제어문(Control Statement)

조건문(Conditional Statement)

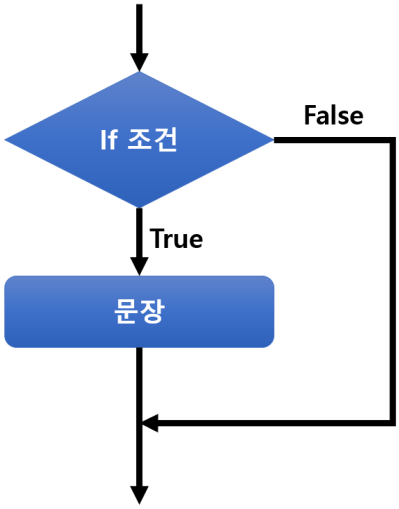
- 조건에 따라 문장을 수행
- 주어진 조건을 판단하고 상황에 맞는 처리가 필요할 때 사용
- 파이썬에서 제공하는 조건문
    - if
    - else
    - elif

if 문

- if 문은 True와 False를 판단하는 조건문
- if 조건 뒤에는 콜론(:)
- if 기본 문법

```
if <조건>:
  <문장>
```

- if 구조도



- if 문을 이용한 미세먼지 측정
    - 미세먼지 농도 pm
    - 35 초과는 미세먼지 농도 나쁨

```
pm = 40
if pm > 35:
  print("미세먼지 농도: 나쁨")
```

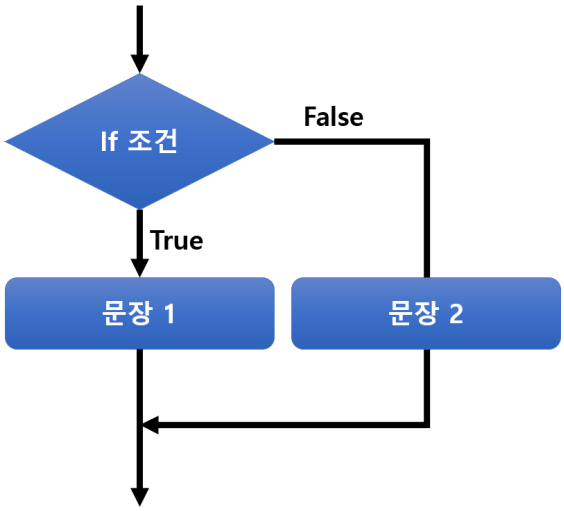
```
미세먼지 농도: 나쁨
```

if-else 문

- if-else 기본 문법

```
if <조건>:
  <문장 1>
else:
  <문장 2>
```

- else 문 뒤에는 콜론(:)
- if-else 구조도



- if-else 문을 이용한 미세먼지 측정
  - 미세먼지 농도 pm
  - 36 이상은 미세먼지 농도 나쁨
  - 35 이하는 미세먼지 농도 좋음

```
pm = 30
if pm >= 36:
  print("미세먼지 농도: 나쁨")
else:
  print("미세먼지 농도: 좋음")
```

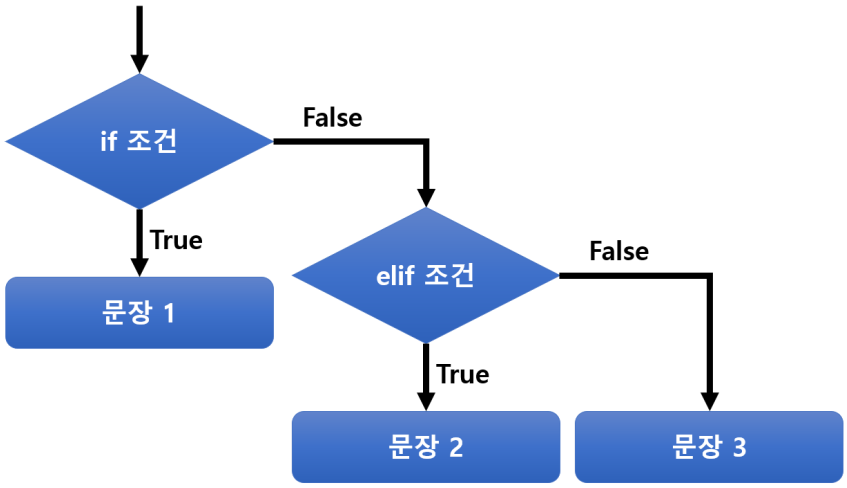
```
미세먼지 농도: 좋음
```

if-elif-else 문

- if-elif-else 기본 문법

```
if <조건>:
  <문장 1>
elif <조건>:
  <문장 2>
else:
  <문장 3>
```

