Review from last week

Solution: Factoring

```
def factor(N):
    $d \leftarrow 2$
    $c \leftarrow N$
    while $d \leq N$ do
        $(q, r) \leftarrow \text{divide}(c, d)$
        if $r = 0$ then
            $\text{print}(d)$
            $c \leftarrow q$
        else
            $d \leftarrow \text{add}(d, 1)$
end def
```

Divide by 2 until you can't divide by 2 anymore. Then divide by 3 until you can't divide by 3 anymore, etc.

Binary search

Number guessing game

The judge picks a number from 1 to 100. The guesser has to guess it. After each guess, the judge will tell you whether your guess was correct, too high, or too low.

What's a good strategy for guessing the number quickly?

Quick review of lists

- Python list datastructure
- Also often called "array"
- Combine multiple pieces of data together.
- Access elements with numeric indexing.
  - Zero-based counting.
  - First element is at index 0, second element is at index 1, third element is at index 2, and so on.

Example:

```
animals = ["cat", "dog", "snake"]

animals[0] == "cat"
animals[2] == "snake"

# Length
len(animals) == 3

# Due to zero-based indexing, the last element is:
animals[len(animals) - 1] == "snake"
```

Linear search

```
*Parameters: Take an array of numbers and a target element 'n' to look for.*
*Return: The index where we found the target.*
```

```
def linear_find(arr, target):
    $i \leftarrow 0$
    result $\leftarrow$ None
    while $i <$ len(arr) do
        if arr[$i$] $=$ target then
            result $\leftarrow i$
            stop looping because we found target
        else if arr[$i$] $>$ target then
            If the array is sorted, stop looping because we've gone past where target could be, to numbers bigger than target.
        end if
        $i \leftarrow i + 1$
    end while
end def
```

If the array is sorted, we can stop looking after we find a number larger than n, even if we didn't find n. And we can return some special value like `None` or `-1` to indicate failure.

Binary search

```
*Parameters: Take an array of numbers and a target element to look for.*
*Return: The index where we found the target.*
def binary_find(arr, target):
  $L \leftarrow 0$. Left boundary
  $R \leftarrow \text{len}(\text{arr}) - 1$. Right boundary
  while $L \leq R$ do
    middle $\leftarrow \text{floor}\left(\frac{L+R}{2}\right)$
    if $\text{arr}[\text{middle}] < \text{target}$ then
      $L \leftarrow \text{middle} + 1$
    else if $\text{arr}[\text{middle}] > \text{target}$ then
      $R \leftarrow \text{middle} - 1$
    else, they're equal, we've found target
      return middle
    end if
  end while
  return None
end def
```

Demo on the board with a list of just numbers 1 through 10, drawing L and R pointers. Crossing out parts of the list beyond L or beyond R.