my_tuple2)

>>> my_tuple3 = '앨리스', 10, 1.0, 1.2
>>> print(my_tuple3)
```

코드 7-2 값이 한 개인 튜플을 만드는 코드

```
>>> my_int = (1)
>>> print(type(my_int))

>>> my_tuple = (1,)
>>> print(type(my_tuple))
```

💘 아래와 같은 출력 결과가 나오도록 빈칸을 채워보세요.

```
>>>
>>> print(nums)
(1, 2, 3)
```

💘 다음 중 출력 결과가 다른 코드를 고르세요.

```
>>> my_var = 1,
>>> print(type(my_var))
```

```
>>> my_var = (1,)
>>> print(type(my_var))
```

```
>>> my_var = (1)
>>> print(type(my_var))
```

```
>>> my_var = 1, 2
>>> print(type(my_var))
```


값 가져오기

튜플[접근할_인덱스]

코드 7-3 튜플에서 값을 가져오는 코드

```
>>> clovers = (‘클로버1’, ‘하트2’, ‘클로버3’)
>>> print(clovers[1])
```

코드 7-4 튜플의 값을 변경하려는 코드

```
>>> clovers = (‘클로버1’, ‘하트2’, ‘클로버3’)
>>> clovers[1] = ‘클로버2’
```

💘 다음 코드의 출력 결과를 적어보세요.

```
01: my_tuple = (3.14, 2.71)
02: print(my_tuple)
03: print(my_tuple[1])
```

↳

패킹과 언패킹

코드 7-5 패킹과 언패킹을 하는 코드

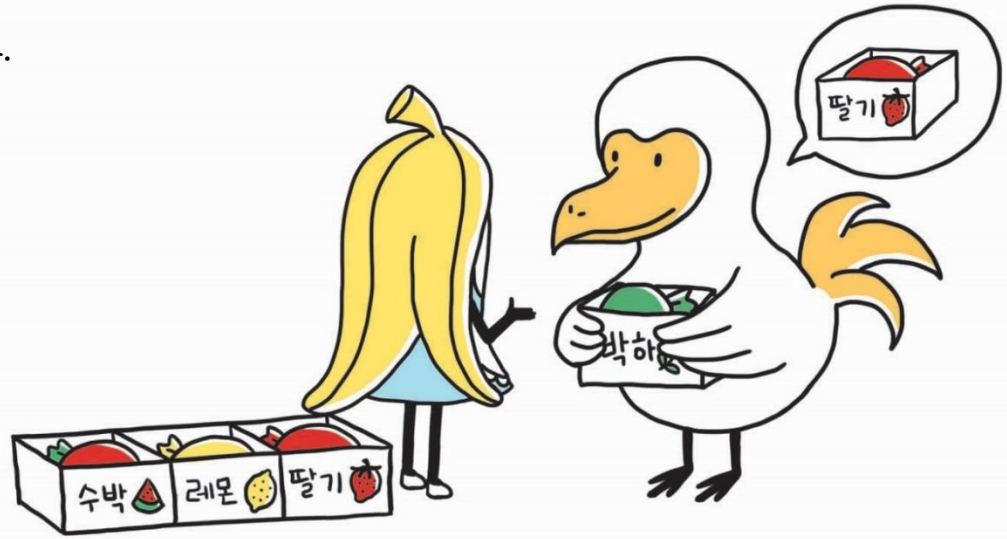
```
>>> clovers = '클로버1', '클로버2', '클로버3'
>>> print(clovers)

>>> alice_blue = (240, 248, 255)
>>> r, g, b = alice_blue
>>> print('R:', r, 'G:', g, 'B:', b)
```

7-1 박하맛 사탕 바꿔주기

도도새와 앨리스는 한동안 말없이 서로를 바라보았어요.

“이 박하맛 사탕은 뭐니?”
앨리스가 물었어요.

“난 박하맛 사탕이 싫어. 딸기맛 사탕으로 바꿔줘.”
도도새가 당연하다는 듯이 대답했어요.



앨리스도 박하맛 사탕을 싫어했지만, 친구와
사이좋게 지내라는 할머니의 말씀에 따르기로 했어
요. 도도새의 박하맛 사탕을 딸기맛 사탕으로 바꿔줍시다.

- dodo에는 '박하맛'이, alice에는 '딸기맛'이 저장되어 있습니다.
- 패킹과 언패킹을 사용해 둘의 사탕을 교환해 보세요.

코드 7-6 두 변수의 값을 서로 교환하는 코드

```
01:
02:
03:
04:
05:
```

↳ 실행화면

```
도도새: 박하맛 앨리스: 딸기맛
도도새: 딸기맛 앨리스: 박하맛
```

💘 다음 코드의 출력 결과를 적어보세요.

