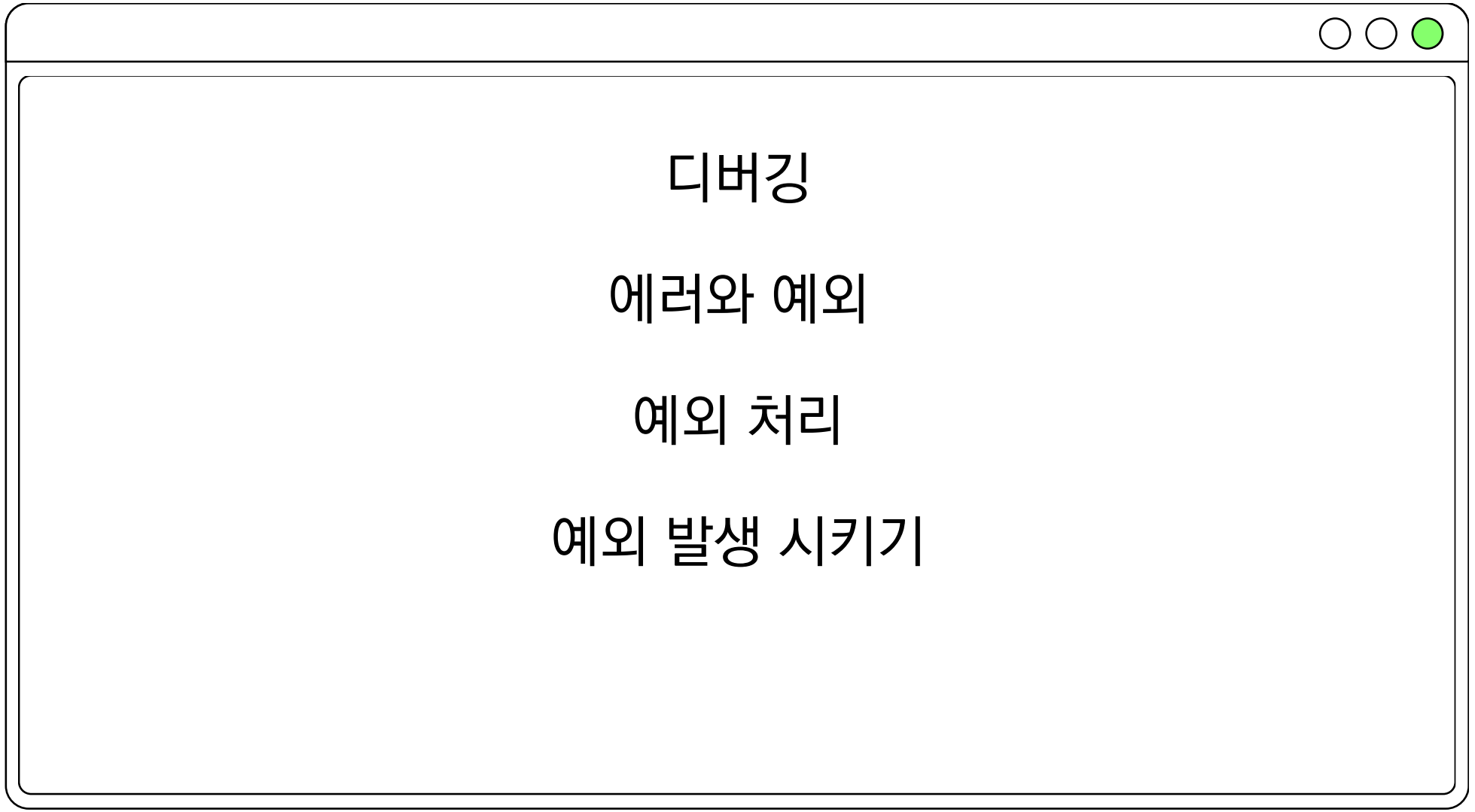
HAPPY HACKING

에러/예외 처리
(Error/Exception Handling)

목차

HAPPY HACKING



HAPPY HACKING

여러분은 어떻게 디버깅하시나요?

