Dynamic Arrays & Amortization

→ in Python: list has length when constructed, but still allows us to add w/ no limit. This abstraction is called a dynamic array.

→ list instance has underlying array that often has a greater capacity

→ if user continues and exhausts the space, the class requests new, larger array

evidenced by:

```
import sys
data = []
for k in range(n):
  a = len(data)
  b = sys.getsizeof(data)    # actual size in bytes
  print('Length: {0:3d}; Size in bytes: {1:4d}'.format(a,b))
  data.append(None)          # increase length by 1
```

Implementing Dynamic Array

→ key to provide means to grow array A that stores the elements:

1.) allocate a new array B with larger capacity

2.) set B[i] = A[i], for i = 0, ..., n-1, where n denotes curr # of items

3.) set A = B; setting B as the array supporting the list

4.) insert new element in new array

→ Code Example

```
import ctypes
Class DynamicArray:
  def __init__(self):
    self._n = 0
    self._capacity = 1
    self._A = self._make_array(self._capacity)

  def __len__(self):
    return self._n

  def __getitem__(self,k):
    if not 0 <= k <= self._n
      raise IndexError('invalid index')
    return self._A[k]

  def _append__(self,obj):
    if self._n = self.capacity:
      self._resize(2 * self.capacity)
    self._A[self._n] = obj
    self._n += 1
```

→