

18. 예외 처리

예외 처리

- Try , Catch

- . 프로그램 을 실행할때 개발자가 미연에 알아차리지 못한 오류 나 예외 상황을 실행 도중 인지 하여 별도의 로직으로 처리 하도록 유도하는 기능
- . 사용자의 실수 또는 프로그램의 오류 로 인한 시스템 의 오작동 을 방지 하고 오류 내용을 완화하여 사용자에게 내용을 전달 함으로서 프로그램에 대한 신뢰도를 높일수 있다
- . 개발자 는 오류 내역이 발생한 부분을 확인 하고 오류를 수정 하거나 사용자가 예외 상황을 발생하지 못하도록 추가 로 로직을 보완 할 수 있다.

```
1 while (True):
2     str = input('점수 를 입력하세요 : ')
3     try :
4         score = int(str)
5         print('입력한 점수 : ', score)
6         break
7     except :
8         print('점수를 올바르게 입력 하지 않았습니다.')
9
```

Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/17.Exception[예외처리].py
점수 를 입력하세요 : 50
입력한 점수 : 50

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/17.Exception[예외처리].py
점수 를 입력하세요 : 오십
점수를 올바르게 입력 하지 않았습니다.
점수 를 입력하세요 : 오십
점수를 올바르게 입력 하지 않았습니다.
점수 를 입력하세요 : 50
입력한 점수 : 50

>3행 : 예외 상황 발생 검출 시작
>4행 : 입력 값을 숫자로 형변환

>7행 : 예외상황이 발생 시 처리할 로직

> 입력한 값을 숫자로 형변환 시 숫자로 바꿀수 없는 데이터 인경우 오류가 발생한다. 이때는 7행의 except : 의 로직을 수행하게 된다.

. Except 의 종류 와 사용법

> try 가 검출 하 는 예외 구문 중 사용 빈도 수가 많은 예외 상황

연산	설명
NameError	명칭이 발견되지 않는다. 초기화 하지 않은 변수 를 사용할 때 발생
ValueError	타입은 맞지만 값의 형식 이 잘못 된 경우 발생
ZeroDivisionError	0 으로 나누기 오류
TypeError	데이터 형식을 잘못 입력 하였을 경우 (문자 -> 숫자)
IndexError	리스트 등의 범위 밖의 index 를 지정 하였을 경우

```
Chap01_intro > 17.Exception[예외처리].py > ...
1 while (True):
2     str = input('점수 를 입력하세요 : ')
3     try :
4         score = int(str)
5         print('입력한 점수 : ', score)
6         break
7     except ValueError:
8         print('값의 형식을 잘못 입력 하였습니다..')
9     except NameError:
10        print('할당 되지 않는 변수를 사용하였습니다..')
11    except TypeError:
12        print('데이터 타입 형 변환에 실패 하였습니다..')
13
```

Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/chap01_intro/17.Exception[예외처리].py
점수 를 입력하세요 : 오십
값의 형식을 잘못 입력 하였습니다..
점수 를 입력하세요 :
점수 를 입력하세요 : 50
입력한 점수 : 50

* Except 구문에 검출된 예외 상황을 나열하지 않아도 예외상황을 검출 할 수 있다
하지만 예외 상황을 미리 예측하여 각 상황 별로 분기 를 만들어 둘 경우 예외 상황 별로 더 세밀하게 프로그램을 관리 할 수 있다.

- rise

. 예외 상황을 발생시키는 구문

```
Chap01_intro > 17.Exception[예외처리].py > ...
1 while (True):
2     str = input('점수 를 입력하세요 : ')
3     try :
4         score = int(str)
5         if score > 100 :
6             raise Exception('100 보다 큰 수 는 입력할 수 없습니다.')
7         print('입력한 점수 : ', score)
8         break
9     except ValueError:
10        print('숫자만 입력 하세요')
11    except Exception as ex:
12        print(ex)
13
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/17.Exceptio
점수 를 입력하세요 : 오십
숫자만 입력 하세요
점수 를 입력하세요 : 120
100 보다 큰 수 는 입력할 수 없습니다.
점수 를 입력하세요 : █
```

- >5행 : 문자를 숫자로 형변환 하는 구문에서 형변환 오류 가 검출 되지 않으면 100 이상의 숫자를 입력 하였는지 확인한다
- >6행 : 100 이상의 수를 입력 하면 예외 상황을 발생시키는데 이때 Exception 형식의 예외를 강제로 일으켜 11 행으로 이동한다.
- >11행 : Exception 예외는 ex 라는 이름으로 사용한다
- >12행 : 6행에서 발생시킨 예외의 메시지를 ex 에 담아 출력 한다.

