

Errors in Python

안녕하세요. 컴퓨팅기초 튜터 정성태입니다.

실습 시간마다 수강생 여러분들께서 다양한 질문을 해주시는데요. 그중 대부분이 에러가 발생했는데 어떻게 대처해야 할지 잘 모르겠다는 질문인 것 같습니다.

그래서 Python으로 프로그래밍을 할 때 자주 만나게 되는 에러에는 어떤 것들이 있으며, 각각의 경우에 어떻게 대처하면 되는지를 간단히 설명해드리려고 합니다.

에러 메시지를 읽어보세요

`NameError` — 그런 이름 몰라요

`ModuleNotFoundError` — 그런 모듈 몰라요

`TypeError` — type이 틀렸어요

`ZeroDivisionError` — 0으로는 못 나뉘요

`ValueError` — 내가 원하는 값이 아니에요

`SyntaxError` — 무슨 말인지 아예 모르겠어요

에러 메시지를 읽어보세요

C를 비롯한 다른 프로그래밍 언어를 배워보신 분이라면 이미 느끼셨을 테지만, Python의 에러 메시지는 굉장히 친절한 편입니다. 대부분의 경우에 에러 메시지를 잘 읽어보기만 해도 작성한 코드의 어느 부분이 어떻게 잘못된 것인지를 쉽게 알아낼 수 있습니다. 에러 메시지가 영어로 쓰여 있어서 여태 읽을 마음이 안 들었을 수도 있지만, 다음에 또 에러를 만나게 된다면 가장 먼저 에러 메시지를 천천히 읽어보세요. 그 안에 답이 들어 있습니다.

`NameError` — 그런 이름 몰라요

```
In [1]: print(num)
-----
NameError                                Traceback (most recent call last)
Input In [1], in <cell line: 1>()
----> 1 print(num)

NameError: name 'num' is not defined
```

정의되지 않은 이름을 사용하려고 할 때 `NameError`가 발생합니다. 사진의 예시에서는 `num`이라는 변수를 선언하지 않은 채로 그 값을 `print`하려고 해서 에러가 발생했습니다. 메시지를 보면 “num이라는 이름이 정의되지 않았다”라고 알려주고 있죠.

NameError를 만나면 변수를 미리 선언해두는 것을 잊어버리지 않았는지, 혹은 변수 이름에 오타가 있지 않은지 확인해보세요.

```
In [1]: num = random.randint(3, 5)
-----
NameError                                Traceback (most recent call last)
Input In [1], in <cell line: 1>()
----> 1 num = random.randint(3, 5)

NameError: name 'random' is not defined
```

수업 시간에 `random` 모듈을 사용하는 방법에 관해서 배웠는데요. 모듈을 사용하기 전에는 반드시 `import random` 이라는 문장을 써서 모듈을 가져와야 합니다. 만약 **import** 를 빠뜨리면 Python이 `random` 이라는 이름을 알아보지 못해서 NameError가 발생하게 됩니다.

ModuleNotFoundError — 그런 모듈 몰라요

```
In [2]: import hello
-----
ModuleNotFoundError                      Traceback (most recent call last)
Input In [2], in <cell line: 1>()
----> 1 import hello

ModuleNotFoundError: No module named 'hello'
```

모듈을 **import** 할 때 존재하지 않는 이름을 쓰게 되면 ModuleNotFoundError가 발생하게 됩니다. NameError와 비슷하게 모듈 이름에 오타가 있기 때문일 가능성이 높겠네요.

TypeError — type이 틀렸어요

```
In [2]: num = 3
In [3]: string = "Hi"
In [4]: print(num + string)
-----
TypeError                                Traceback (most recent call last)
Input In [4], in <cell line: 1>()
----> 1 print(num + string)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

수업 시간에 `int`, `float`, `str` 등 다양한 자료형(type)에 관해 배웠는데요. 각각의 자료형은 각자 알맞은 활용 방식이 정해져 있습니다. 예를 들어 `int` 와 `float` 은 서로 사칙연산이 가능하고, `str` 는 `str` 끼리 더할 수 있습니다.

그런데 이런 규칙을 어기고 변수들을 적절하지 못한 방식으로 사용하려고 하면 `TypeError`가 발생하게 됩니다. 사진의 예시에서는 `int` 와 `str` 변수를 더하려고 하고 있는데요. 이런 연산은 Python 내부에 정의되어 있지 않기 때문에 type이 잘못되었다는 에러가 발생하게 됩니다.

에러 메시지를 보면 ‘operand’라는 말이 나오는데요. 우리는 이미 `+`, `-`, `*` 등 다양한 연산자(operator)를 알고 있습니다. operand는 우리말로 ‘피연산자’ 혹은 ‘논항’이라고 번역되는데요. 예를 들어 `3 + 5` 라는 수식에서는 3과 5가 각각 연산자 `+` 의 operand가 됩니다. 이제 에러 메시지를 다시 보면, “연산자 `+` 의 operand로 `int` 와 `str` type의 값을 쓰는 건 지원되지 않는다”, 즉 “`int` 와 `str` 를 더할 수 없다”라는 뜻을 알 수 있죠.

```
In [5]: num = input("number: ")
number: 3

In [6]: print(num + 7)
-----
TypeError                                 Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 print(num + 7)

TypeError: can only concatenate str (not "int") to str
```

수업 시간에 배운 `input()` 함수를 사용해 얻은 값은 항상 `str` type을 잊어버리고 바로 수와 더하려고 할 때도 `TypeError`를 만나게 되는데요. `str` 와 `int` 를 더하는 것이 적절한 연산이 아니기 때문입니다. 메시지를 보면 “`str` 에는 `str` 만 이어붙일 수 있고 `int` 는 안 된다”라고 알려주고 있네요.

ZeroDivisionError — 0으로는 못 나눕니다

```
In [7]: zero = 0

In [8]: print(3 / zero)
-----
ZeroDivisionError                         Traceback (most recent call last)
Input In [8], in <cell line: 1>()
----> 1 print(3 / zero)

ZeroDivisionError: division by zero
```

임의의 수를 0으로 나누려고 하면 ZeroDivisionError가 발생하게 됩니다. / 뿐 아니라 // 와 % 연산자도 나눗셈을 수행하기 때문에, 두 번째 operand로 0을 쓰지 않도록 항상 주의해야 합니다.

ValueError — 내가 원하는 값이 아니에요

```
In [6]: num = int(input("number: "))
number: 4.55

-----
ValueError                                Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 num = int(input("number: "))

ValueError: invalid literal for int() with base 10: '4.55'
```

앞서 소개한 TypeError는 변수의 type이 잘못되었을 때 발생하는 에러였는데요. 그에 비해 ValueError는 변수의 값 자체가 잘못되었을 때 발생합니다.


사진의 예시에서는 input() 을 통해 입력받은 값을 int type으로 변환하려고 하고 있습니다. 그런데 입력이 4.55로 주어져서 변수 num 에는 "4.55" 라는 str 값이 저장되어버립니다. 이런 경우 이 값을 int 로 변환하는 것이 불가능하겠죠. 그래서 ValueError가 발생하게 됩니다. 에러 메시지를 보면 'literal'이라는 말이 나오는데요. literal은 또 하나의 새로운 개념이지만, 여기서는 그냥 문자열을 가리키는 것이라고 생각하시면 되겠습니다. 그래서 메시지는 "'4.5'라는 문자열은 int() 를 통해 10진수로 바꿀 수 없다"라는 뜻이 됩니다.

SyntaxError — 무슨 말인지 아예 모르겠어요

```
In [7]: num = 10

In [8]: print(f"My number is {num}")
Input In [8]
        print(f"My number is {num}")
              ^
SyntaxError: EOL while scanning string literal
```

```
In [9]: num =
      Input In [9]
      num =
          ^
SyntaxError: invalid syntax
```

여기서 **syntax**는 ‘문법’이라는 뜻입니다. **Python**의 가장 기본적인 문법 규칙을 아예 어겨버려서 코드가 무슨 의미인지를 해석하는 것이 불가능할 때 **SyntaxError**가 발생합니다. 괄호를 열고 나서 닫지 않거나, 문자열의 끝에 따옴표를 쓰는 것을 잊어버리거나, 값을 써야 할 자리에 아무것도 쓰지 않거나... 등의 다양한 경우가 여기에 해당됩니다. 최근 버전의 **Python**은 **SyntaxError**의 에러 메시지에  기호로 어디가 틀렸는지를 표시해줍니다.

이밖에도 **Python**에는 아주 다양한 종류의 에러가 존재하는데요. 처음에 언급했듯이, 대부분의 경우 에러 메시지를 잘 읽기만 해도 큰 어려움 없이 해결할 수 있습니다. **에러 메시지가 중요한 힌트가 된다는 점을 늘 잊지 말아주세요.**

더 궁금한 것이 있다면 언제든지 질문해주세요!