Commonwealth Exceptions

Python 2.x 2.0–2.7 , Python 3.x 3.0–3.6

Here in Stack Overflow we often see duplicates talking about the same errors: "ImportError: No module named '??????' , SyntaxError: invalid syntax or NameError: name '???' is not defined . This is an effort to reduce them and to have some documentation to link to.

Examples

Other Errors

AssertError

The assert statement exists in almost every programming language. When you do:

```
assert condition
```

or:

```
assert condition, message
```

It's equivalent to this:

```
if __debug__:
    if not condition: raise AssertionError(message)
```

Assertions can include an optional message, and you can disable them when you're done debugging.

Note : the built-in variable **debug** is True under normal circumstances, False when optimization is requested (command line option -O). Assignments to **debug** are illegal. The value for the built-in variable is determined when the interpreter starts.

KeyboardInterrupt

Error raised when the user presses the interrupt key, normally Ctrl + C or del .

ZeroDivisionError

You tried to calculate 1/0 which is undefined. See this example to find the divisors of a number:

Python 2.x ≥ 2.0 , ≤ 2.7

```
div = float(raw_input("Divisors of: "))
for x in xrange(div+1): #includes the number itself and zero
    if div/x == div//x:
        print x, "is a divisor of", div
```

Python 3.x ≥ 3.0

```
div = int(input("Divisors of: "))
for x in range(div+1): #includes the number itself and zero
    if div/x == div//x:
        print(x, "is a divisor of", div)
```

It raises ZeroDivisionError because the for loop assigns that value to x . Instead it should be:

Python 2.x ≥ 2.0 , ≤ 2.7

```
div = float(raw_input("Divisors of: "))
for x in xrange(1,div+1): #includes the number itself but not zero
    if div/x == div//x:
        print x, "is a divisor of", div
```

Python 3.x ≥ 3.0

```
div = int(input("Divisors of: "))
for x in range(1,div+1): #includes the number itself but not zero
    if div/x == div//x:
        print(x, "is a divisor of", div)
```

NameError: name '???' is not defined

Is raised when you tried to use a variable, method or function that is not initialized (at least not before). In other words, it is raised when a requested local or global name is not found. It's possible that you misspelt the name of the object or forgot to import something. Also maybe it's in another scope. We'll cover those with separate examples.

It's simply not defined nowhere in the code

It's possible that you forgot to initialize it, specially if it is a constant

```
foo   # This variable is not defined
bar() # This function is not defined
```

Maybe it's defined later:

```
baz()

def baz():
    pass
```

Or it wasn't import ed:

```
#needs import math

def sqrt():
    x = float(input("Value: "))
    return math.sqrt(x)
```

Python scopes and the LEGB Rule:

The so-called LEGB Rule talks about the Python scopes. It's name is based on the different scopes, ordered by the correspondent priorities:

```
Local → Enclosed → Global → Built-in.
```

- **L** ocal: Variables not declared global or assigned in a function.
- **E** nclosing: Variables defined in a function that is wrapped inside another function.
- **G** lobal: Variables declared global, or assigned at the top-level of a file.
- **B** uilt-in: Variables preassigned in the built-in names module.

As an example:

```
for i in range(4):
    d = i * 2
print(d)
```

d is accesible because the for loop does not mark a new scope, but if it did, we would have an error and its behavior would be similar to:

```
def noaccess():
    for i in range(4):
        d = i * 2
noaccess()
print(d)
```

Python says NameError: name 'd' is not defined

TypeErrors

These exceptions are caused when the type of some object should be different

TypeError: [definition/method] takes ? positional arguments but ? was given

A function or method was called with more (or less) arguments than the ones it can accept.

Example

If more arguments are given:

```
def foo(a): return a
foo(a,b,c,d) #And a,b,c,d are defined
```

If less arguments are given:

```
def foo(a,b,c,d): return a += b + c + d
foo(a) #And a is defined
```

Note : if you want use an unknown number of arguments, you can use *args or **kwargs . See *args and **kwargs

TypeError: unsupported operand type(s) for [operand]: '???' and '???'