- elif 문 조건 뒤에는 콜론(:)
- if-elif-else 구조도



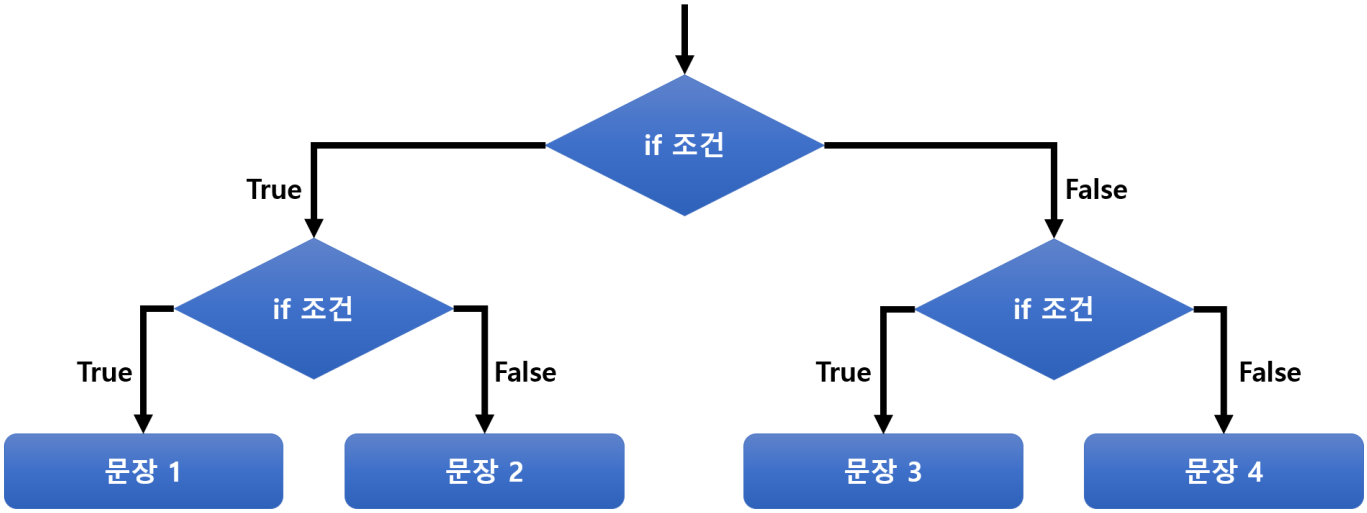
- if-elif-else 문을 이용한 미세먼지 측정
  - 미세먼지 농도 pm
  - 미세먼지 농도 0~15: 좋음
  - 미세먼지 농도 16~35: 보통
  - 미세먼지 농도 36~75: 나쁨
  - 미세먼지 농도 76~: 매우나쁨

```
pm = 40
if pm < 16:
  print("미세먼지 농도: 좋음")
elif pm < 36:
  print("미세먼지 농도: 보통")
elif pm < 76:
  print("미세먼지 농도: 나쁨")
else:
  print("미세먼지 농도: 매우나쁨")
```

```
미세먼지 농도: 나쁨
```

중첩 if 문

- if 문 안에 if 문에 포함된 형태



- 중첩 if 문을 이용한 미세먼지 측정
  - 미세먼지 농도 pm
  - 미세먼지 농도 0~15: 좋음
  - 미세먼지 농도 16~35: 보통
  - 미세먼지 농도 36~75: 나쁨
  - 미세먼지 농도 76~: 매우나쁨

```
pm = 80
if pm < 36:
  if pm < 16:
    print("미세먼지 농도: 좋음")
  else:
    print("미세먼지 농도: 보통")
else:
  if pm < 76:
    print("미세먼지 농도: 나쁨")
  else:
    print("미세먼지 농도: 매우나쁨")
```

```
미세먼지 농도: 매우나쁨
```

if-pass 문

- 조건문은 있지만 실행할 문장이 없는 경우, 오류가 발생하지 않도록 무시하고 넘어가는 기능

```
if 10 > 5:
  print(10)
else:
  pass
```

```
10
```

if 조건 연산자

- 비교연산자: <, >, ==, !=, >=, <=

```
if 2 > 1:
  print(2)

if 3 == 3:
  print(3)

if 1 != 2:
  print(1)
```

```
2
3
1
```

- 논리연산자: and, or, not

```
rain = True
snow = True
sun = False

if rain and snow:
  print("진눈깨비")

if not sun:
  print("흐림")
else:
  print("맑음")
```

```
진눈깨비
흐림
```

- 멤버연산자: in, not in

```
list = ['One', 'Two', 'Three']

if 'One' in list:
  print('One')

if 'Four' not in list:
  print('No')
```

```
One
No
```

- 식별연산자: is, is not

```
if 'One' is 'One':
  print('One')

if 'One' is not 'Two':
  print('One is not Two')
```

```
One
One is not Two
```

조건부 표현식(Conditional Expression)

- 한 라인으로 조건식을 사용한 표현

```
score = 75
msg = "통과" if score >= 70 else "탈락"
print(msg)
```

```
통과
```

[Lab] 학생 종류

- 8세 미만 "학생 아님"
- 14세 미만 "초등학생"
- 17세 미만 "중학생"
- 20세 미만 "고등학생"
- 26세 미만 "대학생"
- 그 외에 "학생 아님"

```
age = 22
if age < 8:
  print("학생 아님")
elif age < 14:
  print("초등학생")
elif age < 17:
  print("중학생")
elif age < 20:
  print("고등학생")
elif age < 26:
  print("대학생")
else:
  print("학생 아님")
```

```
대학생
```

[Lab] 학점 계산

- 95점 이상 "A+"

- 90점 이상 "A"
- 85점 이상 "B+"
- 80점 이상 "B"
- 75점 이상 "C+"
- 70점 이상 "C"
- 65점 이상 "D+"
- 60점 이상 "D"
- 그 외 "F"

```
score = 78
if score >= 95:
  print("A+")
elif score >= 90:
  print("A")
elif score >= 85:
  print("B+")
elif score >= 80:
  print("B")
elif score >= 75:
  print("C+")
elif score >= 70:
  print("C")
elif score >= 65:
  print("D+")
elif score >= 60:
  print("D")
else:
  print("F")
```

```
C+
```

반복문(Repetitive Statement)

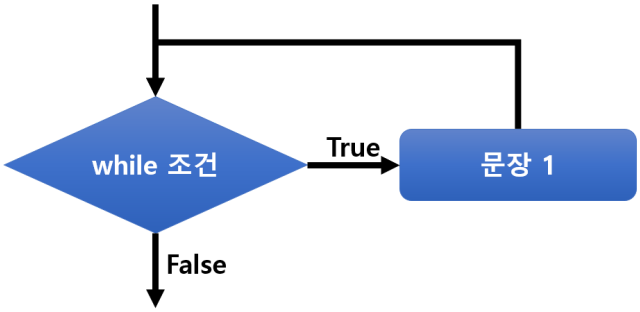
- 문장을 반복적으로 수행
- 정해진 동작을 반복하여 처리할 때 사용
- 파이썬에서 제공하는 반복문
  - while
  - for

while 문

- 어떤 조건이 만족하는 동안 문장을 수행하고 만족하지 않는 경우 수행 중단
- while 문 기본 문법

```
while <조건>:
  <문장>
```

- while 문 구조도



- while 문 예제: 1부터 10까지 반복

```
i = 1
while i <= 10:
  print(i)
  i += 1
```

```
1
2
3
4
5
6
7
8
9
10
```

- while 문 예제: 1부터 10까지 더하기

```
i = 1
sum = 0
while i <= 10:
  sum += i
  i += 1

print(sum)
```

```
55
```

for 문

- 반복 범위를 지정하여 반복 수행
- for 문 기본 문법

```
for 변수 in 리스트, 튜플, 문자열:
  <문장>
```

- 리스트 요소 반복

```
list = ['One', 'Two', 'Three']
for i in list:
  print(i)
```

```
One
Two
Three
```

- 튜플 요소 반복

```
tuple = ('One', 'Two', 'Three')
for i in tuple:
  print(i)
```

```
One
Two
Three
```

- 문자열 요소 반복

```
string = "SuanLab"
for i in string:
  print(i)
```

```
S
u
a
n
L
a
b
```

range()

- 범위 반복에 사용
- range 문법

```
for 변수 in range(시작값, 마지막값, 증가값):
  <문장>
```

- 시작값과 증가값은 생략 가능
- 생략할 때 시작값은 0, 증가값은 1

```
sum = 0
for i in range(101):
  sum += i

print(sum)
```

```
5050
```

```
for i in range(1, 10, 2):
  print(i)
```

```
1
3
5
7
9
```

- 범위 반복 range를 이용한 구구단

```
for i in range(2, 10):
  for j in range(1, 10):
    print('{0} x {1} = {2}'.format(i, j, i * j))
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
5 x 1 = 5
5 x 2 = 10
```

```
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
```

