Lecture 10

Lists (& Sequences)

Announcements for Today

Reading

- Read 10.0-10.2, 10.4-10.6
- Read 5.8 – 5.10 for Tue

- **Prelim, Oct 4th 7:30-9:30**
  - Material up to next Tuesday
  - Study guide next week
- **Conflict with Prelim time?**
  - Submit to Prelim 1 Conflict assignment on CMS
  - Do not submit if no conflict

Assignments

- Assignment 2 Today
  - Hand in at end of class
  - Or scan and put in CMS
  - Put file size must be < 1MB
- Assignment 3 posted
  - Due in two stages
  - Part 1 due Oct. 1 (pass/fail)
  - Part 2 due Oct. 11 (graded)
  - Get help now if you need it

Using Color Objects in A3

- Most types have literals: symbols for values
  - float literals: `1.0`, `-2.3`, `2.34e-30`
  - string literals: `'Hello'`, `'1125kba,re'`
- Mutable objects do not have literals
- Make a mutable object with a **constructor**
  - Function that returns a new version of object
  - Function name is the same as the type name
  - **Example**: `RGB(255,0,0)` makes a red color
- Access components with **attributes**: `rgb.red`

Sequences: Lists of Values

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]

Sequences: Lists of Values

String

- s = 'abc d'

0	1	2	3	4
a	b	c		d

- Put characters in quotes
  - Use \' for quote character
- Access ch
  - s[0] is 'a
  - s[5] causes
  - 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 [ ]
  - mmas
- with []
  - [0] is 5
  - x[6] causes an error
  - x[0:2] is [5, 6] (excludes 2nd 5)
  - x[3:] is [9, 15, 23]

Sequence is a name we give to both

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:

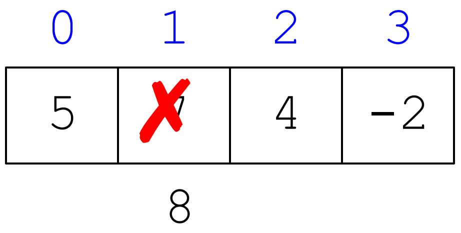
`len(x)`

Lists are Mutable

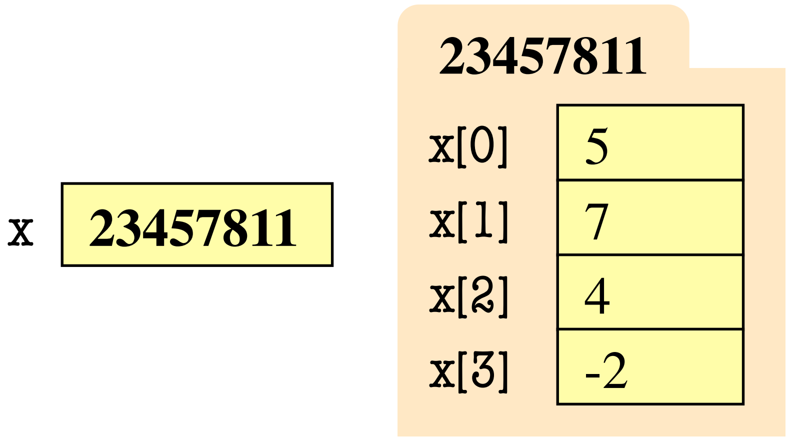
- Can alter their contents
  - Use an assignment:

  <*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?

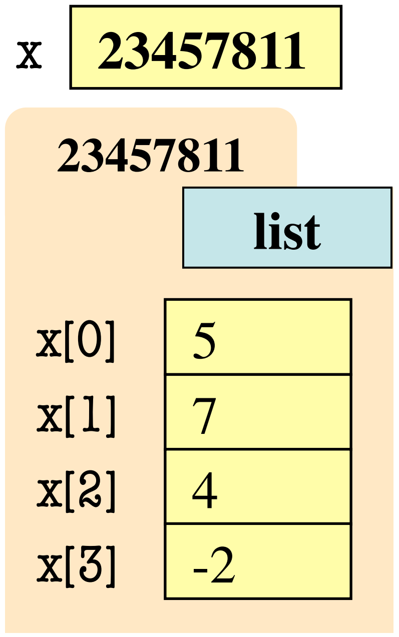
- When the value **contains** other values
  - This is what we are calling ‘objects’
- When the value is **mutable**

