파이썬 프로그래밍

3. 제어문과 조건식

00

Contents

❖ 수업 목표

- 조건식을 이해하고 제어문에 활용할 수 있다.
- 연산자의 종류와 동작을 설명할 수 있다.
- if문의 문법과 동작을 설명할 수 있다.

❖ 세부 목표

00

1. 조건식

❖ 조건식이란?

- 계산의 결과 반환되는 값이 True(참) 또는 False(거짓)인 식

- 논리형 (Boolean, 불리언)
  - 참과 거짓을 표현하는 자료형
  - True / False 2가지 값 가짐

- 연산 결과가 True 또는 False인 연산자 : 비교 연산자, 논리 연산자

2. 비교 연산

❖ 비교 연산자

- 비교에 사용하는 연산자
- <, >, <=, >=, ==, !=
- 비교 연산자 왼쪽과 오른쪽의 피연산자(Operand)를 비교
- 결과는 항상 True 혹은 False
- 조건식은 비교 연산자를 사용하여 구성

00

2. 비교 연산

▪ 비교 연산자 종류

비교 연산자	의미	a=10, b=20	출력 결과
a==b	a와 b는 같다.	print(a == b)	False
a!=b	a와 b는 같지 않다.	print(a != b)	True
a<b	a는 b보다 작다.	print(a < b)	True
a<=b	a는 b보다 작거나 같다.	print(a <= b)	True
a>b	a는 b보다 크다.	print(a > b)	False
a>=b	a는 b보다 크거나 같다.	print(a >= b)	False

00

2. 비교 연산

▪ 비교 연산자 활용 코드

코드 5-2 값을 비교하는 코드

```
>>> print(1 < 2)
True
>>> print(2 < 1)
False
```

코드 5-3 값을 비교하는 코드

```
>>> print(2 > 1)
True
>>> print(2 > 2)
False
```

2. 비교 연산

▪ 비교 연산자 활용 코드

코드 5-6 값을 비교하는 코드

```
>>> print(1 == 1)
True
>>> print(2 == 1)
False
```

코드 5-7 값을 비교하는 코드

```
>>> print(1 != 2)
True
>>> print(2 != 2)
False
```

3. 논리 연산

❖ 논리연산자

- 참, 거짓을 판단하는 연산에 사용하는 **and, or, not** 연산자
- 2개 이상의 조건식을 결합하기 위해 사용

syntax : 논리 연산자

부울 데이터형 논리 연산자 부울 데이터형	True and Fasle

syntax : 정확한 코딩

1. **피연산자** : 부울 데이터형이어야 한다.
2. **실행 결과** : 논리 연산자의 실행 결과는 True 또는 False이다.
3. **부울 데이터형(Boolean Data Type)** : True나 False 두 가지 값을 갖는 데이터형이다.

3. 논리 연산

❖ 논리연산자

- **AND 연산자와 OR 연산자 결과**
  - AND 연산자 : 피연산자가 모두 참(True, 1)일 때만 True, 그 외는 모두 False
  - OR 연산자 : 피연산자가 모두 거짓(False, 0)일 때만 False, 그 외는 모두 Ture

AND 진리표

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR 진리표

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

3. 논리 연산

❖ 논리연산자

▪ and

- 두 조건이 모두 True인지 판단

코드 5-15 and의 결과를 확인하는 코드

```
>>> print(True and True)
True
>>> print(True and False)
False
>>> print(False and True)
False
>>> print(False and False)
False
```

3. 논리 연산

- **or**
  - 두 조건 중 하나라도 True인지 판단

코드 5-16 or의 결과를 확인하는 코드

```
>>> print(True or True)
True
>>> print(True or False)
True
>>> print(False or True)
True
>>> print(False or False)
False
```

- **not**
  - True를 False로, False를 True로 뒤집기

코드 5-17 not의 결과를 확인하는 코드

```
>>> print(not True)
False
>>> print(not False)
True
```

4. 연산자 우선순위

❖ 연산자 우선순위

▪ 산술, 비교, 논리 연산자 활용된 연산의 우선순위

syntax : 연산자 우선순위

A 연산자 (B 연산자 C 연산자 D)	2 * (3-5)

syntax : 정확한 코딩

1. **실행 결과** : 괄호 안의 연산을 가장 먼저 실행하여 결과는 -4가 출력된다.
2. **연산자 우선순위** : 괄호〉 산술 연산자 〉 비교 연산자 〉 논리 연산자

