파이썬 프로그래밍
강의 노트 #10

예외 처리

오류

- 프로그래밍의 오류
  - 문법 오류
  - 논리 오류
  - 실행 오류
- 문법 오류
  - 프로그래밍 언어를 잘못 사용해서 발생
  - 파이썬이 어디서 오류가 발생했는지 알려줌

오류

```
print("오류 발생 확인 프로그램")
n = int(input("정수를 입력하세요: ")
print(f"사용자가 입력한 정수: {n}")
```

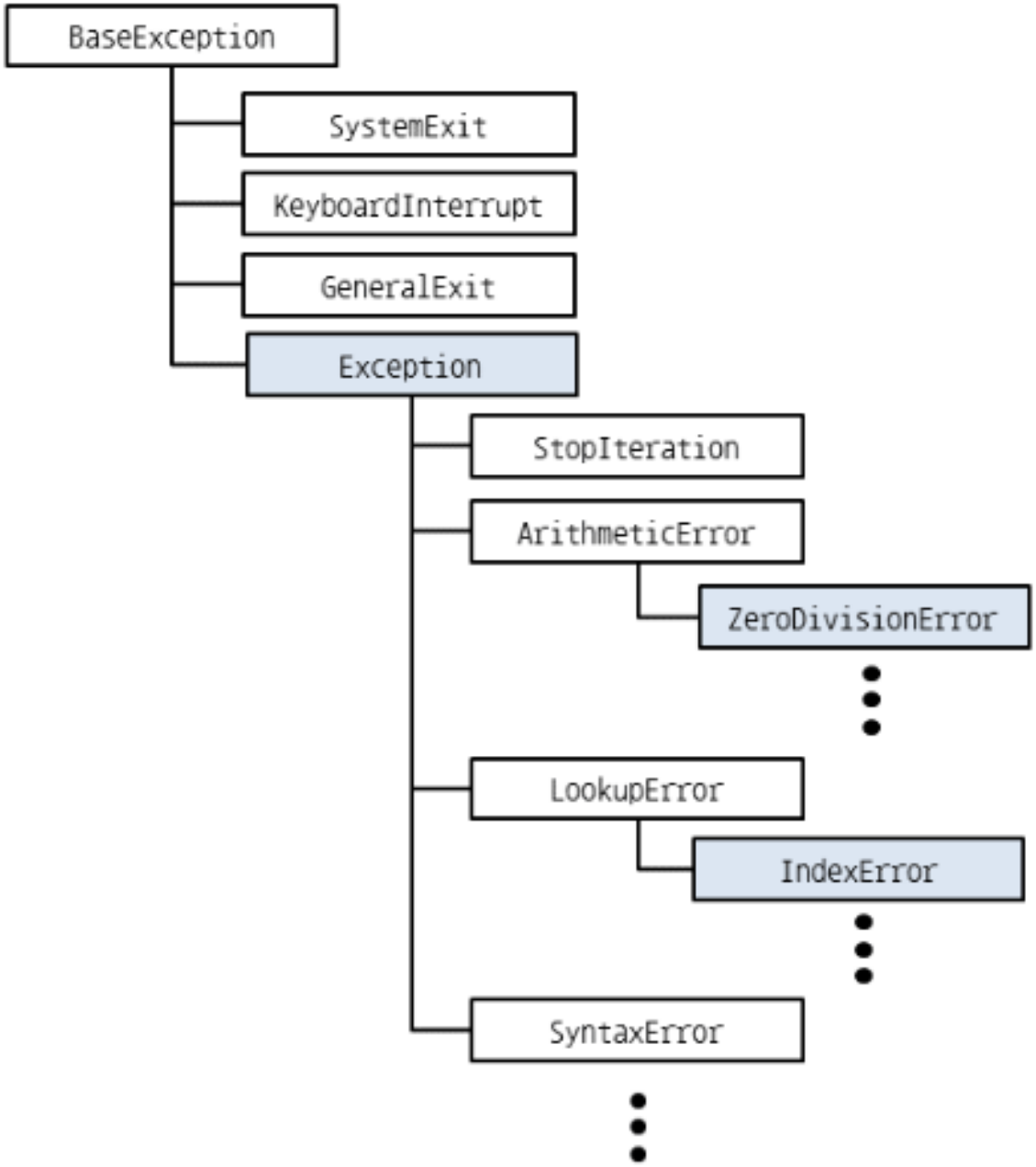
코드10-2

```
[실행 결과]
File "SyntaxError.py", line 2
 n = int(input("정수를 입력하세요: ")
         ^
SyntaxError: '(' was never closed
```