_ 기능

- for문을 사용하면서 iterator 역할을 위해서 i 변수가 필요
- for문 이후에는 iterator 변수 i가 필요하지 않음
- 이후에 사용되지 않을 변수에 어떤 이름을 부여하고 싶지 않을 때 _ 를 사용

```
for i in range(5):
  print("SuanLab")
```

```
SuanLab
SuanLab
SuanLab
SuanLab
SuanLab
```

```
for _ in range(5):
  print("SuanLab")
```

```
SuanLab
SuanLab
SuanLab
SuanLab
SuanLab
```

else 문

- 반복이 종료된 후에 한번 더 실행되는 문장

```
i = 1
sum = 0
while i <= 10:
  sum += i
  i += 1
else:
  print(sum)
```

```
55
```

```
sum = 0
for i in range(11):
  sum += i
else:
  print(sum)
```

```
55
```

break 문

- 반복문 종료

```
i = 0
while i < 100:
  print(i)
  if i == 10:
    break
  i += 1
```

```
0
1
2
3
4
5
6
7
8
9
10
```

```
for i in range(100):
  print(i)
  if i == 10:
    break
```

```
0
1
2
```

```
3
4
5
6
7
8
9
10
```

continue 문

- 반복 조건문으로 이동

```
i = 0
while i < 10:
  i += 1
  if i % 2 == 0:
    continue
  print(i)
```

```
1
3
5
7
9
```

```
for i in range(10):
  if i % 2 == 0:
    continue
  print(i)
```

```
1
3
5
7
9
```

리스트 내포(List Comprehension)

- 리스트 안에 for 문과 if 문 사용

```
list = [1, 2, 3, 4, 5]
print([i * 2 for i in list])
print([i * 2 for i in range(10) if i % 2 == 0])
```

```
[2, 4, 6, 8, 10]
[0, 4, 8, 12, 16]
```

[Lab] 문자열 반전

- 문자열을 역순으로 만들기

```
string = "SuanLab"
reverse = ""
for c in string:
  reverse = c + reverse

print(string)
print(reverse)
```

```
SuanLab
baLnauS
```

에러와 예외(Errors and Exceptions)

- 프로그램에서는 에러가 발생
- 에러에 대한 예외 처리가 필요

에러와 예외 분류

에러	설명
AssertionError	assert 문이 실패할 때 발생
AttributeError	속성에 참조가 대입이 실패할 때 발생
EOFError	EOF(End Of File) 조건을 만날 때 발생
ImportError	import 문이 모듈 로드에 문제가 있을 때 발생
ModuleNotFoundError	import 할 모듈을 찾을 수 없을 때 발생
IndexError	시퀀스 인덱스가 범위를 벗어날 때 발생
KeyError	딕셔너리 키가 키 집합에서 발견되지 않을 때 발생
KeyboardInterrupt	사용자가 인터럽트 키(Ctrl + C 또는 Delete)를 누를 때 발생
MemoryError	작업에 메모리가 부족하지만, 일부 객체를 삭제해서 복구될 수 있는 경우 발생
NameError	지역 또는 전역 이름을 찾을 수 없을 때 발생
NotImplementedError	RuntimeError 에서 내용이 없는 메소드 호출 시 발생
OSError	운영체제에서 에러가 있을 경우 발생
OverflowError	산술 연산의 결과가 너무 커서 표현할 수 없을 때 발생
RuntimeError	다른 범주에 속하지 않는 에러가 감지될 때 발생
SyntaxError	문법 오류가 발견될 때 발생
IndentationError	잘못된 들여쓰기와 관련된 문법 오류가 있을 때 발생
TabError	들여쓰기가 일관성 없이 탭과 스페이스를 사용하는 경우 발생
SystemError	인터프리터 내부에서 에러가 발견될 때 발생
SystemExit	`sys.exit()` 함수가 호출될 때 발생
TypeError	연산이나 함수가 부적절한 형의 객체에 적용될 때 발생
UnboundLocalError	함수나 메소드에서 지역 변수를 참조하지만, 해당 변수에 값이 연결되지 않으면 발생
UnicodeError	유니코드 관련 인코딩/디코딩 에러가 일어날 때 발생
ValueError	연산이나 함수가 올바른 자료형이지만 부적절한 값을 가진 인자를 받았을 때 발생

