Dynamic Array Exercise

In this exercise we will create our own Dynamic Array class!

We'll be using a built in library called ctypes (https://docs.python.org/2/library/ctypes.html). Check out the documentation for more info, but its basically going to be used here as a raw array from the ctypes module. If you find yourself very interested in it, check out: Ctypes Tutorial (http://starship.python.net/crew/theller/ctypes/tutorial.html)

Also...

A quick note on public vs private methods, we can use an underscore _ before the method name to keep it non-public. For example:

In [14]:
```
class M(object):

    def public(self):
        print 'Use Tab to see me!'

    def _private(self):
        print "You won't be able to Tab to see me!"
```

In [15]:
```
m = M()
```

In [17]:
```
m.public()
```

```
Use Tab to see me!
```

In [19]:
```
m._private()
```

```
You won't be able to see me!
```

Check out PEP 8 and the Python docs for more info on this!

Dynamic Array Implementation

In [43]:
```
import ctypes

class DynamicArray(object):
    '''
    DYNAMIC ARRAY CLASS (Similar to Python List)
    '''

    def __init__(self):
        self.n = 0 # Count actual elements (Default is 0)
        self.capacity = 1 # Default Capacity
        self.A = self.make_array(self.capacity)

    def __len__(self):
        """
        Return number of elements sorted in array
        """
        return self.n

    def __getitem__(self,k):
        """
        Return element at index k
        """
        if not 0 <= k <self.n:
            return IndexError('K is out of bounds!') # Check it k index is in bou

        return self.A[k] #Retrieve from array at index k

    def append(self, ele):
        """
        Add element to end of the array
        """
        if self.n == self.capacity:
            self._resize(2*self.capacity) #Double capacity if not enough room

        self.A[self.n] = ele #Set self.n index to element
        self.n += 1

    def _resize(self,new_cap):
        """
        Resize internal array to capacity new_cap
        """

        B = self.make_array(new_cap) # New bigger array

        for k in range(self.n): # Reference all existing values
            B[k] = self.A[k]

        self.A = B # Call A the new bigger array
        self.capacity = new_cap # Reset the capacity

    def make_array(self,new_cap):
        """
        Returns a new array with new_cap capacity
        """
        return (new_cap * ctypes.py_object)()
```

In [44]:
```
# Instantiate
arr = DynamicArray()
```

In [45]:
```
# Append new element
arr.append(1)
```

In [46]:
```
# Check length
len(arr)
```

Out[46]:
```
1
```

In [47]:
```
# Append new element
arr.append(2)
```

In [48]:
```
# Check length
len(arr)
```

Out[48]:
```
2
```

In [50]:
```
# Index
arr[0]
```

Out[50]:
```
1
```

In [51]:
```
arr[1]
```

Out[51]:
```
2
```

Awesome, we made our own dynamic array! Play around with it and see how it auto-resizes. Try using the same **sys.getsizeof()** function we worked with previously!

Great Job!