

프로그램 흐름 제어

흐름 제어 시작 전에

- bool 자료형

```
>>> a = 5 > 3
```

```
>>> a
```

```
True
```

```
>>>
```

```
>>> a = 9 < 7
```

```
>>> a
```

```
False
```

```
>>>
```

```
>>> type(a)
```

```
<class 'bool'>
```

흐름 제어 시작 전에

- 논리 연산자

```
>>>not True
```

```
False
```

```
>>>not False
```

```
True
```

```
>>>not 0
```

```
True
```

```
>>>not -1
```

```
False
```

```
>>>not 1
```

```
False
```

```
>>>not None
```

```
True
```

흐름 제어 시작 전에

- 논리 연산자

```
>>>not 'ABC'    # 비어있지 않은 문자열 부정
```

```
False
```

```
>>>not ''       # 빈 문자열 부정
```

```
True
```

```
>>>not (1, 2, 3) # 비어있지 않은 튜플 부정
```

```
False
```

```
>>>not ()       # 빈 튜플 부정
```

```
True
```

```
>>>not []       # 빈 리스트 부정
```

```
True
```

```
>>>not {}       # 빈 딕셔너리 부정
```

```
True
```

흐름 제어 시작 전에

- 흐름 제어문과 조건문

```
>>> bool(False)
```

```
False
```

```
>>> bool(None)
```

```
False
```

```
>>> bool(0)
```

```
False
```

```
>>> bool(0.0)
```

```
False
```

```
>>> bool("")
```

```
False
```

```
>>> bool([])
```

```
False
```

```
>>> bool(() )
```

```
False
```

```
>>> bool({})
```

```
False
```

흐름 제어 시작 전에

- 흐름 제어문과 조건문

```
>>> bool('Hello')
```

```
True
```

```
>>> bool(123)
```

```
True
```

```
>>> bool([1, 2, 3])
```

```
True
```

```
>>> bool((1, 2, 3))
```

```
True
```

```
>>> bool({1, 2, 3})
```

```
True
```

흐름 제어 시작 전에

- 코드 블록과 들여쓰기

```
if a == 3:
```

```
→ print('조건 만족')
```

```
→ print('변수 a는 3.')
```

```
else:
```

```
→ print('변수 a는 3이 아님.')
```



코드 블록



코드 블록

분기문

- if 문

if 조건문:

→ 명령1

→ 명령2

else:

→ 명령3

→ 명령4

if 뒤에 조건식이 오고 반드시 마지막에 콜론(:)이 온다.

콜론(:) 뒤에는 들여쓰기로 이루어진 코드 블록이 옴.

if문의 조건값이 False일 때 수행이 되고, else 뒤에도 코드 블록이 오기 때문에 콜론(:)을 반드시 삽입.

분기문

- if 문

if 조건1:
→ 코드블록

첫번째 조건은 항상 if로 시작.

...
elif 조건2:
→ 코드블록

두번째 조건부터는 elif를 이용.

...
elif 조건3:
→ 코드블록

...
else:
→ 코드블록
...

마지막 else는 생략 가능.

반복문

- while 문

while 조건:

→ 코드블록

...

- for 문

for 반복변수 in 순서열:

→ 코드블록

...

반복문

- for 문

```
for i in range(0, 5, 1):  
    print(i)
```

```
for i in range(0, 10, 2):  
    print(i)
```

```
for i in range(0, 5):  
    print(i)
```

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

반복문

- continue

```
for i in range(10):  
    if i % 2 == 1:  
        continue  
    print(i)
```

- break

```
i = 0  
while (True):  
    i = i + 1  
    if i == 1000:  
        print('i가 {0}이 되었습니다. 반복문을 중단합니다.'.format(i))  
        break  
    print(i)
```