http://www.cs.cornell.edu/courses/cs1110/2020sp

Lecture 12: Nested Lists, Tuples, and Dictionaries

(Sections 11.1-11.5, 12.1-12)

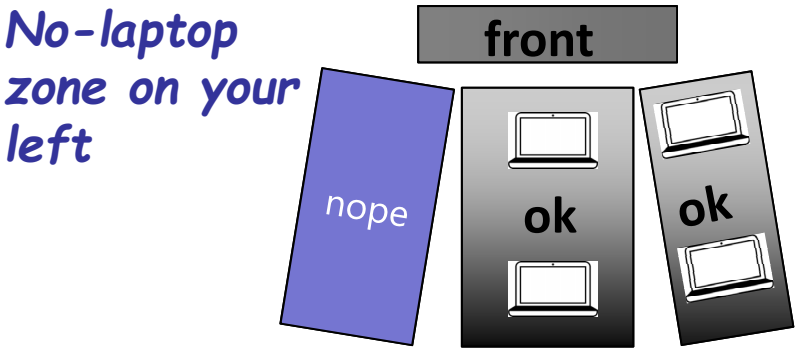
CS 1110

Introduction to Computing Using Python

Dictionaries (type `dict`) will be discussed another time and will not be on Prelim 1

[E. Andersen, A. Bracy, D. Fan, D. Gries, L. Lee,
S. Marschner, C. Van Loan, W. White]

Announcements on Assignments 2 & 3



- A3 out, in 2 parts, see email from CMS. If you visit office/consulting hrs for debugging help, the session will be more effective if you can show the staff what happens on specific test cases.
  - Mon Mar 2 update: minor spec update, more guidance on test cases; students should group on a3tests, staff will propagate grouping to a3fns. See email from CMS.
- A2 grades, solutions released. See email from Gradescope. Check that you are receiving email from Gradescope.

Announcements on Prelim 1

Prelim 1 (Mar 10) info on course website:

Resources → *In-House Resources* → *Exams information*

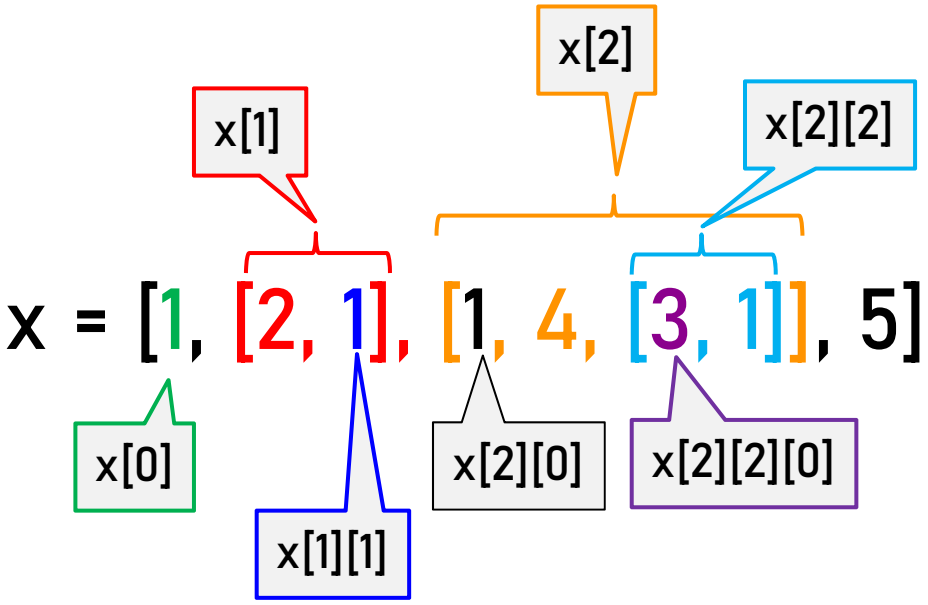
(https://www.cs.cornell.edu/courses/cs1110/2020sp/exams/)

- Read Prelim 1 Study Guide!
- Know your exam room; it is based on your NetID.
- Makeup requests will be answered via email by Friday Noon
- Next lecture will be a review session
- Mar 10 lecture time → office hours