오류가 많이 발생하는 곳은 어딜까요?

어느 부분을 중점적으로 봐야할까요?

제어가 되는 시점
조건/반복, 함수

branches

for loops

while loops

function

branches

모든 조건이 원하는대로 동작하는지

for loops

반복문에 진입하는지, 원하는 횟수만큼 실행되는지

while loops

for loops와 동일, 종료조건이 제대로 동작하는지

function

함수 호출시, 함수 파라미터, 함수 결과

디버깅

HAPPY HACKING

“코드의 상태를 신중하게 출력해가며 심사숙고하는 것보다 효과적인 디버깅 도구는 없습니다.”

— 브라이언 커니핸, Unix for Beginners.

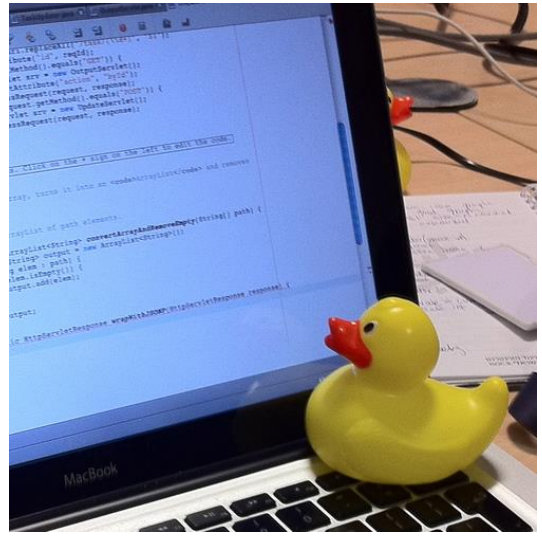
- print 함수 활용
  - 특정 함수 결과, 반복/조건 결과 등 나눠서 생각, 코드를 bisection으로 나눠서 생각
- 개발 환경(text editor, IDE) 등에서 제공하는 기능 활용
  - breakpoint, 변수 조회 등
- Python tutor 활용 (단순 파이썬 코드인 경우)
- 뇌컴파일, 눈디버깅

