Iteration, iterators

- Iteration is basic to computation. One common instance of iteration is over the elements of a collection. Python supports iteration over collection objects by wrapping them inside an iterator object using the `iter` function. The iterator object also has a next() function that produces the next element of the collection.
- To make a collection iterable the class defining the collection object must define the `__iter__()` function or the `__getitem__()` function with sequential indices starting at 0. It must also define a `__next__()` function to get the next element in the collection.
- Classes that implement data types like lists, tuples already have such things built-in and are therefore iterable objects.
- If a collection class is iterable then it can be used in a `for` statement.

Example

```
class Stack:
    def __init__(self):
        self.stack=[]

    def push(self, e):
        self.stack.append(e)

    def pop(self):
        if self.stack!=[]:
            return self.stack.pop(-1)
        else:
            return None

    def isEmpty(self):
        return self.stack==[]

    def __iter__(self):
        return self

    def __next__(self):
        if self.stack!=[]:
            return self.stack.pop(-1)
        else: raise StopIteration()
```