list comprehension.

It makes a new list in place – it is a fast way to write code rather than what you might be familiar with;

```
original_list = [item, item, item]
new_list= []
for item in original_list:
        do something to item
        append item to a new list
```

really cool cos it means you can twrite them inside a method or it just saves lines of code.

e.g. inside a method
pd.DataFrame([write a list comprehension here]).
and it would make a pandas data frame from data you have altered in 1 line of code

```
[ all_p_list[i:i+10] for i in range( 0, len(all_list_p) ]
```

The Green- is the new list

The Yellow- is what is going to be inside the new list
in this scenario each element will be a slice of the existing list all_list_p.
Sliced from what ever number the range iterator is currently on.
So the first element in the list will be a chunk of - all_list_p[0:10].
the second element in the list will be another chunk - all_list_p[1:11].
it keeps going until the range hits then end which is the length of all_list_p.
So if all_list_p has 10 elements in it. The final element in the newly made list would be-all_list_p[10:20]
(This is where the bug is to me because now they've made a slice longer than the original list)

The Orange- is assigning a variable to the current element your looping
like in your normal for loop. You can call 'i' anything you want. It's just a variable.

The Blue- the 'iterator' your working on.
Iterators are anything you can loop through. e.g. A list, a range of numbers, the characters in a string. The keys in a dictionary if you remember dictionary.keys()