CS 116 TUTORIAL

4

[ , , , , ]

LISTS, MUTATION
REMINDER

• Assignment 04 due next Wednesday, June 7th (at 10:00AM)
COMMON LIST FUNCTIONS

• \texttt{len(L)} => produces length of \( L \)
• \texttt{L[i]} => produces element at index \( i \)
• \texttt{L[i:j]} => produces \( L \) from \( i \) to \( j-1 \)
• \texttt{x in L} => produces \( \text{True} \) if \( x \) is in \( L \) and \( \text{False} \) otherwise.
• \texttt{L.append(x)}
• \texttt{L.remove(x)}
• See Module 04 Slide 8 for other list functions and use Python's help function
**MAP, FILTER**

- **map**
  - applies function to each element in list

```python
list(map(function, list))
```

- **Filter**
  - produces an iterator matching the elements in list for which function returns true.

```python
list(filter(function, list))
```

Used to turn it into a list
Lambda

lambda x, y, ..., z: body here

Notes: Can be applied to strings as well

For example:

def just_letter(sentence):
    return list(filter(lambda c: c.isalpha(), sentence))
A Card is a list of length 2 where
- the first item is an integer between 1 and 10, 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
1. Write a function `create_cards` that consumes two lists with same length, which are a list of card values (integers between 1 and 10), and a list of suit values (one of the four suit strings), and produces a sorted list of Card from smallest to largest, created pair-wise from the consumed lists (values and suits).

For example,
```
card create_cards([4,1,10],
                  ["hearts","diamonds","clubs"])
=>[[1, "diamonds"],[4,"hearts"],[10, "clubs"]]
```
2. Write a function `choose_by_colour` that consumes a list of `Card (hand)` and a string "red" or "black" (colour) and produces 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,

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

⇒ `[1, 3]`

Write this function twice. First, use explicit recursion. Then, use abstract list functions.
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`. 
4. Write a function `modify_list` that consumes a list of integers (called `nums`) and a single integer (`n`). The function produces `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`. 
5. Write a function `sanitize` that consumes a string, `s`, and produces a similar string but with any non-alphanumeric characters removed.

For example,

```python
sanitize("@Test@") => "Test"
```

DO NOT use explicit recursion