Ch6. 조건문

- **if 조건문으로 특정 조건일 때 코드 실행하기**
  - 조건문을 사용하면 조건에 따라 다른 코드를 실행할 수 있음

```
if 세탁 완료 소리가 울리면:
    빨래를 꺼내서 말린다.
```

  - 다음과 같이 날씨에 따라 행동할 수도 있음

```
if 비가 온다면:
    우산을 가지고 나간다.

if 날씨가 춥다면:
    코트를 입고 나간다.

if 날씨가 덥다면:
    반소매에 얇은 옷을 입고 나간다.
```

- **if 조건문 사용하기**
  - **if 조건문은 if에 조건식을 지정하고 :(콜론)을 붙이며 다음 줄에 실행할 코드가 옴**
  - **이때 실행할 코드는 반드시 들여쓰기를 해야 함**

```
if 조건식:
    코드
```

  - IDLE의 파이썬 셸에서 if 조건문을 사용해보자

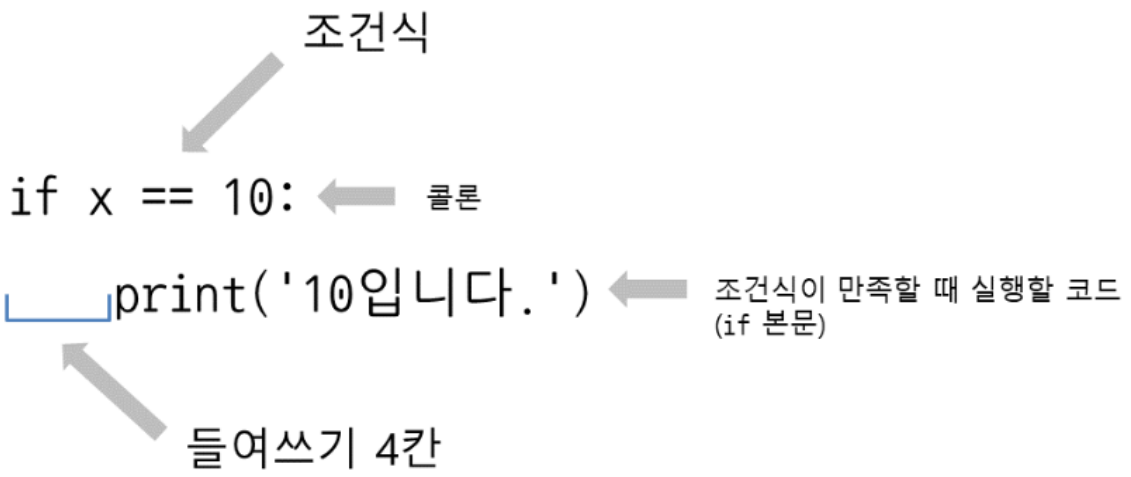
```
>>> x = 10
>>> if x == 10:
...     print('10입니다.')
...
10입니다.
```

