

LABORATORY 6 : Python Lists

OBJECTIVES

- to understand Python Lists
- to understand what "reference" means in Python

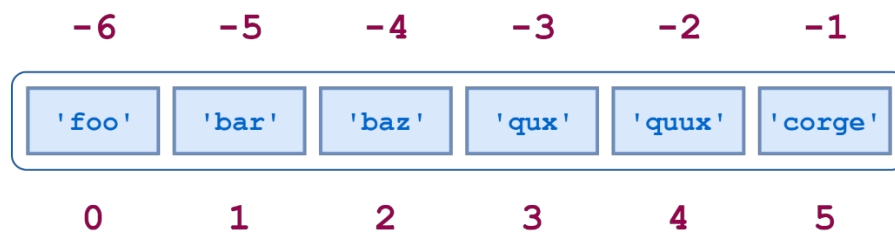
BACKGROUND

❖ Lists

- Basics

A Python list is a collection of arbitrary objects, somewhat similar to an array in many other programming languages but more flexible.

A list can be viewed as comma-separated items between square brackets. Each item occupies one slot in the list. Each slot has a number associated to it. This number is called index. Index is used to refer to item in the slot. List index is an integer starting from 0 to list size – 1. Python allows negative index where index - 1 refers to the right most item. List indexing is shown in the figure below. (Indices are shown in red)



Note that items in a list need not be of the same type.

Examples :

```
list1 = ['January', 'December', 1997, 2018];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]

# accessing item in a list
print "list1[0]: ", list1[0]
# output
# list1[0]:  January

print "list2[1:5]: ", list2[1:5]
# output
# list2[1:4]:  [2, 3, 4]

# update item in a list
list = ['January', 'December', 1997, 2018];
print "Value available at index 2 : "
print list[2]
# output
```

```

# Value available at index 2 :
# 1997

list[2] = 2001;
print "New value available at index 2 : "
print list[2]
# output
# New value available at index 2 :
# 2001

# delete item from a list
del list[2];
print "After deleting value at index 2 : "
print list
# output
# After deleting value at index 2 :
# ['January', 'December', 2018]

```

- Operations

+	concatenation (same as in string)
*	repetition (same as in string)
len(a_list)	number of items in list a_list
x in a_list	membership –check whether x is in list a_list
for x in a_list :	iteration through every item in list a_list

- Indexing and Slicing

a_list[2]	item in a_list at index 2
a_list[-2]	2 nd item from the right (item next to last item in a_list)
a_list[a:]	slice items from index a to end of list
a_list[:a]	slice items from beginning of list to index a - 1
a_list[a:b]	slice items from index a to index b - 1
a_list[a:b:c]	slice items from index a to index b - 1 with step c

- Functions

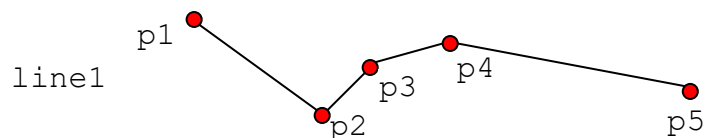
Common functions include

1	<u>list.append(obj)</u> Appends object obj to list
2	<u>list.count(obj)</u>

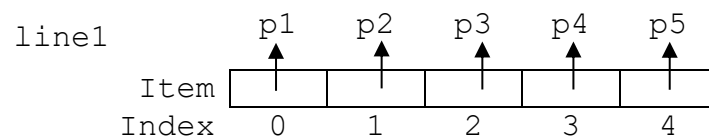
	Returns count of how many times obj occurs in list
3	<u><code>list.extend(seq)</code></u> Appends the contents of seq to list
4	<u><code>list.index(obj)</code></u> Returns the lowest index in list that obj appears
5	<u><code>list.insert(index, obj)</code></u> Inserts object obj into list at offset index
6	<u><code>list.pop(obj=list[-1])</code></u> Removes and returns last object or obj from list
7	<u><code>list.remove(obj)</code></u> Removes object obj from list
8	<u><code>list.reverse()</code></u> Reverses objects of list in place
9	<u><code>list.sort([func])</code></u> Sorts objects of list, use compare function, if given

❖ Line

A line is a series of points (two points, minimum). For example, the line below, named `line1`, is a series of five points.



A line can be viewed as a list of points.



LABORATORY 6: Pre-lab

1. Read about Python List from your favorite Python book.
2. Read Python documentation on List and pseudorandom number generator (`random.Random`). Make yourself familiar with them.

LABORATORY 6: In-lab, Post-lab

1. Use class `Point` from Lab 4. Write a new class, `Line`, representing a line. You may add more function to `Point` as you see fit.

Hint: a line can be represented by a sequence of points

2. In class `Line`, write the following functions (also, you can write more functions)

2.1. `__str__()` :

Returns string representation of a line.

2.2. `join(Line)` :

`line1.join(line2)` connects `line2` to the end of `line1`. All points in `line2` are moved to `line1`. `line2` will be empty after successful operation.

Write two versions of the `join` operation – one using python list's `extend` function, one implementing everything on your own.

