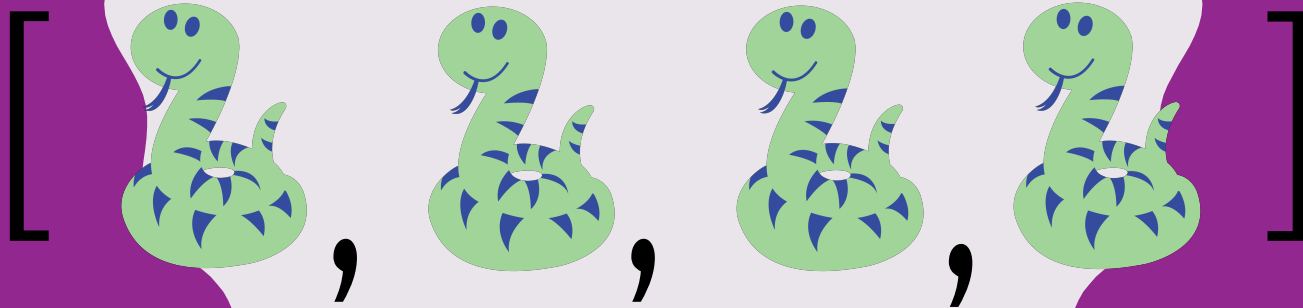


CS 116 TUTORIAL

4



LISTS, MUTATION

REMINDER

- Assignment 04 due next Wednesday, February 12th (at 10:00AM)
- Midterm is on March 2nd at 7 PM
- Final is on April 15th at 4 PM
- Q&A session on February 29 at 2 PM.
 - Share any questions you want to be reviewed with the ISA's on the Piazza post set up for this.

COMMON LIST FUNCTIONS

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

 - `L.append(x)`
 - `L.remove(x)`
 - `L.pop(x)`
 - `L.insert(i, x)`
- Examples of functions that **mutate** lists.
- See Module 04 Slide 8 for other list functions and use Python's help function

CQ 1:

A `Info` is a list of 4 items in the order below:

1. Name (`Str`)
2. User Id (`Str`)
3. Faculty (`Str`)
4. Year (`Nat`)

Example:

```
June_info = ['June K',  
            'k34june',  
            'Science', 3]
```

How do we find June's faculty from `June_info`?

```
# constants:  
faculty = 'Science'
```

- A.** `June_info[2]`
- B.** `June_info[2:3]`
- C.** `June_info[2:3][0]`
- D.** A and B
- E.** A and C

ABSTRACT LIST FUNCTIONS: MAP, FILTER

Note: `fun_name` **typically** have only one **parameter/argument**.

Map applies function to each element in list

❖ Typical ONE parameter case:

```
list(map(fun_name, L))
```



Used to turn
it into a list

```
Ex. def fun_name(x) :  
    return x+2
```

* A good-to-know (**not required** in CS116):

2-parameter example:

```
list(map(fun_name2, L1, L2))
```

- `fun_name2` takes 2 parameters in this case.

ABSTRACT LIST FUNCTIONS: FILTER

- `filter`
 - matches the elements in list for which function `fun_name` returns `True`.

```
list(filter(fun_name, L))
```



Used to turn
it into a list


```
Ex. def fun_name(y):  
    return y==2
```

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 x1, x2, ..., xn: body here
```



Parameters of lambda (no brackets)

Example:

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

```
def triple(numlist):  
    return list(map(lambda x: x * 3, numlist))
```


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



QUESTION 1

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,

```
create_cards([4,1,10],["hearts", "diamonds", "clubs"])  
=>[[4,"hearts"], [1, "diamonds"], [10, "clubs"]]
```

QUESTION 2

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,

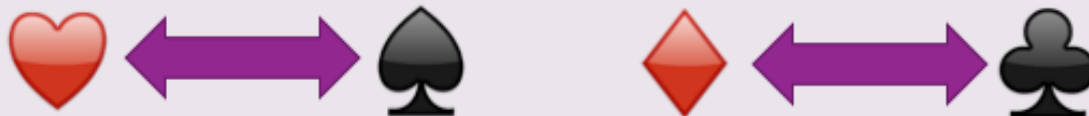
```
choose_by_colour([[1, 'hearts'],  
                 [9, 'spades'],  
                 [3, 'diamonds']], 'red')
```

⇒ [1, 3]

Write this function twice. First, use 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`.



QUESTION 4

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:

```
L = [1, 2, 3]
```

```
modify_list(L, 10) => None
```

```
L = [1, 2, 3, 10]
```

QUESTION 5

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"`

QUESTION 6

Write a function `reversed_list()` that consumes a list of string, `L`, and returns a list containing the elements of `L` in reverse order. Write this function using abstract list functions **ONLY**.

- For example: `reversed_list(['I', 'love', 'cs116']) => (['cs116', 'love', 'I'])`