

파이썬 프로그래밍

12-1: 예외처리

2022.5.17



인하대학교
INHA UNIVERSITY

산업경영공학과 임세실

Overview

- 오류 이해하기
- 예외 처리하기
- 예외의 고급 처리하기



오류 이해하기

- 구문 오류란?
- 예외란?
- 논리 오류란?

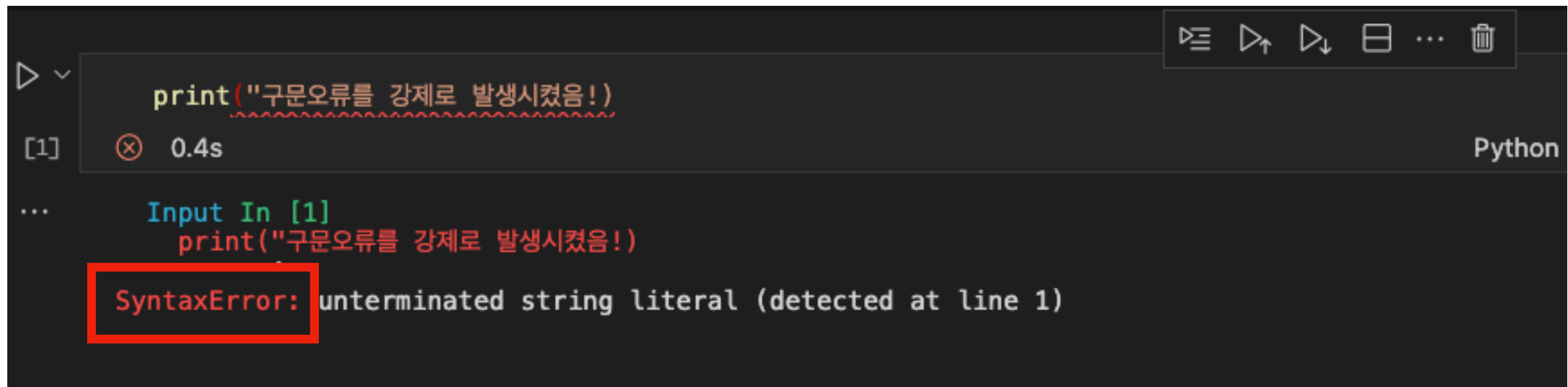


오류(Error)란?

- 프로그램이 올바르게 동작하지 않는 현상
- 오류의 종류
 - 구문 오류 (Syntax error)
 - 예외(Exception) 혹은 런타임 오류 (Runtime error)
 - 논리 오류 (logical error)

구문 오류란?

- 프로그램 실행 전에 발생하는 오류
- 괄호 개수, 들여쓰기, 오타 등 문법 상의 오류
- 해결 안 하면 아예 프로그램 자체가 실행조차 되지 않음



The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there's a toolbar with icons for running, stepping through, and other actions. Below the toolbar, a code cell is shown with the following content:

```
print("구문오류를 강제로 발생시켰음!")
```

The code is highlighted with a red dashed line under the opening quote. Below the code, the output area shows the execution result:

```
[1] × 0.4s Python
```

Below the output, the input prompt is shown:

```
... Input In [1]  
print("구문오류를 강제로 발생시켰음!")
```