4. 연산자 우선순위

❖ 연산자 우선순위

▪ 연산자 우선순위 코드

```
print(9>4 and 3>2) # ----> ①
print(9<4 and 3>2)
print(9<4 or 3<2) # ----> ②
print(9<4 or 3>2)
```

〈화면 출력〉

```
True
False
False
True
```

▪ ① 코드 설명 : 비교 연산자와 논리 연산자 사용

- 비교 연산
  - 9>4 비교 연산의 결과는 True
  - 3>2 비교 연산의 결과는 True
- 논리 연산 : 앞선 비교 연산의 결과로 도출된 2개의 True를 and 연산
  - True and True 의 실행 결과 True 가 출력

4. 연산자 우선순위

❖ 연산자 우선순위

▪ 연산자 우선순위 코드

```
print(9>4 and 3>2) # ----> ①
print(9<4 and 3>2)
print(9<4 or 3<2) # ----> ②
print(9<4 or 3>2)
```

〈화면 출력〉

```
True
False
False
True
```

▪ ② 코드 설명 : 비교 연산자와 논리 연산자 사용

- 비교 연산
  - 9<4 비교 연산의 결과는 False
  - 3<2 비교 연산의 결과는 False
- 논리 연산 : 앞선 비교 연산의 결과로 도출된 2개의 False를 or 연산
  - False and False 의 실행 결과 False 가 출력

5. 제어문

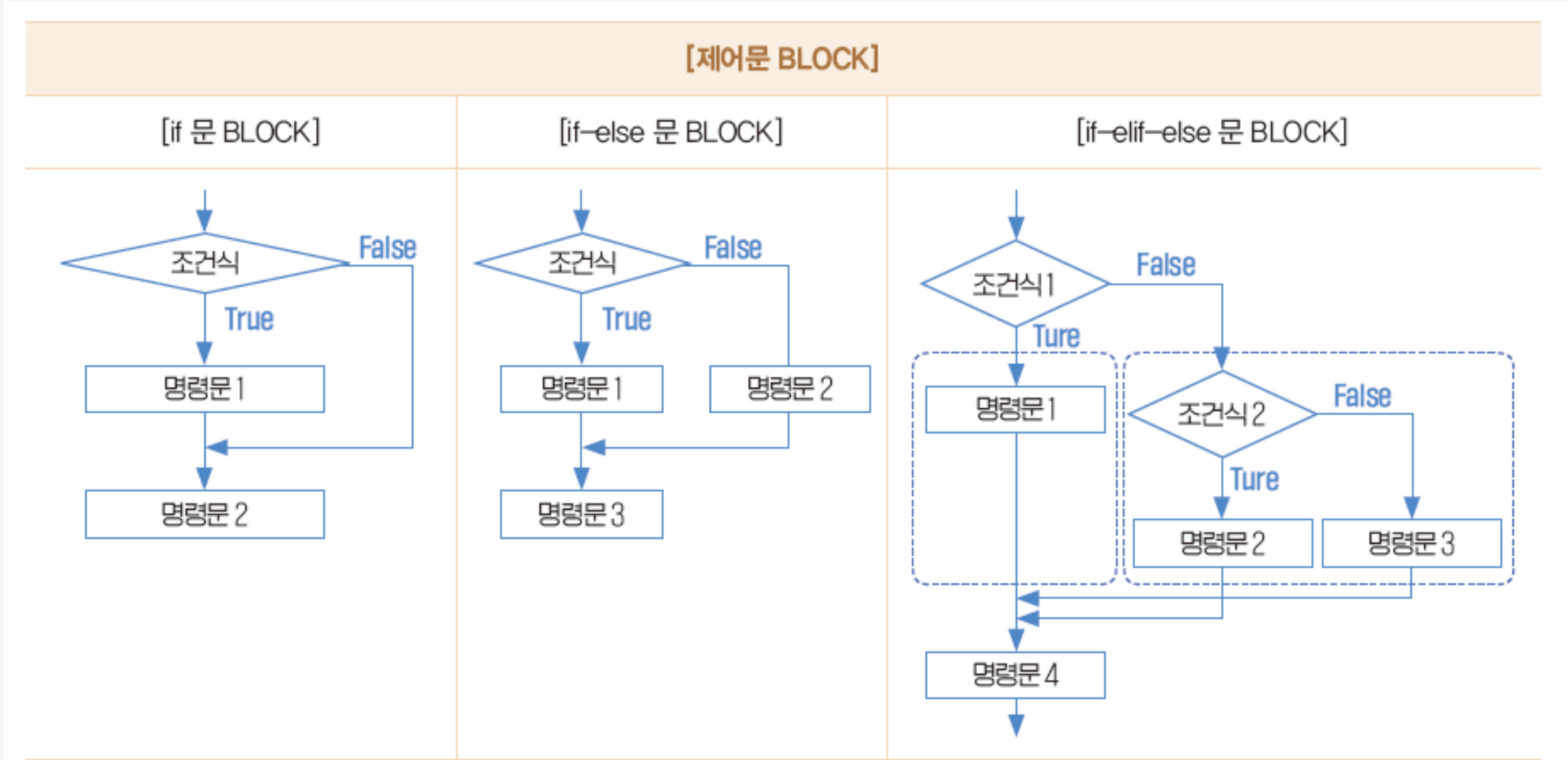
❖ 제어문이란?