```
01: nums = 1, 2, 3
02: for idx in range(3):
03:     print(nums[idx])
```

↳

💘 다음 코드의 출력 결과를 적어보세요.

```
01: diamonds = 1, 5, 6, 7
02: ace, king, queen, jack = diamonds
03: print(queen)
```

↳

🔷 딕셔너리 기본 구조

{키1: 값1, 키2: 값2, …}

코드 7-7 딕셔너리를 만드는 코드

```
>>> my_dict1 = {}
>>> print(my_dict1)

>>> my_dict2 = {0: 1, 1: -2, 2: 3.14}
>>> print(my_dict2)

>>> my_dict3 = {'이름': '앨리스', '나이': 10, '시력': [1.0, 1.2]}
>>> print(my_dict3)
```


키-값 추가하기

```
딕셔너리[추가할_키] = 추가할_값
```

코드 7-8 딕셔너리에 키-값을 추가하는 코드

```
>>> clover = {'나이': 27, '직업': '병사'}
>>> print(clover)

>>> clover['번호'] = 9
>>> print(clover)
```

💘 다음 코드의 출력 결과를 적어보세요.

```
01: alice = {}
02: alice['성별'] = '여'
03: alice['나이'] = 13
04: alice['혈액형'] = 'AB'
05: print(alice)
```

↳

💘 아래와 같은 출력 결과가 나오도록 빈칸을 채워보세요.

```
01:  alice = {'성별': '여', '나이': 13, '혈액형': 'AB'}
02:
03:  print(alice['전화번호'])
```

```
↳    010-1234-5678
```

값에 접근하기

딕셔너리[접근할_키]

딕셔너리.get(접근할_키)

코드 7-9 딕셔너리의 값에 접근하는 코드

```
>>> clover = {'나이': 27, '직업': '병사', '번호': 9}
>>> print(clover['번호'])

>>> clover['번호'] = 6
>>> print(clover['번호'])

>>> print(clover.get('번호'))
```

앨리스의 혈액형을 출력하려고 합니다. 다음 중 올바른 코드를 고르세요.

```
01: alice = {'성별': '여', '나이': 13, '혈액형': 'AB'}
02: print(alice['혈액형'])
```

```
01: alice = {'성별': '여', '나이': 13, '혈액형': 'AB'}
02: print(alice[혈액형])
```

```
01: alice = {'성별': '여', '나이': 13, '혈액형': 'AB'}
02: print(alice['AB'])
```

```
01: alice = {'성별': '여', '나이': 13, '혈액형': 'AB'}
02: print(alice[AB])
```

💘 앨리스의 나이를 14로 바꾸는 코드를 작성하세요.

```
01: alice = {'성별': '여', '나이': 13, '혈액형': 'AB'}
02:
03: print(alice['나이'])
```

```
↳   14
```

키-값 제거하기

```
del 딕셔너리[제거할_키]
```

코드 7-10 딕셔너리에서 키-값을 제거하는 코드

```
>>> clover = {'나이': 27, '직업': '병사', '번호': 6}
>>> print(clover)

>>> del clover['나이']
>>> print(clover)
```

7-2 3월토끼의 라면 가게

3월토끼는 왕국에서 가장 유명한 라면 장인입니다.
3월토끼의 라면 가게는 언제나 손님으로
붐벼서 발 디딜 틈이 없었어요.

“매운라면 주세요!”
“저는 비빔라면요.”
“짜장라면으로 바꿔주세요!”

카드 병사들은 줄도 서지 않고 여기저기서 주문을 하기 시작했어요.
라면을 끓여야 하는 3월토끼를 도와 대신 주문을 받아볼까요?

- 주문1: '스페이드1'은 '비빔라면'을, '다이아2'는 '매운라면'을 주문했어요.
- 주문2: '클로버3'이 '카레라면'을 주문했어요.
- 주문3: '다이아2'가 '매운라면'을 '짜장라면'으로 바꿨어요.
- 주문4: 다이어트 중인 '스페이드1'이 주문을 취소했어요.

코드 7-11 라면 주문을 추가/수정/취소하는 코드