Some types cannot be operated together, depending on the operand.

Example

For example: + is used to concatenate and add, but you can't use any of them for both types. For instance, trying to make a set by concatenating (+ ing) 'set1' and 'tuple1' gives the error. Code:

```
set1, tuple1 = {1,2}, (3,4)
a = set1 + tuple1
```

Some types (eg: int and string) use both + but for different things:

```
b = 400 + 'foo'
```

Or they may not be even used for anything:

```
c = ["a","b"] - [1,2]
```

But you can for example add a float to an int :

```
d = 1 + 1.0
```

TypeError: '???' object is not iterable/subscriptable:

For an object to be iterable it can take sequential indexes starting from zero until the indexes are no longer valid and a IndexError is raised (More technically: it has to have an __iter__ method which returns an __iterator__ , or which defines a __getitem__ method that does what was previously mentioned).

Example

Here we are saying that bar is the zeroth item of 1. Nonsense:

```
foo = 1
bar = foo[0]
```

This is a more discrete version: In this example for tries to set x to amount[0] , the first item in an iterable but it can't because amount is an int:

```
amount = 10
for x in amount: print(x)
```

TypeError: '???' object is not callable

You are defining a variable and calling it later (like what you do with a function or method)

Example

```
foo = "notAFunction"
foo()
```

IndentationErrors (or indentation SyntaxErrors)

In most other languages indentation is not compulsory, but in Python (and other languages: early versions of FORTRAN, Makefiles, Whitespace (esoteric language), etc.) that is not the case, what can be confusing if you come from another language, if you were copying code from an example to your own, or simply if you are new.

IndentationError/SyntaxError: unexpected indent

This exception is raised when the indentation level increases with no reason.

Example

There is no reason to increase the level here:

Python 2.x ≥ 2.0 , ≤ 2.7

```
print "This line is ok"
    print "This line isn't ok"
```

Python 3.x ≥ 3.0

```
print("This line is ok")
```

```
     print("This line isn't ok")
```

Here there are two errors: the last one and that the indentation does not match any indentation level. However just one is shown:

Python 2.x ≥ 2.0 , ≤ 2.7

```
print "This line is ok"
 print "This line isn't ok"
```

Python 3.x ≥ 3.0

```
print("This line is ok")
 print("This line isn't ok")
```

IndentationError/SyntaxError: unindent does not match any outer indentation level

Appears you didn't unindent completely.

Example

Python 2.x ≥ 2.0 , ≤ 2.7

```
def foo():
    print "This should be part of foo()"
   print "ERROR!"
print "This is not a part of foo()"
```

Python 3.x ≥ 3.0

```
print("This line is ok")
 print("This line isn't ok")
```

IndentationError: expected an indented block

After a colon (and then a new line) the indentation level has to increase. This error is raised when that didn't happen.

Example

```
if ok:
doStuff()
```

Note : Use the keyword `pass` (that makes absolutely nothing) to just put an `if` , `else` , `except` , `class` , `method` or `definition` but not say what will happen if called/condition is true (but do it later, or in the case of `except` : just do nothing):

```
def foo():
    pass
```

IndentationError: inconsistent use of tabs and spaces in indentation

Example

```
def foo():
    if ok:
      return "Two != Four != Tab"
        return "i dont care i do whatever i want"
```

How to avoid this error

Don't use tabs. It is discouraged by `PEP8` , the style guide for Python.

1. Set your editor to use 4 **spaces** for indentation.
2. Make a search and replace to replace all tabs with 4 spaces.
3. Make sure your editor is set to **display** tabs as 8 spaces, so that you can realize easily that error and fix it.

See this question if you want to learn more.

Syntax

Parameters

Remarks