- 프로그램의 흐름을 제어하는 구문
- 기본 프로그램의 흐름: 위에서 아래로 순차적 실행
- 선택적으로 실행하거나 반복해서 실행하는 듯 흐름의 제어 발생
- 종류: 조건문, 반복문

6. 조건문

❖ 조건문

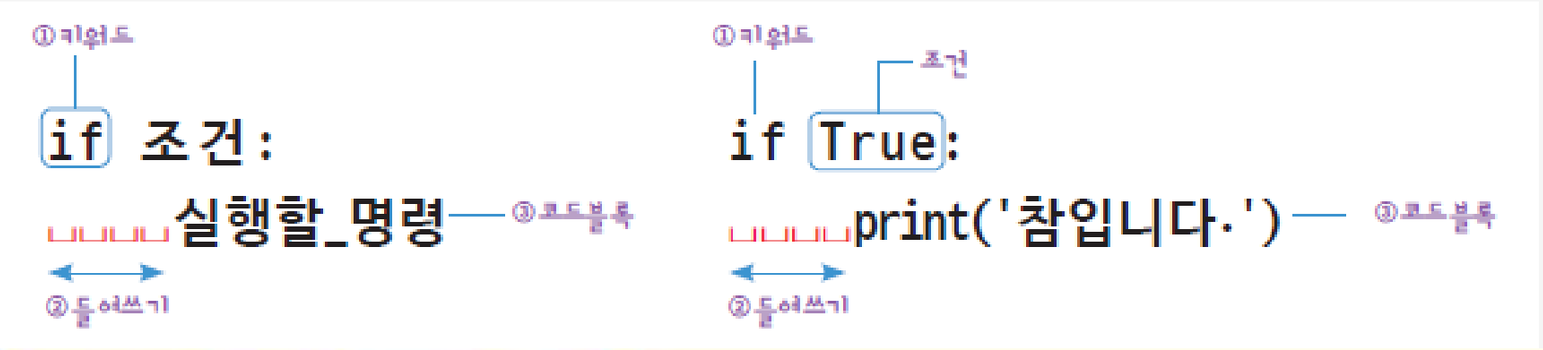
- 조건식의 결과에 따라 실행할 프로그램의 흐름을 선택



6. 조건문: if문

❖ if

- 판단한 결과에 따라 흐름 바꾸기 위해 사용
- 조건이 참인지 거짓인지에 따라 코드블록의 실행 여부 결정



코드 5-10 점수에 따라 합격여부를 출력하는 코드

```
01:  score = 90
02:  if score > 80:
03:      print('합격입니다.')
```

↳ 실행화면

```
합격입니다.
```

6. 조건문: if-else문

❖ else

- if와 달리 조건 없이 사용
- if-else 구조

```
if 조건:
    실행할_명령1 ── ①코드블록
else:
    실행할_명령2 ── ②코드블록
```

조건이 맞다면(True) 실행할_명령1을 실행하고
조건이 틀리다면(False) 실행할_명령2를 실행하세요.

