UNIT 14

else를 사용하여 두 방향으로 분기하기

14 else를 사용하여 두 방향으로 분기하기

else를 사용하여 두 방향으로 분기하기

- if 조건문은 분기(branch)를 위한 문법
- 분기는 "둘 이상으로 갈라지다"라는 뜻으로 프로그램의 흐름을 둘 이상으로 나누는 것을 말함

▼ 그림 도로 이정표

14 else를 사용하여 두 방향으로 분기하기

else를 사용하여 두 방향으로 분기하기

- if에 else를 사용하면 조건식이 만족할 때와 만족하지 않을 때 각각 다른 코드를 실행할 수 있음
- 프로그램이 두 방향으로 분기하는 것임
- 실생활에서 전화가 왔을 때의 예를 들면 다음과 같은 모양이 됨

```
if 광고 전화인가?:
    전화를 끊고, 차단 목록에 등록한다.
else:
    계속 통화한다.
```

14.1 else 사용하기

else 사용하기

- else는 if 조건문 뒤에 오며 단독으로 사용할 수 없음
- if와 마찬가지로 else도 :(콜론)을 붙이며 다음 줄에 실행할 코드가 옴

```
if 조건식:
    코드1
else:
    코드2
```

- 들여쓰기를 맞춰서 파이썬 셸에 코드를 입력해보자

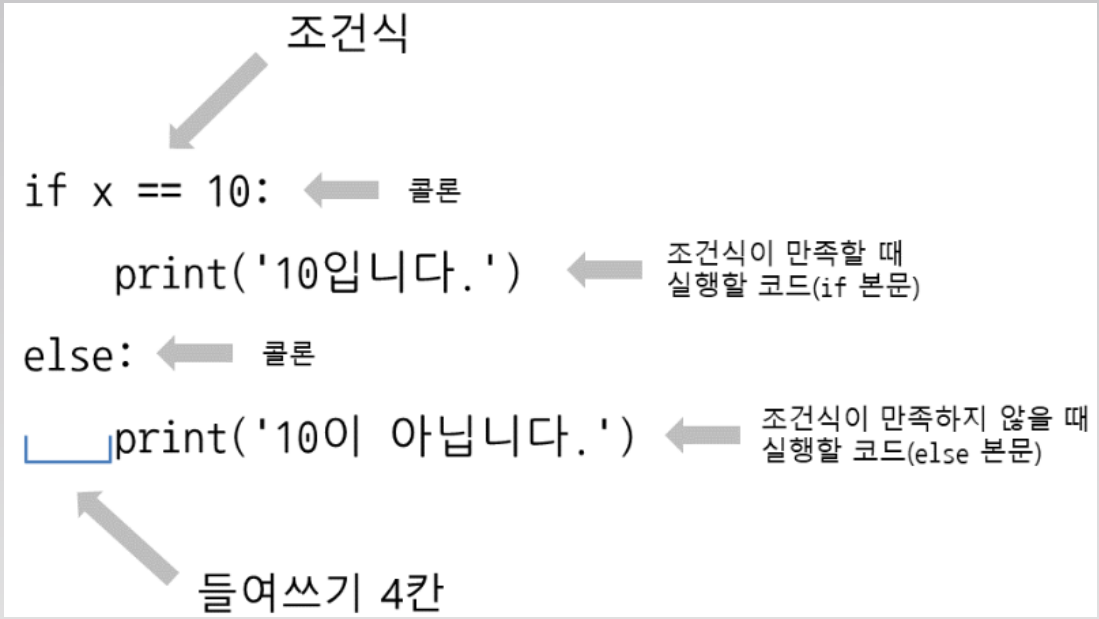
```
>>> x = 5
>>> if x == 10:
...     print('10입니다.')
... else:
...     print('10이 아닙니다.')
...
10이 아닙니다.
```

14.1 else 사용하기

if와 else의 기본 형태와 실행 흐름 알아보기

- else는 if의 조건식이 만족하지 않을 때 코드를 실행함
- x에 5가 들어있어서 x == 10을 만족하지 않으므로 else의 print가 실행되어 '10이 아닙니다.'가 출력됨

