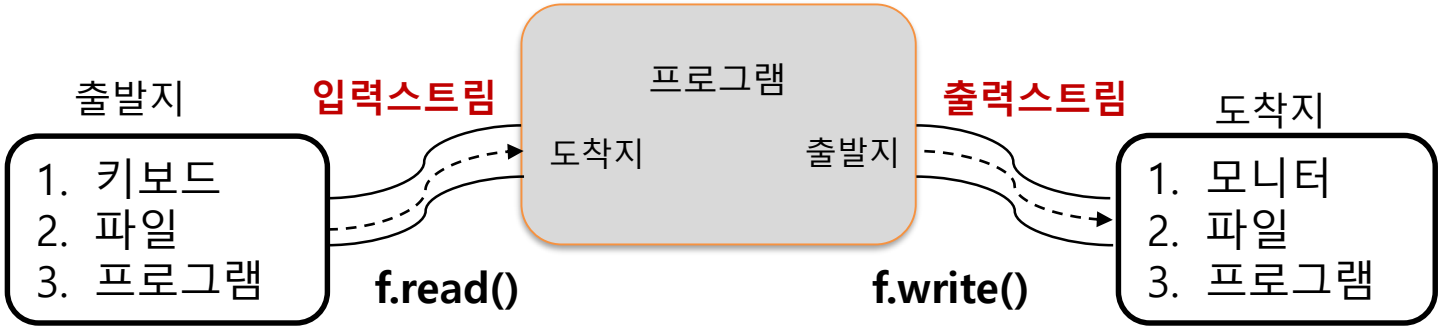
9장. 파일 입출력, 예외처리

입, 출력 스트림

- **스트림(stream)**

  자료흐름이 물의 흐름과 같다는 뜻이다. 입출력 장치는 매우 다양하기 때문에 프로그램 호환성이 떨어짐

  - 입력 스트림 – 동영상을 재생하기 위해 동영상 파일에서 자료를 읽을때 사용함
  - 출력 스트림 – 사용자가 쓴 글을 파일에 저장할 때는 출력 스트림 사용함

