Objectives

- Lists

Review

- What is a list?
- How do we create a list?
  - (What is the syntax?)
- How do we find out the element at position x in the list?
- How do we put 2 lists together?
- How can we iterate through a list? (Two ways)
- How can we find out if some element is in a list?

Practice: Wheel of Fortune

- Modify to keep track of previous guesses
  - If user made that guess before, print message
- What are the data types of the data we're modeling?

Practice: Wheel of Fortune

- Model the wheel
  - Money
  - Bankruptcy, lose a turn, free spin
- Simulate spinning the wheel

Practice: Wheel of Fortune

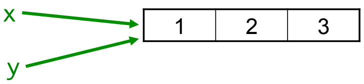
- Read in all puzzles from a file, then randomly select from those puzzles
- Modify: don't allow repeats

Copies of Lists

- What does the following code output?

```
x = [1, 2, 3]
y = x
y[0] = -1
print x
print y
```

List Identifiers are **Pointers**



- y is **not** a copy of x
  - Points to what x points to
- How to make a copy of y?

```
y = x + []   OR   y = []
                  y.extend(x)
```

Empty list

Lists as Parameters to Functions

If a list that is passed as a parameter into a function is **modified** ***in*** **the function**, the list **is modified** ***outside*** **the function**

- Lists are **not** passed-by-value/copied
- Different from immutable types (e.g., numbers, strings)
- Parameter is actually a **pointer** to the list in memory

Problem: Sort a list of 3 numbers, in descending order

- How with list methods?
- Can we do this using only 3 comparisons?

```
# order list such that list3[0] >= list3[1] >= list3[2]
def descendSort3Nums( list3 ):
```

Called as:

```
list = …
descendSort3Nums(list)
print list
```

descendSort.py

Descend Sort a List w/ 3 elements

```
def descendSort3Nums(list3):
  if list3[1] > list3[0]:
     # swap 'em
     tmp = list3[0]
     list3[0] = list3[1]
     list3[1] = tmp

  if list3[2] > list3[1]:
     tmp = list3[1]
     list3[1] = list3[2]
     list3[2] = tmp

  if list3[1] > list3[0]:
     tmp = list3[0]
     list3[0] = list3[1]
     list3[1] = tmp
```

```
def main():
    list = [1,2,3]
    descendSort3Nums(list)
    print list
```

Function does **not** *return* anything.
Simply modifies the `list3` parameter.

Lab 8: *Deal or No Deal* Overview

- Have 26 cases with various amounts of money
  - Amounts are known
- Player selects a case (hope has the big jackpot)
- In each round, player opens up cases
  - Reveals amounts that are not in the case they chose
- Banker makes an offer to buy the case
- Player decides if want to take the deal
  - Is the offer more than what is in the case?
  - Make decision based on amounts that haven't been opened yet
- Game ends when only one more case to open (two amounts on board) or player takes the deal.

Implementing *Deal or No Deal*

- Given: partial solution in code
  - Complete `main()` function, some additional functions
- Your job:
  - Read, understand given code
  - Fill in the functions for a complete solution

Modeling *Deal or No Deal*

- Cases, numbered 0 to 25
  - Have dollar amounts in them

How can we represent when a case has been opened?

1000000	1000	5	…	750000	value
0	1	2		25	case/ position

- Board
  - Which dollar amounts have been chosen, which are still in play

.01	1	5	…	1000000	value
0	1	2		25	position

Modeling *Deal or No De*

CHOSEN = -1
means case opened:
Don't display on board,
Don't allow user to select again

- Cases, numbered 0 to 25
  - Have dollar amounts in them

1000000	1000	5	…	**CHOSEN**	value
0	1	2		25	case/ position

- Board
  - Which dollar amounts have been chosen, which are still in play

.01	**CHOSEN**	5	…	1000000	value
0	1	2		25	position

Functionality

- Read in values contained in cases from a file
  - What data type should these numbers be?
- Have user select from remaining cases
  - Make sure choice is valid
- Display remaining cases
  - Print four to a row
- Display remaining amounts on board
  - Left column is smaller amounts

Where is Documentation Coming From?

- Comes from the code itself in "**doc strings**"
  - i.e., "documentation strings"
- Doc strings are simply strings *after* the function header
  - Typically use triple-quoted strings because documentation goes across several lines

```
def printVerse(animal, sound):
     """ prints a verse of Old
MacDonald, filling in the strings for
animal and sound """
```

How to print remaining cases?

- Cases, numbered 0 to 25
  - Have dollar amounts in them

1000000	1000	5	…	CHOSEN	value
0	1	2		25	case/ position

- Board
  - Which dollar amounts have been chosen, which are still in play

.01	CHOSEN	1000	…	-1	value
0	1	2		25	position

This Week

- Lab 8 due Friday
- Broader Issue: Digital Humanities
  - Read about a new algorithm to detect art fraud or mining metaphors