18장

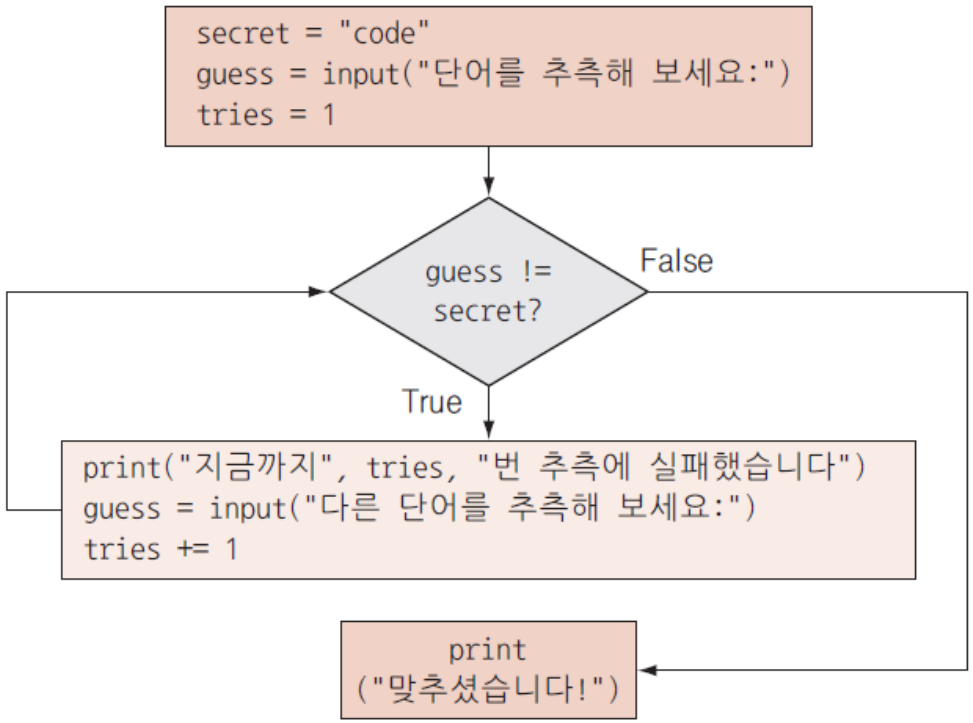
조건이 성립하는 동안 반복하기

18장 조건이 성립하는 동안 반복하기

18.1 조건이 참인 동안에만 루프 반복하기

18.1.1 추측을 위해 루프 돌기

- 단어 맞히기 게임을 해 보자. 여러분이 단어를 하나 정하고 상대방에게 단어를 맞춰보라고 하는 것이다.
- 상대방이 단어를 추측해 말할 때마다 그 단어가 여러분이 생각한 단어인지 말해준다.
- 틀린 경우 상대방은 다시 다른 단어를 물어본다. 상대방이 단어를 맞출 때까지 이 과정을 반복하면서 단어를 몇 번이나 틀렸는지 기록해 둔다.



18.1.1 추측을 위해 루프 돌기


```
secret = "code"
guess = input("단어를 추측해 보세요: ")
tries = 1
while guess != secret: ---- 추측한 단어가 비밀 단어와 다른지 검사한다
    print("지금까지 ", tries, "번 추측에 실패했습니다")
    guess = input("다른 단어를 추측해 보세요: ") ---- 사용자에게 다시 물어본다
    tries += 1
print("맞추셨습니다!") ---- 추측이 올바른 경우 여기 도달한다
```

코드 18-1
while을 사용하는 단어 추측 게임 예제

- 여기서 여러분은 코드 블록 안에 while 루프 조건식의 결과를 바꿀 수 있는 어떤 동작이 들어 있어야 한다는 사실을 눈치챘어야 한다.
- 코드 블록이 while의 조건과 아무 관계가 없다면 루프를 아무리 많이 돌아도 조건이 참인 상황이 변하지 않기 때문에 무한 루프에 빠진다.
- 코드 18-1은 while문의 코드 블록 안에서 사용자에게 새로운 단어를 물어봐 guess를 변경하기 때문에 (사용자가 계속 추측에 실패하지 않는 한) 언젠가는 조건이 거짓이 되고 루프가 끝난다

18.1.2 while 루프

```
while <조건>: ---- 루프 시작을 나타낸다
    <어떤 일을 한다>
```

코드 18-2
while 루프를 작성하는 일반적인 방법

- 파이썬 인터프리터가 처음 while을 만나면 조건이 참인지 검사한다.
- 조건이 참이면 while 루프 코드 블록에 들어가서 그 블록에 들어 있는 문장을 실행한다.
- 코드 블록 실행을 마치면 다시 조건을 검사한다.
- 파이썬 인터프리터는 조건이 참인 동안 코드 블록을 계속 반복 실행한다.