Nested Lists

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

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



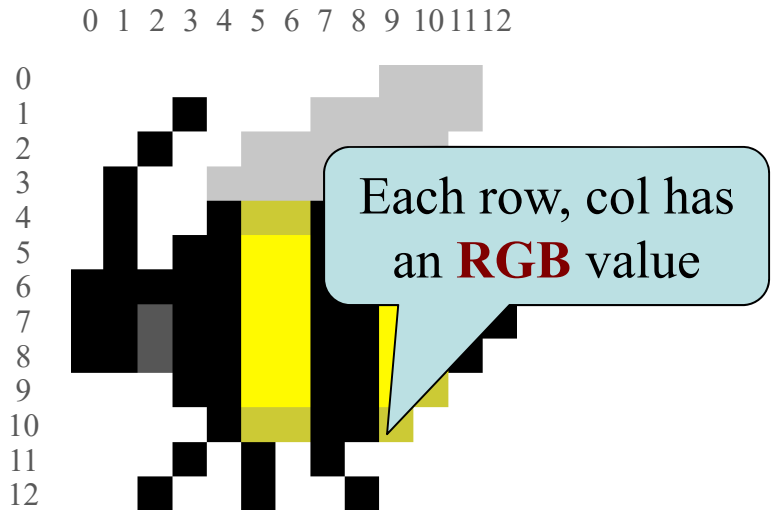
Two Dimensional Lists

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 a list of lists ("**row-major order**")

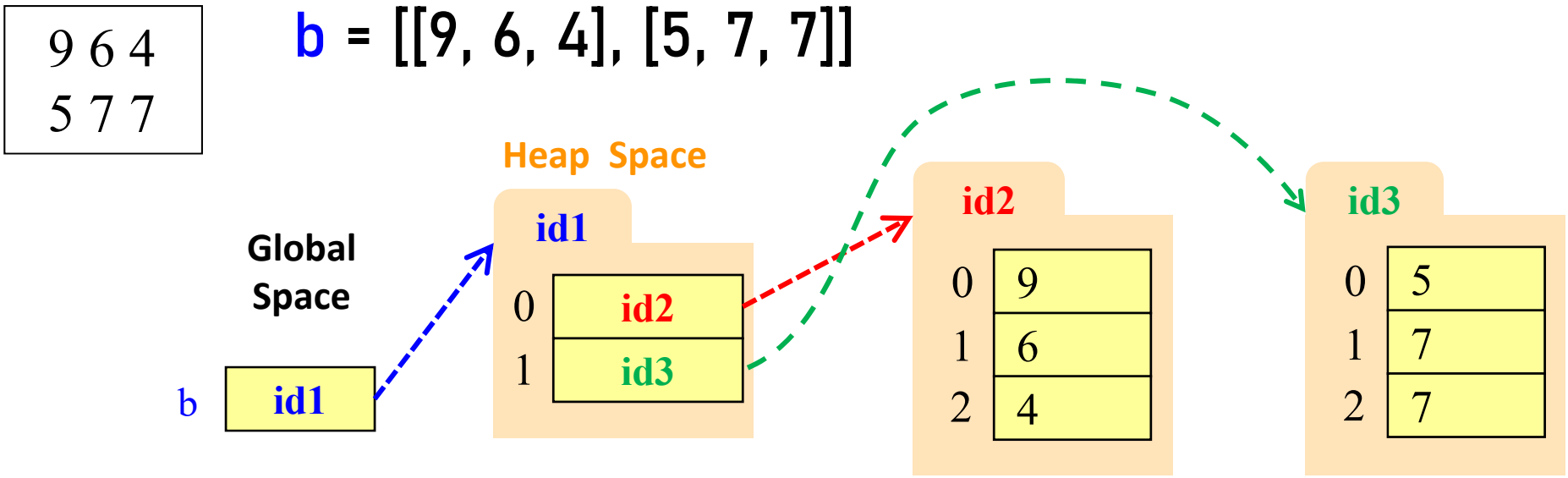
d = [[5,4,7,3],[4,8,9,7],[5,1,2,3],[4,1,2,9],[6,7,8,0]]

Overview of Two-Dimensional Lists

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

```
>>> d = [[5,4,7,3],[4,8,9,7],[5,1,2,3],[4,1,2,9]]
>>> d[3][2]                 Access value at row 3, col 2
2
>>> d[3][2] = 8             Assign value at row 3, col 2
>>> len(d)                  Number of rows of d
4
>>> len(d[2])               Number of cols in row 2 of d
4
>>> d
[[5, 4, 7, 3], [4, 8, 9, 7],[5, 1, 2, 3], [4, 1, 8, 9]]
```