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

Lists vs. Custom Objects

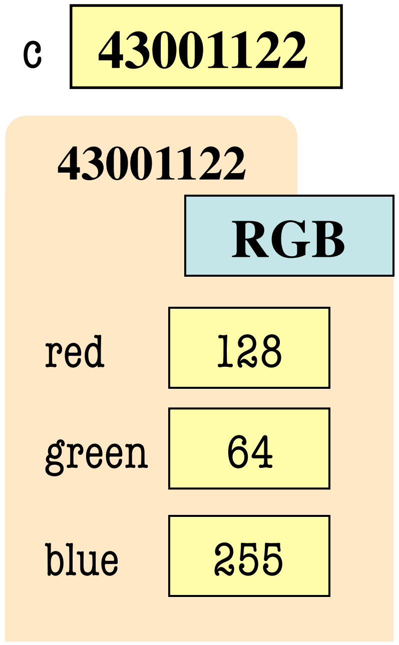
List

- 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

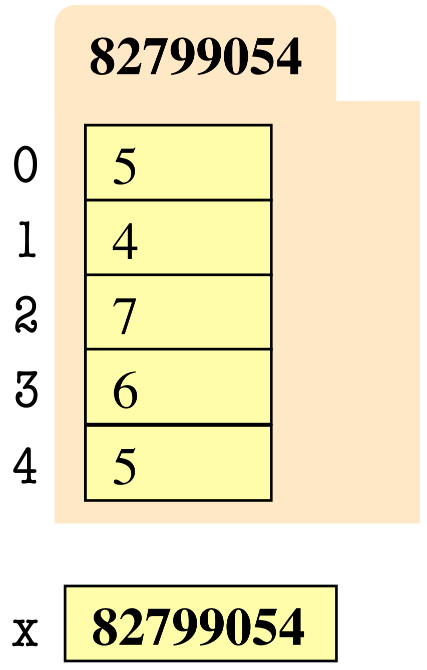
- `append(value)`
  - A **procedure method**, not a fruitful method
  - Adds a new value to the end of list
  - `x.append(-1)` *changes* the list to `[5, 6, 5, 9, -1]`
- `insert(index, value)`
  - Put the value into list at index; shift rest of list right
  - `x.insert(2,-1)` changes the list to `[5,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

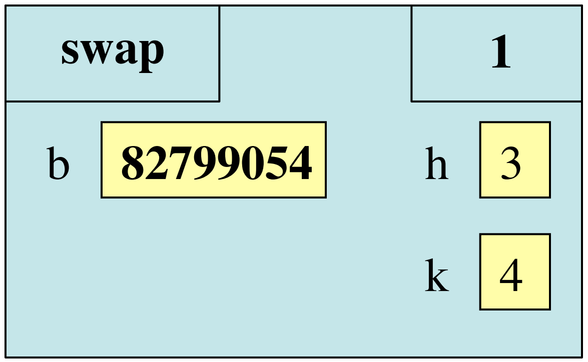
What do you think this does?