코드 5-11 점수에 따라 합격여부를 출력하는 코드

```
01: score = 60
02: if score > 80:
03:     print('합격입니다.')
04: else:
05:     print('불합격입니다.')
```

↳ 실행화면

```
불합격입니다.
```

6. 조건문: if-else문

❖ elif

- if 조건이 False 일 때 elif 조건을 판단하여 다음 동작을 처리
  - if 조건이 True면 if 바로 다음 코드블록 실행 후, 조건문을 빠져나감
  - if조건이 False면 elif 조건을 판단
  - elif 조건이 True이면 elif 다음 코드블록 실행 후, 조건문을 빠져나감
  - elif 조건이 False면 다음 elif 조건을 판단 또는 else 코드블록을 수행

6. 조건문: if-elif-else문

- **if-elif-else 구조**

  - if의 조건이 True면 바로 아래의 첫 코드블록 실행
    - 두 번째 elif와 세 번째 else의 코드블록은 실행하지 않음
  - if의 조건이 False면 elif의 조건으로 넘어감
    - 이 때 elif의 조건이 True면 바로 아래의 두 번째 코드블록 실행
    - False면 세 번째 코드블록으로 넘어감

- **elif와 else는 반드시 if와 함께 사용해야 함**

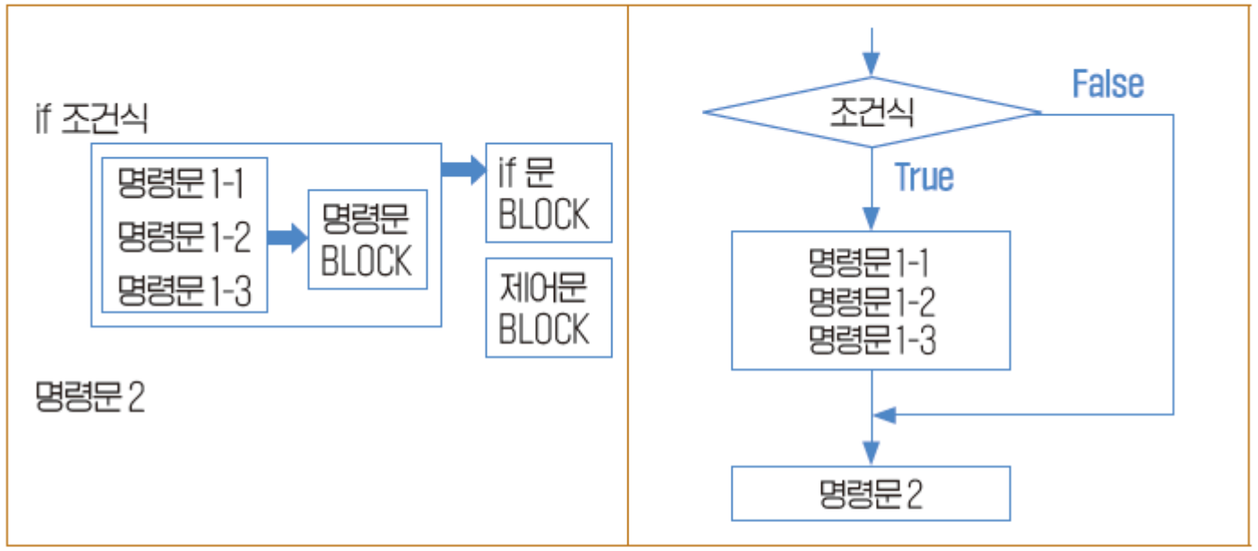
00

6. 조건문: if-elif-else문

❖ if 문

- 조건식이 True인 경우에만 실행할 명령문 BLOCK

syntax : if 문



syntax : 정확한 코딩

1. **if 조건식** : 조건식이 True이면 들여쓰기되어 있는 명령문[if 문 BLOCK]까지 실행된다.
2. 명령문2는 제어문과 상관없이 항상 실행된다.