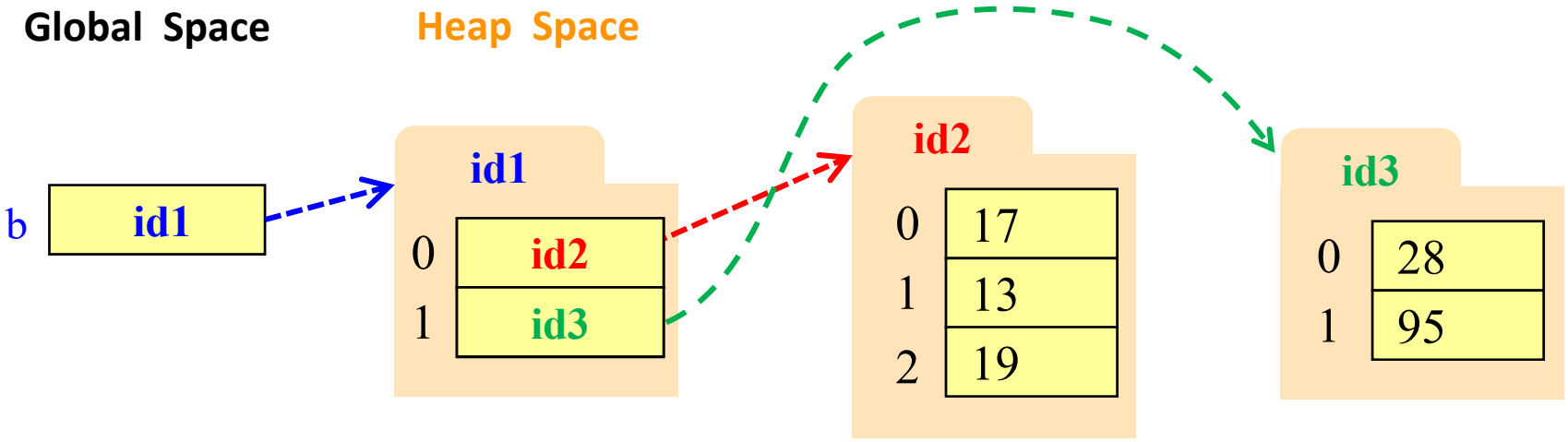
How Multidimensional Lists are Stored



- b holds **id** of a one-dimensional list
  - Has len(b) elements
- b[i] holds **id** of a one-dimensional list
  - Has len(b[i]) elements

Ragged Lists: Rows w/ Different Length

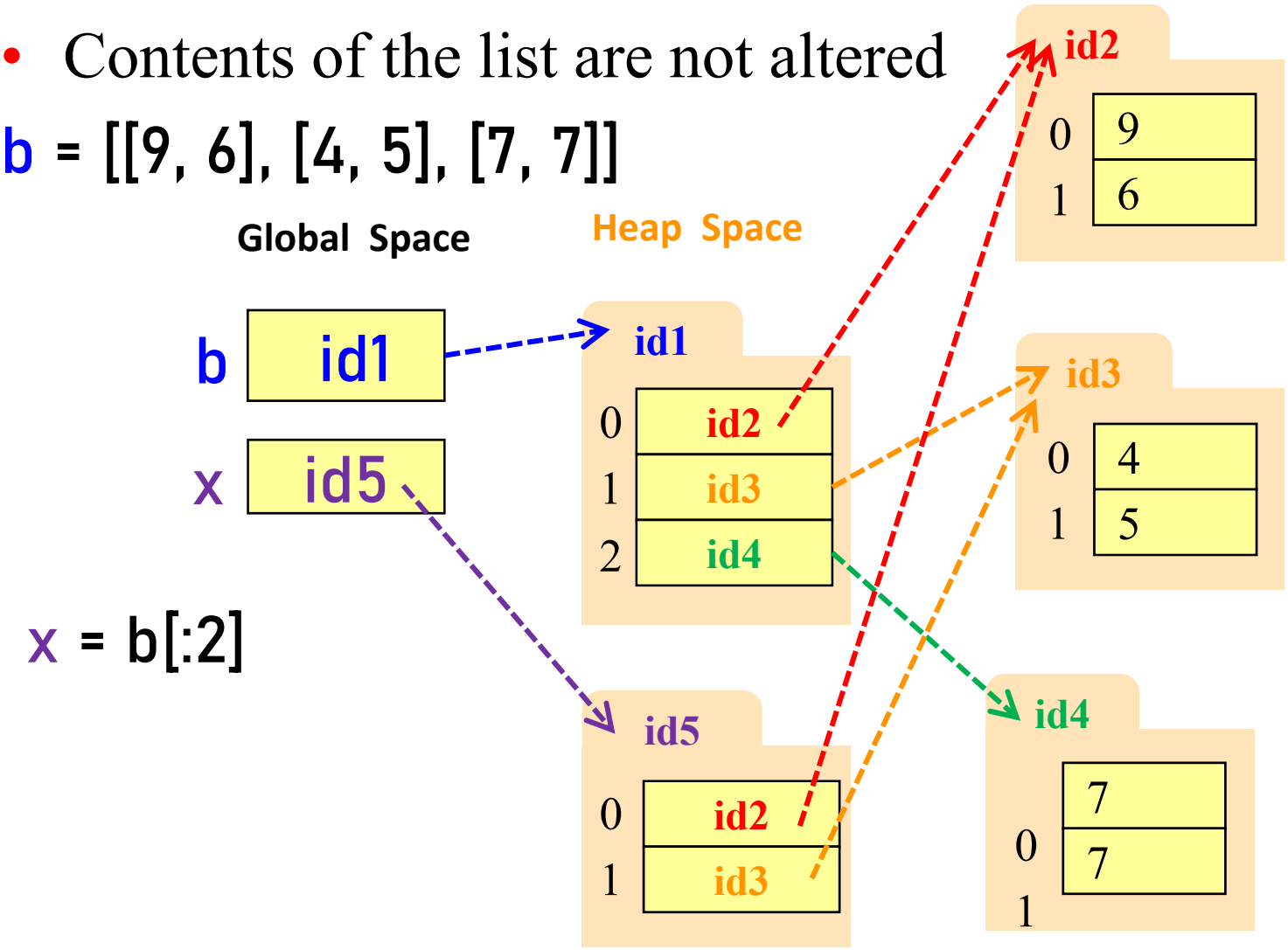
- b = [[17,13,19],[28,95]]