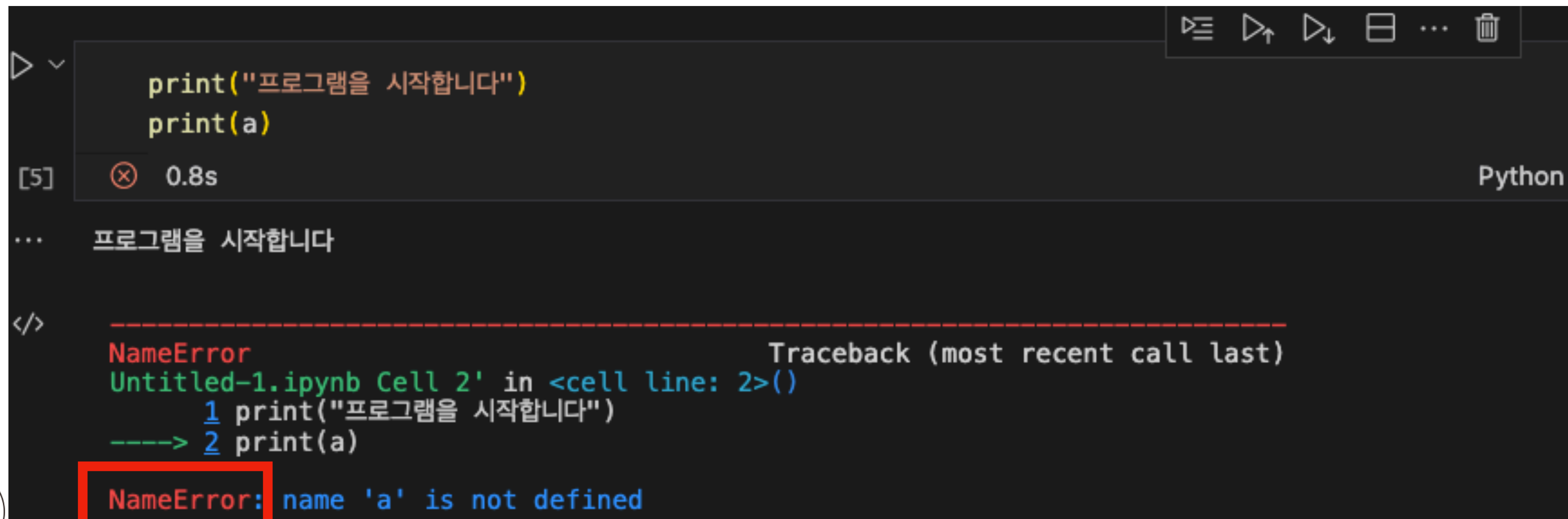
At the bottom, the error message is displayed:

```
SyntaxError: unterminated string literal (detected at line 1)
```

The word "SyntaxError:" is highlighted with a red rectangular box.

예외 혹은 런타임 오류란?

- 프로그램 실행 중에 발생하는 오류
- 실행 중에 비정상적인 값이 발생되어 더 이상 진행이 불가능할 경우 발생
- 예: 없는 변수, 함수, 객체에 접근, 0으로 나눌 때, 변수 타입이 안 맞을 때 등



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with icons for running, stepping through, and other actions. Below the toolbar, the code cell contains two lines of Python code: `print("프로그램을 시작합니다")` and `print(a)`. The output area shows the first line executed successfully, printing "프로그램을 시작합니다". The second line, `print(a)`, has caused an error. The error message is displayed in a red box: `NameError: name 'a' is not defined`. Above the error message, a traceback is shown, indicating the error occurred in 'Untitled-1.ipynb Cell 2' at line 2. The error message is highlighted with a red rectangle.

```
print("프로그램을 시작합니다")
print(a)
```