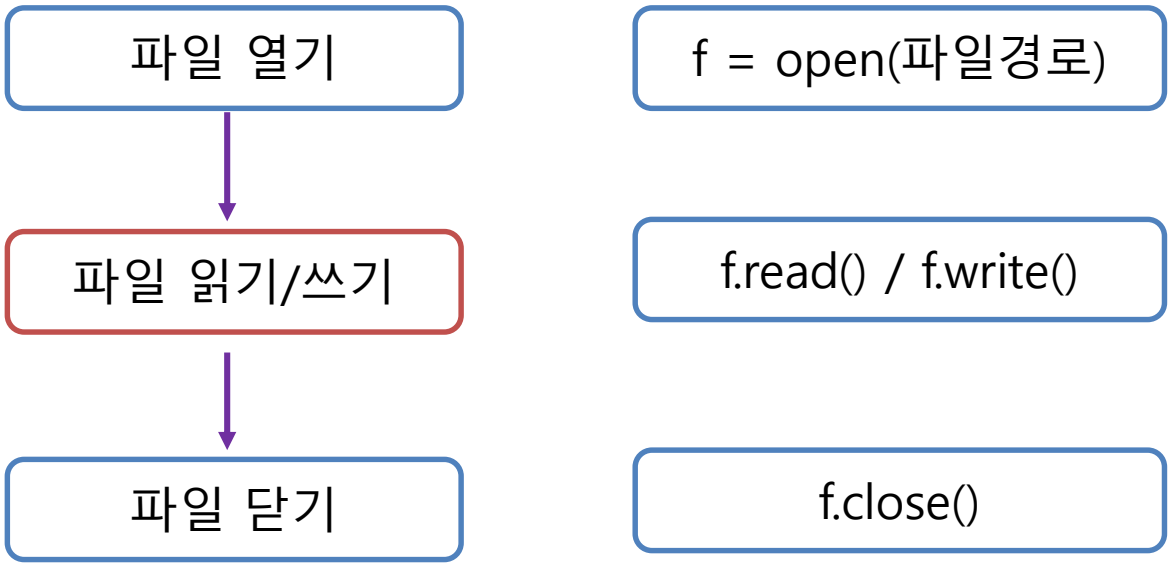


파일 입출력

- **파일 입출력의 필요성**

  프로그램 실행 중에 메모리에 저장된 데이터는 프로그램이 종료되면 사라진다.
  데이터를 프로그램이 종료된 후에도 계속해서 사용하려면 파일에 저장하고
  필요할 때 파일을 읽어서 데이터를 사용할 수 있다.

- **파일 입출력 프로세스**



파일 쓰기

- **파일 관련 주요 메서드**

메서드	모드	기능
open(파일, "w")	w	파일 열기(쓰기)
open(파일, "r")	r	파일 열기(읽기)
open(파일, "a")	a	파일 열기(추가 쓰기)
close()		파일 닫기

파일 쓰기

- **파일 쓰기**

  write() 함수 – 파일에 내용을 쓰는 함수로 문자(열)만 쓰기 가능함

```
# 파일 열기
f = open("C:/pyfile/file.txt", "w")

# 파일 쓰기
f.write("하늘\n")
f.write("cloud\n")
f.write("學生\n")
f.write("30\n")
f.write("3.14\n")

# 숫자 쓰기 불가
# f.write(200)
```

```
# 숫자 쓰기 불가
# f.write(200)

# 변수를 사용하여 쓰기
num = 200 * 3
f.write(f"{num}\n")

# 파일 종료
f.close()
```

```
하늘
cloud
學生
30
3.14
600
```

파일 읽기

- **파일 읽기**

  read() 함수 – 파일의 내용 전체를 읽어서 문자열로 돌려준다.

```
# 파일 열기
f = open("C:/pyfile/file.txt", "r")

# 파일 읽기
data = f.read()
print(data)

# 파일 닫기
f.close()
```

```
하늘
cloud
學生
30
3.14
600
```

파일 쓰기(추가모드)

- **파일 쓰기(추가 모드)**

  'w' 모드는 초기화 되어 저장되므로 추가할때는 'a'모드 사용

```
f = open("C:/pyfile/file.txt", "a")

f.write("Good Luck!\n")
f.write("행운을 빌어요!\n")

f.close()
```

```
하늘
cloud
學生
30
3.14
600
Good Luck!
행운을 빌어요!
```

파일 쓰기 – 리스트형

- 리스트형 자료를 파일에 쓰기

```
try:
    f = open("c:/pyfile/cartlist.txt", "w")

    cartlist = ["계란", "우유", "바나나", "라면"]

    for cart in cartlist:
        f.write(cart + " ")

    f.close()
except FileNotFoundError:
    print("파일을 찾을 수 없습니다.")
```

리스트 랜덤 출력

- 리스트형 자료 랜덤하게 출력

  f.read().split() 사용

```
try:
    f = open("c:/pyfile/cartlist.txt", "r")

    # cartlist = f.read()
    # print(cartlist)

    cartlist = f.read().split()
    print(cartlist)

    cart = random.choice(cartlist)
    print(cart)

    f.close()
except FileNotFoundError:
    print("파일을 찾을 수 없습니다.")
```

```
['계란', '우유', '바나나', '라면']
품목:  우유
```

with ~ as 구문

- 자원누수 방지를 돕는 with ~ as 구문

  f.close()를 사용하지 않음

with open(파일이름) as 파일 객체:
코드 블럭

with ~ as 구문 예제

- 구구단 파일 만들기

```
with open('99.txt', 'w') as f:
    for i in range(2, 10):
        for j in range(1, 10):
            gugudan = "%d x %d = %d" % (i, j, i*j)
            f.write(gugudan)
            f.write('\n')
        f.write('\n')
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
```

```
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
```