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 & Multidimensional Lists (Q1)

- Create a nested list

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

- Get a slice

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

- Append to a row of x

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

- What is now in x?

A: [[9,6,10]]
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

Slices & Multidimensional Lists (A1)

- Create a nested list

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

- Get a slice

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

- Append to a row of x

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

- What is now in x?

A: [[9,6,10]]
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

Slices & Multidimensional Lists (Q2)

- Create a nested 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 nested list

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

- What is now 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

Slices & Multidimensional Lists (A2)

- Create a nested list

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

- Get a slice

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

- Append to a row of x

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

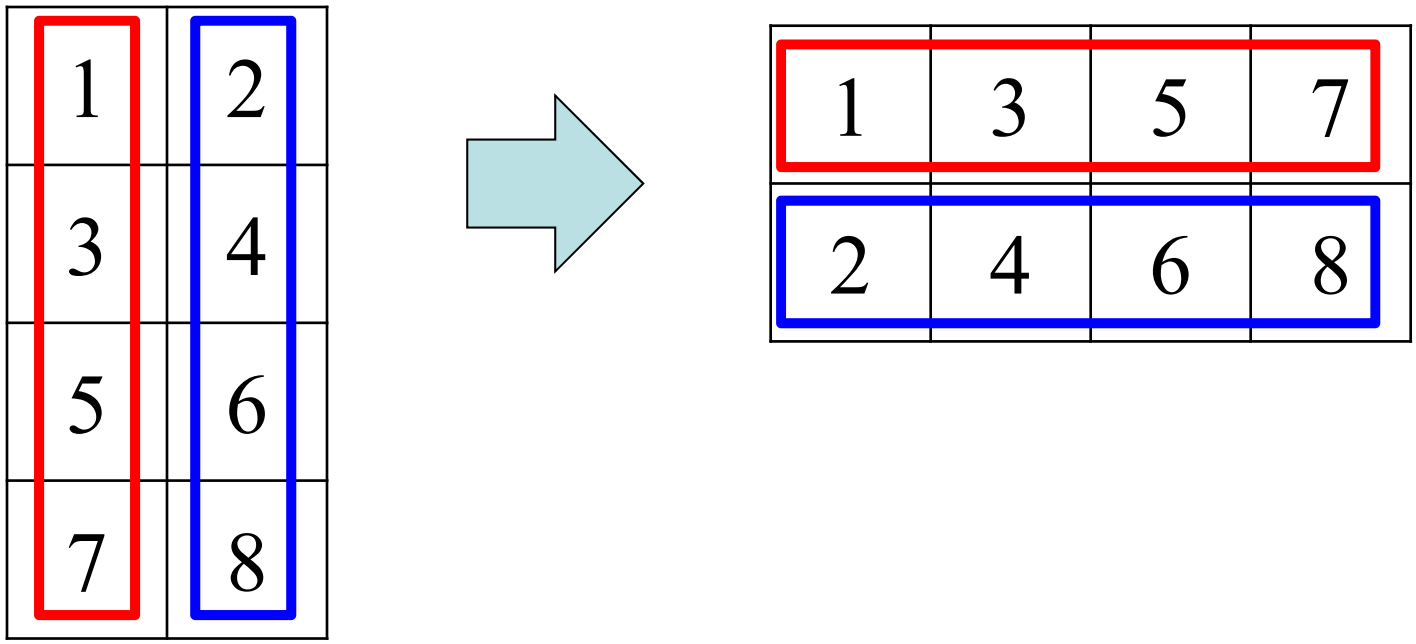
- x now has nested list

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

- What is now 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

Data Wrangling: Transpose Idea



4 lists: 2 elements in each 2 lists: 4 elements in each

How to transpose?

- 1st element of each list gets appended to 1st list
- 2nd element of each list gets appended to 2nd list

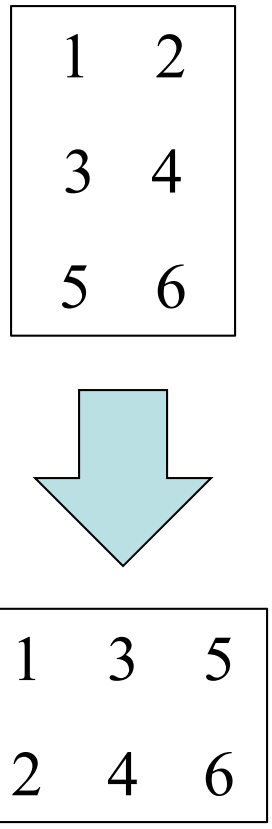
Data Wrangling: Transpose Code

```
def transpose(table):
    """Returns: copy of table with rows and columns swapped