00

6. 조건문: if-elif-else문

- if 문 코드 예시

```
na=21
if na % 2==0:
    print(na,"짝수")
print("if 문 종료 됨")
```

〈화면 출력〉

```
if 문 종료 됨
```

- ① 산술 연산자와 비교 연산자의 우선순위 : 산술 연산자 먼저 실행
- ② na % 2 산술식의 결과는 정수 1
- ③ 정수 1과 정수 0의 비교 연산 실행. 결과는 Fasle
- ④ if 조건문은 들여쓰기 된 첫 번째 print 문까지만 영향을 줌

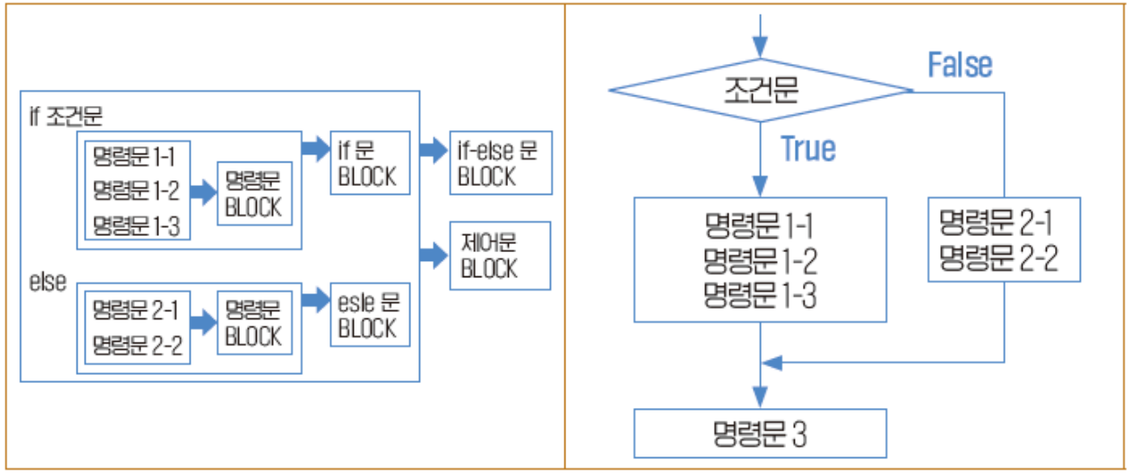
00

6. 조건문: if-elif-else문

❖ if-else 문

- if 조건이 True일 경우와 False일 경우, 각각 실행할 명령문 BLOCK을 정의하고 조건식의 결과에 따라 하나만 선택

syntax : if-else 문



syntax : 정확한 코딩

1. **if 조건문** : if 조건문이 True이면 들여쓰기가 되어 있는 명령문[if 문 BLOCK]까지 실행된다.
2. **else** : else 문은 if 조건문이 True가 아닌 그 외의 모든 경우이기 때문에 조건식을 따로 지정할 필요가 없다. else: 문 아래 들여쓰기가 되어 있는 명령문[else 문 BLOCK]까지만 실행된다. else는 if 문의 조건 이외의 모든 경우이므로 if 문이 없다면 존재할 수 없다.
3. **[if-else 문 문 BLOCK]([제어문 BLOCK])** : [if-else 문 BLOCK]에서는 [if 문 BLOCK] 이나 [else 문 BLOCK] 중 하나만 선택되어 실행된다. 즉, 하나만 선택된 후 [제어문 BOLCK]이 실행 종료되고, 명령문 3이 실행된다.

00

6. 조건문: if-elif-else문

if-else 문 코드 예시

- 앞선 if 문 코드 예제에서 na 변수 값이 짝수인 경우와 홀수인 경우를 구분하여 그 결과를 출력하도록 if-else 문으로 프로그램 수정

```
na=21
if na % 2==0:
    print(na,"짝수") # ---------> ①
else:
    print(na, "홀수") # --------> ②
print("if 문 종료됨") # ---------> ③
```

〈화면 출력〉

```
21 홀수
if 문 종료됨
```

- 산술 연산자 % : 나머지 연산
- 연산자 우선순위 : 산술 연산자 > 비교 연산자
- na % 2의 결과값 정수 1을 정수 0과 비교 연산한 결과가 False => ② 명령문 실행
- ③ 명령문은 [if 문 BLOCK]이 아니므로 if 제어문의 결과와 관계없이 항상 실행