- Finally

- . 예외 상황의 유 무 에 관계 없이 반드시 실행 해야 하는 로직
- . Try / catch 와 한 셋트 로 사용하며 finally 필요에 따라 사용한다.
- . 보통 프로그램이 네트워크 에 접속 후 네트워크 접속 을 종료 하고 로직을 끝낼때 사용한다.

```
Chap01_intro > 17.Exception[예외처리].py > ...
1 print("데이터 베이스에 접속하였습니다.")
2
3 str = input('점수 를 입력하세요 : ')
4 try :
5     score = int(str)
6     if score > 100 :
7         raise Exception('100 보다 큰 수 는 입력할 수 없습니다.')
8     print('입력한 점수 : ', score)
9 except :
10    print('숫자만 입력 하세요')
11 finally :
12    print("데이터 베이스 접속을 종료 합니다.")
13
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/17.Exec
데이터 베이스에 접속하였습니다.
점수 를 입력하세요 : 10
입력한 점수 : 10
데이터 베이스 접속을 종료 합니다.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/17.Exec
데이터 베이스에 접속하였습니다.
점수 를 입력하세요 : 오십
숫자만 입력 하세요
데이터 베이스 접속을 종료 합니다.
```

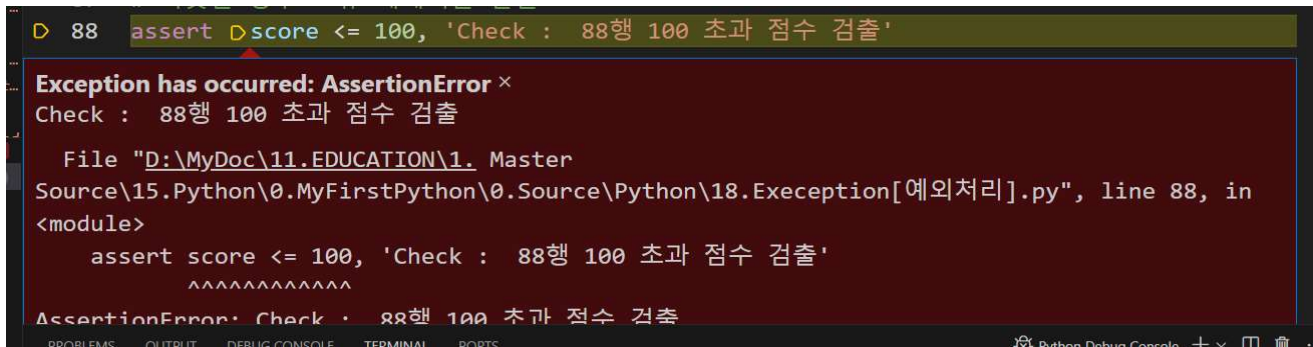
>11행 : 반드시 데이터베이스 접속을 종료함

- assert
 - . 현재 상태가 맞는지 확인 한다.
 - . 점검 할 조건 과 조건이 거짓일때 보여 줄 메시지 를 지정 할 수 있다.
 - > score 에 100 을 초과 하는 값을 입력 할 경우 개발자가 지정한 시스템 오류 메시지를 반환하고 프로그램을 종료 한다.
 - > 프로그램 개발 시 테스트 의 용도로 사용 하기 좋으나 (예외 상황의 시점을 잡아내기 좋다) 자주 사용시 속도가 저하 된다.

```
85 str = input('점수 를 입력하세요 : ')
86 score = int(str)
87 # 거짓일 경우 오류 메시지를 반환
88 assert score <= 100, 'Check : 88행 100 초과 점수 검출'
89 print('입력한 점수 : ', score)
```

> 100 초과 값을 입력하였을 경우 아래와 같은 오류를 반환한다.

* 로직의 중간에 검증 로직으로 사용하는 예



The screenshot shows a Python IDE with a dark theme. The top part shows a code editor with the following code:

```
88 assert score <= 100, 'Check : 88행 100 초과 점수 검출'
```

Below the code editor, an exception message is displayed in a red box:

```
Exception has occurred: AssertionError ×
Check : 88행 100 초과 점수 검출

File "D:\MyDoc\11.EDUCATION\1. Master
Source\15.Python\0.MyFirstPython\0.Source\Python\18.Exception[예외처리].py", line 88, in
<module>
    assert score <= 100, 'Check : 88행 100 초과 점수 검출'
          ^^^^^^^^^^^^^
AssertionError: Check : 88행 100 초과 점수 검출
```

The bottom of the screenshot shows the IDE's interface with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The DEBUG CONSOLE tab is active, showing the exception details.