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"""
1   temp= b[h]
2   b[h]= b[k]
3   b[k]= temp
```

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





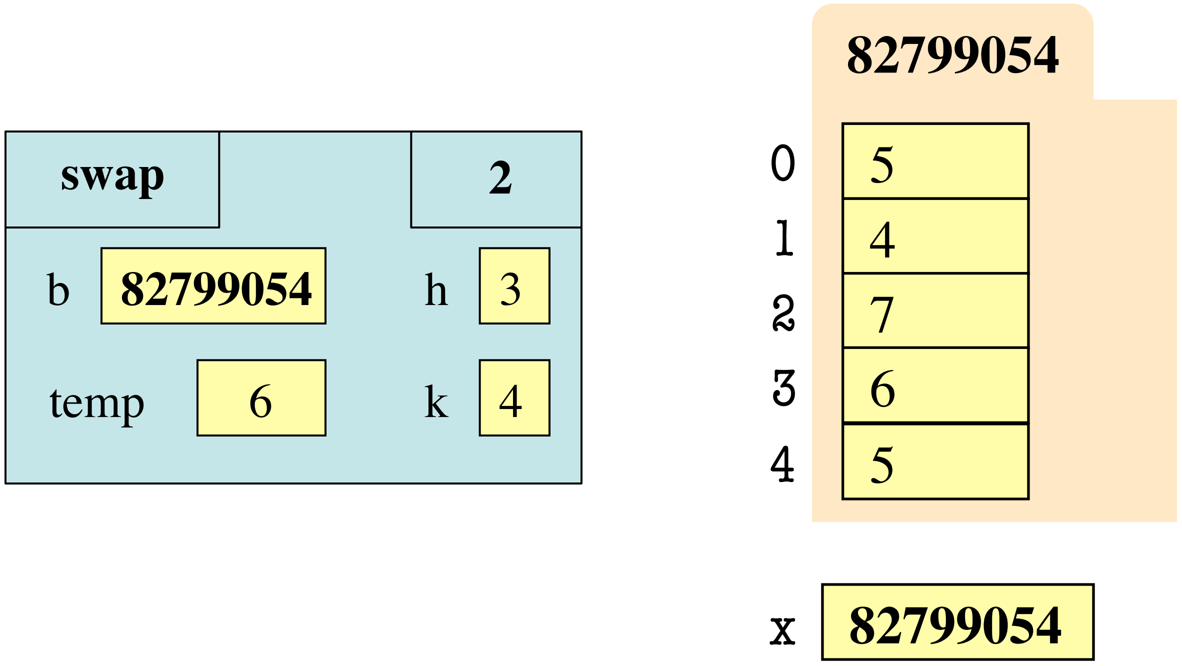
swap(x, 3, 4)

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
```

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

swap(x, 3, 4)

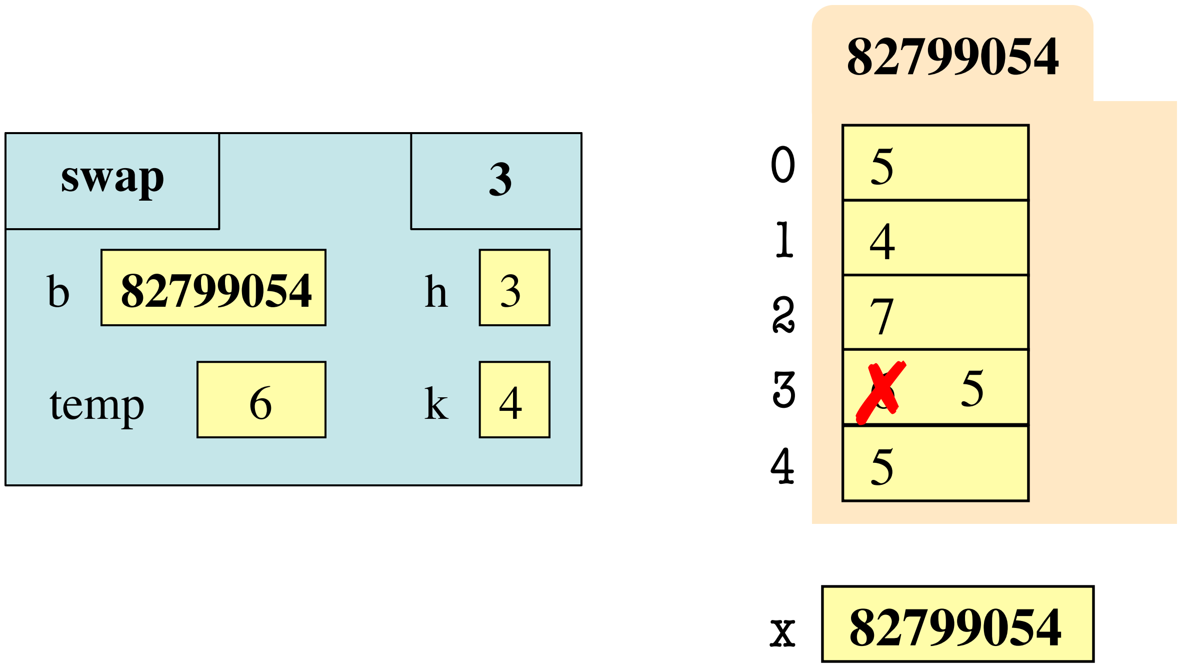


Lists and Functions: Swap

```
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
```