» 셀프 체크 18.1

18.1.3 무한 루프

while 루프를 사용하면 결코 끝나지 않는 코드를 작성할 수 있다.

```
while True:
    print("대체 언제 끝나냐고요!")
```

무한 루프에 들어간 프로그램을 직접 중단하는 방법

- 콘솔 맨 위에 있는 빨간 정사각형을 클릭하라.
- 콘솔 창을 클릭해 입력 포커스가 향하게 한 다음, Ctrl + C 를 눌러라(Ctrl 또는 Control 이라고 쓰인 키를 누른 상태에서 C 키를 누르면 된다).
- 콘솔에서 메뉴를 클릭하고(메뉴는 콘솔창 위 빨간 정사각형 옆에 있는 아이콘 중 오른쪽 끝에 있는 톱니바퀴 모양의 아이콘을 클릭하면 나온다), 커널 재시작(Restart Kernel)을 선택하라.

18.1.3 무한 루프

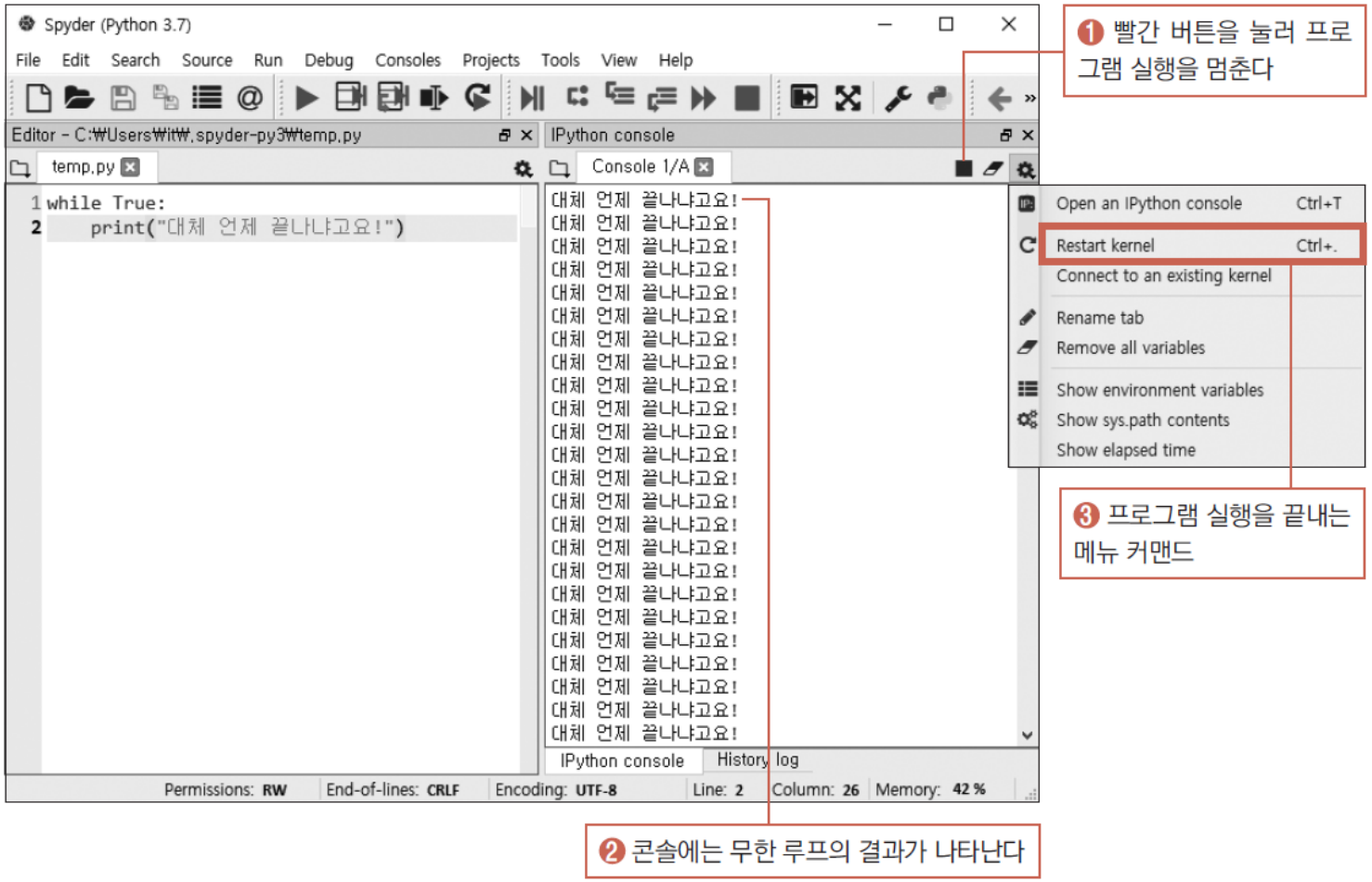


그림 18-2
무한 루프에서 빠져 나오는 방법. 빨간 정사각형 버튼을 클릭하거나, Ctrl + C 를 누르거나, 빨간 정사각형 버튼 옆 오른쪽 끝에 있는 콘솔 메뉴에서 커널 재시작을 선택한다

18.2 for 루프와 while 루프 비교

18.2 for 루프와 while 루프 비교

» for 루프는 while 루프로 바꿀 수 있다. for 루프는 미리 정해진 횟수를 반복한다.

» 이를 while 루프로 바꾸려면 while 조건식에서 검사할 변수를 하나 추가하면 된다.

» while 루프의 코드 블록은 그 변수의 내용을 바꾼다.