오류

- 논리 오류
  - 문제 해결 방법을 잘못 지정한 것
- 실행 오류
  - 문법적/논리적으로 문제가 없는데 오류가 발생하는 것
  - 예
    - 사용자가 정수로 변환할 수 있는 문자열을 입력해야 하는데 잘못 입력
    - 파일을 여는데 이미 삭제되었거나 존재하지 않는 파일일 수 있음
    - 파일에서 데이터를 읽거나 쓸 때, 하드디스크에 오류가 있거나 파일에 문제가 있어 동작하지 않을 수 있음

예외 계층도


BaseException
SystemExit
KeyboardInterrupt
GeneralExit
Exception
StopIteration
ArithmeticError
ZeroDivisionError
LookupError
IndexError
SyntaxError

오류

- 사용자가 정수로 변환할 수 있는 문자열을 잘못 입력한 예

```
>>> n = int(input("정수를 입력하세요: "))
정수를 입력하세요: 23.5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with
base 10: '23.5'
```

- 실행 오류가 발생하는 몇 가지 예
  - int() 명령에 실수 형태로 된 문자열을 전달
  - 문자열에서 index() 함수를 사용하는데 찾는 문자열 없음
  - open() 함수를 이용해서 파일을 열 때 파일이 없는 경우
  - 리스트에 있는 요소 개수보다 큰 범위의 요소를 접근
  - 0으로 나누는 경우

오류 대응 코드의 필요성

- 지금까지 우리는 프로그램을 만들면서 항상 오류가 발생하지 않는다고 가정했음
- 하지만, 프로그램을 사용하다 보면 예기치 못했던 오류가 발생하는 경우가 많음
- 오류를 모두 처리하는 것은 어렵지만, 프로그램 실행이 멈추는 것을 최소화
- 프로그램을 개발할 때 문제가 발생할 수 있는 부분을 생각하고 적절하게 오류에 대응하는 코드를 작성하는 방어적 코딩을 해야 함
- 파이썬에서는 예외처리 기능을 이용해서 다양한 오류 상황을 처리할 수 있도록 지원

예외처리 기법

- 예외(exception)은 프로그램을 실행하면서 발생할 수 있는 오류를 통칭
- 파이썬에서는 try: … except: 구문을 이용해서 예외처리를 지원
- 아래는 기본적인 사용 방법을 보여줌

```
try:
    오류를 발생시킬 수 있는 파이썬 코드
except [오류 종류 [as 오류 변수]]:
    오류가 발생했을 때 실행시킬 코드
[except [오류 종류 [as 오류 변수]]:
    또 다른 종류의 오류가 발생했을 때 실행시킬 코드
…   # 또 다른 except 구문
```

예외처리 기법

- []에 있는 내용은 생략 가능
- try구문은 최소한 한 개 이상의 except 구문과 연동되어야 함
- 코드 10-2에 예외처리 추가

```
print("예외 처리 확인 프로그램")
try:
    s = input("정수를 입력하세요: ")
    n = int(s)                                  # (1)
    print(f"사용자가 입력한 정수: {n}")          # (2)
except:
    print(f"정수로 변환할 수 없습니다. 입력값: {s}") # (3)
print("프로그램 종료")                           # (4)
```

