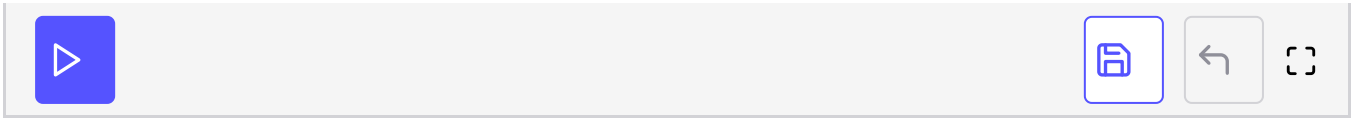
Dictionary comprehensions

We are going to look at Dictionary comprehensions in python

Dictionary comprehensions started life in Python 3.0, but were backported to Python 2.7. They were originally proposed in the Python Enhancement Proposal 274 (PEP 274) back in 2001. They are pretty similar to a list comprehension in the way that they are organized.

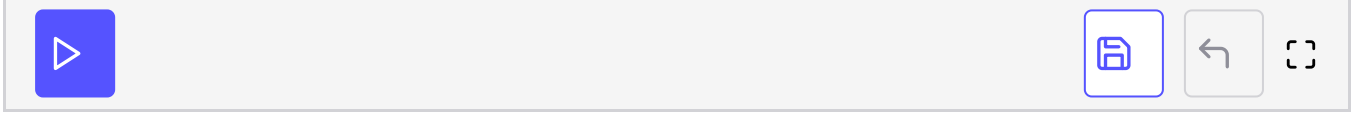
The best way to understand is to just do one!

```
print( {i: str(i) for i in range(5)} )
# {0: '0', 1: '1', 2: '2', 3: '3', 4: '4'}
```



This is a pretty straightforward comprehension. Basically it is creating an integer key and string value for each item in the range. Now you may be wondering how you could use a dictionary comprehension in real life. Mark Pilgrim mentioned that you could use a dictionary comprehension for swapping the dictionary's keys and values. Here's how you would do that:

```
my_dict = {1:"dog", 2:"cat", 3:"hamster"}
print( {value:key for key, value in my_dict.items()} )
# {'hamster': 3, 'dog': 1, 'cat': 2}
```



This will only work if the dictionary values are of a non-mutable type, such as a string. Otherwise you will end up causing an exception to be raised.

I could also see a dictionary comprehension being useful for creating a table out of class variables and their values. However, we haven't covered classes at this point, so I won't confuse you with that here.