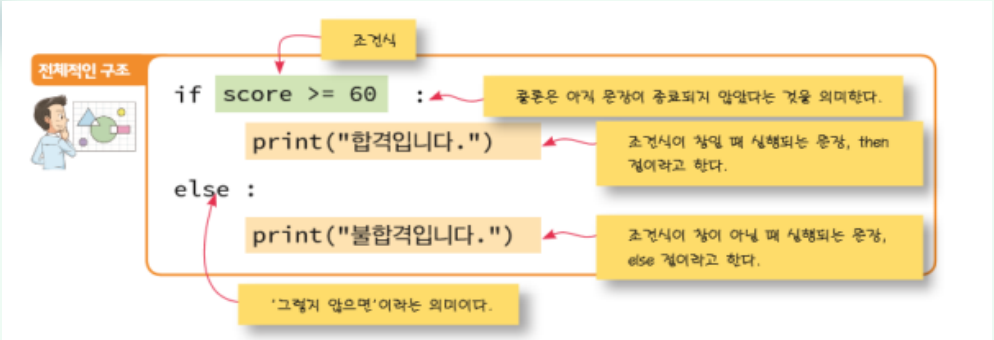
제4장
조건문

1. 조건문의 개요

1) 조건문의 구조

- 우리가 문제를 해결할 때 어떤 조건에 따라서 두 개 또는 여러 개의 실행 경로 가운데 하나를 선택해야 하는 경우가 종종 있다.예를 들면 아래 그림과 같다.

- 프로그램도 외부에서 들어오는 정보에 따라서 많은 선택을 하게 된다. 이런 식으로 조건에 따라 결정을 내리는 문장을 조건문이라고 한다. if – else 문은 조건에 따라서 2개 중에서 하나를 선택해야 하는 경우에 사용되는 문장이다. 예를 들어 성적이 60점 이상이면 합격, 60점 미만이면 불합격으로 처리해야 한다고 하면 코드는 아래와 같을 것이다.



if – else문장은 조건이 참이면 이것을 실행하고 조건이 참이 아니라 거짓이면 저것을 실행해라고 말하는 것과 동일하다. if – else문에서는 조건을 수식으로 표현하는데 이것을 '조건식'이라고한다.하여 조건식의 결과는 참이나 거짓으로 무조건 나오는 것이다.

1. 조건문의 개요

1) 조건문의 구조

- 조건식 뒤에는 콜론(:)이 있다. 콜론(:)은 파이썬 인터프리터에게 “아직 전체 문장이 끝나지 않았으니 잠시 해석을 미뤄달라” 고 요청하는 기호라고 생각하자. if – else문은 주어진 조건식을 계산하여 조건식이 참(true)으로 계산되면 if 아래에 있는 문장을 실행하고 거짓이면 else 아래의 문장을 실행한다.

```
age = 19
if(age >= 19) :
        print(“마트에서 주류를 구입할 수 있습니다.“)
else :
        print(“성인이 되지 않았습니다.다음에 이용해주세요“)

결과 : 마트에서 주류를 구입할 수 있습니다.
```

- 하지만, if – else 구문은 딱 “모 아니면 도“라는 형태이기 때문에 50%의 확률을 지닐 수가 있게 된다.
- 아울러 때에 따라 else문은 생략이 가능하다.

```
like_number = 7
if like_number == 7 :
        print(“내가 제일 좋아하는 숫자는 “, like_number, “입니다”)

결과 : 내가 제일 좋아하는 숫자는 7입니다
```

1. 조건문의 개요

2) 비교(관계) 연산자

- 비교(관계)연산자는 두 개의 피연산자를 비교하는데 사용된다. 예를 들면 "변수 x가 변수 y보다 큰지"를 따지는 데 사용된다. 관계 연산자의 결과는 참(True)아니면 거짓(False)으로 계산된다.
- 비교 연산자의 종류는 아래와 같다.

연산	의미
x == y	x와 y가 같은가?
x != y	x와 y가 다른가?
x > y	x가 y보다 큰가?
x < y	x가 y보다 작은가?
x >= y	x가 y보다 크거나 같은가?
x <= y	x가 y보다 작거나 같은가?