▼ 그림 if와 else

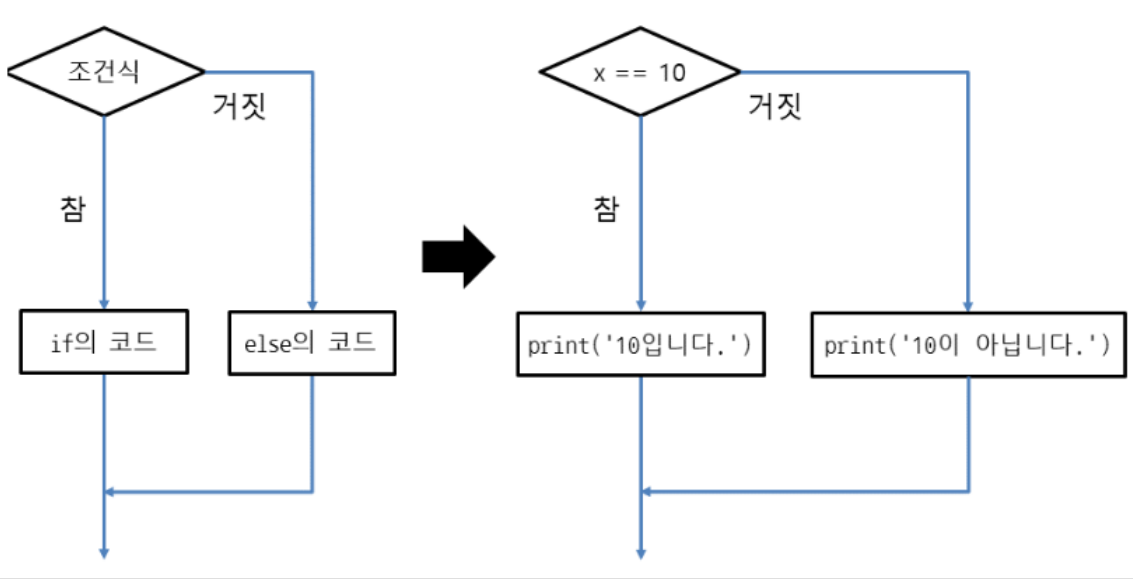


14.1 else 사용하기

if와 else의 기본 형태와 실행 흐름 알아보기

- 조건식이 참(True)이면 if의 코드(if 본문)가 실행되고, 거짓(False)이면 else의 코드(else 본문)가 실행됨
- 코드와 실행 흐름을 비교해보자

▼ 그림 if와 else의 실행 흐름



14.2 else와 들여쓰기

else와 들여쓰기

- else는 if와 들여쓰기 규칙이 같음
- 들여쓰기가 잘못된 코드임

else_indent_error.py

```
x = 5

if x == 10:
    print('10입니다.')
else:
print('x에 들어있는 숫자는')    # unexpected indent 에러 발생
    print('10이 아닙니다.')
```

- else도 코드가 여러 줄일 때는 들여쓰기 깊이가 같게 만들어주어야 함

```
if x == 10:
    print('10입니다.')
else:
    print('x에 들어있는 숫자는')
    print('10이 아닙니다.')
```

14.2 else와 들여쓰기

else와 들여쓰기

- else가 여러 줄일 때는 마지막 줄의 들여쓰기를 하지 않으면 의도치 않은 동작이 됨

else_wrong_indent.py

```
x = 10

if x == 10:    # x가 10이라 조건식이 참
    print('10입니다.')    # 출력
else:
     print('x에 들어있는 숫자는')
print('10이 아닙니다.')    # 출력되지 않아야 하는데 출력됨
```

실행 결과

```
10입니다.
10이 아닙니다.
```

14.2 else와 들여쓰기

else와 들여쓰기

- x가 10이라 조건식이 참이므로 '10입니다.'가 출력됨
- else의 '10이 아닙니다.'도 함께 출력되어버렸죠? 왜냐하면 print('10이 아닙니다.')는 들여쓰기가 없어서 else와는 상관없는 코드가 되었기 때문임
- print를 한 줄 띄워보면 왜 잘못되었는지 알 수 있음

```
x = 10

if x == 10:    # x가 10이라 조건식이 참
    print('10입니다.')
else:
    print('x에 들어있는 숫자는')

print('10이 아닙니다.')
```

14.3 if 조건문의 동작 방식 알아보기

if 조건문의 동작 방식 알아보기

- 조건식이 아닌 값으로 if와 else의 코드를 동작시켜 보자
- IDLE의 소스 코드 편집 창에 입력한 뒤 실행해보자

if_else_boolean_none.py

```
if True:
    print('참')    # True는 참
else:
    print('거짓')

if False:
    print('참')
else:
    print('거짓')    # False는 거짓

if None:
    print('참')
else:
    print('거짓')    # None은 거짓
```

실행 결과

```
참
거짓
거짓
```

14.3 if 조건문의 동작 방식 알아보기

if 조건문의 동작 방식 알아보기

- True는 if의 코드가 실행되고, False는 else의 코드가 실행됨
- None은 False로 취급되므로 else의 코드가 실행됨
- 실제 코드를 작성할 때 변수에 들어있는 값이나 함수의 결과가 None인 경우가 많으므로 이 부분은 꼭 기억해두자