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

swap(x, 3, 4)

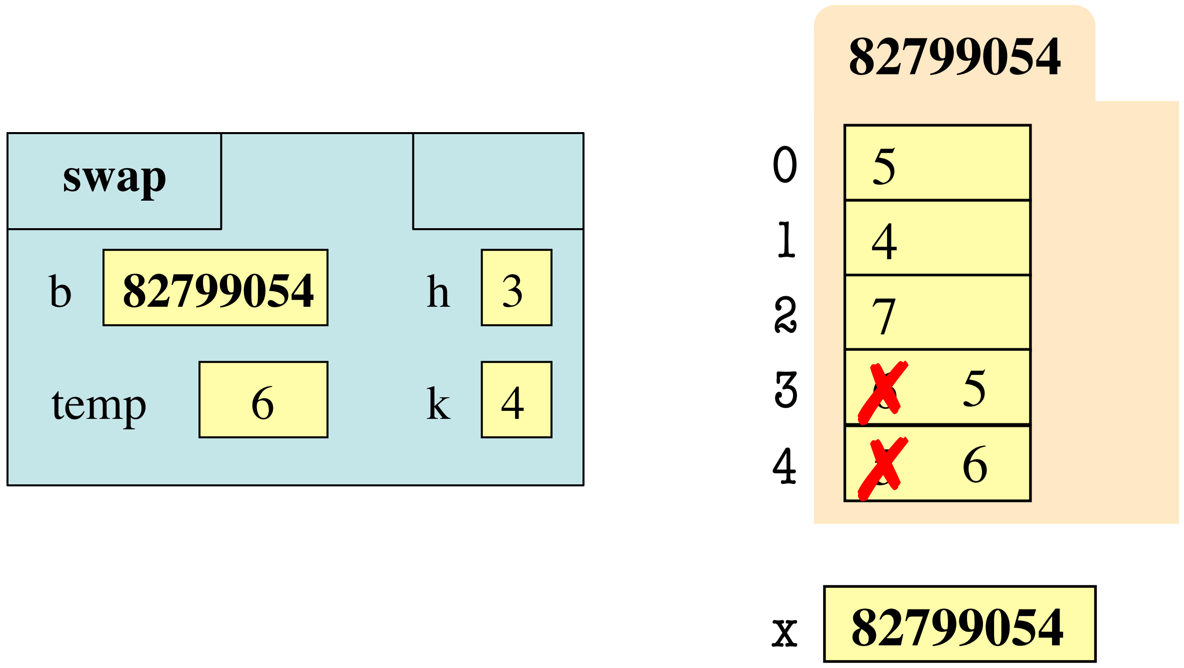


Lists and Functions: Swap

```
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.

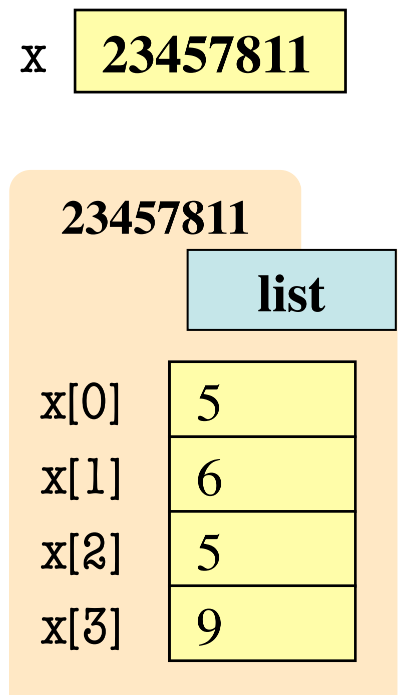
swap(x, 3, 4)

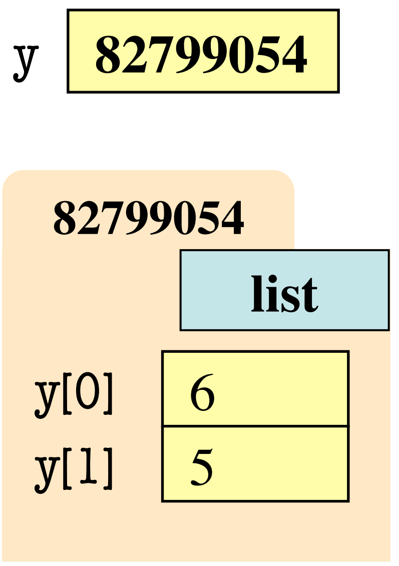


List Slices Make Copies

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

```
y = x[1:3]
```





copy = new folder

Exercise Time

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> x[3] = -1
>>> x.insert(1,2)
```

- What is x[4]?