```
01:
02:
03:
04:
05:
06:
07:
08:
```

↳ 실행화면

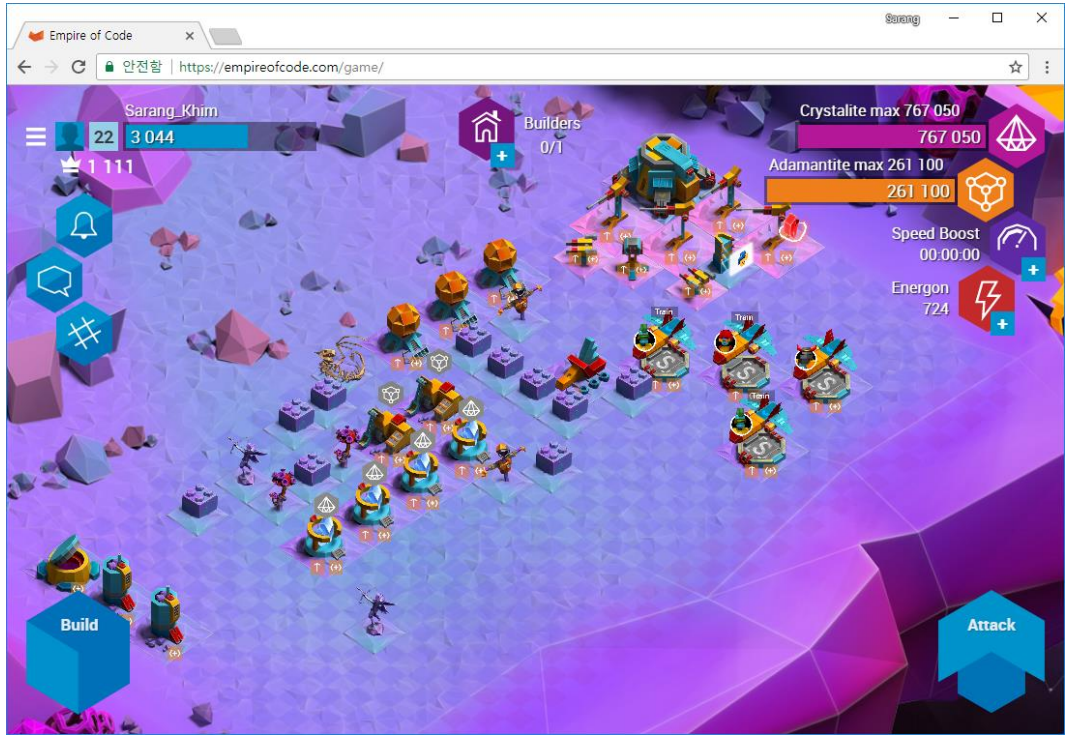
```
{'스페이드1': '비빔라면', '다이아2': '매운라면'}
{'스페이드1': '비빔라면', '다이아2': '매운라면', '클로버3': '카레라면'}
{'스페이드1': '비빔라면', '다이아2': '짜장라면', '클로버3': '카레라면'}
{'다이아2': '짜장라면', '클로버3': '카레라면'}
```

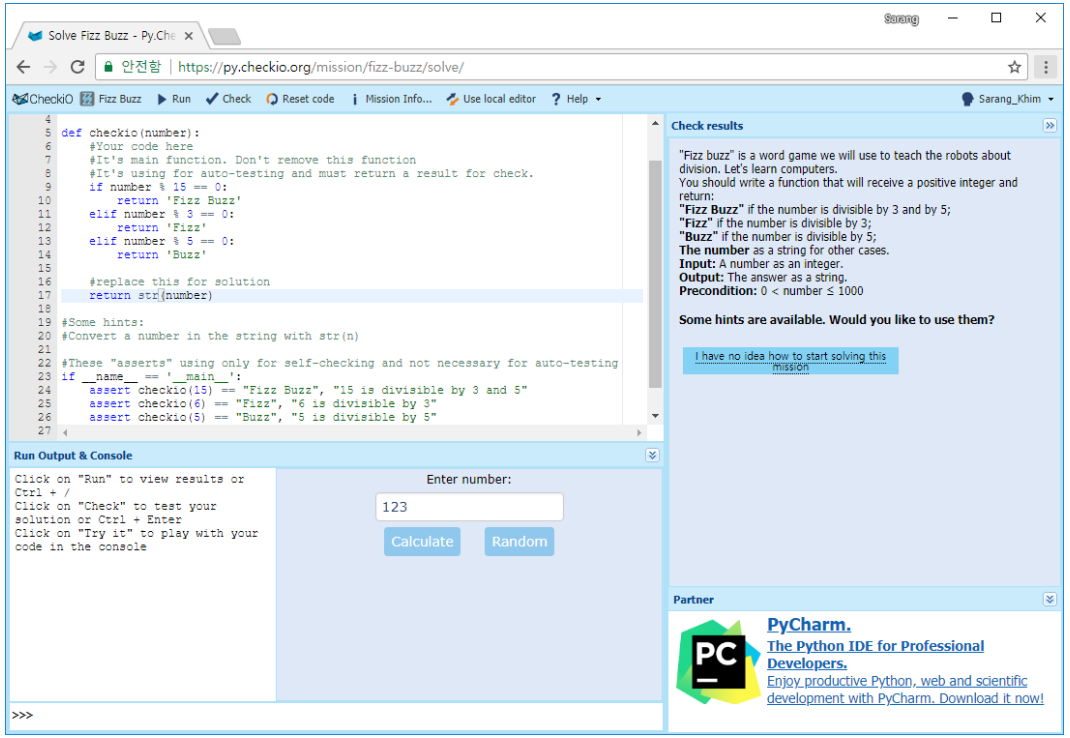
💘 아래와 같은 출력 결과가 나오도록 빈칸을 채워보세요.

