Sometimes it's useful to be able to split a string up based on a particular character or sequence of characters. Python provides a method for doing this. The following examples are intended to provide a crash-course on using the split method of string objects.

```
>>> mystr = "Hello, world"
>>> mystr
Hello, world
```

We have a variable mystr. It's a string. Let's say we'd like to be able to break this string up into words. We can use the split method as follows:

```
>>> mystr.split()
['Hello,', 'world']
```

We can tell from this that:

1) split is a method of the string object (remember, mystr is a string).
2) split returns a list of strings.

So, what is split doing here? When you don't give split any arguments, it breaks up the string on whitespace characters. These include newline characters, tab characters, and spaces. Notice that split removes from the string the characters it used to determine where to split, so the returned strings ('Hello,' and 'world') don't contain any spaces.

Let's look at a longer example.

```
>>> mystr = "cat dog mouse phone laptop clock"
>>> mystr
'cat dog mouse phone laptop clock'
>>> mystr.split()
['cat', 'dog', 'mouse', 'phone', 'laptop', 'clock']
```

Here we have a longer string to start with; it contains 6 words, each separated by just one space. When we call split on mystr, we get back (as the return value) a list containing 6 elements. Each one is a string from the original string. Again notice that none of the strings in the list contains a space.

We can split on other characters also:

```
>>> mystr.split("o")
['cat d', 'g m', 'use ph', 'ne lapt', 'p cl', 'ck']
```

Here, we passed split the argument “o”. The first argument to split is the string that you want to break the bigger string (in this case mystr) up on. We see that split breaks the string on the character “o”; mystr gets broken on the o in “dog”, the o in “mouse”, the o in “phone”, the o in “laptop”, and the o in “clock”. In this case, because we gave split an argument, it didn't use spaces to break up the string. Split leaves the spaces alone, but it removes all occurences of the string “o”.

Here's another example using the string “p”:

```
>>> mystr.split(“p”)
['cat dog mouse ', 'hone la', 'to', ' clock']
```

We can use strings longer than one character to split on:

```
>>> mystr.split("ph")
['cat dog mouse ', 'one laptop clock']
```

Again, split breaks mystr into multiple strings based on the argument we gave it, and deletes all occurrences of the argument. In all of the above cases, note that mystr is NOT modified. Remember, in Python, string objects are immutable.

Split is very handy when trying to handle input from a file. Here's an example input file:

```
word dog cat
chromium wildcats memory
intel
```

I've saved this file as dictionary.txt. Let's say that I open the file and then read the entire contents of the file into a string:

```
>>> infile = open("dictionary.txt", "r")
>>> contents = infile.read()
>>> contents
'word dog cat\nchromium wildcats memory\nintel\n'
```

The representation here shows the newline characters using “\n”, instead of actually printing a new line to the Python shell. How could I use split to break this single string into lines? Well, what character or sequence of characters means that we're going to a new line?

Let's try splitting on the newline character:

```
>>> contents.split(“\n”)
['word dog cat', 'chromium wildcats memory', 'intel', '']
```

That seems to work, but notice that because there's a newline character at the end of the file contents, we get the empty string as the last element in the list. Of course, this is a lot of work to get a list of the lines in a file. If we just want the list of lines, we could use infile.readlines().

Now, what if I just want to get a list of the words that are in the file? How can I use split to do that?

No, really, think about this one for a second.

Well, I want to split on newline characters and spaces, right? What would split return if we did that, and, perhaps the less obvious quesiton, what argument should we pass split?

Remember split's default behavior (when you don't pass it any arguments)?

The default is to split on newlines, tabs, and spaces. This includes newlines and spaces, which is what we want to split on, so let's try that:

```
>>> contents.split()
['word', 'dog', 'cat', 'chromium', 'wildcats', 'memory', 'intel']
```

We can see that that works quite nicely.