- 관계 수식은 참이나 거짓이라는 값을 생성한다. (100 > 50)이라는 관계식이 있다면 이것은 참(True)를 생성한다. 하지만 반대로 (100 < 50)이라면 거짓(False)를 생성한다.

```
>>> 2 == 2
True
>>> 2 == 3
False
>>> 2 == “2”
False
>>> “hello” == “hello”
True
```

1. 조건문의 개요

2) 관계 연산자

- 조건식에는 변수를 사용할 수 있다. 즉 (x > y)와 같은 수식이 가능하다는 것이다. 변수 x가 변수 y보다 크면 이 수식의 값은 1이 된다. 반대로 변수 x의 값이 y보다 작다면 0이 된다. 통상 프로그래밍 언어에서는 0은 False를 의미하면 0이외의 양수는 true로 본다. 하여 양수의 대표값인 1이 true가 되는 것이다.

```
>>> age = 20
>>> age >= 10
True
```

- 아래 코드를 실행하면 아무런 출력이 되질 않는다. 왜 그럴까? 그건 조건문에서 age가 20살 보다 커야만 참이 되어 출력문이 실행되는 것이다. 하지만 age가 정확히 20이기 때문에 출력 문장을 실행하지 않는 것이다.

```
>>> age = 20
>>> if age > 20 :
        print(“20대시군요!”)
```

위와 같을 때는 연산자를 > 대신 >=로 교체하면 출력 문장이 실행될 것이다.

1. 조건문의 개요

3) 부울 변수

- 조건식에의 결과는 항상 True(참) 아니면 False(거짓)이다. 파이썬에서 참과 거짓을 저장하는 변수를 만들 수 있는데 이러한 변수를 부울(bool)변수라고 칭한다. 정수, 실수, 문자열 변수는 많은 값을 가지지만 부울 변수는 딱 2가지 True 또는 False라는 값만을 가진다. 이 부울 변수는 프로그래밍에서 플래그(flag)변수로 많이 사용된다.

```
>>> flag = True
>>> flag
True
```

- 다른 변수처럼 부울값은 수식에서 사용될 수 있으며 변수 안에 저장될 수 있다. 파이썬에서는 참을 표현할 때, 소문자로 시작하는 true가 아니고 대문자로 시작하는 True를 사용한다라는 점은 타 언어와의 차이점이 있다.

```
>>> flag = False
>>> flag
False
```

*참고 : 부울형
“부울형”이란 자료형은 19세기 영국의 수학자였던 George Boole을 기리기 위하여 만들어졌다. 부울은 0과 1(또는 참과 거짓)만을 사용하는 논리 시스템을 고안하였다. 이것이 결국은 현재적인 PC의 초석이 되었다.컴퓨터는 결정을 내리기 위해서는 부울형의 수식을 사용한다. 즉 실행 경로 중에서 하나를 선택할 때는 부울형의 값에 따르는 것이다.