디버깅

HAPPY HACKING

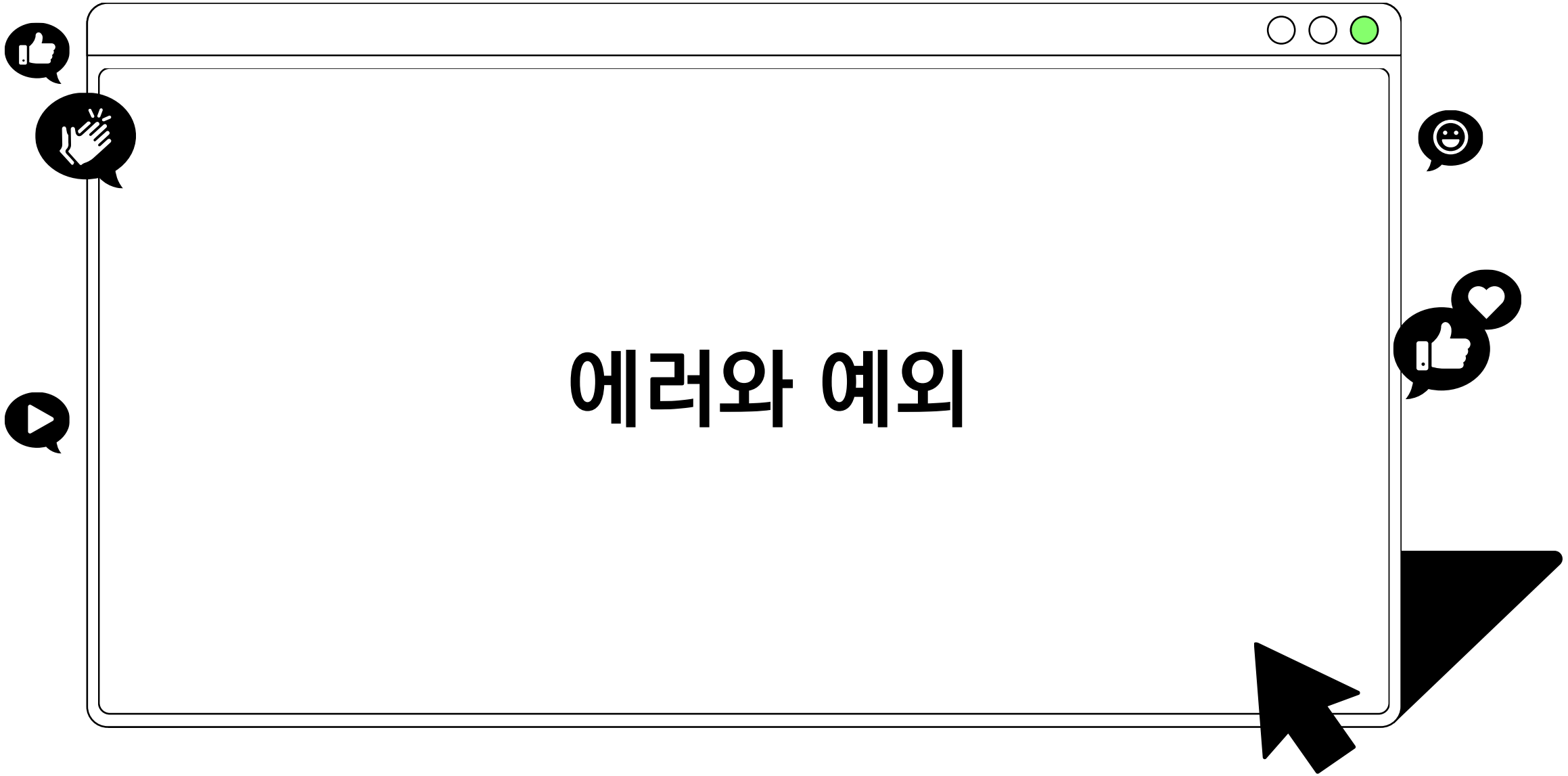
코드를 작성하다가..

- 에러 메시지가 발생하는 경우
  - 해당 하는 위치를 찾아 메시지를 해결
- 로직 에러가 발생하는 경우
  - 명시적인 에러 메시지 없이 예상과 다른 결과가 나온 경우
    - 정상적으로 동작하였던 코드 이후 작성된 코드를 생각해봄
    - 전체 코드를 살펴봄
    - 휴식을 가져봄
    - 누군가에게 설명해봄
    - …



https://en.wikipedia.org/wiki/Rubber_duck_debugging

HAPPY HACKING



에러와 예외

HAPPY HACKING

문법 에러(Syntax Error)

- SyntaxError가 발생하면, 파이썬 프로그램은 실행이 되지 않음
- 파일이름, 줄번호, ^ 문자를 통해 파이썬이 코드를 읽어 나갈 때(parser) 문제가 발생한 위치를 표현
- 줄에서 에러가 감지된 가장 앞의 위치를 가리키는 캐럿(caret)기호(^)를 표시

```
if else
```

```
File "<ipython-input-1-f8a097d0a685>", line 1
if else ^
SyntaxError: invalid syntax
```

HAPPY HACKING

문법 에러(Syntax Error)

- EOL (End of Line)

```
print('hello
# File "<ipython-input-6-2a5f5c6b1414>", line 1
# print('hello
#             ^
# SyntaxError: EOL while scanning string literal
```

- EOF (End of File)

```
print(
# File "<ipython-input-4-424fbb3a34c5>", line 1
# print(
#     ^
# SyntaxError: unexpected EOF while parsing
```

HAPPY HACKING

문법 에러(Syntax Error)

- Invalid syntax

```
while
# File "<ipython-input-7-ae84bbebe3ce>", line 1
# while
#         ^
# SyntaxError: invalid syntax
```

- assign to literal

```
5 = 3
# File "<ipython-input-28-9a762f2c796b>", line
1
# 5 = 3
# ^
# SyntaxError: cannot assign to literal
```