에러	설명
ZeroDivisionError	0으로 다른 숫자를 나누는 경우에 발생
FileExistsError	이미 존재하는 파일이나 디렉토리를 만들려고 할 때 발생
FileNotFoundError	파일이나 디렉토리가 요청되었지만 존재하지 않을 때 발생
InterruptedError	시스템 호출이 들어오는 시그널에 의해 중단될 때 발생
IsADirectoryError	디렉토리에 파일 연산이 요청되었을 때 발생
NotADirectoryError	디렉토리가 아닌 것에 디렉토리 연산이 요청되었을 때 발생
PermissionError	적절한 접근권 없이 연산을 실행하려고 할 때 발생
ProcessLockupError	주어진 프로세스가 존재하지 않을 때 발생
TimeoutError	시스템 함수가 시스템 수준에서 시간이 초과될 때 발생
Warning	오류는 아니고 주의가 필요한 경고 시 발생

에러 예제

- ZeroDivisionError

```
10 / 0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-75-cd759d3fcf39> in <module>()
----> 1 10 / 0

ZeroDivisionError: division by zero

SEARCH STACK OVERFLOW
```

- NameError

```
noname + 3
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-76-e253214df6f7> in <module>()
----> 1 noname + 3

NameError: name 'noname' is not defined

SEARCH STACK OVERFLOW
```

- TypeError

```
'1' + 1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-77-cc892b1f57d5> in <module>()
----> 1 '1' + 1

TypeError: must be str, not int
```

- ValueError

```
int("String")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-80-7bf3cc52d08f> in <module>()
----> 1 int("String")

ValueError: invalid literal for int() with base 10: 'String'
```

SEARCH STACK OVERFLOW

예외 처리(Exception Handling)

try, except 문

- try 문을 통해 예외 발생 가능한 코드에 정의
- except 문을 이용해 예외 처리 방법을 정의

```
try:
  print(2 / 0)
except:
  print("Error")
```

```
Error
```

에러 분류에 따른 except 문

- `ZeroDivisionError` 에 대한 except 처리

```
try:
  print(2 / 0)
except ZeroDivisionError:
  print("ZeroDivisionError")
```

```
ZeroDivisionError
```

에러 메시지가 포함된 except 문

- `ZeroDivisionError` 에 포함된 에러 메시지 `e` 를 출력

```
try:
  print(2 / 0)
except ZeroDivisionError as e:
  print(e)
```

```
division by zero
```

여러 에러 분류가 포함된 except 문

```
try:
  n = int(input("입력: "))
  result = 1 / n
except ZeroDivisionError as e:
  print(e)
except ValueError as e:
  print(e)
else:
  print("결과: " + str(result))
```

```
입력: 1
결과: 1.0
```

에러 발생

raise 문

- 직접적으로 에러를 발생시킬 수 있음
- 에러의 종류와 에러 상황을 정의하고 발생

```
raise ZeroDivisionError("숫자를 0으로 나눔")
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-89-fb78988de7bc> in <module>()
----> 1 raise ZeroDivisionError("숫자를 0으로 나눔")

ZeroDivisionError: 숫자를 0으로 나눔

SEARCH STACK OVERFLOW
```

```
raise NameError("지역 또는 전역 이름을 찾을 수 없음")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
raise TypeError("연산이나 함수가 부적절한 형의 객체에 적용")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-92-90a278017362> in <module>()
----> 1 raise TypeError("연산이나 함수가 부적절한 형의 객체에 적용")

TypeError: 연산이나 함수가 부적절한 형의 객체에 적용
```

SEARCH STACK OVERFLOW

```
raise ValueError("연산이나 함수가 부적잘한 값을 인자로 받음")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-93-d52f59f4de6f> in <module>()
----> 1 raise ValueError("연산이나 함수가 부적잘한 값을 인자로 받음")

ValueError: 연산이나 함수가 부적잘한 값을 인자로 받음
```

SEARCH STACK OVERFLOW

assert 문

- raise 문과 달리 주로 검증을 위한 목적으로 작성
- 상태를 검증하는 목적으로 작성되며 검증식이 포함됨
- 검증식이 `False` 일 경우 `AssertionErro` 발생

```
assert True
```

```
assert False
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-96-a871fdc9ebee> in <module>()
----> 1 assert False

AssertionError:
```

SEARCH STACK OVERFLOW

```
assert False, "에러 메시지"
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-97-1796aa5b1c60> in <module>()
```

```
b = 0
assert b != 0, "상태에 따라서 발생"
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-100-91ab28e2d37f> in <module>()
      1 b = 0
----> 2 assert b != 0, "상태에 따라서 발생"

AssertionError: 상태에 따라서 발생
```

SEARCH STACK OVERFLOW

```
```