#algorithms

Building a Heap

To make things easier, i'm using the bitwise shift in python.

```
return (i << 1) + 1
```

lets say i = 2

- binary: 10

i << 1

- binary: 100
- which is 4

add 1 results in 5 which is the child index.

so that's how the following functions work

- left
- right
- parent

Heapify

- begins at i
- compares with left and right children
- if a child is smaller, we swap
- recursively call heapify down the tree
- stops when the smallest value is i which means no more swapping is needed.

Build min Heap

- Trying to get the last non leaf node
- first we have to find n (length of heap)
- len(self.heap) -1 gets n
    - it becomes heap[-1] which gets us the result

- then we do a bitwise shift >> 1 which is equivalent to /2
- now we have n/2 which is how we get to the leaf nodes in a complete binary tree
- range is a function with (start, stop, step) as parameters
- we just went over start
- stop is -1 so that we include index 0 and no more
- step is -1 so we decrement by 1 moving in reverse order

Push

- starting from the bottom of the tree
- compare to the parent
- swap with parent if it's bigger than it
- repeat until the parent is smaller than itself

Pop

- if empty, return none
- if heap with only one element, return the pop
- otherwise stop the value in "root"
- pop the heap
- heapify the heap
- return root

Peek

- just return heap at index 0