에러와 예외

HAPPY HACKING

예외(Exception)

- 실행 도중 예상치 못한 상황을 맞이하면, 프로그램 실행을 멈춤
  - 문장이나 표현식이 문법적으로 올바르더라도 발생하는 에러
- 실행 중에 감지되는 에러들을 예외(Exception)라고 부름
- 예외는 여러 타입(type)으로 나타나고, 타입이 메시지의 일부로 출력됨
  - NameError, TypeError 등은 발생한 예외 타입의 종류(이름)
- 모든 내장 예외는 Exception Class를 상속받아 이뤄짐
- 사용자 정의 예외를 만들어 관리할 수 있음

https://docs.python.org/ko/3/library/exceptions.html#built-in-exceptions

에러와 예외

HAPPY HACKING

- ZeroDivisionError

```
10/0
```

- NameError

```
print(name_error)
```

에러와 예외

HAPPY HACKING

- ZeroDivisionError : 0으로 나누고자 할 때 발생

```
10/0
# ---------------
# ZeroDivisionError Traceback (most recent call last)
# ----> 1 10/0
# ZeroDivisionError: division by zero
```

- NameError : namespace 상에 이름이 없는 경우

```
print(name_error)
# --------------------------
# NameError Traceback (most recent call last)
# ----> 1 print(name_error)
# NameError: name 'name_error' is not defined
```

HAPPY HACKING

TypeError

```
1 + '1'
```

```
round('3.5')
```

HAPPY HACKING

TypeError : 타입 불일치

```
1 + '1'
# --------------
# TypeError    Traceback (most recent call last)
# ----> 1 1 + '1'
# TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
round('3.5')
# ---------------
# TypeError   Traceback (most recent call last)
# ----> 1 round('3.5')
# TypeError: type str doesn't define __round__ method
```

HAPPY HACKING

TypeError

```
divmod()
```

```
import random
random.sample()
```

HAPPY HACKING

TypeError – arguments 부족

```
divmod()
# ------------
# TypeError  Traceback (most recent call last)
# ----> 1 divmod()
# TypeError: divmod expected 2 arguments, got 0
```

```
import random
random.sample()
# ---------
# TypeError  Traceback (most recent call last)
#       1 import random
# ----> 2 random.sample()
# TypeError: sample() missing 2 required positional arguments:
'population' and 'k'
```

HAPPY HACKING

TypeError

```
divmod(1, 2, 3)
```

```
import random
random.sample(range(3), 1, 2)
```

HAPPY HACKING

TypeError

```
divmod(1, 2, 3)
```

```
import random
random.sample(range(3), 1, 2)
```

TypeError – argument 개수 초과

```
divmod(1, 2, 3)
# ---------
# TypeError Traceback (most recent call last)
# ----> 1 divmod(1, 2, 3)
# TypeError: divmod expected 2 arguments, got 3
```

```
import random
random.sample(range(3), 1, 2)
# --------
# TypeError Traceback (most recent call last)
#       1 import random
# ----> 2 random.sample(range(3), 1, 2)
# TypeError: sample() takes 3 positional arguments but 4 were given
```

HAPPY HACKING

TypeError

```
import random
random.sample(1, 2)
# ------
# TypeError Traceback (most recent call last)
#       1 import random
# ----> 2 random.sample(1, 2)

# ~/.pyenv/versions/3.8.6/lib/python3.8/random.py in sample(self, population, k)
#     357             population = tuple(population)
#     358         if not isinstance(population, _Sequence):
# --> 359             raise TypeError("Population must be a sequence or
set.  For dicts, use list(d).")
#     360         randbelow = self._randbelow
#     361         n = len(population)

# TypeError: Population must be a sequence or set.  For dicts, use list(d).
```

HAPPY HACKING

ValueError

```
int('3.5')
```

```
range(3).index(6)
```

ValueError – 타입은 올바르나 값이 적절하지 않거나 없는 경우

```
int('3.5')
# ------
# ValueError Traceback (most recent call last)
# ----> 1 int('3.5')
# ValueError: invalid literal for int() with base 10:
'3.5'
```

```
range(3).index(6)
# ------
# ValueError Traceback (most recent call last)
# ----> 1 range(3).index(6)
# ValueError: 6 is not in range
```

