## Example Problem Solved with Lists

Prompt user to enter n values from minValue to maxValue. Count how many of each value was entered.

See 04-01-list-of-counters.py

Inefficient – Wastes space.
When valid numbers are 20 to 23, we end up with something like this:
→ [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 5, 0, 1]

We'll fix this later – With a dictionary.

## Functional Programming Tools

### map

Inputs:
1) A function that takes *one argument*
2) A sequence

Output: A new list, constructed from applying function to every element

---

```
def square(x):
   return x * x

def double(x):
    return x + x

M = map(square, range(-5, 5))              →  M: <map object ...>

list(M)                                    →  [25, 16, 9, 4, 1, 0, 1, 4, 9, 16]


M = map(double, [1, 2, 3, 'a', 'b', 'c'])                     →  M: <map object ...>
list(M)                                    →  [2, 4, 6, 'aa', 'bb', 'cc']
```

---

Example using map

Converting a string consisting digits separated by white space into a list of numbers.

1. Read the string of nunbers
   ```
   a = input('Enter numbers separated by spaces: ')
        →      Enter numbers: 1 5 6 5 4 5 6
   a                                    →  '1 5 6 5 4 5 6'
   ```

2. Split the string to get a list of characters, default delimiter is white space
   ```
   b = a.split()                        →  b: ['1', '5', '6', '5', '4', '5', '6']
   ```

3. Map the int function down list to convert string to an int
   ```
   c = map(int, b)
   list(c)                              →  [1, 5, 6, 5, 4, 5, 6]
   ```

reduce – in module functools

Inputs:

1) A function that takes *two arguments*
2) A sequence (list, range, etc.)

Output:

1) Send 1st two elements to function. Get result
2) Send result and 3rd element to function. Get result
3) Send result and 4th element to function. Get result
4) Goes to the end of sequence
5) Return final result

```
from functools import reduce

def add(x, y):
   return x + y

reduce(add, [1, 2, 3, 4])
```

→ 10; (((1 + 2) + 3) + 4)

## filter

Inputs:
1) A Boolean function
2) A sequence

Output:
A new list, consisting of all elements that made the function return True

---

```
def isPositive(number):
   return number > 0

F = filter(isPositive, [1, 2, -3, 4, -5, 0, 6])
     → F: <filter object …>
list(F)                                    → [1, 2, 4, 6]
```

---

Example – Keep only Consonants in a string

```
Import string

def isConsonant(ch):
   return ch in string.ascii_letters and ch not in 'aeiouAEIOU'

F = filter(isConsonant, 'You were not made for comfort, you were made for greatness')

F = list(F)                          →      F: ['Y', 'w', 'r', 'n', 't', 'm', 'd', 'f', 'r', 'c', 'm', 'f',
                                     'r', 't', 'y', 'w', 'r', 'm', 'd', 'f', 'r', 'g', 'r', 't', 'n', 's', 's']

''.join(F)                                 → 'Ywrntmdfrcmfrtywrmdfrgrtnss'
```

---

Example – Keep only digits

```
F = filter(str.isdigit, 'one 1 two 2 x3z 456')

list(F)                                    → ['1', '2', '3', '4', '5', '6']
```

From this point we can either convert it back to the string '123456' or turn it into the list [1, 2, 3, 4, 5, 6]