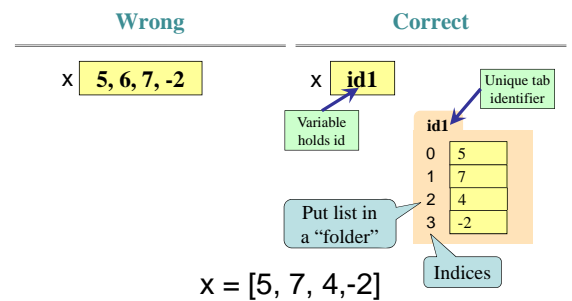## Lecture 10 Announcements

- Prelim 1
  - **Date:** Tuesday, March 14th, 7:30 pm to 9:00 pm
  - Submit conflicts immediately through CMS
- A2: You must scan or take a picture of your work to submit it through CMS
  - Since you have been warned to submit early, do not expect that we will accept work that does not make it onto CMS on time.

Representing Lists

Wrong	Correct

x   5, 6, 7, -2

x   **id1**

Variable holds id

Unique tab identifier

id1

0	5
1	7
2	4
3	-2

Put list in a "folder"

Indices

x = [5, 7, 4,-2]

Sequences: Lists of Values

String	List

String

- s = 'abc d'

0	1	2	3	4
a	b	c		d

- Put characters in quotes
  - Use \' for quote character
- Access characters with []
  - s[0] is 'a'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'

List

- x = [5, 6, 5, 9, 15, 23]

0	1	2	3	4	5
5	6	5	9	15	23

- Put values inside [ ]
  - Separate by commas
- Access **values** with []
  - x[0] is 5
  - x[6] causes an error
  - x[0:2] is [5, 6] (excludes 2nd 5)
  - x[3:] is [9, 15, 23]

Lists vs. Class Objects

List	Objects

List

- Attributes are indexed
  - Example: x[2]

x   **id2**

id2

list

0	5
1	7
2	4
3	-2

Objects

- Attributes are named
  - Example: c.x

c   **id3**

id3

Point3

x	1.0
y	2.0
z	3.0

Lists Have Methods Similar to String

x = [5, 6, 5, 9, 15, 23]

But you get length of a list with a regular function, not method:

len(x)

- <list>.index(<value>)
  - Return position of the value
  - **ERROR** if value is not there
  - x.index(9) evaluates to 3
- <list>.count(<value>)
  - Returns number of times value appears in list
  - x.count(5) evaluates to 2

List Assignment

- **Format**:

  <var>[<index>] = <value>

  - Reassign at index
  - Affects folder contents
  - Variable is unchanged
- Strings cannot do this
  - s = 'Hello World!'
  - s[0] = 'J' **ERROR**
  - String are **immutable**

- x = [5, 7,4,-2]

0	1	2	3
5	✗	4	-2

  8

- x[1] = 8

x   **id1**

id1

0	5
1	✗ 8
2	4
3	-2

List Methods Can Alter the List

x = [5, 6, 5, 9]

See Python API for more

- <list>.append(value)
  - A procedure, not a fruitful method
  - Adds a new value to the end of list
  - x.append(-1) *changes* the list to [5, 6, 5, 9, -1]
- <list>.insert(index, value)
  - Put the value into list at index; shift rest of list right
  - x.insert(2,-1) changes the list to [5, 6, -1, 5, 9,]
- <list>.sort()   What do you think this does?

Exercise Time

- Execute the following:
  ```
  >>> x = [5, 6, 5, 9, 10]
  >>> x[3] = -1
  >>> x.insert(1,2)
  ```
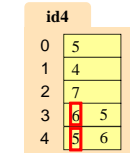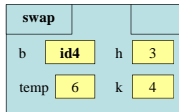- What is x[4]?

- Execute the following:
  ```
  >>> x = [5, 6, 5, 9, 10]
  >>> y = x[1:]
  >>> y[0] = 7
  ```
- What is x[1]?

Lists and Functions: Swap

```
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
    Precondition: b is a mutable list, h
    and k are valid positions in the list"""
1   temp= b[h]
2   b[h]= b[k]
3   b[k]= temp
```

swap(x, 3, 4)

Swaps b[h] and b[k], because parameter b contains name of list.

swap

| b | id4 | h | 3 |
| temp | 6 | k | 4 |

id4

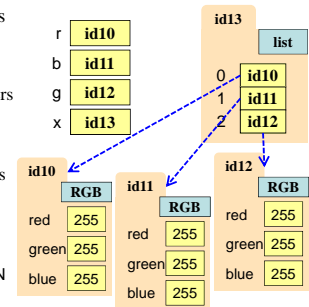| 0 | 5 |
1	4
2	7
3	6	5
4	5	6

x | id4

Lists of Objects

- List positions are variables
  - Can store base types
  - But cannot store folders
  - Can store folder identifiers
- Folders linking to folders
  - Top folder for the list
  - Other folders for contents
- Example:
  ```
  >>> r = colormodel.RED
  >>> b = colormodel.BLUE
  >>> g = colormodel.GREEN
  >>> x = [r,b,g]
  ```
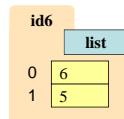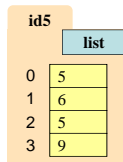
r | id10
b | id11
g | id12
x | id13

id13

	list
0	id10
1	id11
2	id12

id10

RGB
red	255
green	255
blue	255

id11

RGB
red	255
green	255
blue	255

id12

RGB
red	255
green	255
blue	255

List Slices Make Copies

x = [5, 6, 5, 9]   y = x[1:3]

x | id5   y | id6

id5

	list
0	5
1	6
2	5
3	9

id6

	list
0	6
1	5

copy = new folder