00

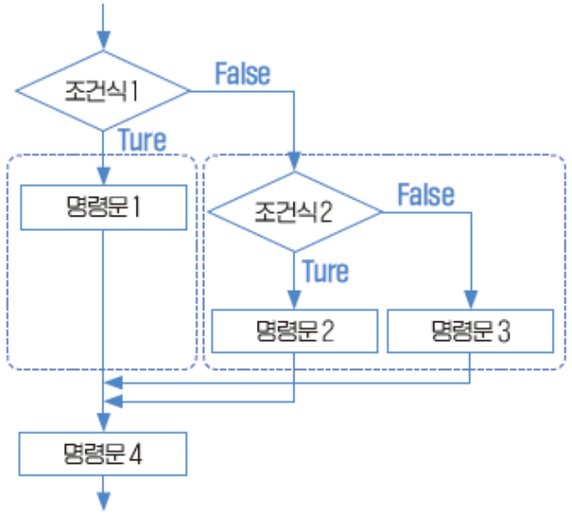
6. 조건문: if-elif-else문

❖ if-elif-else 문

- else는 그 외의 모든 경우로 이해하는 것이 중요
- 여러 가지 경우의 수를 나누어 선택해야 할 때 elif 사용

syntax : if-elif-else 문

```
[제어문 BLOCK]
[if-elif-else 문 BLOCK]
if 조건식1 :
    명령문 1   — if 문 BLOCK
elif 조건식2 :
    명령문 2   — elif 문 BLOCK
esle:
    명령문 3   — else 문 BLOCK
명령문 4
```



syntax : 정확한 코딩

- **if 조건식** : 조건식이 True인 경우만 명령문 1이 실행된다.
- **elif 조건식** : [elif]는 else if를 줄여서 표현한 것으로, 그 외의 모든 경우 중 다시 선택하는 부분으로 조건식이 반드시 필요하다. 이는 if 조건식이 False인 경우에 해당하며, 그 외의 모든 경우를 다시 조건으로 나누는 것이다. elif 문은 들여쓰기된 명령문 [elif 문 BLOCK]까지만 실행된다.
- **else** : else는 그 외의 모든 경우를 의미하기 때문에 조건식이 필요하지 않다. 위의 그림에서 선택할 수 있는 명령문은 3가지로, [제어문 BLOCK]은 이 세 가지 중 하나만 선택할 수 있다.

  if 조건식에 따라 3개의 명령문 중 하나가 선택되고, [제어문 BLOCK]의 실행이 종료되면 명령문 4가 실행된다. [제어문 BLOCK]은 선택된 조건식에 따라 실행되는 것이며, 실행이 종료되면 [제어문 BLOCK] 다음의 명령어가 실행된다. 이 점을 명확히 이해하면 제어문이 길어져도 이해하기 쉬울 것이다.

00

6. 조건문: if-elif-else문

- **if-elif-else 문 코드 예제**
  - **토익 점수를 상/중/하로 900점 이상이면 상위권, 900점 미만이고 700점 이상이면 중위권, 700점 미만이면 하위권으로 분류하는 프로그램 작성**

```
tscore=700
if tscore > 900: # ---------> ②
    print("당신의 토익 점수는", tscore, "상위권 점수입니다.")
elif tscore > 700: # ---------> ①
    print("당신의 토익 점수는", tscore, "중위권 점수입니다.")
else: # ---------> ③
    print("당신의 토익 점수는", tscore, "하위권 점수입니다.")
print("if 문 종료됨")
```

〈화면 출력〉

```
당신의 토익 점수는 700 점으로 하위권 점수입니다.
if 문 종료됨
```

- ① elif 조건식 작성시, 점수가 900점 미만이고 700점 이상 조건을 작성
- ② if tscore > 900을 만족하지 못할 경우에 아래 elif 문으로 제어가 넘어감
- ③ 700점 미만이라는 표현은 상위권, 중위권 외의 모든 경우를 의미함

