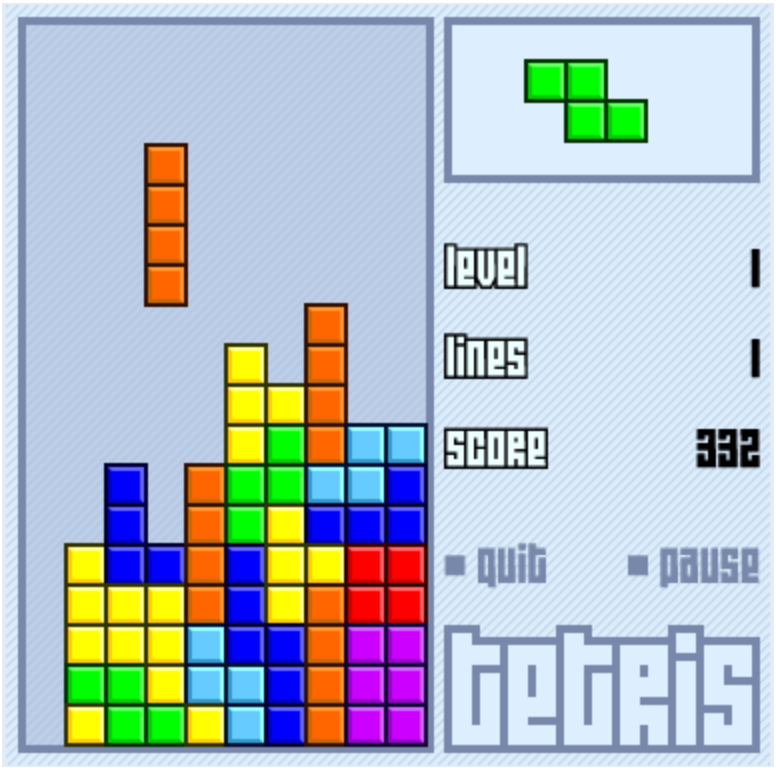
OPERATIONS ON COLLECTIONS & PROJECT PREVIEW

CSSE 120—Rose Hulman Institute of Technology

Project preview

- You will be implementing Tetris
  - http://www.youtube.com/watch?v=G0LtUX_6IXY
  - http://www.youtube.com/watch?v=keeSEJG4XzU&feature=related
  - http://www.youtube.com/watch?v=jwC544Z37qo
- Team size: 3 students
  - You should take/have taken "Team Preference Survey" on Angel
- We will actually begin the project next session
- Due: presentations will be done start of week 8
- Today's homework will help you prepare for project

Game that you will implement



Taken from http://www.socialfiction.org

Lists are sequences…

- Because all of Python's built-in sequence operations apply:

Operator	Meaning
<seq> + <seq>	Concatenation
<seq> * <int-expr>	Repetition
<seq>[ ]	Indexing
len(<seq>)	Length
<seq>[:]	Slicing
for <var> in <seq>:	Iteration
<expr> in <seq>	Membership check (Returns a Boolean)

List-specific methods

- These methods can also be applied to list objects
- Some of them mutate a list

Method	Meaning
<list>.append(x)	Add element x to end of list
<list>.sort()	Sort (order) the list. A comparison function may be passed as a parameter
<list>.reverse()	Reverse the list
<list>.index(x)	Return index of first occurrence of x
<list>.insert(i, x)	Insert x into list at index i
<list>.count(x)	Return number of occurrences of x in list
<list>.remove(x)	Delete first occurrence of x in list
<list>.pop(i)	Delete i^{th} element of list and return its value

What can we do with lists?

- Do the same thing to each object in a list
- Find the largest number in a list of numbers.
- Find the second largest element.
- Find the point in a list that is farthest away from a given point.
- Find the point in a list which, when chosen as the center, can enclose all of the points in the smallest possible circle
- Much more!

Experimenting with list objects

```
colorList = [color_rgb(r,0,255-r) for r in range (0,255,2)] + \
            [color_rgb(255-r,r,0) for r in range (0,255,2)] + \
            [color_rgb(r,255-r,r) for r in range (0,255,2)] + \
            [color_rgb(255,r,255-r) for r in range (0,255,2)]

def moveAllElementsBy(list, dx, dy):
    for obj in list:
        obj.move(dx, dy)

def colorAll(list, color):
    for obj in list:
        obj.setFill(color)

def moveThoseColors(win):
    rectList = []
    for i in range(5):
        rect = Rectangle(Point(i*50, 10), Point(i*50+40, 50))
        rect.draw(win)
        rectList.append(rect)

    for c in colorList:
        time.sleep(.02)
        moveAllElementsBy(rectList, 1, 1)
        colorAll(rectList, c)

    time.sleep(1)
```

Watch the demo

The first two functions are examples of doing the same thing to each element of a list.

Write and test these functions in pairs

1.
```
def doubleAll(list):
    """ returns a list of numbers that are twice
        those in the original list. """
```

2.
```
def largestInList(numList):  # A nonempty list of numbers
    """ returns the largest number in the list. """
```

3.
```
def secondLargest(numList):
    # numList contains at least 2 numbers, all different
    """ returns the second largest number in the list """
```

4.
```
def farthest(pointList, p):
    """return the point in pointList that is
      farthest from point p and its distance"""
```

Do not mutate original lists

“Sliding" Blocks in Tetris

- In Tetris and some block games we need to shift blocks around by sliding them as follows:
  - Slide blocks in the same row or a group of rows to one side of the board, occupying vacant spots
  - Slide blocks in the same column or a group of columns to one side of the board, occupy vacant spots
  - Slide multiple rows of blocks vertically downwards, as when clearing multiple rows in Tetris
- We can call this operation *sliding* a list
  - Before: `['-', 'B', '-', 'B', 'B']`
  - After: `['-', '-', 'B', '-', 'B']`

Sliding blocks in Tetris

- Imagine collection of blocks stored as a list of lists of blocks
- We will be sliding rows or columns in the list of lists
- This idea can be described as sliding lists

List sliding functions

1. `def slideRowToLeft(alist):`

   `""" Slide alist so that each element is slid to the left and the rightmost index contains the empty string """`

2. `def slideRowToRight(alist):`

   `""" Slide alist so that each element is slid to the right and the leftmost index contains the empty string """`

3. `def slideAllRowsDown(atable):`

   `""" Slide atable so that all the columns or each row are slid down by one. The bottom row falls off the edge and the top row is replaced with a row of empty strings """`

You want to mutate original lists in this case

You can move on to rotation functions if you finish these

List rotation functions

1.
```
def rotateFromRight(alist):
    """ Rotates alist so that the rightmost element
      appears at the left end of the list """
```
2.
```
def rotateFromLeft(alist):
    """Rotates alist so that the leftmost element
    appears at the right end of the list """
```
3.
```
def rotateRow(alist, side):
    """Rotates alist from the left or from the right"""
```
4.
```
def rotateAnyRow(atable, rowindex, side):
    """Rotates a row of atable from the left or from
    the right"""
```

Plus 4 more in homework