ITP 115

Lists

Review

Recall: Sequences Have Indices!

- Each individual item in a sequence is automatically given an position number
- This number is called an **index** and tells what position the item is in
- The **first index** is **zero** (**`0`**)
- The **last index** is the **number of items – 1**

Lists

- New type of variable!
- Are sequences like strings, but lists are **mutable**
- Contain all the same type of elements*
  - i.e. all strings or all ints

**Technically, Python allows lists to hold different types of elements. For our class, though, we will only store "like items"*

Lists

- Syntax

  `listVariable = [item1, item2, ...]`

- `item1` could be any type of variable
  - string: `"hello"`
  - int: `7`
  - float: `8.5`
  - another list: `["this is", "another list"]`
  - Any other variable type we will cover

List Methods

Method	Description
`someList.append(value)`	Adds value to end of a list.
`someList.sort()`	Sorts the elements, smallest value first.
`someList.reverse()`	Reverses the order of a list.
`someList.count(value)`	Returns the number of occurrences of value.
`someList.index(value)`	Returns the first position number of where value occurs.
`someList.insert(i, value)`	Inserts value at position i.
`someList.pop([i])`	Returns value at position i and removes value from the list. Providing the position number i is optional. Without it, the last element in the list is removed and returned.
`someList.remove(value)`	Removes the first occurrence of value from the list.
`del someList[i]`	Removes the element at the specified index

`someList.sort()`

- Sorts the elements, smallest value first
  - Sorts the actual list—**it does NOT return a new list**
- Example

```
numbers = [3, 5, -12, 40]
```

0	1	2	3
3	5	-12	40

someList.sort()

- Sorts the elements, smallest value first
  - Sorts the actual list—**it does NOT return a new list**
- Example

```
numbers = [3, 5, -12, 40]
numbers.sort()
```

0	1	2	3
-12	3	5	40

`someList.sort()`

- Sorts the elements, smallest value first
  - Sorts the actual list—**it does NOT return a new list**
- Example

```
drinks = ["coffee", "boba"]
```

0	1
coffee	boba

someList.sort()

- Sorts the elements, smallest value first
  - Sorts the actual list—**it does NOT return a new list**
- Example

```
drinks = ["coffee", "boba"]
drinks.sort()
```

0	1
boba	coffee

someList.reverse()

- Reverses the order of the elements
  - Changes actual list—**it does NOT return a new list**
- Example

```
numbers = [3, 5, -12, 40]
```

0	1	2	3
3	5	-12	40

someList.reverse()

- Reverses the order of the elements
  - Changes actual list—**it does NOT return a new list**
- Example

```
numbers = [3, 5, -12, 40]
numbers.reverse()
```

0	1	2	3
40	-12	5	3

`del someList[index]`

- Removes the element from list at *index*

- Example

```
numbers = [3, 5, -12, 40, 5]
```

0	1	2	3	4
3	5	-12	40	5

del someList[index]

- Removes the element from list at *index*

- Example

```
numbers = [3, 5, -12, 40, 5]
del numbers[2]
```

0	1	2	3
3	5	40	5

`del someList[index]`

- Removes the element from list at *index*

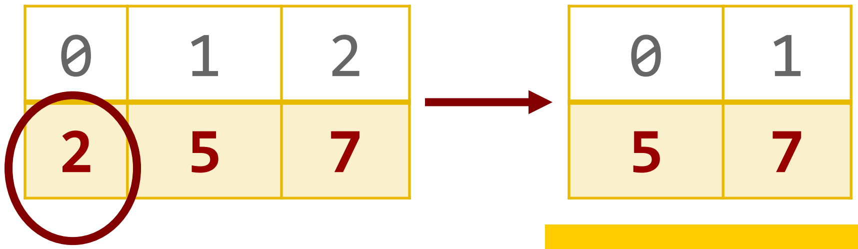
- Example

```
numbers = [3, 5, -12, 40, 5]
del numbers[2]
del numbers[2]
```

0	1	2
3	5	5

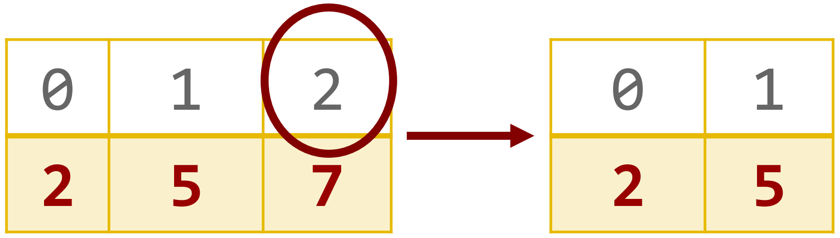
remove vs. del

```
numbers = [2, 5, 7]
numbers.remove(2)
```