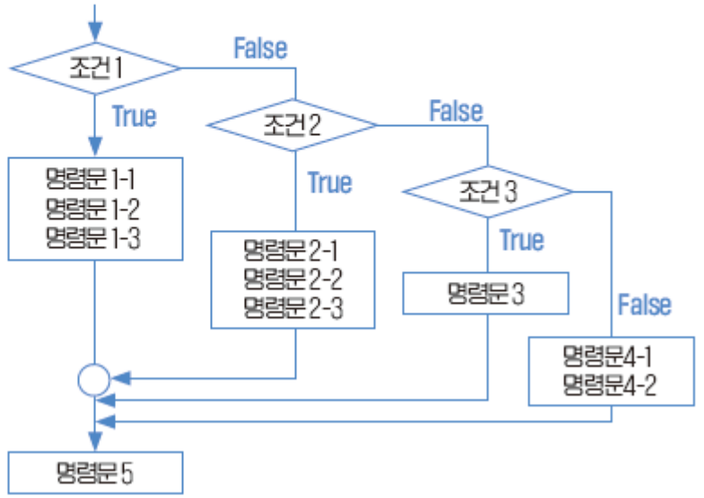
00

6. 조건문: if-elif-else문

❖ 여러 개의 elif 문

syntax : 여러 개의 elif 문

```
if 조건1 :
        명령문1-1
        명령문1-2
elif 조건2 :
        명령문2-1
        명령문2-2
        명령문2-3
elif 조건3 :
        명령문3
else:
        명령문4-1
        명령문4-2
명령문5
```



syntax : 정확한 코딩

- **elif 조건식** : [elif BLOCK]은 여러 개가 존재할 수 있는 제어문이다. 복잡해 보일 수 있지만, 기억해야 할 원칙은 하나이다. 이는 여러 경우 중 단 하나만 선택할 수 있다는 원칙으로, 어떤 부분에서 선택이 이루어지든지 상관없이 [제어문 BLOCK]을 종료하고 다음 명령문을 실행하면 된다는 것이다.

  복잡한 if 제어문 코드를 이해하려면 다음 사실을 기억하는 것도 도움이 된다. if 문은 else가 없어도 상관없지만, else 문은 if 없이는 존재할 수 없다는 것이다. 그러므로 if 문이 긴 코드인 경우, 먼저 else 문을 찾고 if 문을 찾아 전체 [제어문 BLOCK]을 확인한 다음에 하나를 선택하여 선택이 완료된 후에 [제어문 BLOCK] 다음의 명령문을 실행하면 된다.

00

6. 조건문: if-elif-else문

- 여러 개의 elif문 코드 예제
  - 번역 프로그램 코드
    - Input 함수로 문자열 입력
    - 입력 받은 문자열을 요일 문자열과 비교 연산
    - 문자열은 따옴표로 표시

```
print("월요일부터 일요일 중 영어로 번역하고 싶은 요일을 입력하세요.")
yoil=input()
if yoil=="월요일":
    print("monday")
elif yoil=="화요일":
    print("tuesday")
elif yoil=="수요일":
    print("wednesday")
elif yoil=="목요일":
    print("thursday")
elif yoil=="금요일":
    print("friday")
elif yoil=="토요일":
    print("saturday")
elif yoil=="일요일":
    print("sunday")
else:
    print("한글 요일을 잘못 입력했습니다.")
```

〈화면 출력〉

```
월요일부터 일요일 중 영어로 번역하고 싶은 요일을 입력하세요.:월요일
monday
```

00

7. 연습문제

❖ 다음 코드의 출력 결과를 적어보세요

```
01:  num1 = 55
02:  num2 = 13
03:  print(num1 <= num2)
04:  print(num1 != num2)
```

↳ _____
_____

❖ 다음 코드의 출력 결과를 적어보세요

```
01:  MVP = '하트잭'
02:  print(MVP == '공작부인')
```

↳ _____

00

7. 연습문제

❖ 그림과 같은 실행화면을 출력하도록 빈칸을 채워보세요

```
01:  switch = '켜짐'
02:  if                    :
03:      print('조명이 켜졌어요.')
04:  else:
05:      print('조명이 꺼졌어요.')
```

```
↳    조명이 켜졌어요.
```

❖ 출력 결과가 다른 하나를 골라보세요

- print(2 > 5)
- print(2 != 5)
- print(False)
- print(2 == 5)

Notice

❖ 과제

- 1. 조건식 작성하고 값 출력하기
- 2. 비교 연산자 종류 확인하고 코드 작성하기
- 3. 논리 연산자 코드 작성하기
- 4. 비교 연산자와 논리 연산자 함께 사용한 코드 작성하기
- 5. 연산자 우선순위 확인하기
- 6. if-elif-else문 다양한 구조 코드 작성하기

❖ 다음 수업 내용

- 리스트와 딕셔너리
  - 리스트(생성/값변경/값추가), 튜플(생성 및 특성), 딕셔너리(데이터 추가)