CSP1150/CSP5110: Programming Principles
Reading 4.1: Keyword Parameters

As established in Lecture 4, parameters are how you pass data into a function. When you define a function, you can specify the names and (optionally) default values for its parameters. For example:

Python
```python
def repeat(text, multiplier = 2):
    output = ''

    for i in range(multiplier):
        output += text

    return output
```

This function repeats text the specified number of times, and can be called in two different ways:

1. Specifying both parameters: `repeat('Bork', 3)` (result of `'BorkBorkBork'`)
2. Specifying one parameter: `repeat('Meep!  ')` (result of `'Meep!  Meep!  '`)

In the second version, the default value was used for the second parameter since it wasn't specified. All well and good so far. However, problems can occur when we have *multiple optional parameters*.

Let's enhance the function so that we can now specify a *third* parameter to make the function trim off a number of characters from the end result. It has a default of 0 (i.e. don't trim anything):

Python
```python
def repeat(text, multiplier = 2, trim = 0):
    output = ''

    for i in range(multiplier):
        output += text

    if trim > 0:
        return output[0:-trim]
    else:
        return output
```

This means we can get rid of trailing whitespace and commas fairly easily:
e.g. `repeat('No, ', 3, 2)` will result in `'No, No, No'`

However what if we want to specify a value for the third parameter, but not the second one? Doing `repeat('Meep!  ', 2)` still results in `'Meep!  Meep!  '` since the 2 goes to the multiplier parameter instead of the trim parameter.

This method of specifying parameters to functions based on their order is known as "positional", but you can also specify them by "keyword". This is supported in Python and many other languages, and simply requires you to specify the name of the parameter before the value:
e.g. `repeat('Meep!  ', trim=2)` will result in `'Meep!  Meep! '`

You can mix positional and keyword parameters in a call, but you can only use positional ones *before* you use any keyword ones – the "position" of remaining parameters becomes too hard to track.

Specifying parameters by keyword can be a useful way of eliminating ambiguity and ensuring that a function is called in the exact manner that you intend.