- **if 조건문 사용하기**
  - **만약 if 다음 줄에서 들여쓰기를 하지 않으면 들여쓰기 에러가 발생함**

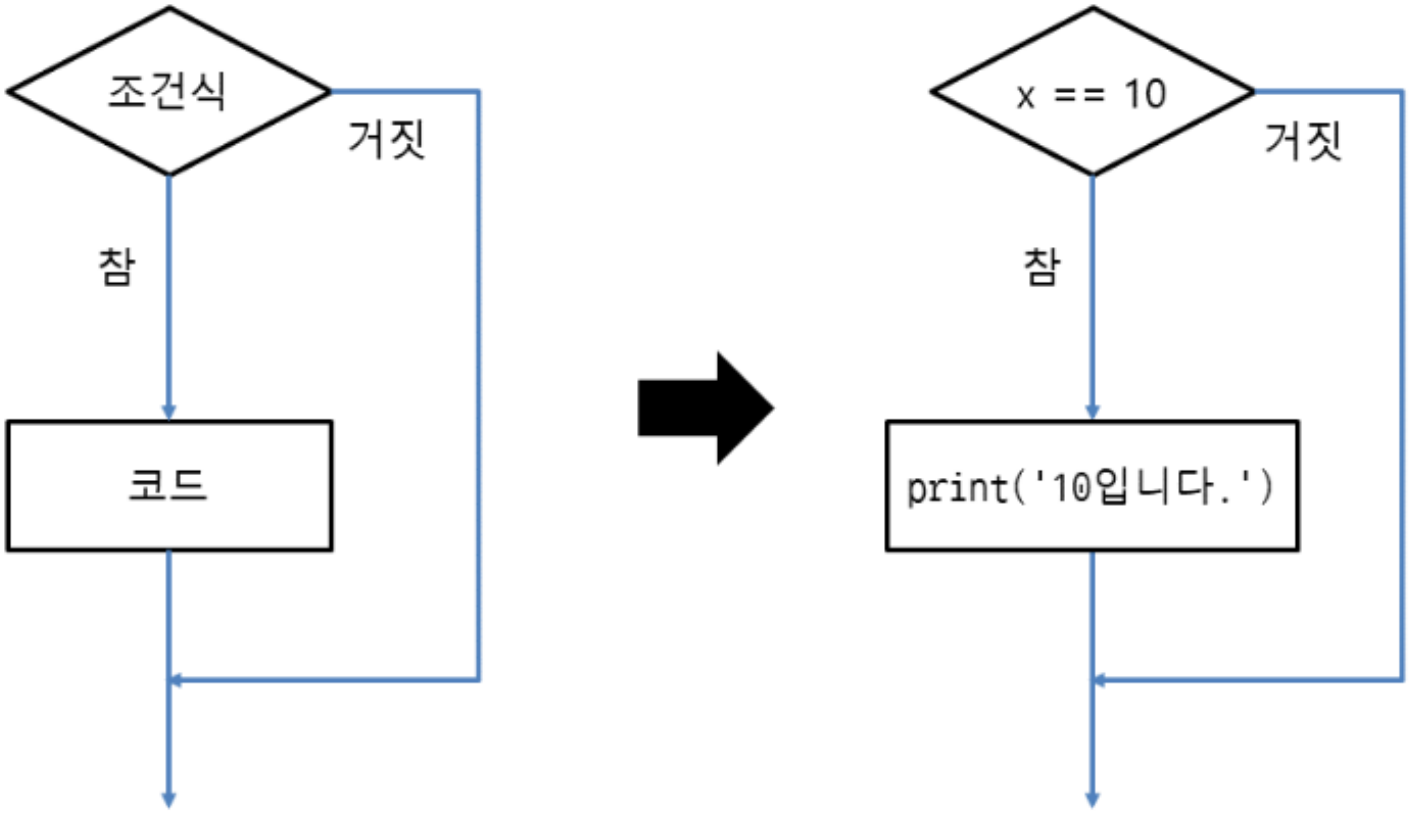
```
>>> x = 10
>>> if x == 10:
... print('10입니다.')
  File "<stdin>", line 2
    print('10입니다.')
        ^
IndentationError: expected an indented block
```

  - 참고로 IDLE의 파이썬 셸에서는 자동으로 들여쓰기가 되지만,
    콘솔(터미널, 명령 프롬프트)에서 실행한 파이썬 셸에서는
    자동으로 들여쓰기가 되지 않으므로 반드시 들여쓰기 필수

- **if 조건문의 기본 형태와 실행 흐름 알아보기**
  - **파이썬에서 if 조건문은 if 조건식: 형식으로 사용하며** 그다음 줄에는 들여쓰기를 한 뒤 조건식이 만족할 때 실행할 코드를 넣음
  - 이 조건식이 만족할 때 실행할 코드를 if 본문(if body)이라고 부름



- **if 실행 흐름**



- **if 조건문을 사용할 때 주의할 점**
  - =은 할당, ==은 비교
  - **if에 =을 사용하면 문법 에러가 발생함**

```
>>> if x = 10:
  File "<stdin>", line 1
    if x = 10:
         ^
SyntaxError: invalid syntax
```

  - **조건식 끝에 :을 빠뜨렸을 때의 경우 에러 발생**

```
>>> if x == 10
  File "<stdin>", line 1
    if x == 10
              ^
SyntaxError: invalid syntax
```

  - 문법 에러가 발생하면 콘솔에서는 잘못된 코드 아래에 ^가 표시되고, IDLE에서는 빨간색으로 표시되므로 자신이 실수한 부분을 쉽게 찾을 수 있음

- **if 조건문에서 코드를 생략하기**

```
>>> x = 10
>>> if x == 10:
...     pass
...
>>>
```

  - if 다음 줄에 pass라는 특별한 키워드를 넣었음
  - pass는 아무 일도 하지 않고 그냥 넘어간다는 뜻
  - pass만 넣고 나중에 할 일은 주석으로 남겨놓는 방식임

```
if x == 10:
    pass    # TODO: x가 10일 때 처리가 필요함
```