A: 10
B: 9
C: -1
D: **ERROR**
E: I don't know

Exercise Time

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> x[3] = -1
>>> x.insert(1,2)
```

- What is x[4]?

-1

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> y = [1:]
>>> y[0] = 7
```

- What is x[1]?

A: 7
B: 5
C: 6
D: **ERROR**
E: I don't know

Exercise Time

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> x[3] = -1
>>> x.insert(1,2)
```

- What is x[4]?

-1

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> y = [1:]
>>> y[0] = 7
```

- What is x[1]?

6

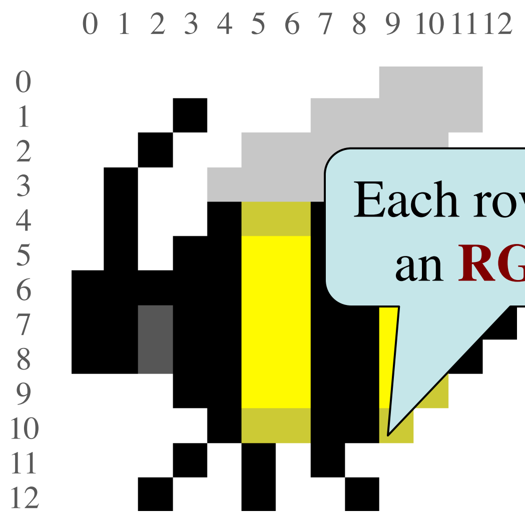
Two Dimensional Lists

Table of Data

	0	1	2	3
0	5	4	7	3
1	4	8	9	7
2	5	1	2	3
3	4	1	2	9
4	6	7	8	0

Each row, col has a value

Images



Each row, col has an **RGB** value

Store them as lists of lists (**row-major order**)

```
d = [[5,4,7,3],[4,8,9,7],[5,1,2,3],[4,1,2,9],[6,7,8,0]]
```

Overview of Two-Dimensional Lists

- Access value at row 3, col 2:

  `d[3][4]`

- Assign value at row 3, col 2:

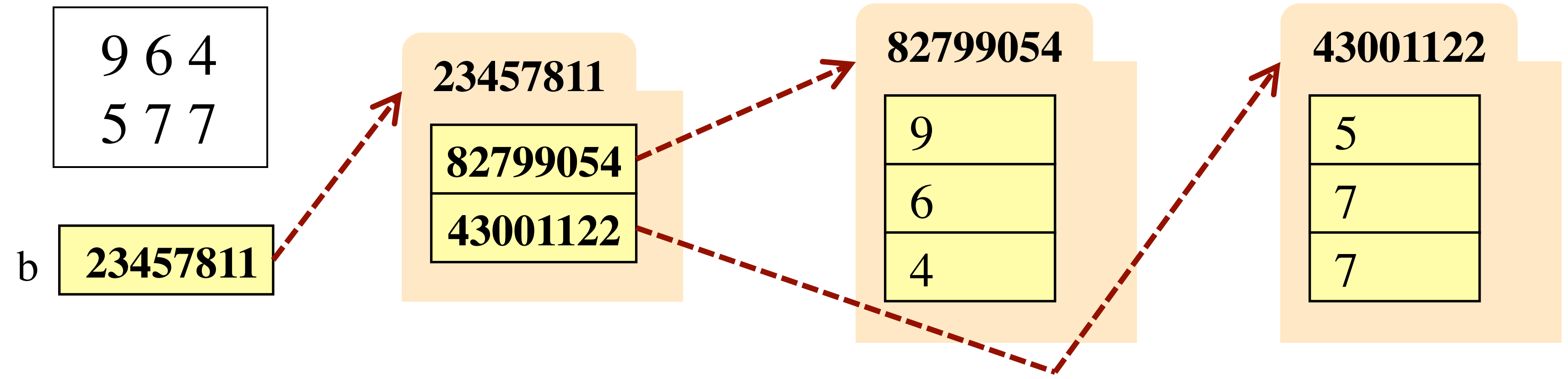
  `d[3][2] = 8`

- **An odd symmetry**
  - Number of rows of d: `len(d)`
  - Number of cols in row r of d: `len(d[r])`