▼ for 루프를 사용하는 경우

```
for x in range(3): ---- 루프 시작 표시
    print("var x is", x)
```

▼ while 루프를 사용하는 경우

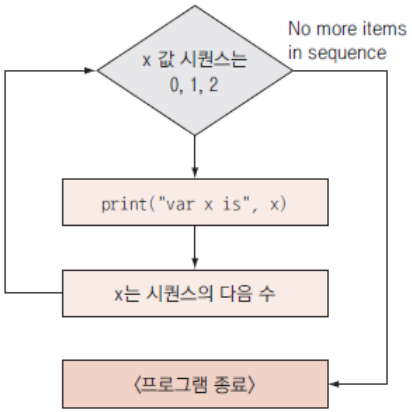
```
x = 0 ---- 루프 변수 초기화
while x < 3:
    print("var x is", x)
    x += 1 ---- 루프 변수 증가
```

코드 18-3
for 루프를 while 루프로 바꿔 쓰기

- while 루프에서는 변수 x를 직접 초기화해야 함에 유의하자.
- 초기화 부분이 없으면 파이썬이 루프 안의 x가 어떤 값을 가리키는지 알 수 없다. 또한, 루프 본문에서 루프 변수 x의 값을 직접 증가시켜줘야 한다.
- for 루프의 경우 파이썬이 알아서 변수를 초기화하고 변수가 가리키는 값을 증가시킨다.
- 코드 18-3에서 while을 사용하는 경우에는 변수를 따로 직접 초기화해야 한다. 그리고 while 루프 안에서 그 변수의 값을 직접 증가시켜야 한다.

18.2 for 루프와 while 루프 비교

A. for 루프를 사용하는 코드의 흐름도



B. while 루프를 사용하는 코드의 흐름도

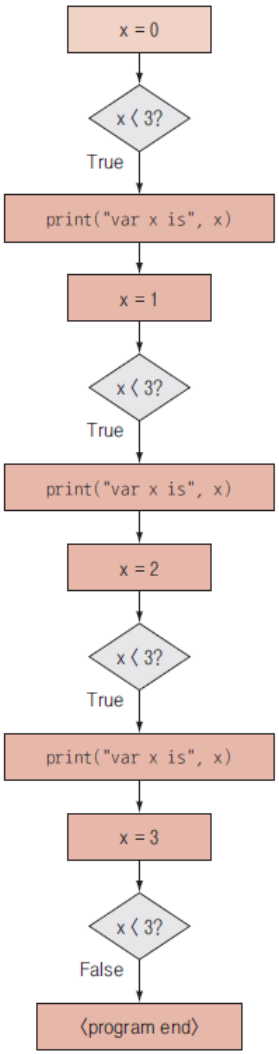


그림 18-3
(A)는 반복할 때마다 루프 변수의 값을 매번 출력하는 for 루프 코드를 흐름도로 시각화한 것이다.