영어 타자 연습 프로그램

- **영어 타자 연습 프로그램**

  게임 방법

  - ✓ 파일 쓰기를 이용하여 word.txt 파일을 생성한다.
  - ✓ 게임이 시작되면 영어 단어가 화면에 표시된다.
  - ✓ 사용자는 최대한 빠르고 정확하게 입력해야 한다.
  - ✓ 바르게 입력했으면 다음 문제로 넘어가고 "통과"를 출력한다.
  - ✓ 오타가 있으면 '오타! 다시 도전!'이 출력되고 같은 단어가 한 번 더 나온다.
  - ✓ 타자 게임 시간을 측정한다.

영어 타자 연습 프로그램

```
[타자 게임]준비되면 엔터!

-문제 1
grape
grape
통과!
-문제 2
potato
potata
오타! 다시 도전!
-문제 2
potato
potato
통과!
-문제 3
grape
```

```
-문제 9
tree
tree
통과!
-문제 10
garlic
garlic
통과!
타자 시간 : 34.46초
```

리스트 랜덤 출력

- word.txt 파일 만들고, 랜덤 추출하기

```
import random

with open("word.txt", 'w') as f:
    word = ['sky', 'earth', 'moon', 'flower', 'tree',
            'strawberry', 'grape', 'garlic','onion', 'potato']
    for i in word:
        f.write(i + ' ')

# 단어를 랜덤 추출
with open("word.txt", 'r') as f:
    data = f.read().split()
    word = random.choice(data)
    print(word)
```

영어 타자 연습 프로그램

```
import random
import time

try:
    # 파일 읽기
    with open("word.txt", "r") as f:
        word = f.read().split()
        #print(word)

    n = 1 #문제 번호

    print("[타자 게임] 준비되면 엔터")
    input()

    start = time.time()
```

영어 타자 연습 프로그램

```
    while n < 11:
        print("\n문제", n)
        q = random.choice(word)
        print(q)  #문제 출제

        u = input() #사용자 입력
        if q == u:
            print("통과!")
            n += 1
        else:
            print("오타! 다시 도전!")

    end = time.time()
    et = end - start
    print(f"게임 소요 시간: {et:.2f}")

except FileNotFoundError:
    print("파일을 찾을 수 업습니다.")
```

파일 쓰기 – 입력 받기

◆ 입력 받아 파일 쓰기

```
# 텍스트 파일 한글 깨짐 방지 - encoding='utf-8'
with open("input.txt", 'a', encoding='utf-8') as f:
    text = input("저장할 내용을 입력해 주세요: ")
    f.write(text)
    f.write('\n')
```

```
sky
바다
오늘도 좋은 하루 되세요~
10000
12.345
```

성적 입력 처리

◆ 파일에 과목의 성적 저장하는 프로그램

```
# vs code 텍스트 파일 한글 깨짐 - encoding='utf-8'
with open('score.txt', 'a', encoding='utf-8') as f:
    name = input("이름 입력: ")
    kor = input("국어 점수: ")
    math = input("수학 점수: ")

    f.write(name + ' ')
    f.write(kor + ' ')
    f.write(math + '\n')
```

```
오상식 90 85
최지능 80 95
```

성적 입력 처리 – 반복

◆ 반복해서 과목의 성적을 저장하는 프로그램

```
성적을 저장할까요?(y/n) : y
이름 입력 : 오상식
국어 점수 : 80
수학 점수 : 70
성적을 저장할까요?(y/n) : y
이름 입력 : 최지능
국어 점수 : 95
수학 점수 : 90
성적을 저장할까요?(y/n) : n
입력을 종료합니다.
```

scorelist.txt

```
1  오상식 80 70
2  최지능 95 90
```

성적 입력 처리 – 반복

◆ 반복해서 과목의 성적을 저장하는 프로그램

```
try:
    with open('scorelist.txt', 'a', encoding='utf-8') as f:
        while True:
            key = input("성적을 저장할까요?(y/n): ")
            if key == 'n' or key == 'N':
                break
            elif key == 'y' or key == 'Y':
                name = input("이름 입력: ")
                kor = input("국어 점수: ")
                math = input("수학 점수: ")

                f.write(name + ' ')
                f.write(kor + ' ')
                f.write(math + '\n')
            else:
                print("잘못된 입력입니다. 다시 입력하세요")
        print("입력을 종료합니다.")
except FileNotFoundError:
    print("파일을 찾을 수 없습니다.")
```