Eliminates by **value**

```
numbers = [2, 5, 7]
del numbers[2]
```



Eliminates by **position**

`someList.index(someValue)`

- Returns the first position number where value occurs

- Example

```
numbers = [3, 5, -12]
```

0	1	2
3	5	-12

someList.index(someValue)

- Returns the first position number where value occurs

- Example

```
numbers = [3, 5, -12]
found = numbers.index(5)

print(found)
```

```
1
```

0	1	2
3	5	-12

`someList.index(someValue)`

- Returns the first position number where value occurs

- Example

```
numbers = [3, 5, -12]
found = numbers.index(5)

print(found)
```

```
1
```

```
found = numbers.index(100)
```

```
Error
```

0	1	2
3	5	-12

Slicing Lists

- We can slice a list just we did with strings
- We can use **slicing** to get <u>multiple items</u> from a sequence

Slicing Lists

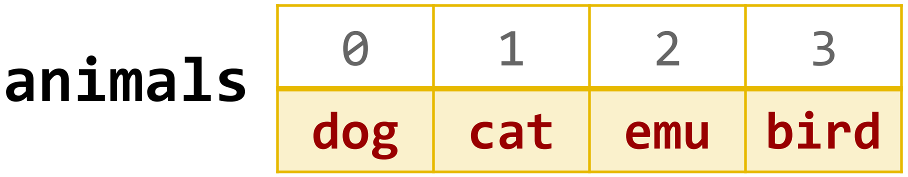
- Syntax

```
someList[startPosition:endPosition]
```

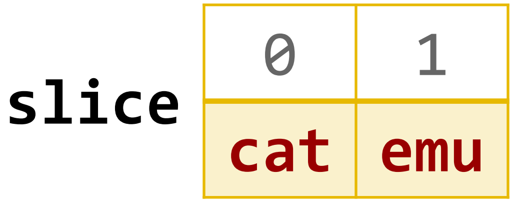
Access from start position

*Go **UP TO BUT NOT INCLUDING** end position*

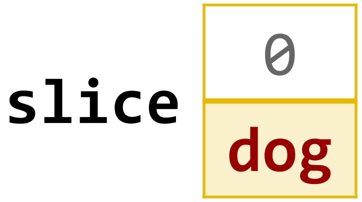
Accessing Lists by Slicing



```
slice = animals[1:3]
```

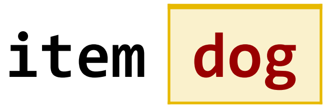


```
slice = animals[0:1]
```



Slice gives you a list

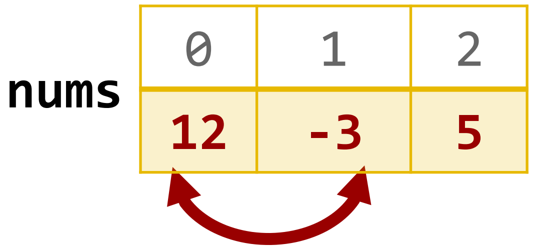
```
item  = animals[0]
```



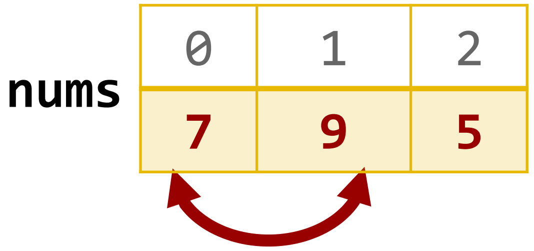
Index gives you one item

Changing Lists by Slicing

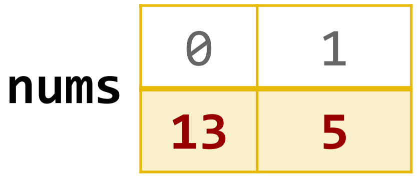
```
nums        = [12, -3, 5]
```



```
nums[0:2] = [7,9]
```



```
nums[0:2] = [13]
```



Slice assignment requires value on right to be a list

Useful Slicing Tricks

0	1	2	3
dog	cat	emu	bird

- Start at beginning

```
print(animals[:2])
```

```
[dog, cat]
```

- Go to end

```
print(animals[1:])
```

```
[cat, emu, bird]
```

- Entire list

```
print(animals[:])
```

```
[dog, cat, emu, bird]
```

Reference only

Note about Slicing Lists

- What is the difference between **a** and **b**?

```
drinks = ["tea", "coffee"]

a = drinks

b = drinks[:]

print(a)
                                   ["tea", "coffee"]
print(b)
                                   ["tea", "coffee"]
```

Reference only

Note about Slicing

- What is the difference between **a** and **b**?

```
drinks = ["tea", "coffee"]

a = drinks

b = drinks[:]
```

- *This means* **a** *is linked to* **drinks**

- *This means* **b** *is NOT linked to* **drinks**
- *This means* **b** *is a copy of* **drinks**

We will revisit this later