- **if 조건문과 들여쓰기**
  - 파이썬은 들여쓰기도 문법으로 정해져 있으며 if 조건문도 들여쓰기가 중요함

if_indent_error.py

```
x = 10

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

  - 위의 예제는 두 번째 print 부분에서 unexpected indent 에러가 발생함
  - 올바른 코드로 고쳐 보면 다음과 같이 두 번째 print도 들여쓰기 4칸으로 수정

if_indent.py

```
x = 10

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

실행 결과

```
x에 들어있는 숫자는
10입니다.
```

- **if 조건문과 들여쓰기**
  - 첫 번째 print만 들여쓰기를 하고, 두 번째 print는 들여쓰기를 하지 않으면 의도치 않은 동작이 됨

if_wrong_indent_true.py

```
x = 10

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

실행 결과

```
x에 들어있는 숫자는
10입니다.
```

if_wrong_indent_false.py

```
x = 5          # x에 5를 할당

if x == 10:    # x가 5라서 조건식을 만족하지 않음
    print('x에 들어있는 숫자는')
print ('10입니다.')
```

실행 결과

```
10입니다.
```

- **if 조건문과 들여쓰기**

```
x = 5

if x == 10:    # x가 5라서 조건식이 만족하지 않음
    print('x에 들어있는 숫자는')

print('10입니다.')    # 위의 if와는 상관없는 코드
```

- **중첩 if 조건문 사용하기**
  - 변수의 값이 10 이상이면 '10 이상입니다.'를 출력한 뒤 15이면 '15입니다.', 20이면 '20입니다.'를 출력함

if_if.py

```
x = 15

if x >= 10:
    print('10 이상입니다.')

    if x == 15:
        print('15입니다.')

    if x == 20:
        print('20입니다.')
```

실행 결과

```
10 이상입니다.
15입니다.
```

- **중첩 if 조건문 사용하기**
  - 들여쓰기가 된 if x == 15:와 if x == 20:은 처음에 나온 if x >= 10:에 속한 코드임
  - if x >= 10:의 조건식이 만족해야만 실행되는 코드임
  - 안쪽의 if에 속한 print는 들여쓰기를 한 번 더 실행

```
if x >= 10:
    print('10 이상입니다.')

    if x == 15:
        print('15입니다.')

    if x == 20:
        print('20입니다.')
```

- **사용자가 입력한 값에 if 조건문 사용하기**

if_input.py

```
x = int(input())          # 입력받은 값을 변수에 저장

if x == 10:               # x가 10이면
    print('10입니다.')    # '10입니다.'를 출력

if x == 20:               # x가 20이면
    print('20입니다.')    # '20입니다.'를 출력
```

  - 스크립트 파일을 실행한 뒤 10을 입력하고 엔터 키를 누르자

실행 결과

```
10 (입력)
10입니다.
```

  - if 조건문은 조건식이 만족했을 때 코드를 실행한다는 점이 중요함
  - if 조건문은 들여쓰기에 따라 문법 에러가 발생하거나, 의도치 않은 동작이 나올 수 있으므로 들여쓰기 규칙을 정확히 사용

- **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
  - 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.
```