```
01:  alice = {'성별': '여', '나이': 14, '혈액형': 'AB', '전화번호': '010-1234-5678'}
02:
03:  print(alice)
```

```
↳    {‘성별’: ‘여’, ‘나이’: 13, ‘혈액형’: ‘AB’}
```

신나는 게임으로 배우는 파이썬 #2

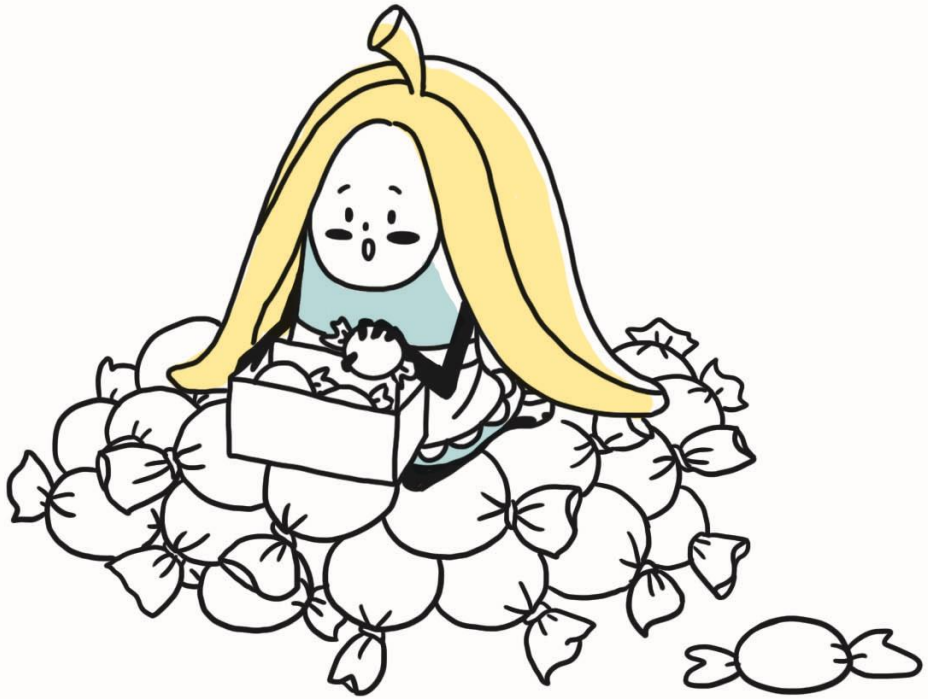




프로젝트 : 가위바위보 3단계

이번에는 7장에서 배운 튜플과 딕셔너리를 활용하겠습니다.
튜플과 딕셔너리로 대결 결과를 미리 저장해두면 이전
게임에서의 대결 결과를 판단하는 부분을 크게 줄일
수 있습니다.

어떻게 하면 튜플과 딕셔너리로 대결 결과를
저장할 수 있을지 한 번 생각해보세요.



- 대결 결과를 저장하는 딕셔너리를 만드세요.
- 이 딕셔너리를 사용해 대결 결과를 판단하세요.

코드 project3 딕셔너리를 사용한 가위바위보

