

# 구문 오류와 예외

- 오류(error)와 예외

오류

Traceback (most recent call last):

File "test.py", line 16, in <module>

print(fibonacci(10))

File "test.py", line 6, in fibonacci

counter += 1

UnboundLocalError: local variable 'counter' referenced before assignment

# 오류의 종류

- 오류 (error)
  - 구문 오류 (syntax error)
    - 프로그램 실행 전에 발생하는 오류
  - 런타임 오류 (runtime error) / 예외 (exception)
    - 프로그램 실행 중에 발생하는 오류
- 구문 오류

## 구문 오류가 발생하는 코드

# 프로그램 시작

```
print("# 프로그램이 시작되었습니다!")
```

# 구문 오류 발생 코드

```
print("# 예외를 강제로 발생시켜 볼게요!")
```

→ 닫는 따옴표로 문자열을 닫지 않았습니다.

# 오류의 종류

## ❗ 오류

SyntaxError: EOL while scanning string literal

- **SyntaxError**
  - 구문에 문제가 있어 프로그램 실행부터 불가능한 경우

## 구문 오류 해결

# 프로그램 시작

```
print("# 프로그램이 시작되었습니다!")
```

# 구문 오류 발생 코드

```
print("# 예외를 강제로 발생시켜 볼게요!")
```

→ 닫는 따옴표로 문자열을 닫아서 해결합니다.

# 오류의 종류

- 예외 / 런타임 오류
  - 실행 중에 발생하는 오류

## 예외가 발생하는 코드

```
# 프로그램 시작
print("# 프로그램이 시작되었습니다!")

# 예외 발생 코드
list_a[1]
```

# 프로그램이 시작되었습니다! → 여기까지는 프로그램이 정상으로 실행되었다는 것을 확인할 수 있습니다.

Traceback (most recent call last):

File "test.py", line 5, in <module>

list\_a[1]

**NameError: name 'list\_a' is not defined**

# 기본 예외 처리

- 예외 처리 (exception handling)
  - 조건문을 사용하는 방법
    - 기본 예외 처리
  - try 구문을 사용하는 방법
- 예외 상황 확인하기

예외가 발생할 수 있는 코드

```
# 숫자를 입력받습니다.  
number_input_a = int(input("정수 입력> "))  
  
# 출력합니다.  
print("원의 반지름:", number_input_a)  
print("원의 둘레:", 2 * 3.14 * number_input_a)  
print("원의 넓이:", 3.14 * number_input_a * number_input_a)
```

# 기본 예외 처리

- 정수를 입력하지 않았을 경우

정수 입력> 7센티미터  Enter → 정수로 변환할 수 없는 문자열을 입력했습니다.

Traceback (most recent call last):

File "test.py", line 2, in <module>

number\_input\_a = int(input("정수 입력> "))

ValueError: invalid literal for int() with base 10: '7센티미터'

# 기본 예외 처리

- 조건문으로 예외 처리하기
  - 위 슬라이드의 경우

isdigit() 함수 사용하여 숫자로만 구성된 글자인지 확인

---

```
01  # 숫자를 입력받습니다.
02  user_input_a = input("정수 입력> ")
03
04  # 사용자 입력이 숫자로만 구성되어 있을 때
05  if user_input_a.isdigit():
06      # 숫자로 변환합니다.
07      number_input_a = int(user_input_a)
08      # 출력합니다.
09      print("원의 반지름:", number_input_a)
10      print("원의 둘레:", 2 * 3.14 * number_input_a)
11      print("원의 넓이:", 3.14 * number_input_a * number_input_a)
12  else:
13      print("정수를 입력하지 않았습니다.")
```

---



# 기본 예외 처리

- 정수 입력하면 정상적인 값 출력

정수 입력> 8

원의 반지름: 8

원의 둘레: 50.24

원의 넓이: 200.96

- 정수로 변환할 수 없는 문자열 입력하는 경우

정수 입력> yes!!

정수를 입력하지 않았습니다.

# try except 구문

- try except 구문

- 예외 처리할 수 있는 구문

```
try:
```

예외가 발생할 가능성이 있는 코드

```
except:
```

예외가 발생했을 때 실행할 코드

- 어떤 상황에 예외가 발생하는지 완벽하게 이해하고 있지 않아도 프로그램이 강제로 죽어버리는 상황은 막을 수 있음

# try except 구문

- 예시

```
01  # try except 구문으로 예외를 처리합니다.  
02  try:  
03      # 숫자로 변환합니다.  
04      number_input_a = int(input("정수 입력> ")) → 예외가 발생할 가능성이 있는 구문  
05      # 출력합니다.  
06      print("원의 반지름:", number_input_a)  
07      print("원의 둘레:", 2 * 3.14 * number_input_a)  
08      print("원의 넓이:", 3.14 * number_input_a * number_input_a)  
09  except:  
10      print("무언가 잘못되었습니다.") → 예외가 발생했을 때 실행할 구문
```

정수 입력> yes!!