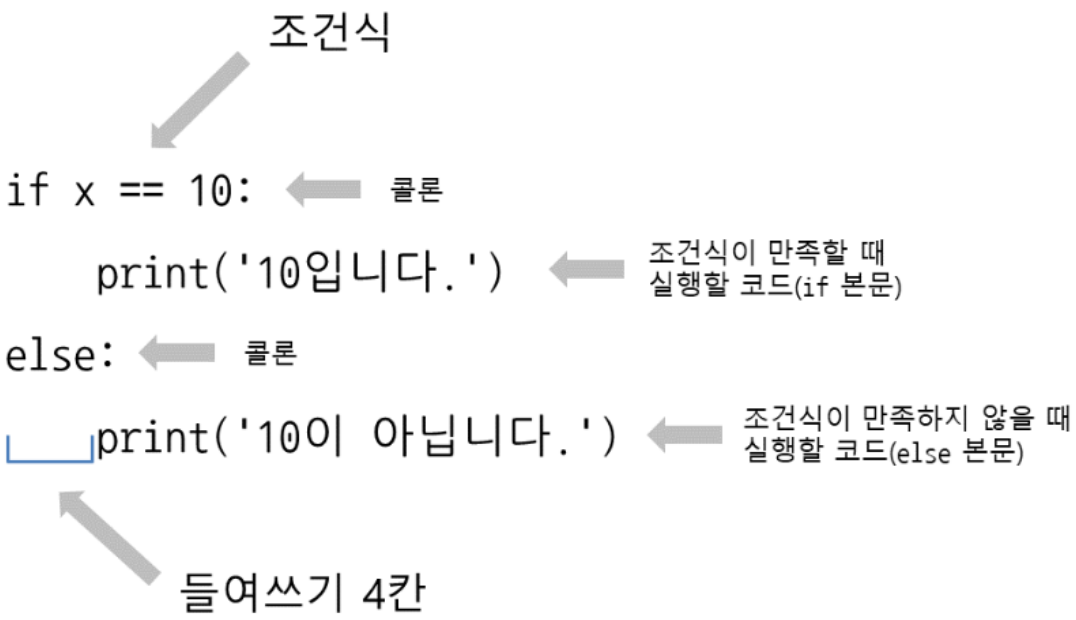
- **else 사용하기**
  - **else는 if 조건문 뒤에 오며 단독으로 사용할 수 없음**
  - **if와 마찬가지로 else도 :(콜론)을 붙이며 다음 줄에 실행할 코드가 옴**

```
if 조건식:
    코드1
else:
    코드2
```

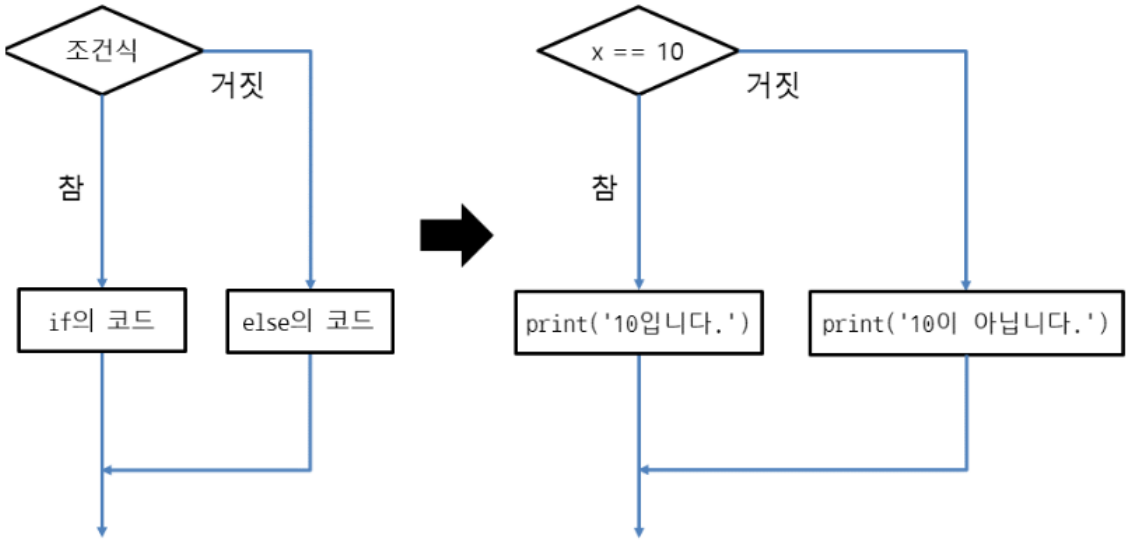
  - 들여쓰기를 맞춰서 파이썬 셸에 코드를 입력해보자

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

- **if와 else의 기본 형태와 실행 흐름 알아보기**
  - else는 if의 조건식이 만족하지 않을 때 코드를 실행함



- **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



- **else와 들여쓰기**
  - else는 if와 들여쓰기 규칙이 같음

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.')
```

- **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.
```

```
x = 10

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

print('10이 아닙니다.')
```

- **if 조건문의 동작 방식 알아보기**
  - 조건식이 아닌 값으로 if와 else의 코드를 동작
  - IDLE의 소스 코드 편집 창에 입력한 뒤 실행
  - None은 False로 취급되므로 else의 코드가 실행됨

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
```

- **if 조건문에 숫자 지정**
  - 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
```

