Announcements

- HW4 due tomorrow
- Get EC for transcribing MT1 by Friday
- Maps project due Thursday 9/27

LAB 4: LISTS & DATA ABSTRACTION

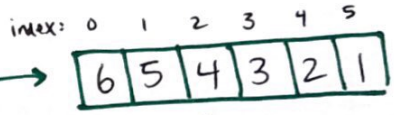
A list is a data structure that can store multiple values. Each value in the list can be any type (it can even be another list!)

→ Each element in a list has an index (start at 0!)

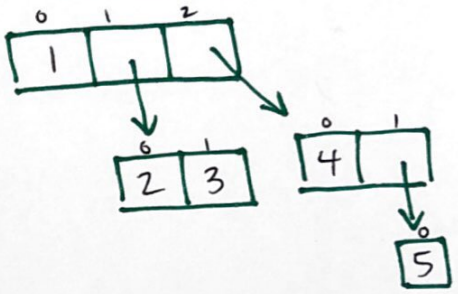
→ Lists are represented using box and pointer diagrams

```
>>> lst = [6,5,4,3,2,1]
>>> lst2 = [1, [2,3], [4, [5]]]
```

Diagram: lst →



lst2 →



* To retrieve an element, use indexing:

```
>>> lst[0]
6

>>> lst2[1]
[2,3]
```

* To find the length of a list, use the len(lst) function:

```
>>> len(lst)
6

>>> len(lst2)
3

>>> len([])
0
```

* To create a copy of some portion of the list, use list slicing.

Syntax: lst [start_index : end_index]

↳ The new list will include the element at start_index and go up to but not include the element at end_index.

```
>>> lst[1:4]
[5,4,3]

>>> lst[:3]
[6,5,4]

>>> lst[2:]
[4,3,2,1]

>>> lst[:]
[6,5,4,3,2,1]
```

List comprehensions

You can create new lists out of sequences!

Syntax: `[ <expr> for <elem> in <sequence> if <cond> ]`

(`if <cond>` is optional)

Example:

```
>>> [ i**2 for i in [1,2,3,4,5,6] if i%2==0]
[4, 16, 36]
```

Iterating through a List

We can use a for loop to iterate through every element in a list!

Syntax:

```
for <elem> in <list>:
    #do stuff
```

example:

```
sum = 0
for i in [1,2,3,4]:
    sum += i
```

At the end of this loop, sum will equal 10.

Data Abstraction

We use Abstract Data Types (ADT) to abstract away information. We can use ADTs without knowing how functions work and can just assume they work correctly.

Abstract data types consist of:

1. constructors: functions that build the ADT
2. selectors: functions that retrieve information from the ADT

DO NOT VIOLATE THE ABSTRACTION BARRIER!

↳ Always use constructor(s) and selector(s) whenever possible instead of assuming the ADT's implementation.