

## 오류

- 함수나 메소드가 처리 도중 다음 명령문을 실행할 수 없는 상황
- 오류 중 처리가능한 것을 Exception(예외) 라고 한다. 그리고 그 예외를 처리하는 것을 Exception Handling 이라고 한다.

친구를 만난다()	예외 상황
1. 약속시간 1시간 전에 집에서 나온다. 2. 버스 정류장에 간다. 3. XXX번 버스를 탄다. 4. 약속장소 근처 정류장에 내린다. 5. 약속장소인 카페를 찾아 간다. 6. 친구를 만난다.	- 3 에서 버스가 오지 않는 경우 (예외) <b>처리</b> : 택시를 타고 간다. (예외 처리) - 5 에서 약속장소가 쉬는 날인 경우 (예외) <b>처리</b> : 친구에게 전화해서 장소를 변경 (예외 처리)

## 오류의 종류

- 파이썬 문법을 어겨서 발생하는 오류
  - 코드 상 100% 발생하는 오류
  - 코드를 수정해 야한다.
  - 보통 이런 오류는 컴파일 방식 언어의 경우 컴파일 때 에러를 내서 수정하도록 한다.
- 실행 환경의 문제로 발생하는 오류
  - 코드상에서는 Exception의 발생여부를 확신할 수 없다.
  - 만약 발생할 경우 어떻게 처리할지를 구현해야 한다.

In [ ]:



## Exception handling

Exception이 발생되어 프로그램이 더이상 실행될 수 없는 상황을 처리(handling)해서 정상화 시키는 작업을 말한다.

try - except 구문을 이용해 처리한다.

### try, except 구문

```
try:
    Exception 발생가능한 코드 블록
except [Exception클래스 이름 [as 변수]] :
    처리 코드
```

- try block**
  - Exception 발생 가능성 있는 코드와 그 코드와 연결된 코드들을 블록으로 묶는다.
    - 연결된 코드란 Exception이 발생 안해야만 실행되는 코드를 말한다.
- except block**
  - 발생한 Exception을 처리하는 코드 블록을 작성한다.

- try block의 코드를 실행하다 exception이 발생하면 except block이 실행된다. Exception이 발생하지 않으면 실행되지 않는다.
- try block에서 발생한 모든 Exception을 처리하는 경우 except: 로 선언한다.
- try block에서 발생한 특정 Exception만 따로 처리할 경우 except Exception클래스 이름을 선언한다.
  - 모든 Exception들은 클래스로 정의 되어 있다. 그 클래스 이름을 적어준다.
  - **Exception 들 별로 각각 처리할 수 있으면 이 경우 except 구문(처리구문)을 연속해서 작성하면 된다.**
- try block에서 발생한 특정 Exception만 따로 처리하고 그 Exception이 왜 발생했는지 등의 정보를 사용할 경우 except Exception 클래스 이름 as 변수명 으로 선언하고 변수명을 이용해 정보를 조회한다.

In [ ]:



## finally 구문

- 예외 발생여부, 처리 여부와 관계없이 무조건 실행되는 코드블록
  - try 구문에 반드시 실행되어야 하는 코드블록을 작성할때 사용한다.
  - 보통 프로그램이 외부자원과 연결해서 데이터를 주고 받는 작업을 할때 마지막 연결을 종료하는 작업을 finally 블록에 넣는다.
- finally 는 except 보다 먼저 올 수 없다.
  - 구문순서
    1. try - except - finally
    2. try - except
    3. try - finally

In [ ]:



## Exception 발생 시키기

### 사용자 정의 Exception 클래스 구현

- 파이썬은 Exception 상황을 클래스로 정의해 사용한다.
  - Exception이 발생하는 상황과 관련된 attribute들과 메소드들을 정의한 클래스
- 구현
  - Exception 클래스를 상속받는다.
  - 클래스 이름은 Exception 상황을 설명할 수 있는 이름을 준다.

In [ ]:



## Exception 발생시키기

- 함수나 메소드가 더이상 작업을 진행 할 수 없는 조건이 되면 Exception을 강제로 발생시킨다.
- **Call Stack Mechanism**
  - 발생한 Exception은 처리를 하지 않으면 caller에게 전달된다.
    - 발생한 Exception에 대한 처리가 모든 caller에서 안되면 결국 파이썬 실행환경까지 전달되어 프로그램은 비정상적으로 종료 되게 된다.

In [ ]:



## raise 구문

- Exception을 강제로 발생시킨다.
  - 업무 규칙을 어겼거나 다음 명령문을 실행할 수 없는 조건이 되면 진행을 멈추고 caller로 요청에게 작업을 처리 못했음을 알리며 돌아가도록 할때 exception을 발생시킨다.
  - 구문

```
raise Exception객체
```

- **raise와 return**
  - 함수나 메소드에서 return과 raise 구문이 실행되면 모두 caller로 돌아간다.
  - return은 정상적으로 끝나서 돌아가는 의미이다. 그래서 처리결과가 있으면 그 값을 가지고 돌아간다.
    - caller는 그 다음작업을 이어서 하면 된다.
  - raise는 실행도중 문제(Exception)가 생겨 비정상적으로 끝나서 돌아가는 의미이다. 그래서 비정상적인 상황 정보를 가지는 Exception객체를 반환값으로 가지고 돌아간다.
    - caller는 try - except구문으로 발생한 exception을 처리하여 프로그램을 정상화 하거나 자신도 caller에게 exception을 발생시키는 처리를 한다.

In [ ]:

