Processing lists: builtins

- `sum(x)` adds up all the elements in the list `x`
  - they had better be numbers!
- `min(x)` or `max(x)` find the minimum resp. maximum value in the list `x`
  - they use the same ordering as sort()
- `range(`*n*`)` produces [0, 1, 2, …, *n*]
  - optional arguments to start somewhere other than zero
- `list(x)` converts `x` (a string for example) to a list
  - e.g. `list('mimsy')` produces [“m”, “i”, “m”, “s”, “y”]

Processing lists: The `map` Function

General form: `map(`⟨*function*⟩, ⟨*list*⟩`)`

if `x` is a list of *n* items and
`f` is a function with one parameter:

`map(f, x)`

⇩

`[f(x[0]), f(x[1]), ..., f(x[n–1])]`

calls the function once for each list item

if `x` is a list of *n* items and
`m` is a method with no parameters:

`map(m, x)`

⇩

`[x[0].m(), x[1].m(), ..., x[n–1].m()]`

examples:

`map(len, ['a', 'bc', 'defg'])` produces [1, 2, 4]
`map(str.strip, ['a ', ' bc', ' defg '])` produces [“a”, “bc”, “defg”]

Processing lists: The `for` Statement

General form:

for ⟨*variable*⟩ in ⟨*list*⟩:
    ⟨*statements*⟩

executes the body once for each list item

```
for a in x:
    print 3 * a
```

⇨

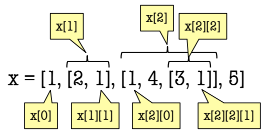
```
print 3 * x[0]
print 3 * x[1]
print 3 * x[2]
...
print 3 * x[n–1]
```

when the body is executed, the value of `a` is the current list item

Nested Lists

- Lists can hold any objects
- Lists are objects
- Therefore lists can hold other lists!

```
a = [2, 1]
b = [3, 1]
c = [1, 4, b]
x = [1, y, z, 5]
```



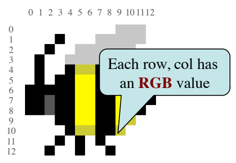
Two Dimensional Lists

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



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:

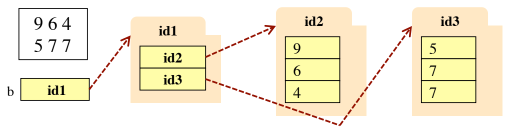
  `d[3][2]`
- Assign value at row 3, col 2:

  `d[3][2] = 8`
- Getting array dimensions:
  - Number of rows of d: `len(d)`
  - Number of cols in row r of d: `len(d[r])`

d	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

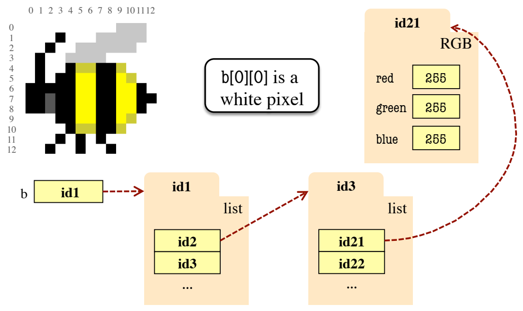
How Multidimensional Lists are Stored

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



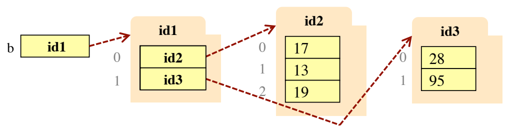
- b holds name of a one-dimensional list
  - Has len(b) elements
  - Its elements are (the names of) 1D lists
- b[i] holds the name of a one-dimensional list (of ints)
  - Has len(b[i]) elements

Image Data: 2D Lists of Pixels



Ragged Lists: Rows w/ Different Length

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

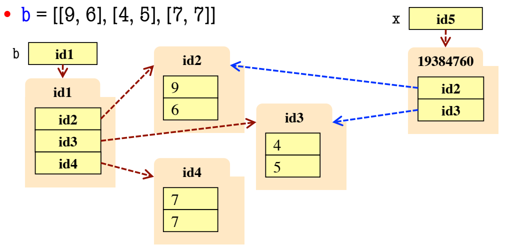


- Will see applications of this later

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