[5] 0.8s Python

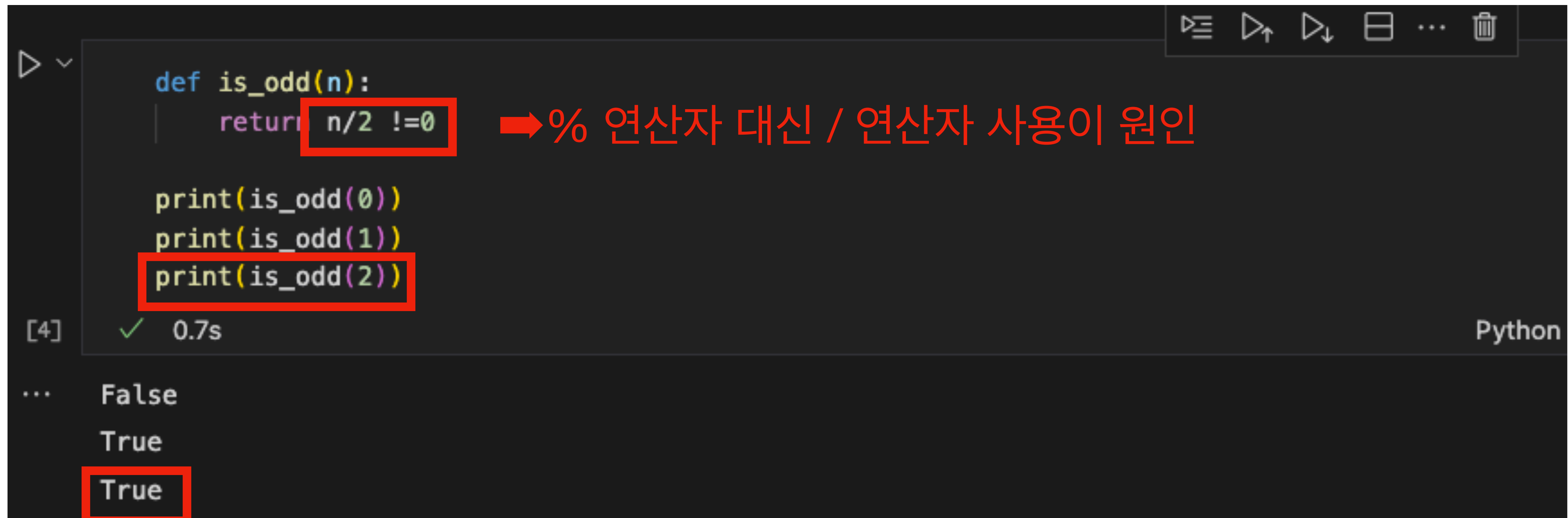
... 프로그램을 시작합니다

</>

```
-----
NameError                                Traceback (most recent call last)
Untitled-1.ipynb Cell 2' in <cell line: 2>()
      1 print("프로그램을 시작합니다")
----> 2 print(a)
NameError: name 'a' is not defined
```

논리 오류란?

- 구문 오류나 예외가 발생하지는 않았지만, 그릇된 결과가 발생한 것
- 알고리즘 설계가 잘못되거나 연산자, 수, 변수 이름 표기 실수가 원인



The screenshot shows a Jupyter Notebook interface with a Python code cell. The code defines a function `is_odd(n)` that returns `n/2 != 0`. Below the function definition, there are three print statements: `print(is_odd(0))`, `print(is_odd(1))`, and `print(is_odd(2))`. The output of the cell is shown below the code, with the results `False`, `True`, and `True`. A red box highlights the `True` result for `is_odd(2)`. A red arrow points from the `n/2` expression in the function definition to the text `⇒ % 연산자 대신 / 연산자 사용이 원인` (Reason: use / operator instead of % operator).

```
def is_odd(n):  
    return n/2 != 0  
  
print(is_odd(0))  
print(is_odd(1))  
print(is_odd(2))
```

⇒ % 연산자 대신 / 연산자 사용이 원인

[4] ✓ 0.7s Python

... False
True
True

오류 고치기

- 코드 검토하기
- 오류 메시지에서 힌트 찾기
- 여러 값으로 테스트해보기
- 선례 찾아보기
 - 스택 오버플로 (www.stackoverflow.com)



예외 처리하기

- 조건문 사용해 예외 처리하기
- try~except 구문 사용하기
- try~except~else 구문 사용하기
- finally 구문 사용하기



예외의 예시 (1)

- 원의 반지름 입력받아 둘레와 넓이 구하기

```
import math
a = int(input("정수를 입력하세요:"))
print("원의 반지름:{}".format(a))
print("원의 둘레:{:.2f}".format(2*a*math.pi))
print("원의 넓이:{:.2f}".format(a*a*math.pi))
```

[12] ✓ 1.3s Python

... 원의 반지름:7
 원의 둘레:43.98
 원의 넓이:153.94

예외의 예시 (2)

- 원의 반지름 입력받아 둘레와 넓이 구하기 (C'd)

```
import math
a = int(input("정수를 입력하세요:"))
print("원의 반지름:{}".format(a))
print("원의 둘레:{:.2f}".format(2*a*math.pi))
print("원의 넓이:{:.2f}".format(a*a*math.pi))
```

[13] 2.7s Python

```
-----
ValueError                                Traceback (most recent call last)
Untitled-1.ipynb Cell 4' in <cell line: 2>()
----> 1 import math
      2 a = int(input("정수를 입력하세요:"))
      3 print("원의 반지름:{}".format(a))
      4 print("원의 둘레:{:.2f}".format(2*a*math.pi))

ValueError: invalid literal for int() with base 10: 'a'
```

예외의 처리

- 예외가 발생했을 때에도 프로그램이 정상적으로 종료되도록 하는 기법
- 예외 처리하는 방법
 - 조건문 사용하기
 - try~except 구문 사용하기

