Appendix D Common Error Messages in Python

Error messages in Python can often be confusing. Here is a list of common error messages you may find, along with a plain English explanation. These error messages are the result of runtime errors. They will immediately crash your Here are the error messages explained (your error messages may be slightly different but mean the same thing):

- SyntaxError: invalid syntax
- ImportError: No module named raandom
- SyntaxError: EOL while scanning string literal
- AttributeError: 'str' object has no attribute 'lowerr'
- IndentationError: expected an indented block
- IndentationError: unexpected indent
- IndentationError: unindent does not match any outer indentation level
- TypeError: bad operand type for abs(): 'str'
- TypeError: abs() takes exactly one argument (2 given)
- IndexError: list index out of range
- KeyError: 'spam'

SyntaxError: invalid syntax

This is the most generic error message the Python interpreter will give you. It means that Python was expecting something that isn't there, or there is something there that it didn't expect. Maybe you forgot to include or inserted an extra character. Here are some examples:

```
if guess = 5:
```

In the above case, the programmer used = (the assignment operator) instead of == (the equals comparator operator). Python never expects assignment statements where there should be a condition.

```
def foo(:
```

In the above case, the programmer forgot to match the ending) closing parenthesis.

```
def foo()
```

In the above case, the programmer forgot to put the colon at the end of the `def` statement. This can also happen with `for`, `while`, `if`, `elif`, and `else` statements.

ImportError: No module named raandom

This error shows up when you try to import a module that does not exist. Most likely, you have a typo in the module name. For example, you may have typed `raandom` instead of `random`.

SyntaxError: EOL while scanning string literal

```
print('Hello world!)
print("Hello world!')
```

This error happens when you do not have two quote marks for a string, or you use different quote marks for the same string. Look at these two examples:

AttributeError: 'str' object has no attribute 'lowerr'

```
'Hello'.lowerr()
'Hello'.append('x')
```

This error appears when you call a method or access an attribute that does not exist. This is most likely because 1) you have a typo in the method or attribute name, or 2) you are calling the method or attribute on a value that is the wrong data type. For example, strings have a method named `lower()`, but not `lowerr()` (that is a typo). And the `append()` method is a list method, so calling it on a string value will cause this error.

IndentationError: expected an indented block

```
def foo():
print('Hello world!')
```

This error happens if you fail to indent your code for a block. In the above example the `print()` call is at the same level of indentation as the `def` statement, when it should have a larger indentation.

IndentationError: unexpected indent

```
def foo():
    print('Hello world!')
     print('Goodbye')
```

An unexpected indent error happens when you add an indentation for no reason. You should only add indentation after a `def`, `if`, `else`, `elif`, `while`, or `for` statment (or any statement that ends with a colon.)

IndentationError: unindent does not match any outer indentation level

```
def foo():
    print('Hello world!')
   print('Goodbye')
```

This indentation error appears when you are decreasing the indentation, but not decreasing it to the same level as the previous indentation. The `print('Goodbye')` call should either be at the same indentation as the other `print()` call (and be inside the `if` block) or at the same indentation as the `if` statement (and be outside of the `if` block).

TypeError: bad operand type for abs(): 'str'

```
abs('Hello')
```

This error occurs when the value of an argument you pass to a function or method is of the wrong data type. In the above example, the `abs()` function takes an integer or floating point number. Passing a string for the argument results in an error.

TypeError: abs() takes exactly one argument (2 given)

```
abs(42, 50)
```

This error appears when you pass the wrong number of arguments to a function or method, either too many or too few. The `abs()` function takes exactly one (and only one) argument. In our example we pass two arguments, which results in this error.

IndexError: list index out of range

```
myList = ['spam', 'fizz', 'eggs']
print(myList[3])
```

The IndexError happens when the index you use is larger than or equal to the number of actual items in the list. In our above example, the `myList` list only has 3 items in it, so the only valid indexes to use are `0`, `1`, and `2`. The index `3` (or any other index larger than `2`) is larger than any of these indexes, so the code results in an IndexError.

KeyError: 'spam'

```
myDict = {'fizz':42, 'eggs':100}
myDict['spam']
```

The KeyError happens when you try to access a key in a dictionary object that does not exist. Either the key was never added to the dictionary, was deleted previously with the `del` operator, or the key you are using has a typo in it.