예외처리 기법

- 예외가 발생했을 때의 실행 순서

```
print("예외 처리 확인 프로그램")
try:
    s = input("정수를 입력하세요: ")
    n = int(s)                                                    # (1)
    print(f"사용자가 입력한 정수: {n}")                              # (2)
except:
    print(f"정수로 변환할 수 없는 값을 입력했습니다. 입력값: {s}") # (3)
print("프로그램 종료")                                            # (4)
```

① 23.5

② 예외 발생

③

④

- 예외가 발생하지 않을 때는?

여러 가지 오류가 발생하는 코드

- 여러 가지 오류가 발생할 수 있음

```
lst = [1, 2, 3]
idx = int(input("변경할 요소의 위치(인덱스)를
입력하세요: "))
n = int(input("새로운 값을 입력하세요: "))     # (1)
lst[idx] = n # idx 위치의 요소 값을 n으로 변경 # (2)
print(lst)
```

  - (1)에서 정수가 아닌 문자열이 입력될 때 ValueError
  - (2)에서 idx가 3 이상일 때 IndexError

여러 오류에 대응하는 예외 처리 코드 작성

- 여러 가지 오류가 발생할 수 있는 경우, 예외 처리 방법
  - 모든 오류에 대해서 한꺼번에 대응
  - 오류마다 각각 대응
  - 여러 가지 오류 중 일부만 대응하고, 나머지는 무시하거나 한꺼번에 대응

여러 오류에 대응하는 예외 처리 코드 작성

- 모든 오류를 함께 처리
  - except를 오류 정보 없이 사용

```
try:
    lst = [1, 2, 3]
    idx = int(input("변경할 요소의 위치(인덱스)를 입력하세요: "))
    n = int(input("새로운 값을 입력하세요: "))
    lst[idx] = n # idx 위치의 요소 값을 n으로 변경
    print(lst)
except: # 모든 오류를 한 곳에서 대응
    print("오류: 인덱스 범위를 벗어났거나 정수가 아닌 문자열을 입력했음")
```

여러 오류에 대응하는 예외 처리 코드 작성

- 각 오류를 따로 처리
  - except에 각 오류 정보를 명시

```
try:
    lst = [1, 2, 3]
    idx = int(input("변경할 요소의 위치(인덱스)를
입력하세요: "))
    n = int(input("새로운 값을 입력하세요: "))
    lst[idx] = n # idx 위치의 요소 값을 n으로 변경
    print(lst)
except ValueError: # ValueError에 대응
    print("오류: 정수가 아닌 문자열을 입력했음")
except IndexError: # IndexError에 대응
    print("오류: 인덱스 범위를 벗어남")
```

여러 오류에 대응하는 예외 처리 코드 작성

- 일부 오류만 대응하고 나머지는 무시

```
try:
    lst = [1, 2, 3]
    idx = int(input("변경할 요소의 위치(인덱스)를
입력하세요: "))
    n = int(input("새로운 값을 입력하세요: "))
    lst[idx] = n # idx 위치의 요소 값을 n으로 변경
    print(lst)
except ValueError: # ValueError에 대응
    print("오류: 정수가 아닌 문자열을 입력했음")
```

여러 오류에 대응하는 예외 처리 코드 작성

- 일부 오류만 대응하고 나머지는 한꺼번에 처리

```
try:
    lst = [1, 2, 3]
    idx = int(input("변경할 요소의 위치(인덱스)를
입력하세요: "))
    n = int(input("새로운 값을 입력하세요: "))
    lst[idx] = n # idx 위치의 요소 값을 n으로 변경
    print(lst)
except ValueError: # ValueError에 대응
    print("오류: 정수가 아닌 문자열을 입력했음")
except: # 나머지 (IndexError, NameError)에 대응
    print("오류 발생")
```