    Precondition: table is a (non-ragged) 2d List"""
    n_rows = len(table)
    n_cols  = len(table[0]) # All rows have same no. cols
    new_table = [] # Result accumulator




    return new_table

d = [[1,2],[3,4],[5,6]]
d_v2 = transpose(d)
```

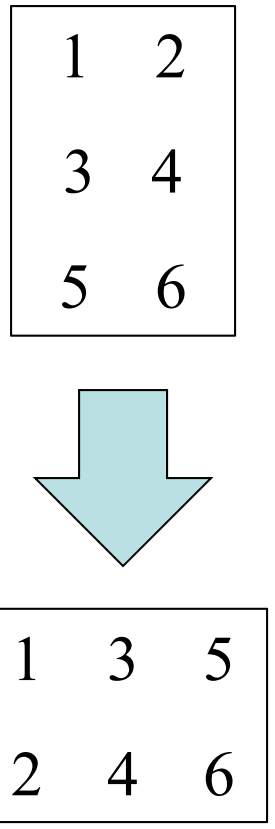


Data Wrangling: Transpose Code

```
def transpose(table):
    """Returns: copy of table with rows and columns swapped
    Precondition: table is a (non-ragged) 2d List"""
    n_rows = len(table)
    n_cols  = len(table[0]) # All rows have same no. cols
    new_table = [] # Result accumulator

    for c in range(n_cols):
        row = [] # Single row accumulator
        for r in range(n_rows):
            row.append(table[r][c]) # Build up new row
        new_table.append(row) # Add new row to new table
    return new_table

d = [[1,2],[3,4],[5,6]]
d_v2 = transpose(d)
```



Tuples

strings:	tuples*:	lists:
immutable sequences of **characters**	**immutable** sequences of **any objects**	mutable sequences of **any objects**

* "tuple" generalizes "pair," "triple," "quadruple," …

- Tuples fall between strings and lists
  - write them with just commas: **42, 4.0, 'x'**
  - often enclosed in parentheses: **(42, 4.0, 'x')**

Use **lists** for:
- long sequences
- homogeneous sequences
- variable length sequences

Use **tuples** for:
- short sequences
- heterogeneous sequences
- fixed length sequences

Returning multiple values

- Can use lists/tuples to **return** multiple values

```
INCHES_PER_FOOT = 12

def to_feet_and_inches(height_in_inches):
    feet = height_in_inches // INCHES_PER_FOOT
    inches = height_in_inches % INCHES_PER_FOOT
    return (feet, inches)

all_inches = 68
(ft,ins) = to_feet_and_inches(all_inches)
print(You are “+str(ft)+” feet, “+str(ins)+” inches.”)
```