```
01: import random
02:
03: rps = ['가위', '바위', '보']
04:
05:
06:
07:
08:
09:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27:
28:
29:
30:
```

- 4_13_b.py
- 6_1_b.py
- 6_2_b.py
- 6_3_b.py
- 6_4_a.py
- 6_5_b.py
- 6_6_a.py
- 6_7_b.py
- 6_8_a.py
- 6_9_a.py
- 6_10_a.py
- 6_11_a.py
- 6_12_b.py,

chapter 8

모아서 다시 쓰기

누가 파이를 훔쳤나

하트 여왕의 파이를 훔친 범인을 찾는 재판이 열렸어요. 흰 토끼가 나팔을 힘차게 세 번 불더니 양피지 두루마리를 펼치고 다음과 같이 읽었어요.

"하트 여왕님께서 어느 여름날 파이를 만드셨다.
누군가 그 파이를 훔쳐서 멀리 달아났도다."

재판정은 증언을 하려는 카드 병사들로 시끌시끌했어요.
다들 재판의 시작을 기다리고 있었는데 하트 여왕은
증언을 듣기도전에 판결을 내리기 시작했어요.

코드 8-1 모든 카드 병사에게 유죄 판결을 내리는 코드

```
01: print('하트 1 유죄!')
02: print('하트 2 유죄!')
03: print('하트 3 유죄!')
04: print('클로버 1 유죄!')
05: print('클로버 2 유죄!')
06: print('클로버 3 유죄!')
07: print('스페이드 1 유죄!')
08: print('스페이드 2 유죄!')
09: print('스페이드 3 유죄!')
```

↳ 실행화면

```
하트 1 유죄!
하트 2 유죄!
하트 3 유죄!
클로버 1 유죄!
클로버 2 유죄!
클로버 3 유죄!
스페이드 1 유죄!
스페이드 2 유죄!
스페이드 3 유죄!
```

함수의 종류

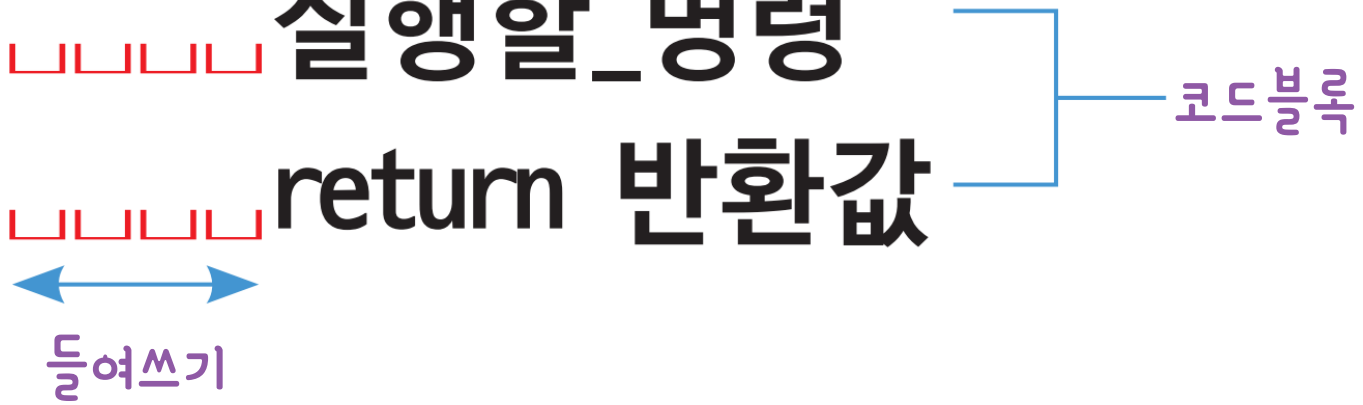
내장 함수

모듈의 함수

사용자 정의 함수

함수의 기본 구조

```
def 함수_이름(인수):
    실행할_명령
    return 반환값
```



코드 8-2 함수를 사용해 문자열을 출력하는 코드

```
01: def my_func():
02:     print('토끼야 안녕!')
03:
04: my_func()
```

↳ **실행화면**

코드 8-3 함수를 사용해 두 개의 숫자를 더하는 코드