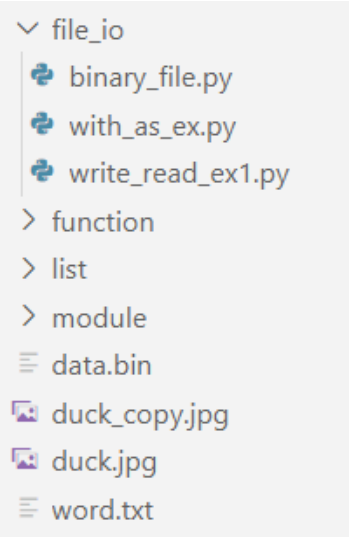
바이너리 파일 읽고 쓰기

◆ 바이너리 파일

바이너리 파일이란 화상, 음성 등의 대부분의 파일로 0과 1로 이루어진 파일이다.

open("파일 위치", 'wb')

모드	설명
wb	쓰기
rb	읽기



바이너리 파일 읽고 쓰기

◆ 바이너리 파일

```
# 바이너리 파일 읽고 쓰기
with open("data.bin", "wb") as f:
    text = "비가 내린다."
    f.write(text.encode())

with open("data.bin", "rb") as f:
    data = f.read()
    print(data.decode())
```

바이너리 파일 읽고 쓰기

◆ 이미지 복사하기

이미지 파일 읽어와서 다른 이름으로 쓰기

```
# 이미지 파일 읽기
with open("duck.jpg", "rb") as f1:
    data = f1.read()

# 이미지 파일 쓰기
with open(".duck_copy.jpg", "wb") as f2:
    f2.write(data)
```

pickle 모듈

◎ **pickle** 모듈

- 객체의 형태를 그대로 유지하면서 파일에 저장하고 불러올 수 있는 모듈이다.
- 이때 객체란, 리스트나 딕셔너리등의 자료구조도 포함한다.

모드	설명
pickle.dump	쓰기
pickle.load	읽기

pickle 모듈

```
import pickle

try:
    with open("object.dat", "wb") as f:
        lis = ['강아지', '고양이', '닭']
        dic = {1: '강아지', 2: '고양이', 3: '닭'}
        pickle.dump(lis, f)
        pickle.dump(dic, f)
except FileNotFoundError:
    print("파일을 찾을 수 없습니다.")


try:
    with open("object.dat", "rb") as f:
        lis = pickle.load(f)
        dic = pickle.load(f)
        print(lis)
        print(dic)
except FileNotFoundError:
    print("파일을 찾을 수 없습니다.")
```

```
['강아지', '고양이', '닭']
{1: '강아지', 2: '고양이', 3: '닭'}
```

실습 문제 – 파일 입출력

실행 결과가 표시되도록 코드 작성란을 완성하세요.

☞ 실행 결과

```
봄
여름
가을
겨울
```

```
seasons = ["봄", "여름", "가을", "겨울"]

with open("season.txt", "w", encoding='utf-8') as f1:

    # 코드 작성
```

```
with open("season.txt", "r", encoding='utf-8') as f2:

    #코드 작성
```

에러(Error)와 예외(Exception)

에러(Error)

- **구문(syntex) 오류** : 문법에 맞지 않거나 오타가 났을 경우 발생하는 오류
  IDE에서 실행 전에 알 수 있음

예외(Exception)

- **실행(runtime) 오류** : 문법적인 오류는 없지만 실행(run) 될 때 에러가 발생하는 것을 말한다.

  예) 파일을 읽어 사용하려는데 파일이 없는 경우,
  리스트 값을 출력하려는데 리스트 요소가 없는 경우 등..

