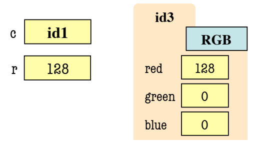
Using Color Objects in A3

- 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 int in 0..255
  - Get an error if you violate



```
>>> import colormodel
>>> c = colormodel.RGB(128,0,0)
>>> r = c.red
>>> c.red = 500 # out of range
AssertionError: 500 outside [0,255]
```

How to Do the Conversion Functions

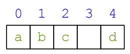
```
def rgb_to_cmyk(rgb):
    """Returns: color rgb in space CMYK
    Precondition: rgb is an RGB object"""
    # DO NOT CONSTRUCT AN RGB OBJECT
    # Variable rgb already has RGB object
    # 1. Access attributes from rgb folder
    # 2. Plug into formula provided
    # 3. Compute the new cyan, magenta,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

Only time you will ever call a constructor

Sequences: Lists of Values

String

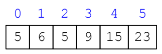
- s = 'abc d'



- 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]



- 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 Have Methods Similar to String

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

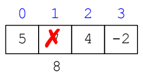
- `index(value)`
  - Return position of the value
  - **ERROR** if value is not there
  - `x.index(9)` evaluates to 3
- `count(value)`
  - Returns number of times value appears in list
  - `x.count(5)` evaluates to 2

But you get length of a list with a regular function, not method: `len(x)`

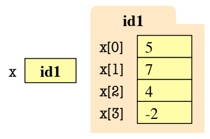
Lists are Mutable

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

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



- x[1] = 8



When Do We Need to Draw a Folder?

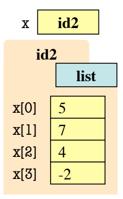
- When the value **contains** other values
  - This is what we are calling 'objects'
- When the value is **mutable**

Type	Container?	Mutable?
int	No	No
float	No	No
str	Yes*	No
Point	Yes	Yes
RGB	Yes	Yes
list	Yes	Yes

Lists vs. Class Objects

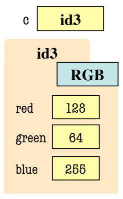
List

- Attributes are indexed
  - Example: x[2]



RGB

- Attributes are named
  - Example: c.red



List Methods Can Alter the List

x = [5, 6, 5, 9]

See Python API for more

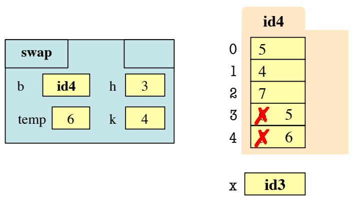
- append(value)
  - A **procedure method**, not a function method
  - Adds a new value to the end of list
  - x.append(-1) *changes* the list to [5, 6, 5, 9, -1]
- insert(index, value)
  - Put the value into list at index; shift rest of list right
  - x.insert(2,-1) changes the list to [5, 6, -1, 5,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

Lists and Functions: Swap

```
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
```

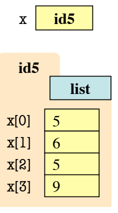
Swaps b[h] and b[k], because parameter b contains name of list.

swap(x, 3, 4)

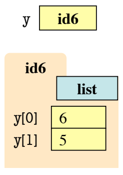


List Slices Make Copies

x = [5, 6, 5, 9]



y = x[1:3]



copy = new folder

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:

```
>>> a = [1+2,3+4,5+6]
>>> a
[3, 7, 11]
```

- Execute the following:

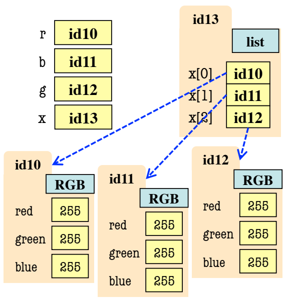
```
>>> a = 5
>>> b = 7
>>> x = [a, b, a+b]
```

- What is x[2]?

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
x[0] id10
x[1] id11
x[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