IndexError

```
empty_list = []
empty_list[2]
# ------
# IndexError Traceback (most recent call last)
#       1 empty_list = []
# ----> 2 empty_list[2]
# IndexError: list index out of range
```

HAPPY HACKING

KeyError

```
song = {'IU': '좋은날'}
song['BTS']
# ------
# KeyError Traceback (most recent call last)
#       1 song = {'IU': '좋은날'}
# ----> 2 song['BTS']
# KeyError: 'BTS'
```

HAPPY HACKING

ModuleNotFoundError

```
import nonamed
```

HAPPY HACKING

ModuleNotFoundError – 존재하지 않는 모듈을 import 하는 경우

```
import nonamed
# ------
# ModuleNotFoundError Traceback (most recent call last)
# ----> 1 import nonamed
# ModuleNotFoundError: No module named 'nonamed'
```

HAPPY HACKING

ImportError

```
from random import samp
```

HAPPY HACKING

ImportError – Module은 있으나 존재하지않는 클래스/함수를 가져오는 경우

```
from random import samp
# ------
# ImportError Traceback (most recent call last)
# ----> 1 from random import samp
# ImportError: cannot import name 'samp' from 'random'
```

HAPPY HACKING

IndentationError – Indentation이 적절하지 않는 경우

```
for i in range(3):
print(i)
# File "<ipython-input-56-78291925d94f>", line 2
# print(i)
# ^
# IndentationError: expected an indented block
```

HAPPY HACKING

KeyboardInterrupt – 임의로 프로그램을 종료하였을 때

```
while True:
    continue
# ------
# KeyboardInterrupt Traceback (most recent call last)
# <ipython-input-55-6a65cf439648> in <module>
#       1 while True:
# ----> 2     continue
# KeyboardInterrupt:
```

