Using Color Objects in A3

- New classes in `colormodel`
  - RGB, CMYK, and HSV
- Each has its own attributes
  - RGB: red, blue, green
  - CMYK: cyan, magenta, yellow, black
  - HSV: hue, saturation, value
- Attributes have invariants
  - Limits the attribute values
  - Example: red is an int in 0..255
  - Get an error if you violate

```python
>>> import colormodel
>>> c = colormodel.RGB(128,0,0)
>>> r = c.red
```

How to Do the Conversion Functions

```python
def rgb_to_cmyk(rgb):
    # Returns: color rgb in space CMYK
    # Precondition: rgb is an RGB object
    # DO NOT CONSTRUCT AN RGB OBJECT
    # Variable rgb already has RGB object
    # 1. Access attributes from rgb folder
    # 2. Plug into formula provided
    # 3. Compute the new cyan, magenta, etc. values
    # 4. Construct a new CMYK object
    # 5. Return the newly constructed object
```

Sequences: Lists of Values

<table>
<thead>
<tr>
<th>String</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>s = 'abc d'</td>
<td>x = [5, 6, 9, 15, 23]</td>
</tr>
</tbody>
</table>
- Put characters in quotes
  - Use \' for quote character
- Access characters with []
  - s[0] is 'a'
  - s[3] causes an error
  - s[3] is 'c' (excludes c)
- s[4] is ' d'

Lists Have Methods Similar to String

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

Lists are Mutable

- Can alter their contents
  - Use an assignment:
    ```python
    <var>[<index>] = <value>
    ```
  - Index is position, not slice
- Does not work for strings
  - s = 'Hello World!'
  - s[0] = 'H' ERROR
- Represent list as a folder
  - Variable holds tab name
  - Contents are attributes
  - x[1] = 8

When Do We Need to Draw a Folder?

- When the value contains other values
  - This is what we are calling 'objects'
- When the value is mutable

<table>
<thead>
<tr>
<th>Type</th>
<th>Container?</th>
<th>Mutable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>float</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>str</td>
<td>Yes*</td>
<td>No</td>
</tr>
<tr>
<td>Point</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RGB</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>list</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Lists vs. Class Objects

### List
- Attributes are indexed
  - Example: `x[2]`

### RGB
- Attributes are named
  - Example: `c.red`

---

List Methods Can Alter the List

- **append(value)**
  - A **procedure method**, not a function method
  - Adds a new value to the end of list
  - `x.append(-1)` changes the list to `[5, 6, 5, 9, -1]`

- **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,]`

- **sort()**
  - What do you think this does?

---

Lists and Functions: Swap

```python
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"
    temp= b[h]  
    b[h]= b[k] 
    b[k]= temp 
swap(x, 3, 4)
```

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

---

List Slices Make Copies

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

---

Lists and Expressions

- List brackets `[]` can contain expressions
- This is a list expression
  - Python must evaluate it
  - Evaluates each expression
  - Puts the value in the list
- Example:
  ```python
  >>> a = [1+2,3+4,5+6]
  >>> b = [7]
  >>> x = [a, b, a+b]
  >>> x[2]
  12
  ```

---

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:
  ```python
  >>> r = colormodel.RED
  >>> b = colormodel.BLUE
  >>> g = colormodel.GREEN
  >>> x = [r, b, g]
  ```

---

Lists and Functions: Swap

Swaps `b[h]` and `b[k]` in `b`.

Precondition: `b` is a mutable list, `h` and `k` are valid positions in the list.

```python
def swap(b, h, k):
    temp= b[h]  
    b[h]= b[k] 
    b[k]= temp 
```

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

---

List Slices Make Copies

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

---

Lists and Expressions

- List brackets `[]` can contain expressions
- This is a list expression
  - Python must evaluate it
  - Evaluates each expression
  - Puts the value in the list
- Example:
  ```python
  >>> a = [1+2,3+4,5+6]
  >>> b = [7]
  >>> x = [a, b, a+b]
  ```

---

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:
  ```python
  >>> r = colormodel.RED
  >>> b = colormodel.BLUE
  >>> g = colormodel.GREEN
  >>> x = [r, b, g]
  ```