오류의 예

오류 종류	설명
ValueError	인자로 다른 자료형의 값을 전달
IndexError	자료구조에서 인덱스의 범위를 벗어남
NameError	값이 저장되지 않은 변수가 사용
UnicodeDecodeError	파일이 저장된 인코딩 방식과 지정된 인코딩 방식이 다름
TypeError	허용되지 않은 자료형 사용
FileNotFoundError	파일이 없음
ZeroDivisionError	0으로 나눔

UnicodeDecodeError

- data.utf8.txt파일이 utf-8 방식으로 한글 문자열을 저장했다고 가정할 때, cp949 형식으로 열고 읽으면 UnicodeDecodeError 발생

```
>>> f = open("data.utf8.txt")
>>> f.read()
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'cp949' codec can't decode byte
0xed in position 0: illegal
multibyte sequence
```

FileNotFoundError

- 파일 작업할 때 파일이 없으면 발생

```
filename = "C:\\temp\\noname.txt"
f = open(filename)
f.close()
```

- C:₩temp₩noname.txt가 없다고 가정할 때 실행 결과

```
Traceback (most recent call last):
File "exception8.py", line 2, in <module>
f = open(filename)
FileNotFoundError: [Errno 2] No such file or
directory: 'C:\\temp\\noname.txt'
```

스크립트 실행 중에 종료하기

- sys모듈의 exit함수를 이용하면 스크립트 실행 중에 종료하기

```
import sys
try:
    fname = "c:\\temp\\a.ini"
    f = open(fname)
    lines = f.readlines()
    f.close()
except FileNotFoundError: # FileNotFoundError 발생
    print("Could not open " + fname)
    sys.exit()
except:   # 다른 오류 일괄 처리
    print("Other error occurred")
print("End of the program")
```

실습문제 1

- 문제
  - 사용자로부터 두 수를 입력받아 나눗셈을 수행하고 결과를 출력하는 프로그램을 작성하세요. 0으로 나누는 오류를 처리하세요.

- 요구사항
  - ZeroDivisionError 예외처리를 사용

실습문제 1

- 최종 코드

```
def basic_exception_handling():
    try:
        a = int(input("분자를 입력하세요: "))
        b = int(input("분모를 입력하세요: "))
        result = a / b
        print("결과:", result)
    except ZeroDivisionError:
        print("오류: 0으로 나눌 수 없습니다.")

# 함수 호출
basic_exception_handling()
```

예외 발생시키기

- raise 예외(“에러메시지”)

```
try:
      x = int(input("3의 배수 입력:"))
      if x % 3 != 0:
            raise Exception("3의 배수아님")
      print(x)
except Exception as e:
    print("에러 발생  : ",e)
print("End of the program")
```

예외 발생시키기

- raise 예외("에러메시지")
- except 안에서 raise를 사용하면 현재 예외를 다시 발생시킴(re-raise)

```
def three_div():
    try:
        x = int(input("3의 배수 입력:"))
        if x % 3 != 0:
            raise Exception("3의 배수아님")
        print(x)
    except Exception as e:
        print("함수 에러 발생  : ",e)
        raise
try:
       three_div()
except Exception as e :
       print("코드 에러 발생  : ",e)
```

실습문제 2

- 문제
  - 사용자로부터 숫자를 입력받아 정수로 변환한 뒤, 역수를 출력하는 프로그램을 작성하세요.

- 요구사항
  - 입력 오류와 0으로 나누는 오류를 처리하세요.
  - ValueError
  - ZeroDivisionError

실습문제 2

- 최종 코드

```
def multiple_exception_handling():
    try:
        num = int(input("숫자를 입력하세요: "))
        inverse = 1 / num
        print("역수:", inverse)
    except ValueError:
        print("오류: 유효한 숫자가 아닙니다.")
    except ZeroDivisionError:
        print("오류: 0으로 나눌 수 없습니다.")

# 함수 호출
multiple_exception_handling()
```

예외 처리로 pass문 사용

- 실행 오류가 발생했을 때, 오류를 처리하지 않고 프로그램 실행을 지속시키기 위해 pass문을 사용하는 경우가 있음

```
try:
    print(3 / 0)   # 오류 발생
except ZeroDivisionError:
    pass           # 예외 처리로 아무것도 안함
print("오류가 발생함에도 여기까지 출력됨")
```

실습문제 3

- 문제
  - 사용자로부터 정수를 입력받아 100을 입력받은 수로 나누는 프로그램을 작성하세요.