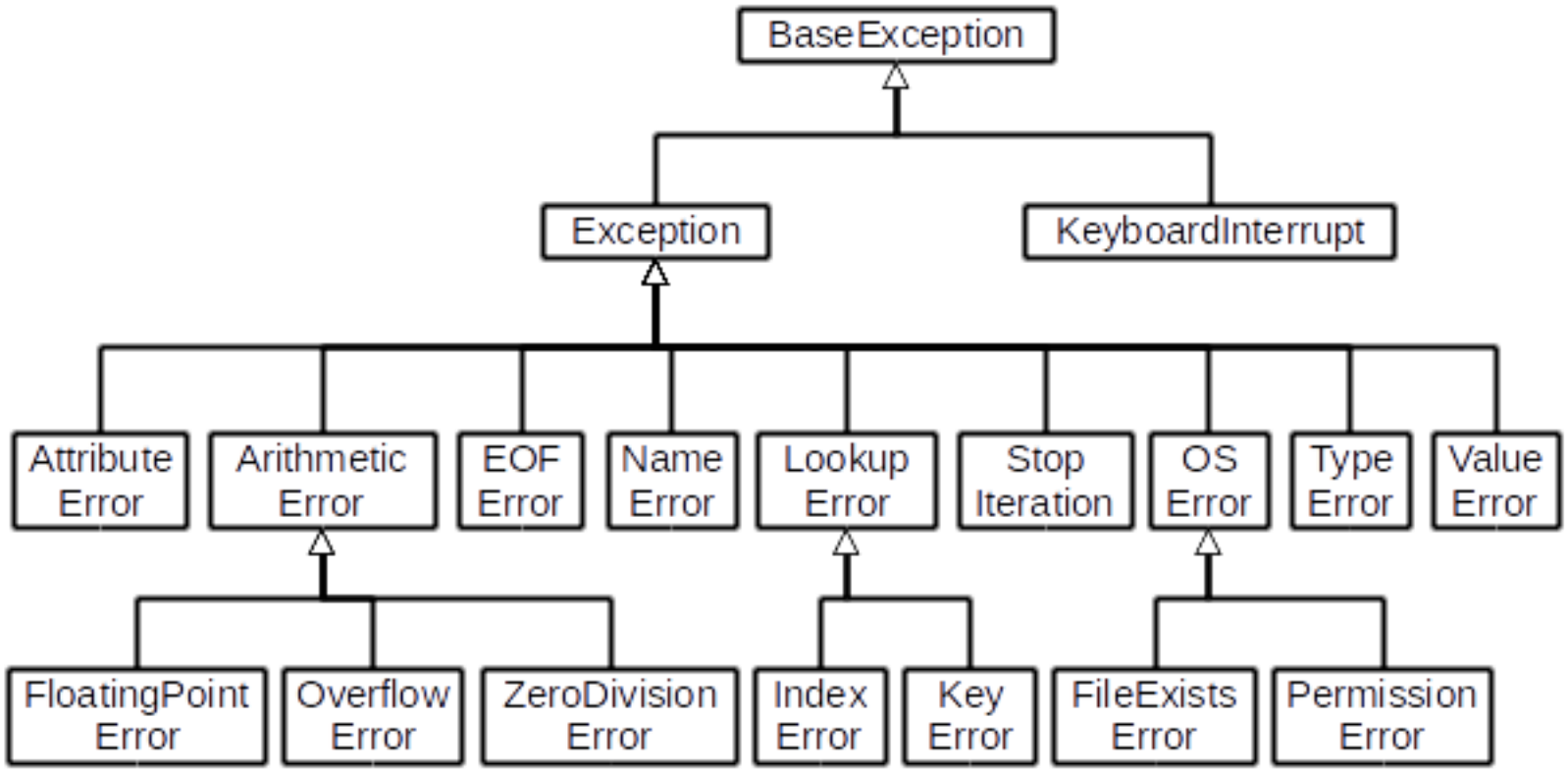
에러와 예외

HAPPY HACKING

파이썬 내장 예외의 클래스 계층 구조

https://docs.python.org/ko/3/library/exceptions.html#exception-hierarchy

- 파이썬 내장 예외 (built-in-exceptions)



HAPPY HACKING

예외처리

HAPPY HACKING

예외 처리

- try 문(statement) / except 절(clause)을 이용하여 예외 처리를 할 수 있음
- try문
  - 오류가 발생할 가능성이 있는 코드를 실행
  - 예외가 발생되지 않으면, except 없이 실행 종료
- except 문
  - 예외가 발생하면, except 절이 실행
  - 예외 상황을 처리하는 코드를 받아서 적절한 조치를 취함

예외 처리

HAPPY HACKING

- 처리 순서

예외없는 정상종료

```
try:
    try-명령문
except 예외:
    예외처리 명령문
else:
    else 명령문
Finally:
    최종 정리문(clean-up)
```

예외처리 할 경우

```
try:
    try-명령문
except 예외:
    예외처리 명령문
else:
    else 명령문
Finally:
    최종 정리문(clean-up)
```

예외처리 하지 못한 경우

```
try:
    try-명령문
except 예외:
    예외처리 명령문
else:
    else 명령문
Finally:
    최종 정리문(clean-up)
```

오류메시지 출력

예외 처리

HAPPY HACKING

- **작성 방법**

⚠ 주의사항
try문은 반드시 한 개 이상의 except문이 필요

```
try:
    try 명령문
except 예외그룹-1 as 변수-1 :
    예외처리 명령문 1
except 예외그룹-2 as 변수-2 :
    예외처리 명령문 2
finally:
    finally명령문
```

선택사항

HAPPY HACKING

예외 처리 예시

```
num = input('숫자입력 :')
print(int(num))
# 숫자입력 :3
# 3
```

```
num = input('숫자입력 :')
print(int(num))
# 숫자입력 :안녕
# ------
# ValueError Traceback (most recent call last)
#       1 num = input('숫자입력 :')
# ----> 2 print(int(num))
# ValueError: invalid literal for int() with
base 10: '안녕'
```

예외 처리 예시

```
try:
    num = input('숫자입력 :')
    print(int(num))
except:
    print('숫자가 아닙니다')
```