```
01:
02:
03:
04:
```

```
def add(num1, num2):
    return num1 + num2

print(add(2, 3))
```

↳ **실행화면**

코드 8-4 함수를 사용해 두 개의 숫자를 더하고 곱하는 코드

```
01: def add_mul(num1, num2):
02:     return num1 + num2, num1 * num2
03:
04: print(add_mul(2, 3))
```

↳ **실행화면**

8-1 좀 더 효율적인 판결

하트 여왕이 증언은 듣지도 않고 끊임 없이 판결만 내리더니 결국 지쳐버리고 말았어요.

“꼬마야, 이제부터는 네가 판결을 내려라. 난 골치가 너무 아프구나!”

하트 여왕은 앨리스에게 이렇게 말하고는 그대로 바닥에 누워버렸어요. 하트 여왕을 대신해 좀 더 효율적인 방법으로 판결을 내려볼까요?

- 입력받은 카드 병사에게 유죄 판결을 내리는 judge_cards 함수를 만드세요.
- 이 함수를 사용해 하트 1~3, 클로버 1~3, 스페이드 1~3에게 판결을 내리세요.

코드 8-5 함수를 사용해 판결을 내리는 코드

```
01:
02:
03:
04:
05:
06: judge_cards('하트')
07: judge_cards('클로버')
08: judge_cards('스페이드')
```

↳ 실행화면

```
하트 1 유죄!
하트 2 유죄!
하트 3 유죄!
클로버 1 유죄!
클로버 2 유죄!
클로버 3 유죄!
스페이드 1 유죄!
스페이드 2 유죄!
스페이드 3 유죄!
```

💘 다음 코드의 출력 결과를 적어보세요.

```
01: def welcome():
02:     print('이상한 나라에 오신 것을 환영합니다.')
03:
04: welcome()
```

↳

💘 다음 코드의 출력 결과를 적어보세요.

```
01: def welcome(name):
02:     print(name, '님 이상한 나라에 오신 것을 환영합니다.')
03: 
04: welcome('앨리스')
05: welcome('도도새')
```

↳

💘 아래와 같은 출력 결과가 나오도록 빈칸을 채워보세요.

```
01:  def draw_stars(num):
02:
03:
04:  draw_stars(3)
05:  draw_stars(2)
06:  draw_stars(1)
```

```
↳    ***
     **
     *
```

💘 아래와 같은 출력 결과가 나오도록 빈칸을 채워보세요.

```
01:
02:
03:
04: print(concat('빨주노초', '파남보'))
```

```
↳    빨주노초파남보
```

💘 정수 두 개를 입력 받아 두 정수의 뺀 값과 나눈 값을 돌려주는 함수 sub_div()를 만들어 보세요.

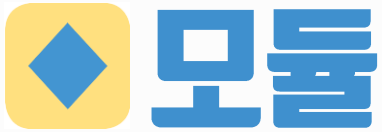
```
01:
02:
03:
04: print(sub_div(6, 3))
```

```
↳   (3, 2.0)
```

💘 카드의 이름과 수를 입력으로 받아 판결을 내리는 함수를 만들어 보세요.

```
01:
02:
03:
04:
05: judge_cards('하트', 2)
06: judge_cards('클로버', 3)
```

```
↳   하트 1 유죄!
    하트 2 유죄!
    클로버 1 유죄!
    클로버 2 유죄!
    클로버 3 유죄!
```


모듈

import 모듈_이름

랜덤하게 뽑기

random

random.choice(리스트)

코드 8-6 하나의 값을 임의로 선택하는 코드

```
01: import random
02: animals = [‘체셔고양이’, ‘오리’, ‘도도새’]
03: print(random.choice(animals))
```

↳ **실행화면**

random.sample(리스트, 뽑을_개수)

코드 8-7 여러 개의 값을 임의로 선택하는 코드

```
01: import random
02: animals = [‘체셔고양이’, ‘오리’, ‘도도새’]
03: print(random.sample(animals, 2))
```

↳ **실행화면**

