LISTS, MUTATION
REMINDER

• Assignment 04 due next Wednesday, June 12\textsuperscript{th} at 10:00AM

• Midterm is on June 24\textsuperscript{th} at 7 PM
COMMON LIST FUNCTIONS

• \texttt{len(L)} => returns length of \( L \)
• \texttt{L[i]} => returns element at index \( i \)
• \texttt{L[i:j]} => returns \( L \) from \( i \) to \( j-1 \)
• \texttt{x in L} => returns \text{True} if \( x \) is in \( L \) and \text{False} otherwise.

• \texttt{L.append(x)}
• \texttt{L.remove(x)}

Examples of functions that \textit{mutate} lists.

• See Module 04 Slide 8 for other list functions and use Python's help function
ABSTRACT LIST FUNCTIONS: 
MAP, FILTER

- **map**
  - applies function to each element in list
    \[
    \text{list(map(fun\_name, L))}
    \]

- **filter**
  - matches the elements in list for which function returns True.
    \[
    \text{list(filter(fun\_name, L))}
    \]

**Note:** fun\_name must have only one parameter/argument.

**Note:** map and filter both return an iterator, and we need to convert that to a list.

map and filter can also be applied to strings.
**LAMBDA**

Lambda \(x, y, \ldots, z: \text{body here}\)

Example:

```python
def non_zero(numlist):
    return list(filter(lambda x: x != 0, numlist))

def triple(numlist):
    return list(map(lambda x: x * 3, numlist))
```
We can apply `map` and `filter` to strings with `lambda`.

For example:

```python
def just_letter(sentence):
    loc = list(filter(lambda c: c.isalpha(), sentence))
    return "".join(loc)
```
ITEM DEFINITION

A Card is a list of length 2 where
- the first item is an integer between 1 and 13, inclusive, representing the value of the card, and
- the second item is a string ("hearts", "spades", "clubs", or "diamonds") representing the suit of the card.

Example: [1, "hearts"] represents the ace of hearts
Write a function `create_cards` that consumes two lists with same length, which are a list of card values (integers between 1 and 13), and a list of suit values (one of the four suit strings), and returns a list of Card, created pair-wise from the consumed lists (values and suits).

- For example,

```python
create_cards([4, 1, 10], ['hearts', 'diamonds', 'clubs'])
=> [[4, 'hearts'], [1, 'diamonds'], [10, 'clubs']]
```
Write a function `choose_by_colour` that consumes a list of `Card` (hand) and a string "red" or "black" (colour) and returns a list of the values of the `Card` in hand of the appropriate colour (spades and clubs are "black", hearts and diamonds are "red").

For example,

```python
choose_by_colour([[1,'hearts'],
                  [9,'spades'],
                  [3,'diamonds']], 'red')
⇒ [1,3]
```

Write this function twice. First, use structural recursion. Then, use abstract list functions.
QUESTION 3

a) Write a function `flip_colour` that consumes a `Card`, `c`, and **mutates** the suit of that `Card` to a different colour: if `c` is a heart, it is mutated to a spade (and vice versa), while if `c` is a club, it is mutated to a diamond (and vice versa).

b) Write a function `flip_hand` that consumes a list of `Card` (hand), and **mutates** the suit of each `Card` in the list so that their colours are flipped in the same way as in `flip_colour`. 
Write a function `modify_list` that consumes a list of integers (called `nums`) and a single integer (`n`). The function returns `None`, but **mutates** the list in the following way:

- If `n` does not appear in `nums` then add it to the end of `nums`.
- If `n` appears once, then remove `n` from `nums`.
- If `n` appears at least twice, remove the first and last occurrences of `n`.

For example:

```python
L = [1, 2, 3]
modify_list(L, 10) => None
L = [1, 2, 3, 10]
```
Write a function sanitize that consumes a string, s, and returns a similar string but with any non-alphanumeric characters removed. Write this function using abstract list functions that operate on the consumed string.

• For example: `sanitize("@Test@") => "Test"`