무언가 잘못되었습니다.


# try except 구문

- try except 구문과 pass 키워드 조합하기
  - 예외가 발생하면 일단 처리해야 하지만, 해당 코드가 딱히 중요한 부분이 아닌 경우 프로그램 강제 종료부터 막는 목적으로 except 구문에 아무 것도 넣지 않고 try 구문 사용
  - pass 키워드를 빈 except 구문에 넣음

```
try:  
    예외가 발생할 가능성이 있는 코드  
except:  
    pass
```

- 예시 – 숫자로 변환되는 것들만 리스트에 넣기

```
01  # 변수를 선언합니다.
02  list_input_a = ["52", "273", "32", "스파이", "103"]
03
04  # 반복을 적용합니다.
05  list_number = []
06  for item in list_input_a:
07      # 숫자로 변환해서 리스트에 추가합니다.
08      try:
09          float(item) # 예외가 발생하면 알아서 다음으로 진행은 안 되겠지?
10          list_number.append(item) # 예외 없이 통과했으면 리스트에 넣어줘!
11      except:
12          pass
13
14  # 출력합니다.
15  print("{} 내부에 있는 숫자는".format(list_input_a))
16  print("{}입니다.".format(list_number))
```

 실행결과

['52', '273', '32', '스파이', '103'] 내부에 있는 숫자는  
['52', '273', '32', '103']입니다.

# try except else 구문

- try except 구문 뒤에 else 구문 붙여 사용하면  
예외가 발생하지 않았을 때 실행할 코드 지정할 수 있음

```
try:  
    예외가 발생할 가능성이 있는 코드  
except:  
    예외가 발생했을 때 실행할 코드  
else:  
    예외가 발생하지 않았을 때 실행할 코드
```

- 이 때, 예외 발생 가능성 있는 코드만 try 구문 내부에 넣고  
나머지는 모두 else 구문으로 빼는 경우 많음

# try except else 구문

- 예시

```
01  # try except else 구문으로 예외를 처리합니다.  
02  try:  
03      # 숫자로 변환합니다.  
04      number_input_a = int(input("정수 입력> "))  
05  except:  
06      print("정수를 입력하지 않았습니다.")  
07  else:  
08      # 출력합니다.  
09      print("원의 반지름:", number_input_a)  
10      print("원의 둘레:", 2 * 3.14 * number_input_a)  
11      print("원의 넓이:", 3.14 * number_input_a * number_input_a)
```

실행결과 1

정수 입력> 7

원의 반지름: 7

원의 둘레: 43.96

원의 넓이: 153.86

실행결과 2

정수 입력> yes!!

정수를 입력하지 않았습니다.

# finally 구문

- finally 구문

- 예외 처리 구문에서 가장 마지막에 사용할 수 있는 구문
- 예외 발생 여부와 관계없이 무조건 실행할 경우 사용

```
try:
```

```
    예외가 발생할 가능성이 있는 코드
```

```
except:
```

```
    예외가 발생했을 때 실행할 코드
```

```
else:
```

```
    예외가 발생하지 않았을 때 실행할 코드
```

```
finally:
```

```
    무조건 실행할 코드
```