```
random.randint(시작_값, 끝_값)
```

코드 8-8 지정한 범위에서 임의의 정수를 선택하는 코드

```
01: import random
02: print(random.randint(5, 10))
```

↳ **실행화면**

8-2 하트 여왕의 판결

“지금부터는 제비뽑기로 판결을 내리겠다!”
다시 기운을 차린 하트 여왕이 말했어요.

“하지만 그건 불공평해요!”
앨리스가 반박했어요.

“이건 가장 오래된 규칙이야. 어서 판결을 내려라!”
하트 여왕이 근엄한 목소리로 말했어요.

random 모듈을 사용해서 제비뽑기로 판결을 내려봅시다.

- `random.choice()`를 사용해 `cards`에서 임의로 카드 병사 한 명을 뽑으세요.
- 뽑힌 카드 병사에게 유죄 판결을 내리세요.

코드 8-9 임의로 카드 하나를 뽑는 코드

```
01:
02:
03:
04:
```

↳ 실행화면

```
클로버  유죄!
```

💘 다음은 무작위로 한 장의 카드를 뽑는 코드입니다. 빈칸을 채워보세요.

```
01:
02: clovers = ['클로버1', '클로버2', '클로버3']
03:
```

```
↳    클로버2
```

💘 중복되지 않는 카드 두 장을 뽑도록 빈칸을 채워보세요.

```
01:
02: clovers = ['클로버1', '클로버2', '클로버3']
03:
```

```
↳    클로버2, 클로버3
```

💘 다음 중 출력 결과가 다른 하나를 고르세요.

```
01: import random
02: print(random.choice([1, 2, 3]))
```

```
01: import random
02: print(random.choice(range(1, 4))
```

```
01: import random
02: print(random.randint(1, 3))
```

```
01: import random
02: print(random.randint(1, 4))
```

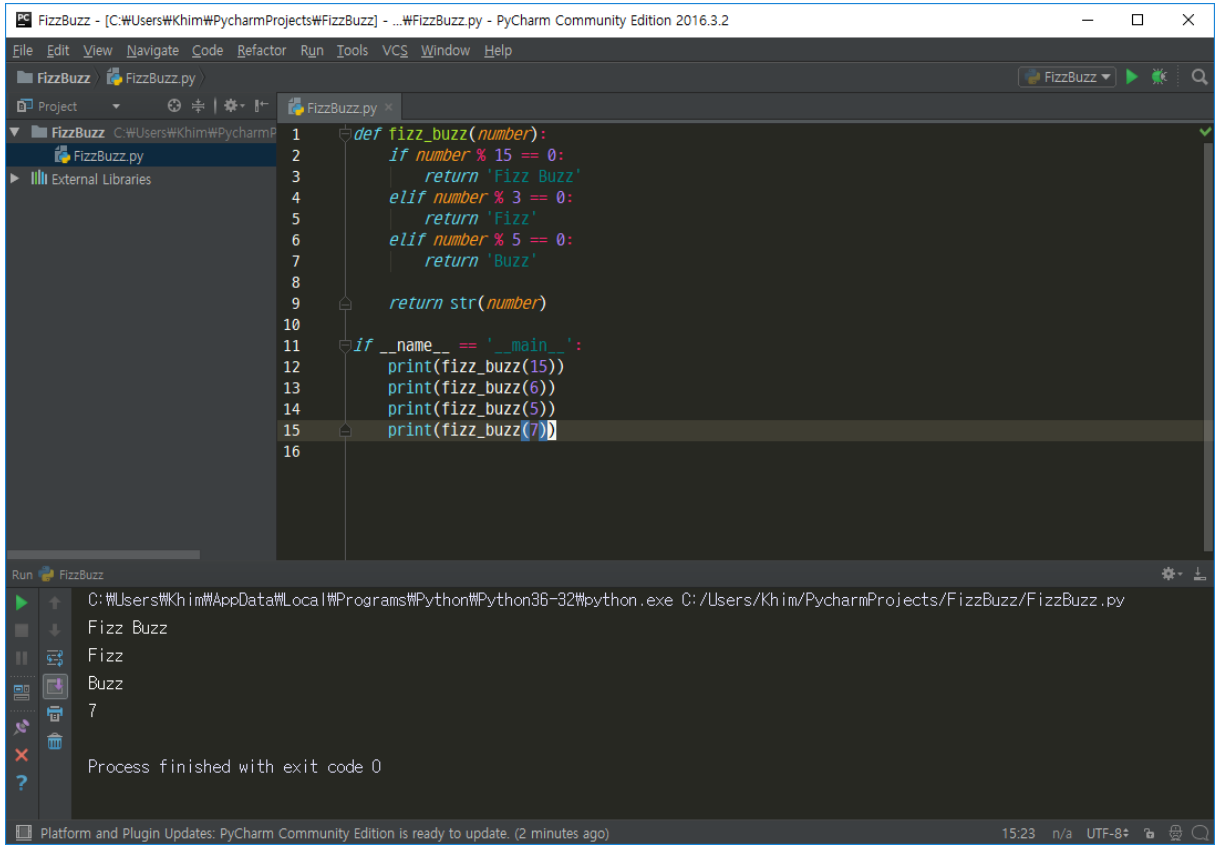
💘 인디언들은 태어난 날에 따라 이름을 짓는다고 합니다. birth_year는 태어난 연도의 1의 자리, birth_month는 태어난 월, birth_date는 태어난 일을 의미합니다. random 모듈을 사용해 랜덤으로 인디언식 이름을 한번 지어 볼까요?