  에러가 발생되면 프로그램의 동작이 중지 또는 종료된다

예외(Exceptions)

❖ python.org > Tutorial(자습서)

8.2. Exceptions

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called *exceptions* and are not unconditionally fatal: you will soon learn how to handle them in Python programs. Most exceptions are not handled by programs, however, and result in error messages as shown here:

Copy

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    10 * (1/0)
         ~^~
ZeroDivisionError: division by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    4 + spam*3
        ^^^^
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    '2' + 2
    ~~~~^~~
TypeError: can only concatenate str (not "int") to str
```

예외(Exceptions)

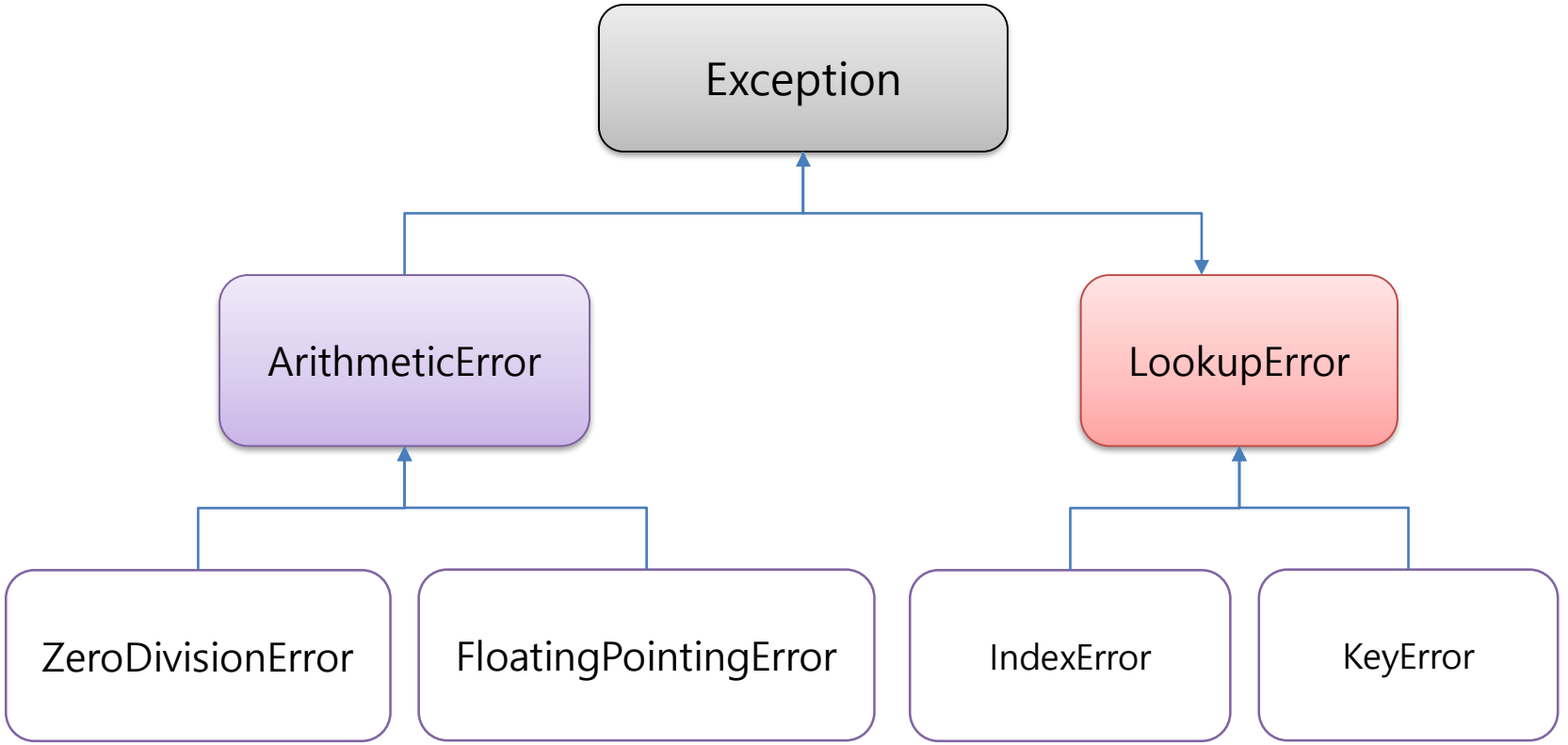
❖ 여러가지 예외 발생 코드

