A Short Guide to Common Errors in Python

A Short Guide to Common Errors in Python

This guide is an attempt to compile a small set of example programs that illustrate common errors in Python and then explain those errors in English. (Our thanks to Emily, a class mentor, who drafted an early version of this guide.) While it is by no means complete or comprehensive, we've added it here as a resource that you can consult when you encounter an error. For a more comprehensive list of Python errors, feel free to consult the [official Python Docs](https://docs.python.org/3/library/exceptions.html). **Note that you won't encounter many of these errors until later in the specialization.**

**SyntaxError: "I don't recognize what you just wrote as Python code."**

'Syntax' refers to the rules that dictate how your Python program should be structured. For example, one rule states that you may not begin a variable name with a digit in Python. A SyntaxError will pop up if you violate any of Python's syntax rules. This could be forgetting to close a parenthesis or quote, mistakes in indentation, forgetting colons after function headers, etc.

Note that syntax errors can be tricky - sometimes the real error is actually before or after the line that the error is reported on. When you go hunting for syntax errors, try looking at the entire block of code that surrounds the line where the error appears. See the last example below that illustrates this point.

* SyntaxError: bad input ('"Hello world!"') - <http://py3.codeskulptor.org/#user300_6QYHe623Yr_0.py>
* Syntax Error: bad token('"') - <http://py3.codeskulptor.org/#user300_q58d5mVZvS_2.py>
* Syntax Error: bad input('print') - <http://py3.codeskulptor.org/#user300_p7riICJqyf_2.py>
* Syntax Error: 'return' outside function - <http://py3.codeskulptor.org/#user300_YwkuUJ8sQE_0.py> (???)
* Syntax Error: can't assign to literal - <http://py3.codeskulptor.org/#user300_HPhwY3lyZs_1.py>
* Syntax Error: bad input (' ') - <http://py3.codeskulptor.org/#user300_q58d5mVZvS_0.py>
* SyntaxError: bad input ('surprise') - <http://py3.codeskulptor.org/#user300_0eCps4tcH3_1.py>

**NameError: "What in the world is X?"**

A NameError is triggered when Codeskulptor doesn't recognize the variable, function name, etc. that you're referring to. A NameError is Codeskulptor's way of saying "I've never heard of that before!" Common causes of this are misspellings or forgetting to declare variables entirely.

* NameError: name 'some\_variable\_name' is not defined - <http://py3.codeskulptor.org/#user300_rvErPx9lFA_8.py>
* NameError: name 'some\_function\_name' is not defined-<http://py3.codeskulptor.org/#user300_DSrwc3QXrb_2.py>

**TypeError: "You're trying to use some of those things in a way that wasn't intended."**

This error can come up when you're trying to do perform an operation on something, but the operation and the thing just don't go together. One example is trying to multiply a dictionary and an integer - it just won't work. A helpful tool in debugging type errors is the Python \color{red}{\verb|type()|}type() function. Adding print statements like "\color{red}{\verb|print(type(my\_variable))|}print(type(my\_variable))" will help you figure out what's going wrong.

* TypeError: cannot concatenate 'str' and 'int' objects (or any other data type): - <http://py3.codeskulptor.org/#user300_O8c065gLl1_12.py>
* TypeError: function\_name() takes exactly x arguments (y given) - <http://py3.codeskulptor.org/#user300_U4kDkOG61t_0.py>
* TypeError: 'int' object is not callable (could be a different data type) - <http://py3.codeskulptor.org/#user300_BRRaiVlwpD_1.py>
* TypeError: 'int' object is not iterable (could be a different data type) - <http://py3.codeskulptor.org/#user300_Vd3ft6yTVH_1.py>

**IndentationError : "Several statements in the same block of code are not indented the same."**

Python uses indentation to group the statements that form the body of a function definition or the clauses of an if-elif-else statement. In particular, all of the statements in this body should have the same level of indentation. Python gives this error when it detects two statements that should be at the same level of indentation.

* IndentationError: unindent does not match any outer indentation - <http://py3.codeskulptor.org/#user300_DM85TuO1tN5P1fJ_0.py>

**IndexError: "That list/dictionary/tuple (etc.) doesn't have that many items in it."**

An IndexError happens when you try to access an index that doesn't actually exist. It's like telling someone to take 13 eggs out of a 12-egg carton - it won't work, because there are only 12 spaces. Printing out your index values, along with \color{red}{\verb|len(the\_list\_in\_question)|}len(the\_list\_in\_question) will help you in debugging these errors. Important to note: negative indices \*are\* possible in Python! See the video lecture on lists for more information.

* IndexError: list index out of range - <http://py3.codeskulptor.org/#user300_eget014sAQ_10.py>

**TokenError: "You probably forgot to close a bracket." \***

Very simply, tokens are things that stand for other things - sort of like variables, except they're used at a more structural level. Some examples of tokens are EOF (End Of File) and EOL (End Of Line). These are the two most common ones you will come across in Python, but they are used everywhere in programming. The TokenErrors that you will see in this course are usually solved by remembering to close your brackets. See the example for a more in-depth explanation of why this is.

* TokenError: EOF in multi-line statement - <http://py3.codeskulptor.org/#user300_TwsBlWnsiG_4.py>

\*Not always what it means - but a more detailed explanation is more appropriate for later courses. This is the most common causes of TokenErrors in this class.

**ValueError: "There's something wrong with the value of one of those arguments (but the type is ok)."**

A ValueError is raised when a function receives an argument that looks ok on the surface (e.g., it receives a character, and it was looking for a character), but the value of that argument is unexpected (e.g., the function was only built to handle digits, and it received a letter 'a'). This type of error can be solved by checking the documentation for whatever function you're trying to use and making sure that whatever you put inside the parentheses, the function was built to handle it.

* ValueError: invalid literal for int() with base 10: ' ' - <http://py3.codeskulptor.org/#user300_v7yqnKyAPa_4.py>

**AttributeError: "That object doesn't know how do to what you asked it to do"**

In object-oriented programming (OOP), objects have "attributes" - things that they're aware of, and/or know how to do. In terms of Python, attributes are an objects properties and the methods defined by its class. An AttributeError will be thrown when you ask an object to do something or access something that isn't in its class definition.

* Attribute Error: '[Object X] has no attribute [attribute Y] - <http://py3.codeskulptor.org/#user300_Qvjnv5BNTg_1.py>

**Miscellaneous section - these errors are either self-explanatory or not common in the level of programming done in this class**

* ZeroDivisionError - you guessed it - you're trying to divide by zero somewhere
* ImportError - caused by trying to import a module that doesn't exist. Check your spelling.
* HierarchyError - Internet Explorer is not supported by CodeSkulptor3. Use Chrome or Firefox.
* JavaScript section - these are not Python errors. They are related to your browser, not your code.