$$\mathtt{d} \quad \begin{array}{c|cccc|} & 0 & 1 & 2 & 3 \\ \hline 0 & 5 & 4 & 7 & 3 \\ 1 & 4 & 8 & 9 & 7 \\ 2 & 5 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 & 9 \\ 4 & 6 & 7 & 8 & 0 \\ \hline \end{array}$$

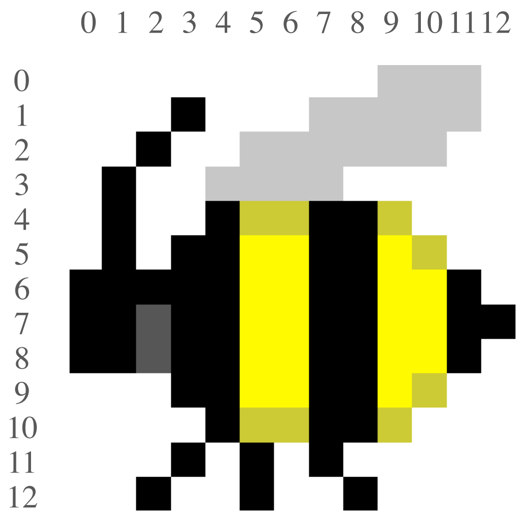
How Multidimensional Lists are Stored

- `b = [[9, 6, 4], [5, 7, 7]]`

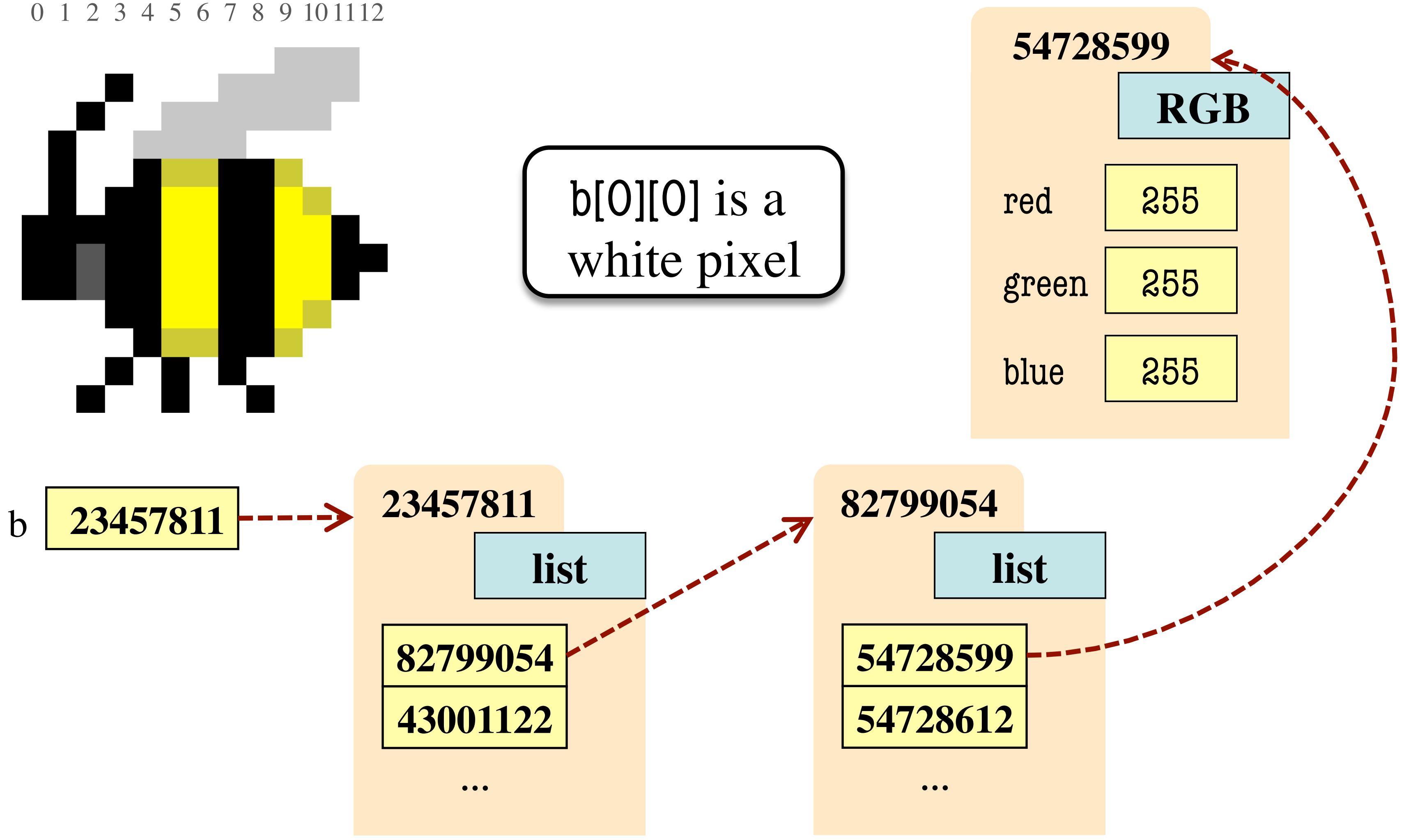


- b holds name of a one-dimensional list
  - Has len(b) elements
  - Its elements are (the names of) 1D lists
- b[i] holds the name of a one-dimensional list (of ints)
  - Has len(b[i]) elements

