Reading documentation for python functions can be hard. There is a whole new language used for documentation. I'll talk through a few examples:

arange Let's start by looking at the documentation for the numpy function `arange`.

The first thing it shows is the form of a call to the function:

```
numpy.arange([start, ]stop, [step, ]dtype=None, *, like=None)
```

1. The words "start", "stop", etc. are the names of arguments (or parameters).
2. Square brackets around an argument indicate that it is optional. So you don't have to specify the start, and you don't have to specify the step.
3. An argument with an "=" sign is also optional, but it's optional because it has a default value. In this case, `dtype=None` means that the default `dtype` is `None`. That probably doesn't help you too much, but you can usually ignore most arguments with default values. Assume the default is reasonable.

After this, the parameters are individually described. This can get pretty verbose, but is sometimes pretty helpful. Then the documentation talks about the return value (or values).

Towards the bottom you can sometimes find helpful examples, e.g.

```
>>> np.arange(3)
array([0, 1, 2])
>>> np.arange(3.0)
array([ 0.,  1.,  2.])
np.arange(3,7)
>>> array([3, 4, 5, 6])
>>> np.arange(3,7,2)
array([3, 5])
```

This can be helpful, but beware that often these examples are not well constructed for learning. In particular, they frequently double as tests that the function is working correctly, which causes them sometimes to be contorted in confusing ways.

errorbar Now let's consider the matplotlib `errorbar` function.

1. We have lots of default values for `errorbar`.
2. The `, *,` which seems so weird separates ordinary arguments from ones that can only be used as "keyword arguments." Similarly the `, **kwargs` at the very end indicates that maybe other keyword arguments could be used.

   A keyword argument is a parameter to a function that is specified by name.

print See the print function.

You can try

```
print(5, 2, 4)
print(5, 3, 4, sep='x', end=' silly!')
print('the end')
```