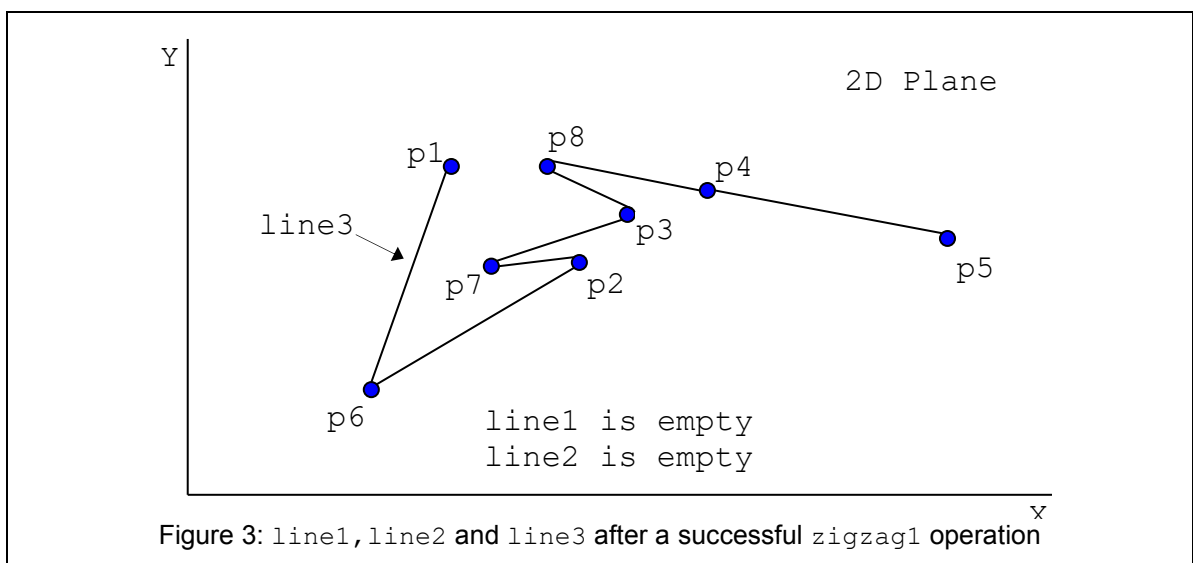
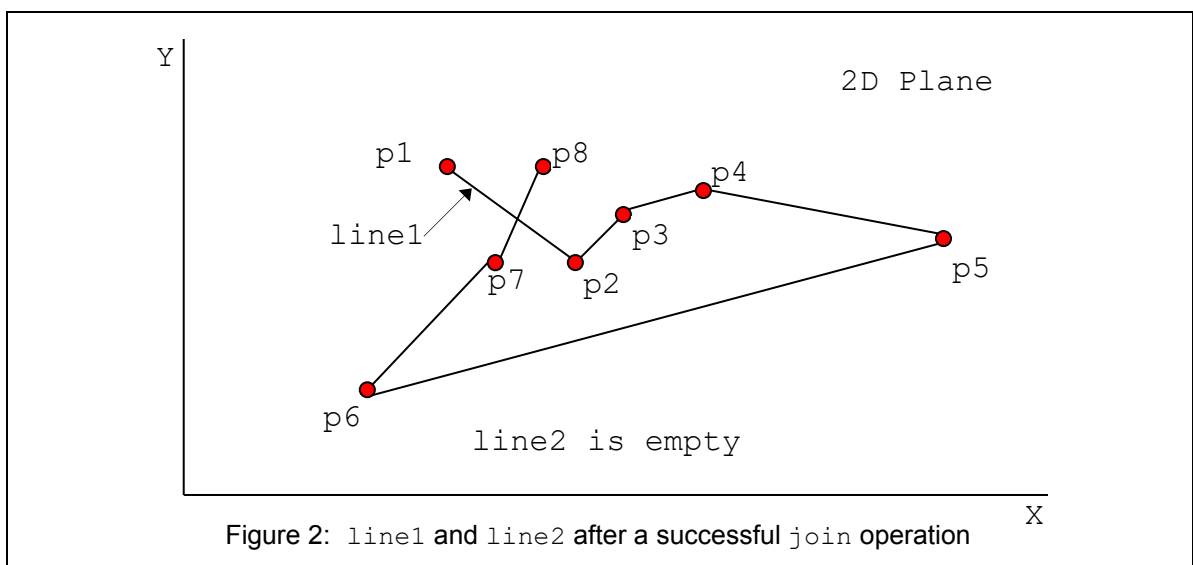
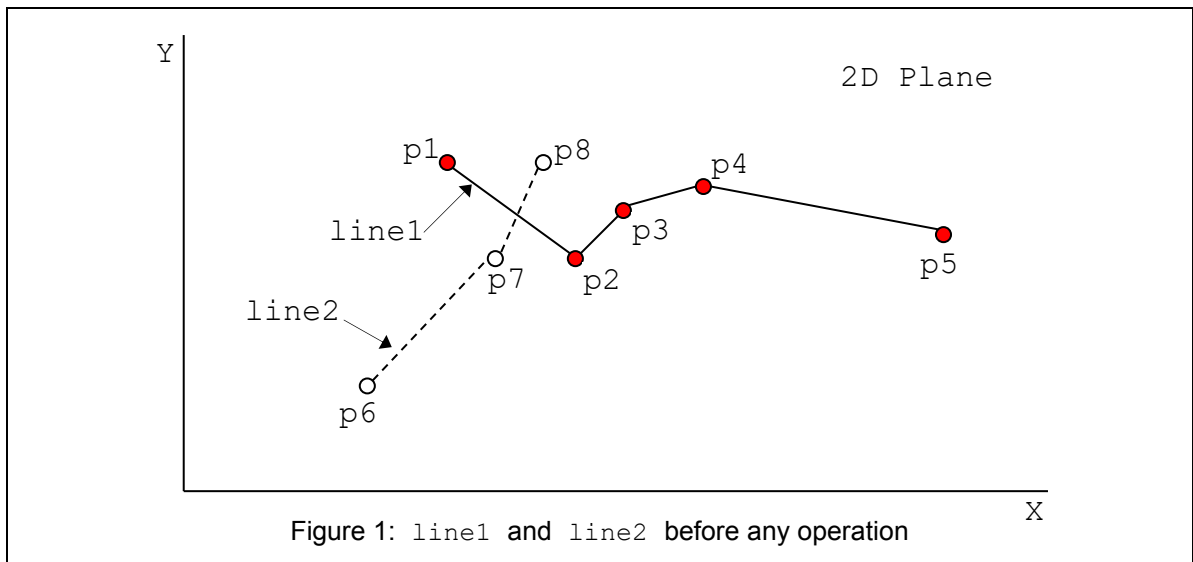
2.3. `zigzag1(Line)` :