2. 순서도

1) 순서도

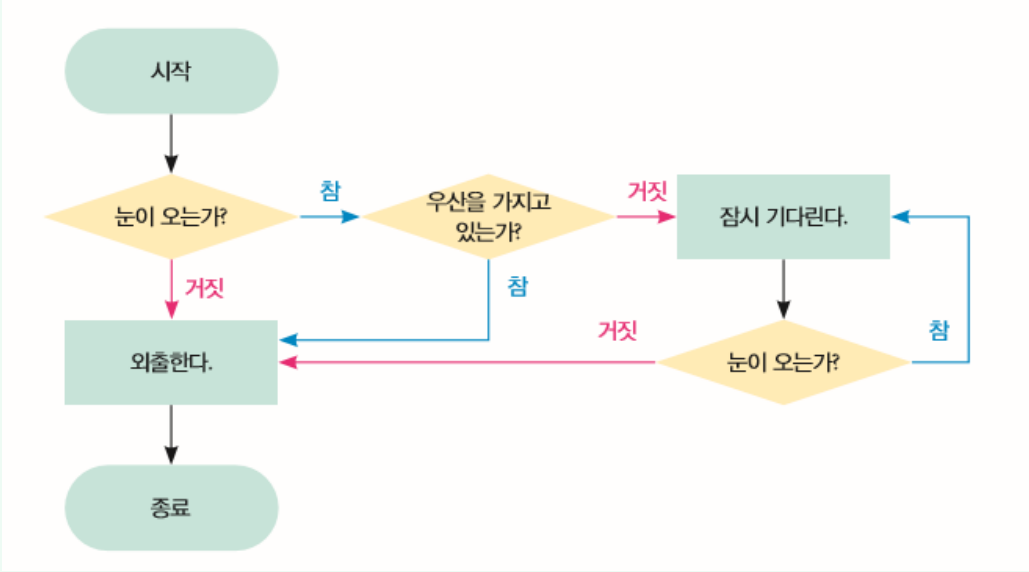
- 순서도는 프로그램을 작성하기 전에 그림으로 한번 그려보는 것을 말한다. 물론 현재의 객체지향 프로그램에 있어서는 좀 퇴색이 되는 경향이 없지 않아 있지만 프로그램의 초창기 절차지향적 프로그래밍을 할 때는 많이 사용되었다. 하지만 아직까지도 컴퓨터 구조론에 나오고 자격증 시험에도 나오기 때문에 알아두면 좋다.

- 순서도의 기호는 아래와 같다.

기호	설명
	시작/종료
	도형들을 연결한다.
	입력과 출력
	처리
	판단

* 문제

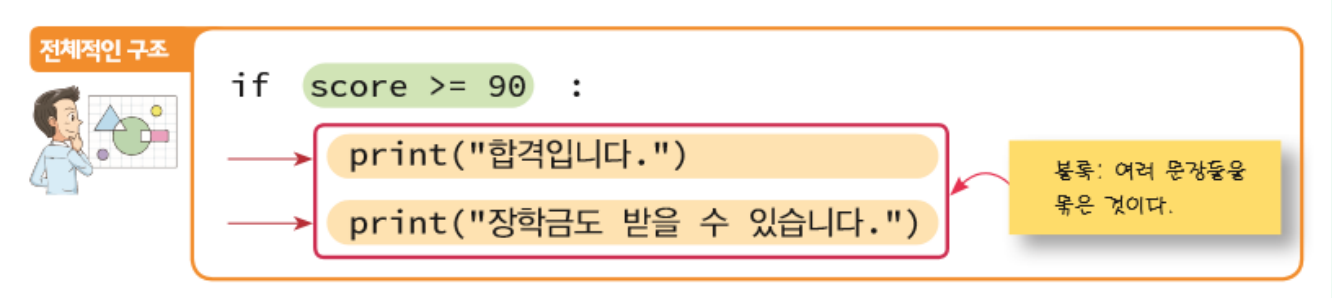
비가 올 때 어떻게 대처할 것인지를 순서도로 그려보라. 비가 오지 않으면 외출한다. 비가 오면 우산을 가지고 있는지 검사한다. 우산을 가지고 있다면 외출한다. 우산을 가지고 있지 않다면 무한정 비가 그칠 때까지 기다린다.



3. 블록

1) 블록(block)을 만드는 방법

- 만약 조건이 참인 경우에 여러 개의 문장이 실행되어야 한다면 어떻게 해야 하는지에 대해서 알아보도록 하자. 예를 들어 성적이 90점 이상이면 합격과 동시에 장학금도 받을 수 있다고 출력하려면 이런 경우는 2개의 출력 함수를 이용해서 2줄이 작성되어야 할 것이다. 이런 경우는 아래의 형태로 프로그래밍을 하면 될 것이다.