(B)는 (A)에 흐름도로 그려둔 for 루프를 while 루프로 변환한 코드에서 변수의 값이 어떻게 변하는지 보여 주기 위해 시각화한 것이다.

while을 사용하는경우에는 루프를 시작하기 전에 변수를 직접 초기화하고 while 루프 본문 안에서 직접 증가시켜야 한다.

또한, 루프 변수 x의 값에 따라 결과가 달라지게 while의 조건문을 만들어야 while 루프가 3번만 실행될 수 있다

18.2 for 루프와 while 루프 비교

- for 루프는 모두 while 루프로 바꿔 쓸 수 있다. 하지만 while 루프를 모두 for 루프로 바꿔 쓸 수있는 건 아니다.
- while은 반복 횟수를 알든 모르든 사용할 수 있으므로 반복 횟수를 모를 때는 for 루프로 바꿀 수 없다.
- 셀프 체크 18.2

18.3.1 루프를 빨리 끝내기

» 키워드 break를 사용하면 루프를 빨리 끝낼 수 있다. break를 실행하면 파이썬은 현재 실행 중인 while 루프의 조건이 설령 참이라 할지라도 루프를 중단하고 루프 다음에 있는 문장을 실행한다.

```
secret = "code"
max_tries = 100
guess = input("단어를 추측해 보세요: ")
tries = 1
while guess != secret:
    print("지금까지 ", tries, "번 추측에 실패했습니다")
    if tries == max_tries:
        print("추측할 수 있는 기회를 다 썼습니다")
        break ---- max_tries 이상 추측을 시도한 경우 루프를 깨고 루프 밖으로 나온다
    guess = input("다른 단어를 추측해 보세요: ")
    tries += 1
if tries <= max_tries and guess == secret: ---- 루프에서 나온 이유를 알아낸다
    print("맞추셨습니다!")
```

코드 18-4
break 키워드
사용하기

18.3.1 루프를 빨리 끝내기

코드 18-4는 사용자가 단어를 100번 추측했는지 검사하는 부분이 더 들어갔다.

- 이 조건이 참이면 메시지를 출력하고 루프를 중단하고 나온다. break 문을 실행한 즉시 루프가 끝난다.
- 따라서 break로 루프를 중단하는 경우, 루프 본문 안 break 뒤에 있는 문장들은 하나도 실행되지 않는다.
- 이제 사용자가 단어를 제대로 맞히는 경우 외에 루프에서 나갈 수 있는 경우가 하나 더 늘어난 것이다.
- 따라서 루프가 끝난 다음, 루프에서 빠져나온 이유가 무엇인지 검사하는 조건문이 더 필요하다

사용자가 비밀 단어를 맞힌 경우와 사용자가 max_tries보다 더 많은 시도를 한 경우 어떤 일이 벌어지는지 생각해 보자.

- 두 경우 모두 while 루프를 빠져나와서 루프 바로 뒤에 있는 문장을 실행한다.
- 하지만 축하 메시지를 표시하고 싶은 경우는 사용자가 비밀 단어를 맞춰서 루프를 빠져나온 경우뿐이다.
- 축하 메시지를 표시하는 코드는 루프가 끝난 뒤에 올 수 밖에 없다. 하지만 루프를 빠져나왔다고 메시지를 그냥 찍어버릴 수는 없다.
- while 루프를 빠져나온 정확한 이유를 모르는 상황이기 때문이다. 사용자가 너무 많이 틀려서 루프에서 빠져 나왔을 가능성도 있다.