예외 처리 예시

```
try:
    num = input('숫자입력 :')
    print(int(num))
except ValueError:
    print('숫자가 아닙니다')
```

HAPPY HACKING

복수의 예외처리 실습

- 100을 사용자가 입력한 값으로 나누고 출력하는 코드를 작성해보시오.
  - 먼저, 발생 가능한 에러가 무엇인지 예상해보시오.

```
num = input('100으로 나눌 값을 입력하시오 : ')
print(100/int(num))
```

HAPPY HACKING

복수의 예외처리 실습

- 100을 사용자가 입력한 값으로 나누고 출력하는 코드를 작성해보시오.
  - 먼저, 발생 가능한 에러가 무엇인지 예상해보시오.

```
num = input('100으로 나눌 값을 입력하시오 : ')
print(100/int(num))
```

문자인 경우? 0인 경우?

HAPPY HACKING

복수의 예외처리 실습

- 100을 사용자가 입력한 값으로 나누고 출력하는 코드를 작성해보시오.
  - 먼저, 발생 가능한 에러가 무엇인지 예상해보시오.

```
try:
    num = input('100으로 나눌 값을 입력하시오: ')
    100/int(num)
except (ValueError, ZeroDivisionError):
    print('제대로 입력해줘')
```

발생 가능한 에러를 모두 명시

HAPPY HACKING

복수의 예외처리 실습

- 100을 사용자가 입력한 값으로 나누고 출력하는 코드를 작성해보시오.
  - 먼저, 발생 가능한 에러가 무엇인지 예상해보시오.

```
try:
    num = input('값을 입력하시오: ')
    100/int(num)
except ValueError:
    print('숫자를 넣어주세요.')
except ZeroDivisionError:
    print('0으로 나눌 수 없습니다.')
except:
    print('에러는 모르지만 에러가 발생하였습니다.')
```

에러 별로 별도의 에러처리

HAPPY HACKING

정리

- try
  - 코드를 실행함
- except
  - try 문에서 예외가 발생 시 실행함
- else
  - try 문에서 예외가 발생하지 않으면 실행함
- finally
  - 예외 발생 여부와 관계없이 항상 실행함

예외처리 예시

- 파일을 열고 읽는 코드를 작성하는 경우
  - 파일 열기 시도
    - 파일 없는 경우 ⇒ ‘해당 파일이 없습니다.’ 출력
    - 파일 있는 경우 ⇒ 파일 내용을 출력
  - 해당 파일 읽기 작업 종료 메시지 출력