- 요구사항
  - 예외가 발생할 경우, 예외의 이름과 설명을 출력하세요.

실습문제 3

- 최종 코드

```
def print_exception_info():
    try:
        num = int(input("숫자를 입력하세요: "))
        result = 100 / num
        print("결과:", result)
    except Exception as e:
        print(f"오류: {type(e).__name__}, 설명: {e}")

# 함수 호출
print_exception_info()
```

실습문제 4

- 문제
  - 파일을 열어 내용을 읽은 후 파일을 닫는 프로그램을 작성하세요.

- 요구사항
  - 파일 처리 도중 예외가 발생하더라도 파일이 정상적으로 닫히도록 finally 블록을 사용하세요

실습문제 4

- 최종 코드

```
def read_file_with_finally():
    try:
        file = open("output.txt", "r")
        print(file.read())
    except FileNotFoundError:
        print("오류: 파일을 찾을 수 없습니다.")
    finally:
        file.close()
        print("파일이 닫혔습니다.")

# 함수 호출
read_file_with_finally()
```

실습문제 5

- 문제
  - 사용자로부터 나이를 입력받아, 나이가 0보다 작거나 같으면 예외를 발생시키는 프로그램을 작성하세요.

- 요구사항

실습문제 5

- 최종 코드

```
def raise_exception_on_age():
    age = int(input("나이를 입력하세요: "))
    if age <= 0:
        raise ValueError("나이는 0보다 커야 합니다.")
    print("입력된 나이:", age)

# 함수 호출
try:
    raise_exception_on_age()
except ValueError as e:
    print(f"오류: {e}")
```

실습문제 6

- 문제
  - 나이 입력에서 부적절한 값이 들어왔을 때 이 예외를 발생시키는 프로그램을 작성하세요.

- 요구사항
  - Raise 활용

실습문제 6

- 최종 코드

```
def custom_exception_for_age():
    age = int(input("나이를 입력하세요: "))
    if age <= 0:
        raise Exception("나이는 0보다 커야 합니다.")
    print("입력된 나이:", age)

# 함수 호출
try:
    custom_exception_for_age()
except Exception as e:
    print(f"오류: {e}")
```

실습문제 7

- 문제
  - 사용자로부터 파일 이름을 입력받아 파일을 열고 내용을 읽는 '예외 체인' 프로그램을 작성하세요.

- 요구사항
  - 파일을 찾을 수 없는 경우, 사용자에게 더 친절한 메시지를 제공하면서 원래의 예외도 함께 출력하세요.

https://docs.python.org/ko/3/library/exceptions.html

실습문제 7

- 최종 코드

```
def exception_chaining():
    try:
        filename = input("파일 이름을 입력하세요: ")
        with open(filename, "r") as file:
            print(file.read())
    except FileNotFoundError as e:
        raise FileNotFoundError("파일을 열 수 없습니다. 파일 이름을
확인하세요.") from e

# 함수 호출
try:
    exception_chaining()
except FileNotFoundError as e:
    print(e)
    print("원래의 예외:", e.__cause__)
```

실습문제 8

- 문제 - 리소스 관리
  - 파일을 안전하게 열고, 파일 내용을 읽은 후, 자동으로 파일을 닫는 with 문을 사용하는 프로그램을 작성하세요.