조건문 사용해 예외 처리하기

- 예: 반지름 입력받아 둘레와 넓이 구하기

```
import math
a = input("정수를 입력하세요:")
if a.isdigit():
    int_a = int(a)    ➡ 문제가 없을 경우
    print("원의 반지름:{}".format(int_a))
    print("원의 둘레:{:.2f}".format(2*int_a*math.pi))
    print("원의 넓이:{:.2f}".format(int_a*int_a*math.pi))
else:
    print("정수를 입력하지 않았습니다!") ➡ 문제가 있을 경우
```

7 입력

원의 반지름:7
원의 둘레:43.98
원의 넓이:153.94

abc 입력

정수를 입력하지 않았습니다!

try~except 구문 사용하기 (1)

- 조건문으로 모든 예외를 다 처리할 수 없음
 - 어떤 상황에서 예외 발생하는지 모두 파악하기 힘들
- try~except 구문 사용하면 프로그램 강제종료되는 일 없이 예외처리 가능

try:

오류 발생 가능성이 있는 코드 + 문제 없을 경우 실행될 코드

except:

문제 생겼을 때 실행되는 코드

try~except 구문 사용하기 (2)

- 예: 반지름 입력받아 둘레와 넓이 구하기

```
import math
try:
    a = int(input("정수를 입력하세요:"))
    print("원의 반지름:{}".format(a))
    print("원의 둘레:{:.2f}".format(2*a*math.pi))
    print("원의 넓이:{:.2f}".format(a*a*math.pi))
except:
    print("예외가 발생했습니다!")
```

7 입력

원의 반지름:7
원의 둘레:43.98
원의 넓이:153.94

abc 입력

예외가 발생했습니다!

try~except~else 구문 사용하기 (1)

- 문제없을 때 실행할 코드를 else에 작성
- 예외가 발생할 가능성이 있는 코드를 확실히 한 두줄로 좁힐 수 있을 때, 그 부분을 분리해서 try에 작성

try:

오류 발생 가능성이 있는 코드

except:

문제 생겼을 때 실행되는 코드

else:

문제 없을 때 실행될 코드

try~except~else 구문 사용하기 (2)

- 예: 반지름 입력받아 둘레와 넓이 구하기

```
import math
try:
    a = int(input("정수를 입력하세요:"))
except:
    print("예외가 발생했습니다!")
else:
    print("원의 반지름:{}".format(a))
    print("원의 둘레:{:.2f}".format(2*a*math.pi))
    print("원의 넓이:{:.2f}".format(a*a*math.pi))
```

7 입력

원의 반지름:7
원의 둘레:43.98
원의 넓이:153.94

abc 입력

예외가 발생했습니다!



finally 구문 사용하기 (1)

- 예외가 있든 없든 무조건 가장 마지막에 실행되는 구문

try:

오류 발생 가능성이 있는 코드

except:

문제 생겼을 때 실행되는 코드

else:

문제 없을 때 실행될 코드

finally:

무조건 실행되는 코드

finally 구문 사용하기 (2)

- 예: 반지름 입력받아 둘레와 넓이 구하기

```
import math
try:
    a = int(input("정수를 입력하세요:"))
except:
    print("예외가 발생했습니다!")
else:
    print("원의 반지름:{}".format(a))
    print("원의 둘레:{:.2f}".format(2*a*math.pi))
    print("원의 넓이:{:.2f}".format(a*a*math.pi))
finally:
    print("프로그램이 종료됩니다!")
```

7 입력

원의 반지름:7
원의 둘레:43.98
원의 넓이:153.94
프로그램이 종료됩니다!

abc 입력

예외가 발생했습니다!
프로그램이 종료됩니다!

finally 구문 사용하기 (3)

```
print("프로그램이 시작되었습니다")

while True:
    try:
        print("try 구문이 시작되었습니다")
        break
        print("try의 break 다음입니다")
    except:
        print("예외 상황입니다")
    finally:
        print("finally 구문이 실행되었습니다")
        print("while 반복문의 마지막 줄입니다")
print("프로그램이 종료되었습니다")
```

[27] ✓ 0.3s

... 프로그램이 시작되었습니다
try 구문이 시작되었습니다
finally 구문이 실행되었습니다
프로그램이 종료되었습니다

참고: 함수와 try~except 사용

- 예: 반지름 입력받아 둘레와 넓이 구하기

```
import math
def circle(a):
    return 2*a*math.pi, a*a*math.pi

try:
    a = int(input("정수를 입력하세요:"))
except:
    print("예외가 발생했습니다!")
else:
    print("원의 반지름:{}".format(a))
    print("원의 둘레:{:.2f}".format(circle(a)[0]))
    print("원의 넓이:{:.2f}".format(circle(a)[1]))
finally:
    print("프로그램을 종료합니다!")
```

예외의 고급 처리하기

- 예외 객체 사용하기
- 예외 강제로 발생시키기



예외 객체란?