HAPPY HACKING

예외처리 예시

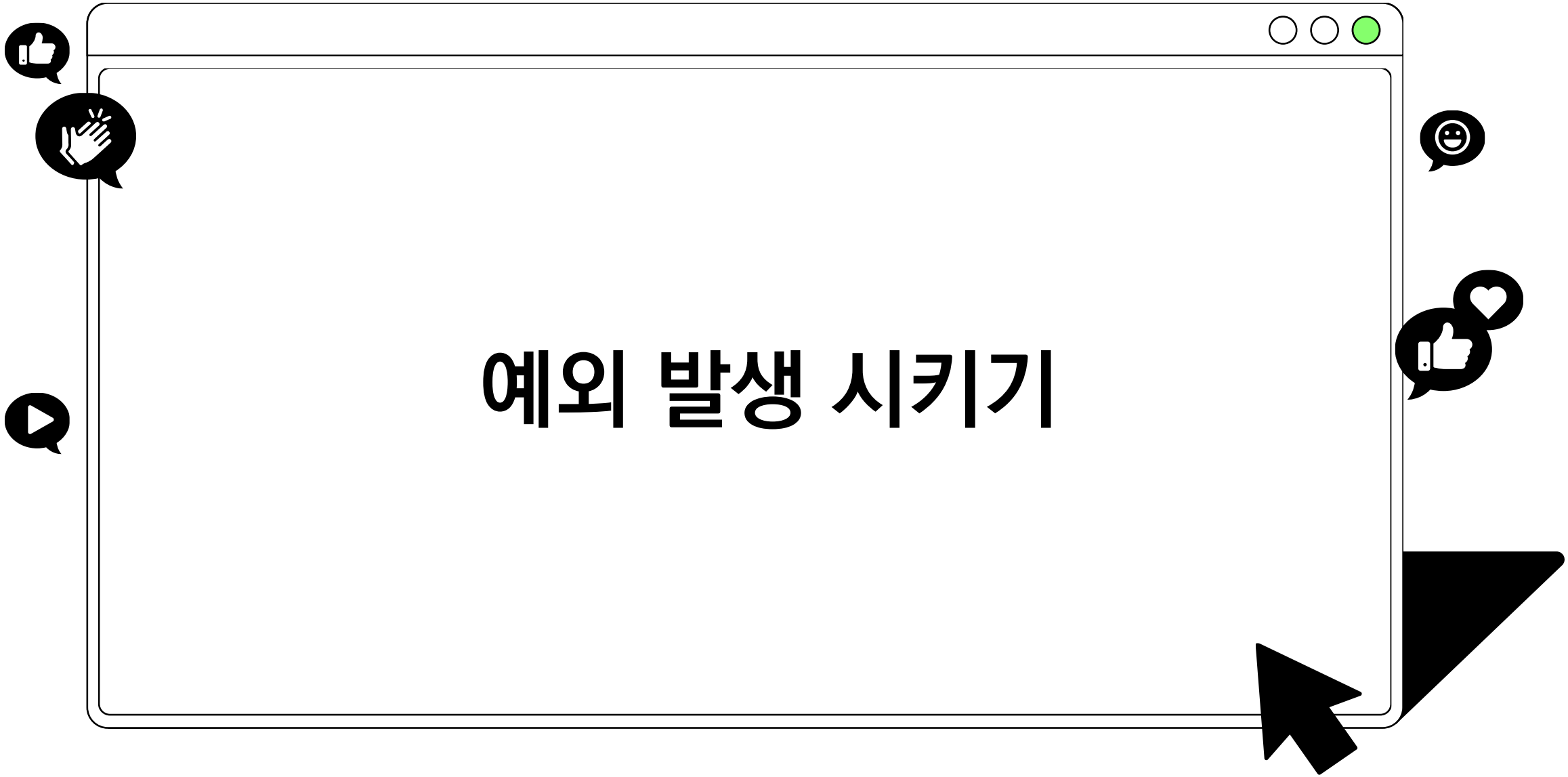
- 파일을 열고 읽는 코드를 작성하는 경우
  - 파일 열기 시도
    - 파일 없는 경우 ⇒ '해당 파일이 없습니다.' 출력 (except)
    - 파일 있는 경우 ⇒ 파일 내용을 출력 (else)
  - 해당 파일 읽기 작업 종료 메시지 출력 (finally)

```
# 파일이 없는 경우
try:
    f = open('nooofile.txt')
except FileNotFoundError:
    print('해당 파일이 없습니다.')
else:
    print('파일을 읽기 시작합니다.')
    print(f.read())
    print('파일을 모두 읽었습니다.')
    f.close()
finally:
    print('파일 읽기를 종료합니다.')
```

```
# 파일이 있는 경우
try:
    f = open('file.txt')
except FileNotFoundError:
    print('해당 파일이 없습니다.')
else:
    print('파일을 읽기 시작합니다.')
    print(f.read())
    print('파일을 모두 읽었습니다.')
    f.close()
finally:
    print('파일 읽기를 종료합니다.')
```

모두 실행

HAPPY HACKING



예외 발생 시키기

HAPPY HACKING

raise statement

- raise를 통해 예외를 강제로 발생

```
raise <표현식>(메시지)
```


예외 타입 지정
(주어지지 않을 경우 현재 스코프에서
활성화된 마지막 예외를 다시 일으킴)

HAPPY HACKING

raise statement

- raise를 통해 예외를 강제로 발생

```
raise
# -------
# RuntimeError Traceback (most recent call last)
# ----> 1 raise
# RuntimeError: No active exception to reraise
```