```
01: import random
02: birth_year = ['시끄러운 ', '푸른 ', '적색 ', '조용한 ', '웅크린 ', '백색 ', '지혜로운 ', '용감한 ', '날카로운 ', '욕심많은 ']
03: birth_month = ['늑대', '태양', '양', '매', '황소', '불꽃', '나무', '달빛', '말', '돼지', '하늘', '바람']
04: birth_date = ['와(과) 함께 춤을', '의 기상', '은(는) 그림자 속에', '', '', '', '의 환생', '의 죽음', ' 아래에서', '을(를) 보라', '이(가) 노래하다', ' 그림자', '의 일격', '에게 쫓기는 남자', '의 행진', '의 왕', '의 유령', '을(를) 죽인자', '은(는) 맨날 잠잔다', '처럼', '의 고향', '의 전사', '은(는) 나의 친구', '의 노래', '의 정령', '의 파수꾼', '의 악마', '와(과) 같은 사나이', '을(를) 쓰러트린자', '의 혼', '은(는) 말이 없다']
05: random_name = random.choice(birth_year) + random.choice(birth_month) + random.choice(birth_date)
06: print('당신의 인디언식 이름은', random_name, '입니다.')
```

↳

♣ 모듈을 사용하는 이유

본격! 파이썬 개발 준비


FizzBuzz - [C:₩Users₩Khim₩PycharmProjects₩FizzBuzz] - ...₩FizzBuzz.py - PyCharm Community Edition 2016.3.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
FizzBuzz FizzBuzz.py
Project
FizzBuzz C:₩Users₩Khim₩PycharmP
FizzBuzz.py
External Libraries
FizzBuzz.py
def fizz_buzz(number):
if number % 15 == 0:
return 'Fizz Buzz'
elif number % 3 == 0:
return 'Fizz'
elif number % 5 == 0:
return 'Buzz'
return str(number)
if __name__ == '__main__':
print(fizz_buzz(15))
print(fizz_buzz(6))
print(fizz_buzz(5))
print(fizz_buzz(7))
Run FizzBuzz
C:₩Users₩Khim₩AppData₩Local₩Programs₩Python₩Python36-32₩python.exe C:/Users/Khim/PycharmProjects/FizzBuzz/FizzBuzz.py
Fizz Buzz
Fizz
Buzz
7
Process finished with exit code 0
Platform and Plugin Updates: PyCharm Community Edition is ready to update. (2 minutes ago)
15:23 n/a UTF-8

프로젝트 : 가위바위보 4단계

마지막 가위바위보 게임입니다. 이번에는 완전히 다른 방식의 가위바위보 게임을 만들어 보겠습니다. '가위', '바위', '보' 문자열이 아닌 0, 1, 2 정수를 입력받는 방법입니다. 숫자 사이의 관계에 따라 대결 결과를 판단할 수 있습니다.

8장에서 배운 함수도 적용해보면 더욱 좋습니다.

- 가위, 바위, 보를 정수 `0`, `1`, `2`로 입력받으세요.
- 컴퓨터도 정수 `0`, `1`, `2` 중 랜덤으로 선택하게 하세요.
- 두 값의 차에 따라 결과를 판단하는 함수를 만드세요.

코드 project4 함수를 넣은 가위바위보

```
01:
02:
03:
04:
05:
06:
07:
08:
09:
10:
11:
12:
```

```
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
```

- 8_1_b.py
- 8_2_b.py
- 8_3_a.py
- 8_6_b.py
- 8_9_b.py
- 8_10_b.py
- 8_11_a.py
- 8_4_a.py
- 8_5_a.py
- 8_7_a.py
- 8_8_a.py
- 8_12_a.py