- 예외 상황 발생 시 관련 정보가 객체에 저장
 - 예외 상황의 종류, 관련 오류 메시지, 오류 발생 위치 등
- 예외 객체를 만드는 클래스는 매우 다양
 - 예: IndexError (리스트 인덱스 오류), ZeroDivisionError (0으로 나누기 오류) 등
- 모든 예외 클래스는 Exception 클래스로부터 상속받음

예외 객체 사용하기 (1)

- 예외 상황 발생 시 생성된 객체를 변수에 저장

try:

오류 발생 가능성이 있는 코드 + 문제 없을 경우 실행될 코드

except 예외의 종류 as 변수 이름:

문제 생겼을 때 실행되는 코드

예외 객체 사용하기 (2)

- 리스트 인덱스 범위 벗어난 예외 상황에 대한 객체 생성

```
list1=[1,2,3]
try:
    i = int(input())
    print(list1[i])
except IndexError as err:
    print("type(err):", type(err))
    print("err:",err)
```

[29] ✓ 1.5s

... type(err): <class 'IndexError'>
err: list index out of range

예외 객체 사용하기 (3)

- 예외 구분해서 실행 다르게 하기

```
list1=[1,2,3]
try:
    i = int(input("정수 입력!"))
    print(list1[i])
except IndexError as err:
    print("리스트의 인덱스가 벗어났습니다!")
    print("err:",err)
except ValueError as err:
    print("정수를 입력해주세요!")
    print("err:",err)
```

5 입력

리스트의 인덱스가 벗어났습니다!
err: list index out of range

a 입력

정수를 입력해주세요!
err: invalid literal for int() with base 10: 'a'

예외 객체 사용하기 (4)

- 객체 사용 없이 메시지만 출력할 경우

```
list1=[1,2,3]
try:
    i = int(input("정수 입력!"))
    print(list1[i])
except IndexError:
    print("리스트의 인덱스가 벗어났습니다!")
except ValueError:
    print("정수를 입력해주세요!")
```

5 입력

리스트의 인덱스가 벗어났습니다!

a 입력

정수를 입력해주세요!

예외 객체 사용하기 (5)

- 모든 예외 다 포함하기: Exception 클래스

```
list1=[1,2,3]
try:
    i = int(input("정수 입력!"))
    print(list1[i])
    print(k)
except IndexError:
    print("리스트의 인덱스가 벗어났습니다!")
    print("err:",err)
except ValueError as err:
    print("정수를 입력해주세요!")
    print("err:",err)
except Exception as err:
    print("그외 미리 파악 못한 예외가 발생했습니다")
    print("type(err)",type(err))
    print("err:",err)
```

[44] ✓ 1.4s

... 2

```
그외 미리 파악 못한 예외가 발생했습니다
type(err) <class 'NameError'>
err: name 'k' is not defined
```

예외 강제로 발생시키기

- 아직 구현되지 않은 부분이어서 강제로 예외 발생시켜 종료시키고 싶을 때

```
▶ a = int(input("정수입력"))

if a>0:
    #아직 미구현
    raise NotImplementedError
else:
    #아직 미구현
    raise NotImplementedError

[47] (x) 2.8s

...
-----
NotImplementedError                                Traceback (most recent call last)
Untitled-1.ipynb Cell 12' in <cell line: 3>()
      1 a = int(input("정수입력"))
      3 if a>0:
      4     #아직 미구현
----> 5     raise NotImplementedError
      6 else:
      7     #아직 미구현
      8     raise NotImplementedError

NotImplementedError:
```


실습 (5/19)

- 강의실에서 출석 확인 예정
 - 온라인 출석확인 대상자는 과제 제출 여부로 출석 확인
- 실습과제는 실습시간 정시에 iClass에 업로드될 예정
- 5월 20일 자정까지 제출해야 과제 점수 인정