- 위의 코드는 score의 값이 90 이상이라면 print()함수를 호출하는 2개의 문장이 실행된다. 단 주의해야 할 것은 if 문 아래 코드는 동일한 개수의 공백을 가지고 있어야 한다는 것이다. 이들 모두 동일한 블록(block)에 속해 있는 것이다. 하나의 블록에 속하는 문장들은 모두 같이 실행된다. 파이참은 알아서 블록 형태를 자동으로 만들어 주기 때문에 특별히 신경 쓸 필요는 없다.하지만 알고는 있어야 한다는 것이다.

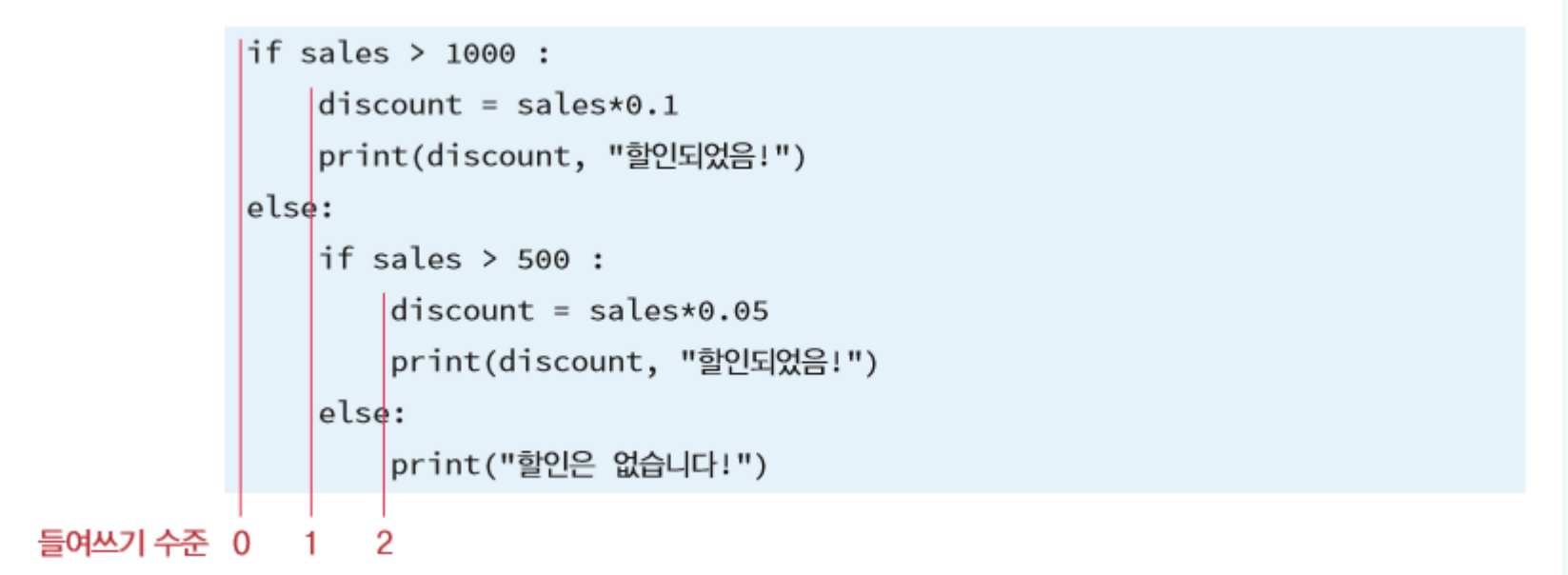
```
if score >= 90:
        print(“합격입니다.)
  print(“장학금도 받을 수 있습니다“)
```

결과 -> IndentationError: unindent does not match any outer indentation level

3. 블록

2) 블록(block)의 들여쓰기

- 블록 안에 다시 새로운 블록을 만드는 것도 가능하다. 아래 그림을 참조하여 블록을 이해하도록 하자.



- 소스의 가독성을 높이려면 들여쓰기를 할 때 일관된 방법을 사용하는 것이 좋다. 즉 TAB 키를 사용하였다면 프로그램의 나머지 부분에서 TAB키를 사용하여서 블록의 공간을 똑같이 해주는 것이 편리하다는 것이다.

