E: Errors (also called Exceptions)

Gareth McCaughan Revision 1.8, May 14, 2001

Credits

For the LaTeX source of this sheet, and for more information on LiveWires and on this course, see the LiveWires web site at `http://www.livewires.org.uk/python/`

Introduction

When something goes wrong, Python responds by giving you a rude message that might look something like this:

```
Traceback (innermost last):
  File "<stdin>", line 1, in ?
  File "<stdin>", line 1, in f
TypeError: illegal argument type for built-in operation
```

You can usually ignore everything except the last line, although the earlier lines may – if you look at them closely – give some hints about where the trouble happened.

This sheet is a brief guide to some of the commoner things you might see on that last line, what they mean and what you might have done to provoke them.

Error messages

Attribute Errors, Key Errors, Index Errors

These have messages starting "`AttributeError:`", "`KeyError:`" or "`IndexError:`". They all mean something rather similar: you were trying to get at a part of an object (one element of a list, for instance), but you asked for a non-existent part of the object. So, if `x` is the list `[1,2,3]` and you ask for `x[100]` you'll get an `IndexError`.

Name Errors

These all have messages starting "`NameError:`". They mean "I've never heard of this thing". The "thing" Python's never heard of is what comes after "`NameError:`". This might mean that you mistyped something: `primt x` instead of `print x`, say. It might also mean that you used a variable (i.e., a name) before it was defined. Or that you used a *function* before it was `def`ed.

One way this can sometimes happen is if you forget the quotation marks around a string.

Syntax Errors

These all have messages starting "`SyntaxError:`". What they have in common is that you said something Python couldn't even begin to make sense of. If someone says "I apple eat like the to" then that's a syntax error!

```
SyntaxError: invalid syntax
```

This can mean lots of things. Here are some examples of things that provoke it.

`+ + +`	*Add* `what` *to* `what`*?*
`1z6`	*Is it a number? Is it a variable? Is it a mistake?*
`for`	*After* `for` *there ought to be some more stuff*
`while if:`	`if` *isn't a thing that can be true or false!*
`def 1(x):`	`1` *is a number. You can't use it as a name for a function*
`f([)`	*The* `[` *starts a list, but you never finished it*
`if 1==2`	*You missed out the ''':''' at the end of the line*
`if x=y:`	*That should say* `==`*, not* `=`

One way in which you can get slightly surprising syntax errors is if you mess up the indentation (spaces at starts of lines), so if you can't find anything else wrong it's worth checking that the indentation is right.

```
SyntaxError: invalid token
```

This usually means that you messed up when typing a string. Maybe you missed off the final quotation mark, or tried to put apostrophes into a string surrounded by single quotes, or something like that.

Type Errors

These all have messages starting "`TypeError:`". What they have in common is that Python was expecting one kind of object (a number, maybe) and you gave it another (a string, or a list, perhaps).

A lot of these things can happen in non-obvious ways. For instance, if you hand something that isn't a number to one of the graphics functions – `move()`, `draw()` etc – then you are likely to get a `TypeError`.

```
TypeError: illegal argument type for built-in operation
```

Means: You asked Python to do a "built-in operation" (i.e., something like `+`) on the wrong sort of object. For instance, you might have asked it to add a number to a string.

```
TypeError: len() of unsized object
```

You asked for the length of something that doesn't have a length. `len()` makes sense for strings, lists and tuples (you don't need to know what a "tuple" is), and not much else.

```
TypeError: not enough arguments; expected 1, got 0
```

You tried to use a function, but the function was defined with more arguments than you gave it. You probably forgot one!

```
TypeError: number coercion failed
```

You tried to do something using numbers, but one of the things involved wasn't a number. You can get this by doing `4+'ouch'`.

```
TypeError: too many arguments; expected 1, got 3
```

You tried to use a function, but the function was defined with fewer arguments than you gave it.

```
TypeError: unsubscriptable object
```

You tried to do something like `a[3]` where `a` was the wrong kind of thing. This can easily happen if you use a function that expects a string and give it something else instead.