14.3 if 조건문의 동작 방식 알아보기

if 조건문에 숫자 지정하기

- 숫자는 정수(2진수, 10진수, 16진수), 실수와 관계없이 0이면 거짓, 0이 아닌 수는 참임

if_else_number.py

```
if 0:
    print('참')
else:
    print('거짓')    # 0은 거짓

if 1:
    print('참')    # 1은 참
else:
    print('거짓')

if 0x1F:    # 16진수
    print('참')    # 0x1F는 참
else:
    print('거짓')

if 0b1000:    # 2진수
    print('참')    # 0b1000은 참
else:
    print('거짓')

if 13.5:    # 실수
    print('참')    # 13.5는 참
else:
    print('거짓')
```

실행 결과

```
거짓
참
참
참
참
```

14.3 if 조건문의 동작 방식 알아보기

if 조건문에 문자열 지정하기

- 문자열은 내용이 있을 때 참, 빈 문자열은 거짓임

if_else_string.py

```
if 'Hello':    # 문자열
    print('참')    # 문자열은 참
else:
    print('거짓')

if '':    # 빈 문자열
    print('참')
else:
    print('거짓')    # 빈 문자열은 거짓
```

```
참
거짓
```

- 불값 True와 True로 취급하는 10진수, 16진수, 2진수, 실수, 문자열로 if를 동작시킴
- 값 자체가 있으면 if는 동작함
- 반대로 0, None, ''은 False로 취급하므로 else가 동작함

14.4 조건식을 여러 개 지정하기

조건식을 여러 개 지정하기

- 만약 조건이 복잡할 때는 어떻게 해야 할까?
- 예를 들어 인터넷 포털의 중고나라에 글을 올리려면 먼저 포털 사이트의 회원이면서 중고나라 카페의 회원이라야 함
- 이 조건을 if 조건문으로 나타내면 다음과 같은 모양이 됨

```
if 포털 사이트 회원인지? 그리고 중고나라 회원인지?:
    글쓰기 화면 표시
else:
    포털 사이트 또는 중고나라 회원이 아니므로 글을 쓸 수 없다는 경고 문구 표시
```

14.4 조건식을 여러 개 지정하기

조건식을 여러 개 지정하기

- if 조건문에는 논리 연산자를 사용하여 조건식을 여러 개 지정할 수 있음
- IDLE의 소스 코드 편집 창에 입력한 뒤 실행해보자

if_else_multiple_cond_exp.py

```
x = 10
y = 20

if x == 10 and y == 20:     # x가 10이면서 y가 20일 때
    print('참')
else:
    print('거짓')
```

실행 결과

```
참
```

- x == 10 and y == 20처럼 and 논리 연산자를 사용하면 x가 10이면서 y가 20일 때 if의 코드가 실행됨
- 만약 둘 중 하나라도 만족했을 때 '참'이 출력되도록 하려면 or 논리 연산자를 사용하면 됨

14.4 조건식을 여러 개 지정하기

중첩 if 조건문과 논리 연산자

- 그럼 이런 논리 연산자를 어디에 사용할까?
- 보통 여러 조건을 판단할 때 if를 계속 나열해서 중첩 if 조건문으로 만드는 경우가 많음
- 예를 들어 x가 양수이면서 20보다 작은지 판단하려고 함

```
if x > 0:
    if x < 20:
        print('20보다 작은 양수입니다.')
```

- if로 x가 0보다 큰지 검사하고(0보다 크면 양수), 다시 if로 20보다 작은지 검사함
- 중첩 if 조건문은 and 논리 연산자를 사용해서 if 하나로 줄일 수 있음

```
if x > 0 and x < 20:
    print('20보다 작은 양수입니다.')
```

14.4 조건식을 여러 개 지정하기

중첩 if 조건문과 논리 연산자

- x가 0보다 크면서 20보다 작을 때처럼 and 논리 연산자를 사용해서 두 조건을 모두 만족하면 '20보다 작은 양수입니다.'를 출력하도록 만들었음
- 파이썬에서는 이 조건식을 더 간단하게 만들 수 있음

```
if 0 < x < 20:
    print('20보다 작은 양수입니다.')
```

- 0 〈 x 〈 20처럼 부등호를 연달아서 사용함
- 여기서는 0이 앞에 왔으므로 0보다 큰지 판단하는 부등호는 방향이 반대로 바뀌었음
- 조건식을 만들 때는 부등호의 방향과는 관계 없이 조건의 뜻만 만족하면 됨
- x 〉 0과 0 〈 x의 뜻은 같음
- if의 조건식이 참일 때는 if의 코드를 실행하고, 거짓일 때는 else의 코드를 실행한다는 점이 중요함
- 조건식이 여러 개일 때는 논리 연산자를 활용한다는 점도 기억해두자