- **if 조건문에 문자열 지정**
  - 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
```

  - **값 자체가 있으면 if는 동작함**
  - **반대로 0, None, ''은 False로 취급하므로 else가 동작함**

- **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
```

- **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 사용

- **중첩 if 조건문과 논리 연산자**
  - 그럼 이런 논리 연산자를 어디에 사용할까?
  - 보통 여러 조건을 판단할 때 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.')
```

- **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 사용**

- **elif를 사용하여 여러 방향으로 분기하기**
  - 프로그램을 만들다 보면 참, 거짓으로만 분기하는 것은 한계가 있음
  - 실제로는 두 가지 이상의 다양한 상황이 발생함
  - 여러 가지 상황을 처리하는 대표적인 예는 음료수 자판기가 있음
  - 자판기 안에는 각각 다른 종류의 음료수가 들어있고, 버튼을 누르면 해당 버튼에 해당하는 음료수가 나옴

```
if 콜라 버튼을 눌렀다면:
    콜라를 내보냄
elif 사이다 버튼을 눌렀다면:
    사이다를 내보냄
elif 환타 버튼을 눌렀다면:
    환타를 내보냄:
else:
    제공하지 않는 메뉴
```

- **elif 사용하기**
  - **if, else와 마찬가지로 조건식 끝에 :(콜론)을 붙여야 하고, elif 단독으로 사용할 수 없음**

```
if 조건식:
    코드1
elif 조건식:
    코드2
```

  - 들여쓰기를 맞춰서 파이썬 셸에 코드를 입력해보자

```
>>> x = 20
>>> if x == 10:
...     print('10입니다.')
... elif x == 20:
...     print('20입니다.')
...
20입니다.
```

  - elif로 만들면 다음과 같은 모양이 됨

- **if, elif, else를 모두 사용하기**

```
if 조건식:
    코드1
elif 조건식:
    코드2
else:
    코드3
```

if_elif_else.py

```
x = 30

if x == 10:             # x가 10일 때
    print('10입니다.')
elif x == 20:           # x가 20일 때
    print('20입니다.')
else:                   # 앞의 조건식에 모두 만족하지 않을 때
    print('10도 20도 아닙니다.')
```

실행 결과

```
10도 20도 아닙니다.
```

- **if, elif, else를 모두 사용하기**

```
if x == 10:
    print('10입니다.')
else:
    print('10도 20도 아닙니다.')
elif x == 20:    # elif 앞에 else가 오면 잘못된 문법
    print('20입니다.')
```

- **음료수 자판기 만들기**
  - 음료수 자판기를 만들어보자
  - 버튼 1번은 '콜라', 2번은 '사이다', 3번은 '환타'이고 각 버튼에 따라 음료수 이름을 출력한다고 함(1, 2, 3이외의 숫자는 '제공하지 않는 메뉴' 출력)
  - 코드를 보기 전에 어떻게 만들면 될 지 머리 속으로 한 번 생각해보자

vending_machine.py

```
button = int(input())

if button == 1:
    print('콜라')
elif button == 2:
    print('사이다')
elif button == 3:
    print('환타')
else:
    print('제공하지 않는 메뉴')
```

실행 결과

```
1 (입력)
콜라
```

- **cf) 리스트, if문 적용**

```
pocket=['paper', 'cellphone', 'money' ]
if 'money' in pocket:
    print('택시 타고간다')
else :
    print('걸어 간다')
```

```
택시 타고간다
>>>
```

```
pocket = ['paper', 'cellphone']
card = True
if 'money' in pocket:
    print("택시를 타고가라")
elif card:
    print("택시를 타고가라")
else:
    print("걸어가라")
```

```
택시를 타고가라
>>>
```

- cf) if 조건문만 사용할 때 elif를 사용할 때 차이점
- 1) if 조건문만 사용

```
a, b, c = 10, 20, 30
if a==10:
    print('10')

if b==20:
    print('20')

if c==30:
    print('30')
```

  - 모든 if 조건문의 조건식 검사 후 코드 실행

```
10
20
30
```

- cf) if 조건문만 사용할 때 elif를 사용할 때 차이점
- 2) elif 사용

```
a, b, c = 10, 20, 30
if a==10:
    print('10')

elif b==20:
    print('20')

elif c==30:
    print('30')
```

  - if 뒤에 elif가 연결된 경우, 첫번째 if의 조건식이 만족하면 다음을 건너뜀

```
10
```

  - if, elif는 여러 코드 중에서 하나만 실행할 때 사용