Image Data: 2D Lists of Pixels

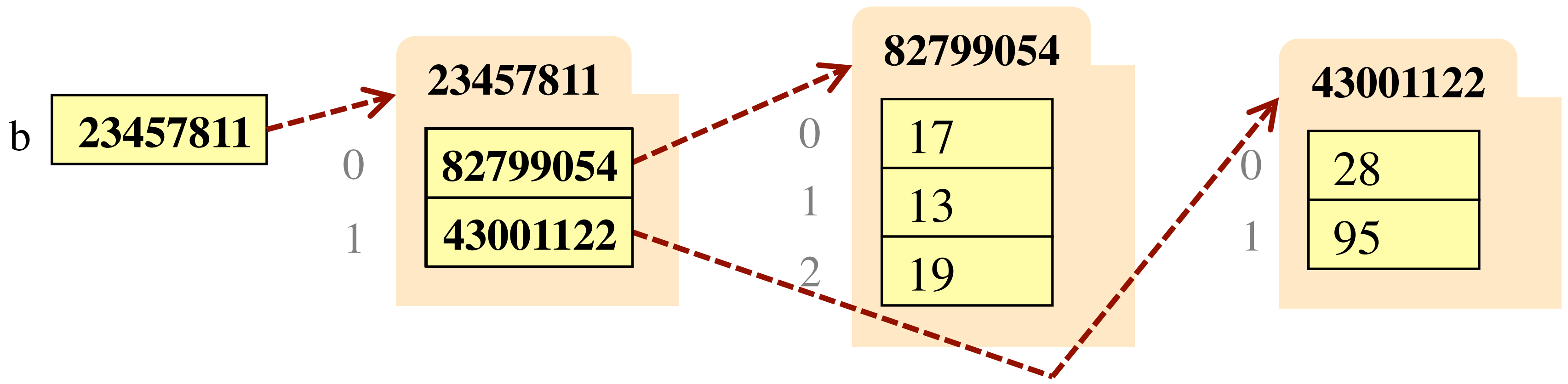


b[0][0] is a white pixel



Ragged Lists: Rows w/ Different Length

- `b = [[17,13,19],[28,95]]`

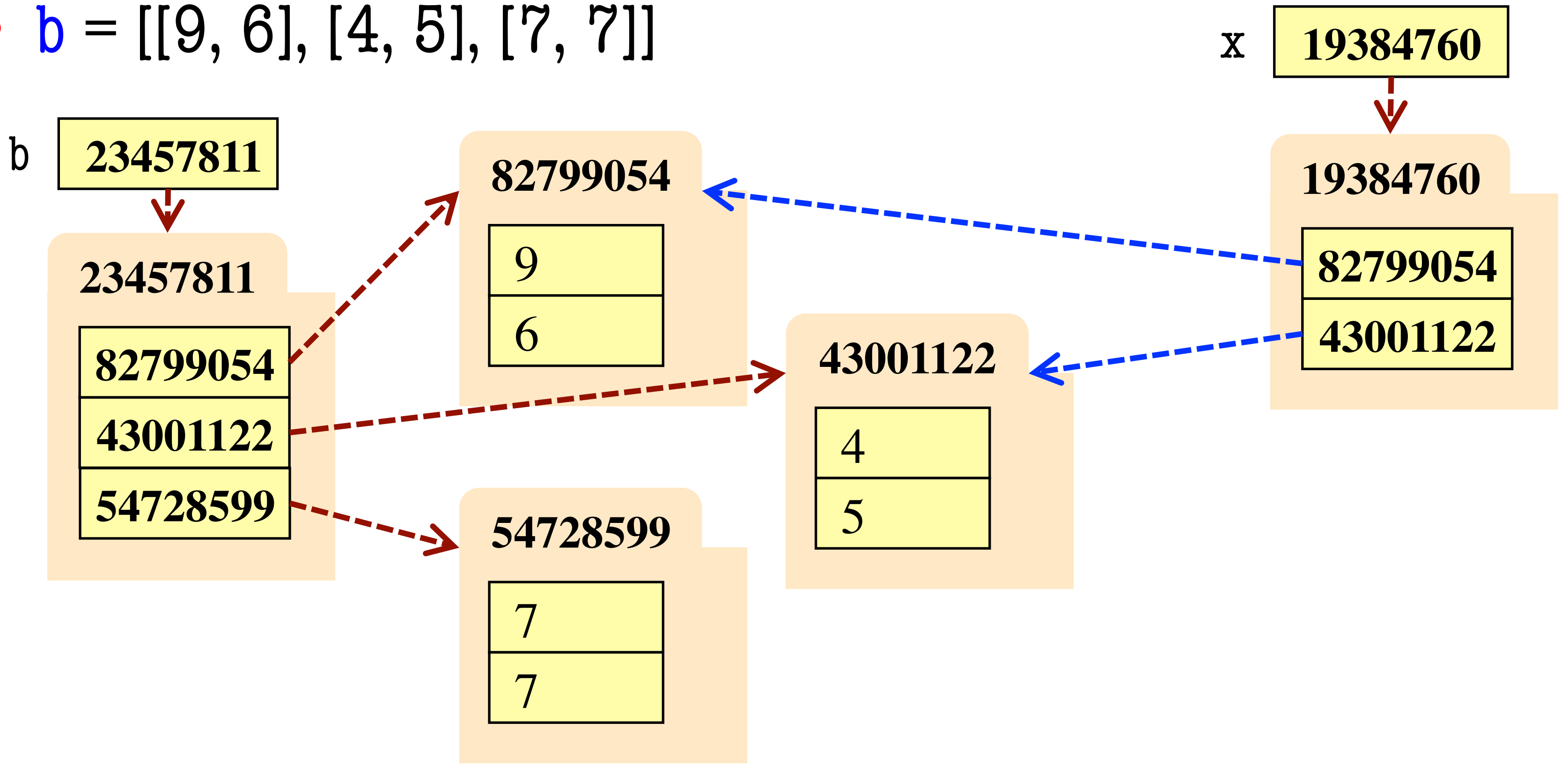


- Will see applications of this later

Slices and Multidimensional Lists

- Only "top-level" list is copied.
- Contents of the list are not altered
- `b = [[9, 6], [4, 5], [7, 7]]`

`x = b[:2]`



Slices and Multidimensional Lists

- Create a 2D List

```
>>> b = [[9,6],[4,5],[7,7]]
```

- Get a slice

```
>>> x = b[:2]
```

- Append to a row of x

```
>>> x[1].append(10)
```

- x now has the 2D list

```
[[9, 6], [4, 5, 10]]
```

- What are the contents of the list (with name) in b?

A: [[9,6],[4,5],[7,7]]
B: [[9,6],[4,5,10]]
C: [[9,6],[4,5,10],[7,7]]
D: [[9,6],[4,10],[7,7]]
E: I don't know