- 요구사항
  - 파일이 존재하지 않는 경우 예외 처리를 포함하세요

실습문제 8

- 최종 코드

```
def safe_file_handling():
    try:
        with open("data.txt", "r") as file:
            print(file.read())
    except FileNotFoundError:
        print("오류: 파일을 찾을 수 없습니다.")

# 함수 호출
safe_file_handling()
```

실습문제 9

- 문제
  - 사용자로부터 "yes" 또는 "no"만을 입력받아야 하는 프로그램을 작성하세요.

- 요구사항
  - 잘못된 입력이 들어오면 예외를 발생시키고, 다시 입력받도록 처리하세요. (while)

실습문제 9

- 최종 코드

```
def get_yes_or_no():
    while True:
        try:
            response = input("yes 또는 no를 입력하세요: ")
            if response.lower() not in ["yes", "no"]:
                raise ValueError("잘못된 입력입니다. yes 또는 no만
입력해야 합니다.")
            break  # 올바른 입력을 받으면 반복문을 종료
        except ValueError as e:
            print(e)
    print("올바른 응답:", response)

# 함수 호출
get_yes_or_no()
```

실습문제 10

- 문제
  - 사용자로부터 두 개의 숫자를 입력받아 더하는 프로그램을 작성하세요. 입력이 숫자가 아닐 경우 타입 에러를 처리하세요.

- 요구사항
  - 타입 에러 처리

실습문제 10

- 최종 코드

```
def add_numbers():
    try:
        num1 = int(input("첫 번째 숫자를 입력하세요: "))
        num2 = int(input("두 번째 숫자를 입력하세요: "))
        print("두 숫자의 합:", num1 + num2)
    except ValueError:
        print("오류: 숫자만 입력해야 합니다.")

# 함수 호출
add_numbers()
```

실습문제 11

- 문제
  - 사용자로부터 파일 이름을 입력받아 내용을 출력하는 프로그램을 작성하세요. 파일 엑세스 에러(존재하지 않거나, 읽을 수 없는 파일)를 처리하세요.

- 요구사항
  - 파일 엑세스 에러 처리

실습문제 11

- 최종 코드

```
def read_file_content():
    try:
        filename = input("읽을 파일 이름을 입력하세요: ")
        with open(filename, 'r') as file:
            print(file.read())
    except FileNotFoundError:
        print("오류: 파일을 찾을 수 없습니다.")
    except PermissionError:
        print("오류: 파일 읽기 권한이 없습니다.")

# 함수 호출
read_file_content()
```

실습문제 12

- 문제
  - 입력 중에 발생할 수 있는 예외를 무시하고 프로그램을 계속 진행하게 만드는 프로그램을 작성하세요.

- 요구사항
  - 예외무시

실습문제 12

- 최종 코드

```
def ignore_exception():
    try:
        num = int(input("숫자를 입력하세요: "))
        print(f"입력된 숫자: {num}")
    except ValueError:
        pass  # 예외를 무시하고 진행
    print("프로그램 종료")

# 함수 호출
ignore_exception()
```

실습문제 13

- 문제
  - 사용자로부터 비밀번호를 입력받습니다. 비밀번호가 8자 미만이거나 숫자를 포함하지 않을 경우 ValueError를 발생시키는 프로그램을 작성하세요.

- 요구사항
  - 비밀번호 복잡성 검사
  - 문자열의 길이 확인과 문자열 내 숫자 포함 여부를 검사하여 비밀번호 복잡성을 검증

실습문제 13

- 최종 코드

```
def check_password_complexity():
    password = input("비밀번호를 입력하세요: ")
    if len(password) < 8 or not any(char.isdigit() for char in
password):
        raise ValueError("비밀번호는 최소 8자 이상이어야 하며,
숫자를 하나 이상 포함해야 합니다.")
    print("비밀번호가 적합합니다.")

# 함수 호출
try:
    check_password_complexity()
except ValueError as e:
    print(e)
```


NVIDIA