따라서 사용자의 재시도 횟수가 아직 남아있는지 검사하고 사용자가 비밀 단어를 맞췄는지 검사 하는 조건문을 추가해야 한다

18.3.2 루프를 시작하는 부분으로 돌아가기

- 루프 본문에 남아있는 나머지 문장을 생략하고 루프의 시작 부분으로 돌아가 조건을 다시 검사해서 루프를 더 진행하거나 종료하고 싶을 때는 continue 키워드를 사용한다.

- 코드를 깔끔하게 만들려고 continue를 쓰는 경우도 자주 있다.

18.3.2 루프를 시작하는 부분으로 돌아가기


```
### 코드 버전 1 ###
x= 0
for x in range(100):
    print("x는", x)
    if x > 5:
        print("x가 5보다 큽니다")
        if x%10 != 0:
            print("x를 10으로 나눌 수 없습니다")
            if x==2 or x==4 or x==16 or x==32 or x==64:
                print("x는 2의 거듭제곱 수입니다")
                # 코드가 더 올 수 있음

### 코드 버전 2 ###
x = 0
for x in range(100):
    print("x는", x)
    if x <= 5:
        continue
    print("x가 5보다 큽니다") ---- x > 5인 경우 여기로 진행함
    if x%10 == 0:
        continue
    print("x를 10으로 나눌 수 없습니다") ---- x%10 != 0인 경우 여기로 진행함
    if x!=2 and x!=4 and x!=16 and x!=32 and x!=64:
        continue
    print("x는 2의 거듭제곱 수입니다") ---- x가 2, 4, 16, 32, 64 중 하나인 경우 여기로 진행함
    # 코드가 더 올 수 있음
```

루프 본문의 나머지 문장 실행 생략

코드 18-5
continue를 사용하지 않는 코드와 사용하는 코드 비교

18.3.2 루프를 시작하는 부분으로 돌아가기

- continue를 사용하는 코드에서 조건식이 참일 때 루프 본문의 나머지 문장을 건너뛴다. 그리고 루프의 시작으로 가서 루프문이 실행될 때처럼 다음 값을 x에 가져온다.

- continue를 사용하지 않는 코드에서는 continue를 사용하는 코드보다 조금 더 복잡하게 일을 처리해야 한다.

- 이 예제처럼 여러 가지로 내포된 조건을 검사해야 하는 경우에는 continue 키워드를 사용하는 쪽이 더 편리하다.

18.4 요약

18.4 요약

- while 루프는 조건이 성립하는 동안 문장을 반복 수행한다.
- for 루프는 언제나 while 루프로 바꿔 쓸 수 있다. 하지만 while 루프를 언제나 for로 바꿀 수는 없다.
- 루프를 일찍 끝내고 싶을 때 break 문을 사용한다.
- 루프 본문의 나머지 문장을 건너 뛰고, while 루프의 조건을 다시 검사하거나 for 루프 시퀀스의 다음 원소를 가져와 루프를 다시 반복하고 싶을 때 continue 문을 사용한다.
- 프로그램에서 for나 while 루프 중 하나를 시도해 보고, 그 선택이 여러분의 프로그램에 적합한지 알아내도 아무 문제가 없다. 여러 가지를 시도해 보라. 그런 시도가 코딩이라는 퍼즐을 맞추기 위해 이런저런 요소를 끼어 맞추는 것과 같다고 생각하라.

18.4 요약

» (Q18.1) 다음 프로그램에는 버그가 있다. 한 줄만 변경해서 무한 루프를 없애라. 몇 가지 입력을 시험해 보면서 프로그램이 어떻게 작동해야 할지 생각해보고, 실제 프로그램이 생각대로 작동하는지 확인해 보라.

```
num = 8
guess = int(input("내가 생각한 수를 맞춰 보세요: "))
while guess != num:
    guess = input("다시 맞춰 보세요: ")
print("Right!")
```

18.4 요약

» (Q18.2) 사용자에게 게임을 하고 싶은지 물어보는 프로그램을 작성하라. 사용자가 y나 yes를입력하면 1 이상 10 이하인 수를 생각하고 있다고 출력한 후, 사용자에게 그 수를 맞추게 하라. 프로그램은 사용자가 수를 맞출 때까지 계속 새 값을 입력하라고 사용자에게 요청해야 한다. 사용자가 값을 맞추면 축하 메시지를 표시하고 게임을 다시 하고 싶은지 물어보라. 사용자가 y나 yes를 입력하면 전체 과정을 반복하라.