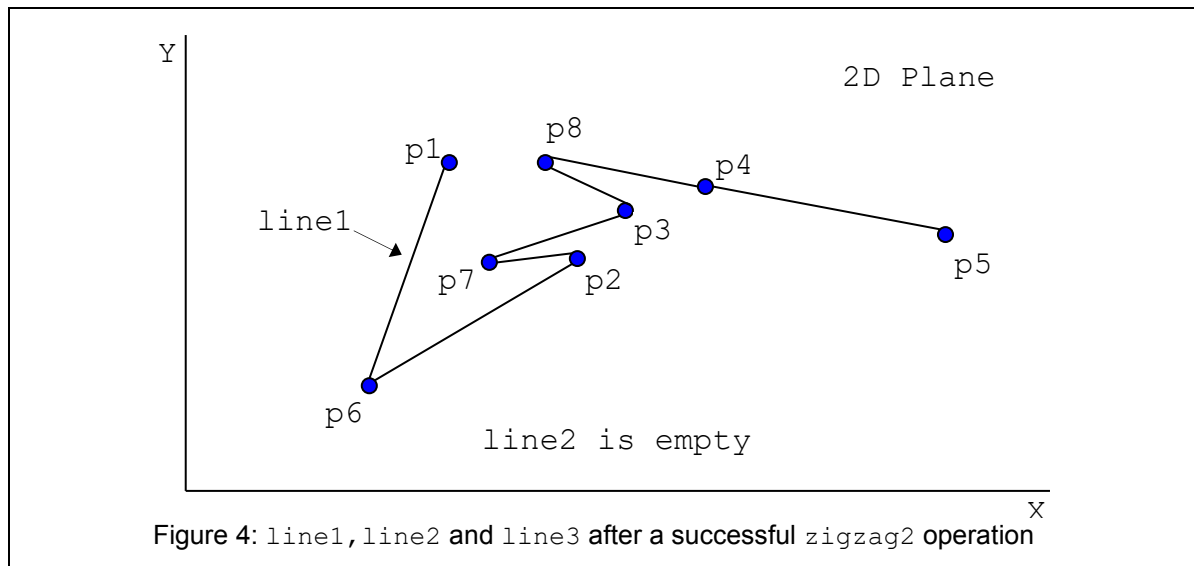
`line3 = line1.zigzag1(line2)` shuffles points in `line1` and `line2` to `line3` in a way that one point removed from `line1` is added to `line3` and then one point removed from `line2` is added to `line3`; and so forth (please see Figure 3 below). After `zigzag1` operation finishes, `line3` will contain all points from `line1` and `line2`. `line1` and `line2` are empty after a successful operation. `line3` is returned.

2.4. `zigzag2(Line)` :

`line1.zigzag2(line2)` shuffles points in `line1` and `line2` to `line1` in a way that points are removed from `line2`, one by one, then inserted into `line1` at zigzag positions (please see Figure 4 below). All points in `line2` are shuffled into `line1`. `line2` will be empty after a successful operation.

Figures below illustrate initial lines and lines after successful `join`, `zigzag1` and `zigzag2` operations.





Please note that in this example `line1` contains more number of points; however, `line2` can contain more points or number of points in both lines can be equal.

3. Draw your lines with `Turtle`. (Turtle's odometer should relate to length of line)
4. Write a test plan. Write a class `LineTester` to test your `Line` class.

Submission:

Due date: in-lab, post-lab : as stated in Canvas.

You are to demonstrate your test plan and program. Prepare to answer some questions individually.