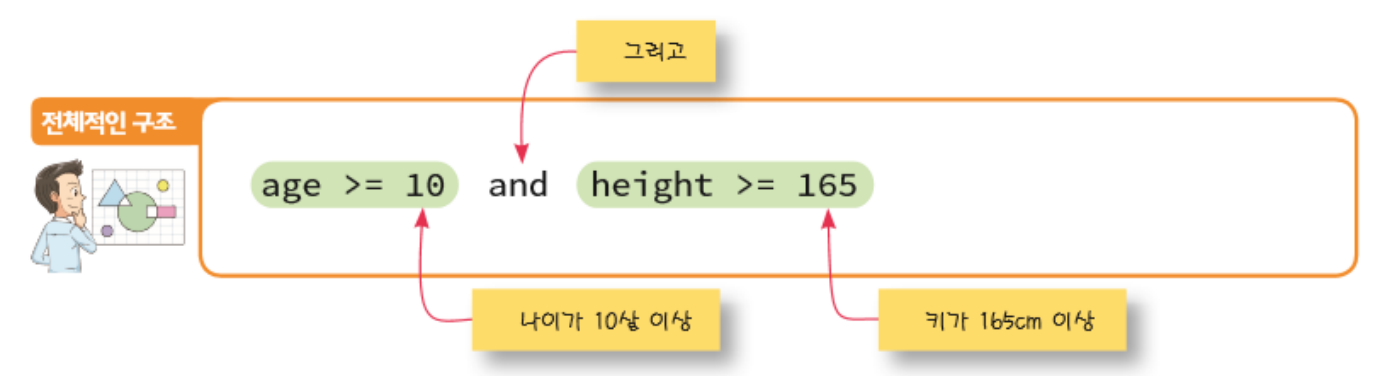
4. 논리 연산자

1) 논리 연산자의 개요

- 논리 연산자(logical operator)는 여러 개의 조건을 조합하여 참인지 거짓인지를 따질 때 사용한다.

  AND(논리곱), OR(논리합), NOT(논리부정)이 있다.

-놀이공원에서 놀이기구를 탈 수 있는 조건을 논리 수식으로 작성하여 보면 다음과 같다.



- 논리 연산자는 아래의 표와 같다.

연산	의미
x and y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
x or y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
not x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

4. 논리 연산자

2) 논리 연산자의 예시

```
>>> age = 20
>>> height  = 180
>>> if( (age>=10) and (height>=165)) :
        print("놀이 기구를 탈 수 있습니다.")
else :
        print("놀이 기구를 탈 수 없습니다.")

결과 -> 놀이 기구를 탈 수 있습니다.
```

- 주의해야 될 점이 있는데, 논리 연산자를 작성할 때 and연산자의 경우, 여러 개의 조건 중에서 처음 조건이 거짓이라면 다른 조건들은 전혀 검사조차 하지 않는다 라는 점이다. 그 이유는 첫 번째 조건이 어차피 거짓이니깐 나머지 조건들을 계산하지 않아도 전체 수식은 거짓이 되기 때문이다. 예를 들면 아래와 같다.

  (2 > 22) and (x < 5) 라면 벌써 첫 번째 조건이 거짓이므로 (x < 5)라는 조건식은 검사를 안한다 는 것이다.

  이런 계산을 단축 계산이라고도 한다.