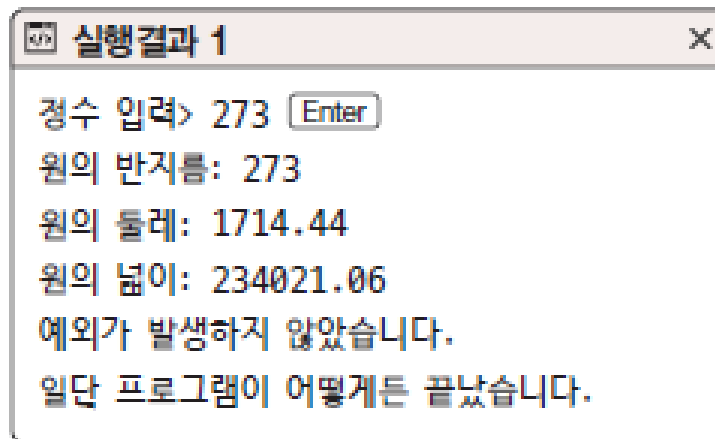


# finally 구문

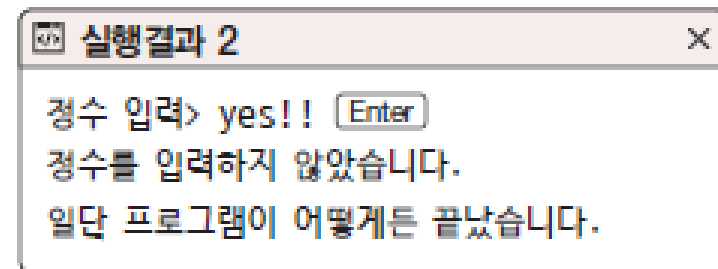
- finally 구문

---

```
01  # try except 구문으로 예외를 처리합니다.
02  try:
03      # 숫자로 변환합니다.
04      number_input_a = int(input("정수 입력> "))
05      # 출력합니다.
06      print("원의 반지름:", number_input_a)
07      print("원의 둘레:", 2 * 3.14 * number_input_a)
08      print("원의 넓이:", 3.14 * number_input_a * number_input_a)
09  except:
10      print("정수를 입력해달라고 했잖아요?!")
11  else:
12      print("예외가 발생하지 않았습니다.")
13  finally:
14      print("일단 프로그램이 어떻게든 끝났습니다.")
```



```
실행결과 1
정수 입력> 273 [Enter]
원의 반지름: 273
원의 둘레: 1714.44
원의 넓이: 234021.06
예외가 발생하지 않았습니다.
일단 프로그램이 어떻게든 끝났습니다.
```



```
실행결과 2
정수 입력> yes!! [Enter]
정수를 입력하지 않았습니다.
일단 프로그램이 어떻게든 끝났습니다.
```


- try, except, finally 구문의 조합
  - try 구문은 단독으로 사용할 수 없으며,  
반드시 except 구문 또는 finally 구문과 함께 사용해야 함
  - else 구문은 반드시 except 구문 뒤에 사용해야 함

- try + except 구문 조합
- try + except + else 구문 조합
- try + except + finally 구문 조합
- try + except + else + finally 구문 조합
- try + except 구문 조합

- 오류 경우

## try + else 구문 조합

```
# try except 구문으로 예외를 처리합니다.  
try:  
    # 숫자로 변환합니다.  
    number_input_a = int(input("정수 입력> "))  
    # 출력합니다.  
    print("원의 반지름:", number_input_a)  
    print("원의 둘레:", 2 * 3.14 * number_input_a)  
    print("원의 넓이:", 3.14 * number_input_a * number_input_a)  
else:  
    print("프로그램이 정상적으로 종료되었습니다.")
```

 오류

SyntaxError: Invalid syntax

# finally 구문

- try 구문 내부에서 return 키워드를 사용하는 경우

```
01  # test() 함수를 선언합니다.
02  def test():
03      print("test() 함수의 첫 줄입니다.")
04      try:
05          print("try 구문이 실행되었습니다.")
06          return
07          print("try 구문의 return 키워드 뒤입니다.")
08      except:
09          print("except 구문이 실행되었습니다.")
10      else:
11          print("else 구문이 실행되었습니다.")
12      finally:
13          print("finally 구문이 실행되었습니다.")
14      print("test() 함수의 마지막 줄입니다.")
15
16  # test() 함수를 호출합니다.
17  test()
```

finally 구문은 무조건 실행됩니다. ←

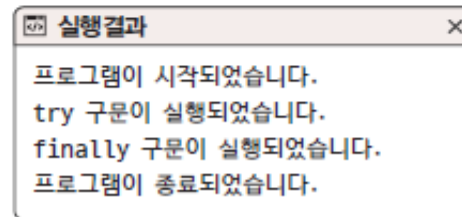
**실행결과**

test() 함수의 첫 줄입니다.  
try 구문이 실행되었습니다.  
finally 구문이 실행되었습니다.

