**1. 조건문**

* **특정 상황에 따라 프로그램의 행동을 바꿀 필요가 있음**
* **if()문**
* **조건**을 만족하면 if문의 **중괄호 안의 코드를 실행**
* 조건식을 참이나 거짓을 반환해야 함
* 이 조건식을 불리언 표현식(Bollean Expression)이라고 말함
* **표현식**
* ex)  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* 하나 이상의 피연산자와 0개 이상의 연산자의 조합
* 어떤 특정 값이나, 개체, 메소드 혹은 네임스페이스로 평가 가능
* **구문과 표현식의 관계**
* **표현식**은 평가(evaluate)되며, 값을 반환함(void 반환도 포함)  
  - 그러나 일부 표현식은 단독으로 사용할 수 없음  
  - ex)  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* 구문은 한 줄 이상의 코드로 실행되는 집합  
  - 세미콜론으로 종결되는 문장이나 중괄호로 묶인 코드 블록  
  - 실행의 최소 단위라고도 말할 수 있음  
  - 구문 안에는 한 개 이상의 표현식이 포함될 수 있음  
  - ex)  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* **불리언 표현식**
* 불리언 표현식을 실행하면 참 또는 거짓의 결과가 도출됨  
  - 아래와 같은 표현식은 당연히 true나 false가 도출되지 않음  
  텍스트이(가) 표시된 사진

  자동 생성된 설명

**2. 관계 연산자, 비교 연산자 (Relational Operator)**테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

**3. if/else 문과 코딩 표준**

* **else 문**
* if 문의 조건식이 참이 아닐 때 수행함
* **else if 문**
* 이전 if 문의 조건식이 참이 아니고, 다시 새로운 조건문을 수행하여 참이면 실행함
* 다음 if 문이나 else 문으로 이동함  
  - 이때 다음 if 문이라면 새로운 해당 조건식을 다시 검사하게 됨
* **if / else if 문 조건식을 만들 때 주의점**
* 조건문의 순서가 올바른지 확인할 것  
  - 논리적으로 말이 되는지 안 되는지 확인할 것
* **중괄호 생략**
* 문법상 로직 코드가 한 줄일 때 가능함
* 하지만 코드를 보고 누구나 실수(혼란)이 올 수도 있음
* 줄 바꿈으로 **범위를 제어**하려고 하는 것은 굉장히 나쁜 생각임  
  - 파이썬의 경우, 줄 바꿈을 탭이나 스페이스로 할 수 있음  
  - 이 탭과 스페이스를 섞어 쓰면 오류가 발생함  
  - 탭과 스페이스는 육안으로 구분하기 매우 힘듦
* **범위가 명백하도록 (코딩 표준: 중괄호는 항상 사용할 것)**
* 본인이 아무리 조심해도 다른 사람이 실수할 수 있음
* 중괄호를 이용해서 범위를 명확하게 보여줘야 함
* 언제나 if 문을 쓸 때마다 예외 없이 중괄호를 사용해야 함
* 중괄호가 새로운 줄에 있는 것이 더 나음  
  - 여는 중괄호가 위에 있는 방식은 구시대적 방식
* **공통된 규칙은 생산성을 높여줌**
* 모두가 공통된 규칙으로 변수명을 짓고, 중괄호 위치를 정하면 코드를 읽기 편해짐  
  - 즉, 코드 속에서 문제점을 찾기 쉬움
* 그렇기 때문에 회사마다 코딩 표준이 있음