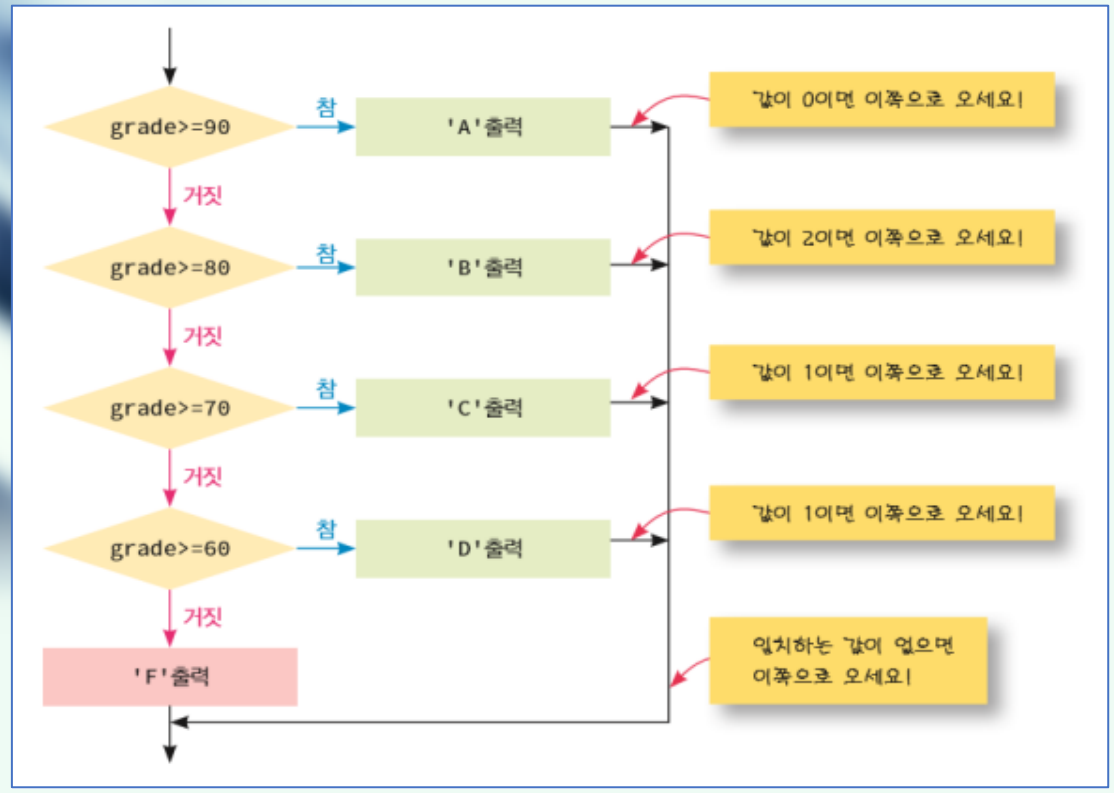
3) 논리 부정 연산자

- 논리 부정을 나타내는 not 연산자는 조건이 참이면 전체 수식을 거짓으로 만들고, 조건이 거짓이면 전체 수식 값을 참으로 만들어준다. 예를 들면 not(1==0)는 참이 된다. (1==0)는 원래 거짓이지만 not가 참으로 만들어 주는 것이다.

5. if ~ elif 문

1) if ~ elif 문의 개요

- 종종 우리는 조건에 따라서 다중으로 분기되는 결정을 내려야 하는 경우가 있다.
- 예를 들어 학생들의 성적을 받아서 학점을 출력하는 프로그램을 작성하여 실행하여 보자. 성적이 90점 이상이면 A학점, 80점 이상이고 90점 미만이면 B학점, 70점 이상이고 80점 미만이면 C학점과 같이 결정하는 것이다.




```
score = int(input("성적을 입력하시오: "))

if score >= 90 :
        print("학점 A")
elif score >= 80 :
        print("학점 B")
elif score >= 70 :
        print("학점 C")
elif score >= 60 :
        print("학점 D")
else :
        print("학점 F");
```