# finally 구문

- 반복문과 함께 사용하는 경우

```
01 print("프로그램이 시작되었습니다.")
02
03 while True:
04     try:
05         print("try 구문이 실행되었습니다.")
06         break
07         print("try 구문의 break 키워드 뒤입니다.")
08     except:
09         print("except 구문이 실행되었습니다.")
10     finally:
11         print("finally 구문이 실행되었습니다.")
12         print("while 반복문의 마지막 줄입니다.")
13 print("프로그램이 종료되었습니다.")
```



실행결과

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

- break 키워드로 try 구문 전체 빠져나가도 finally 구문 실행

- 예외 객체 (exception object)

- 예외 발생 시 예외 정보가 저장되는 곳

```
try:
```

예외가 발생할 가능성이 있는 구문

```
except 예외의 종류 as 예외 객체를 활용할 변수 이름:
```

예외가 발생했을 때 실행할 구문

# 예외 객체

- Exception
  - 모든 예외의 상위 예외 객체
  - 예시

---

```
01  # try except 구문으로 예외를 처리합니다.
02  try:
03      # 숫자로 변환합니다.
04      number_input_a = int(input("정수 입력> "))
05      # 출력합니다.
06      print("원의 반지름:", number_input_a)
07      print("원의 둘레:", 2 * 3.14 * number_input_a)
08      print("원의 넓이:", 3.14 * number_input_a * number_input_a)
09  except Exception as exception:
10      # 예외 객체를 출력해봅니다.
11      print("type(exception):", type(exception))
12      print("exception:", exception)
```

---



# 예외 객체

정수 입력> yes!!

```
type(exception): <class 'ValueError'>
```

```
exception: invalid literal for int() with base 10: 'yes!!'
```

- 다양한 예외들이 발생할 때 그 정보를 메일 등으로 보내도록 해서 수집하면 큰 규모의 웹서비스 등에서 프로그램 개선에 큰 도움이 됨

# 예외 구분하기

- 예외 구분하기

- except 구문 뒤에 예외 종류 입력해서 구분할 수 있음

```
try:
```

```
    예외가 발생할 가능성이 있는 구문
```

```
except 예외의 종류A:
```

```
    예외A가 발생했을 때 실행할 구문
```

```
except 예외의 종류B:
```

```
    예외B가 발생했을 때 실행할 구문
```

```
except 예외의 종류C:
```

```
    예외C가 발생했을 때 실행할 구문
```

# 예외 구분하기

- 예시 – ValueError와 IndexError

---

```
01  # 변수를 선언합니다.
02  list_number = [52, 273, 32, 72, 100]
03
04  # try except 구문으로 예외를 처리합니다.
05  try:
06      # 숫자를 입력받습니다.
07      number_input = int(input("정수 입력> "))
08      # 리스트의 요소를 출력합니다.
09      print("{}번째 요소: {}".format(number_input, list_number[number_input]))
10  except ValueError:
11      # ValueError가 발생하는 경우
12      print("정수를 입력해 주세요!")
13  except IndexError:
14      # IndexError가 발생하는 경우
15      print("리스트의 인덱스를 벗어났어요!")
```

---

# 예외 구분하기

- 정수 아닌 값 입력해 ValueError 발생시키는 경우

정수 입력> yes!!

정수를 입력해 주세요!

- 리스트의 인덱스를 넘는 숫자 입력해 IndexError인 경우

정수 입력> 100

리스트의 인덱스를 벗어났어요!

# 예외 구분하기

- 예외 구분 구문과 예외 객체
  - as 키워드 사용
  - 각각의 except 구문 뒤에 예외 객체 붙여 예외 구분에 활용

---

```
01  # 변수를 선언합니다.
02  list_number = [52, 273, 32, 72, 100]
03
04  # try except 구문으로 예외를 처리합니다.
05  try:
06      # 숫자를 입력 받습니다.
07      number_input = int(input("정수 입력> "))
08      # 리스트의 요소를 출력합니다.
09      print("{}번째 요소: {}".format(number_input, list_number[number_input]))
10  except ValueError as exception:
11      # ValueError가 발생하는 경우
12      print("정수를 입력해 주세요!")
13      print("exception:", exception)
14  except IndexError as exception:
15      # IndexError가 발생하는 경우
16      print("리스트의 인덱스를 벗어났어요!")
17      print("exception:", exception)
```

---

- raise 키워드

- 예외를 강제로 발생시킴
- 프로그램 개발 단계에서 아직 구현되지 않은 부분에 일부러 예외를 발생시켜 잊어버리지 않도록 함

raise 예외 객체

```
# 입력을 받습니다.  
number = input("정수 입력> ")  
number = int(number)  
  
# 조건문 사용  
if number > 0:  
    # 양수일 때: 아직 미구현 상태입니다.  
    raise NotImplementedError  
else:  
    # 음수일 때: 아직 미구현 상태입니다.  
    raise NotImplementedError
```

# 정리합시다

- 오류의 종류
- 기본 예외 처리
- try except 구문
- try except else 구문
- finally 구문
- 예외 객체
- 예외 구분하기
- raise 구문