```
#TypeError: can only concatenate str (not "int") to str
print('2' + 2)

# NameError: name 'num' is not defined.
calc1 = 4 + num * 3
print(calc1)

# ZeroDivisionError: division by zero
calc2 = 10 * (1/0)
print(calc2)
```

Exception 계층도



예외(Exceptions)

❖ **예외 처리 방법** : **try ~ except 구문**

try:

예외가 발생할 가능성이 있는 코드

except 예외 클래스:

예외가 발생했을 경우 실행 코드

예외(Exceptions)

- 숫자를 입력할 곳에 문자를 입력하여 예외 발생

```
숫자를 입력하세요 : ㅂ
Traceback (most recent call last):
  File "C:/pyworks/cho8/try_except/value_error.py", line 3,
    x = int(input("숫자를 입력하세요 : "))
ValueError: invalid literal for int() with base 10: 'ㅂ'
```

```
while True:
    x = input("숫자를 입력하세요(q 종료): ")
    if x == 'q':
        print("프로그램 종료!")
        break

    try:
        num = int(x)
        print(num)
    except ValueError:
        print("유효한 숫자가 아닙니다. 다시 입력해 주세요")
```

예외(Exceptions)

숫자 추측 게임

- 예외 처리

```
import random

com = random.randint(1, 30) #컴퓨터의 난수
# print(com)

while True:
    x = input("맞혀보세요(입력: 1 ~ 30): ")
    guess = int(x) # 사용자가 추측한 수

    if guess < 1 or guess > 30:
        print("범위를 초과했어요. 다시 입력하세요")
    elif guess == com:
        print("정답!")
        break
    elif guess > com:
        print("너무 커요")
    else:
        print("너무 작아요")
```

다중 예외(Exceptions)

- 다중 try ~ except 구문

```
try:
    실행 코드
except 오류 Type1 as e
    문제 발생시 실행코드
except 오류 Type2 as e
    문제 발생시 실행코드
```

오류 메시지 표시

다중 예외(Exceptions)

- 다중 try ~ except 구문

```
try:
    data= [50, 40, 80, 60]
    x = input("정수 입력(0~4): ")
    num = int(x)
    print(data[num])
except IndexError as e:
    print("범위를 초과했어요. 0~ 4까지 입력하세요")
except ValueError as e:
    print("유효한 숫자가 아닙니다. 다시 입력바랍니다.")
```

예외(Exceptions)

❖ try ~ except ~ finally 구문

finally 구문을 반드시 실행한다.

```
def divide(x, y):
    try:
        result = x / y
        print(result)
    except ZeroDivisionError:
        print("0으로 나눌수 없습니다.")
    finally:
        print("여기는 반드시 수행되는 구간입니다.")

#divide(2, 1)
divide(2, 0)
```

예외 처리 미루기

➢ raise 사용

예외 처리를 raise 명령어로 연기했다가 코드를 사용하는 곳에서 예외 처리

```
class Animal:
    def cry(self):
        raise NotImplementedError("구현되지 않는 메서드가 있습니다.")

class Dog(Animal):
    def cry(self):
        print("왈~ 왈~")

class Cat(Animal):
    pass
    # def cry(self):
    #     print("야~ 옹!")

try:
    dog = Dog()
    dog.cry()

    cat = Cat()
    cat.cry()
except NotImplementedError as e:
    print(f"오류: {e}")
```

반드시 구현하도록 만드는 예외 처리