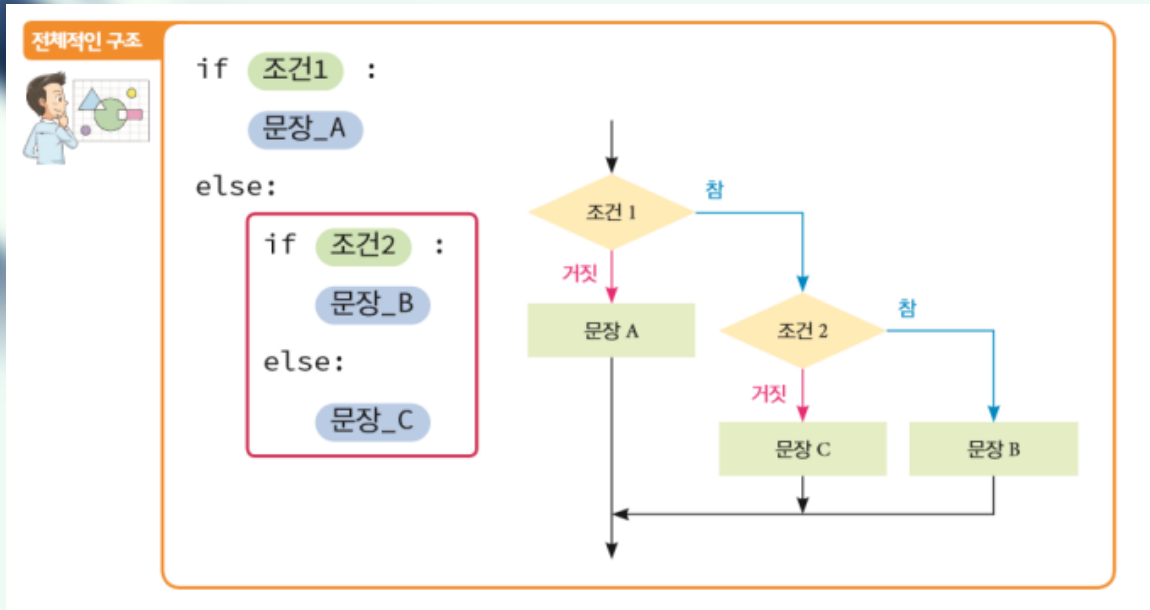
```
성적을 입력하시오: 90
학점 A
```

기억해야 할 점은 if ~ elif 구문에서는 다중 조건 중 하나만 만족하게 된다면 그 이후는 실행되지 않는다 라는 점이다.

6. 중첩 if ~ else 문

1) 중첩 if ~ else 문의 개요

- 지금까지는 if~else 문장만을 사용하였다. 하지만 만약 조건문 안에 또 다른 조건문을 넣어야 한다면 이제 중첩 if~else 구문을 사용해야 한다.
- if~else 구문 안에는 여러 개(무제한)의 if~else 구문이 포함될 수 있다. 들여쓰기로 중첩의 수준을 알 수 있다.
- 이러한 구조는 자칫하면 프로그래밍을 한 자신도 못 알아볼 수 있으니, 많이 사용하지 않는 것이 바람직하며, 통상 2개의 if~else 구문 정도로만 해도 거의 다 문제는 해결된다. 물론 조건을 어떻게 생각하느냐에 달려 있는 것이다.



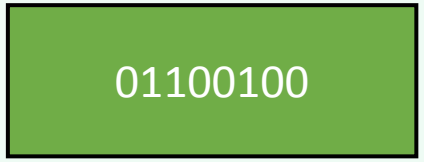

```
appleQuality = input("사과의 상태를  입력하시오: ")
applePrice = int(input("사과 1개의 가격을 입력하시오: "))
if appleQuality == "신선":
        if applePrice < 1000:
                print("10개를 산다")
        else:
                print("5개를 산다")
else:
         print("사과를 사지 않는다.")
```

7. 문자열과 숫자의 저장 구조

1) 문자열과 숫자 저장 구조

- 숫자 100과 문자열 "100"은 어떤 차이가 있을까? 유사해 보이지만 컴퓨터에서는 이것을 상당히 다르게 처리한다.
  우리가 조건식을 만들 때도 문자열과 숫자를 구별하여야 한다.
- 앞서 이미 위의 내용에 대해서 학습을 하였고 실습도 진행한 바 저장 구조만 설명하겠다.

[문자열 "100"의 저장구조]



[숫자 100의 저장구조]

- if문에서의 조건식에서도 위의 개념이 그대로 적용된다는 사실을 필히 기억하도록 하자.

```
>>>s = "100"
>>>number = int(s)
>>>if number==100:
        print("숫자 100입니다. ")

결과 -> 숫자 100입니다.
```

감사합니다.