Lecture 4
Strings and Lists

Recap last lecture

- Boolean expressions
- Condional execution
- Accumulator pattern
- Iteration using `while`
- Blackjack

Strings

Strings

```
“I am a string”
```

- strings are **collections** of characters
- strings with no characters are **empty**

Mathematical operations

+

concatenation

*

repetition

Indexing

M	**O**	**N**	**T**	**Y**		**P**	**Y**	**T**	**H**	**O**	**N**
0	1	2	3	4	5	6	7	8	9	10	11
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
someString[2]
```

Slicing

M	**O**	**N**	**T**	**Y**		**P**	**Y**	**T**	**H**	**O**	**N**
0	1	2	3	4	5	6	7	8	9	10	11

```
someString[begin:end]
someString[:end]
someString[begin:]
someString[:]
```

String methods

`.upper(), .lower()`

    change case

`.strip()`

    Strip newlines and spaces from
    beginning and end of string

`.find()`

    find index of substring

`.replace()`

    replace substring with another
    substring

Comparing strings

==, >, <

they all work, following the characters are numbers rationale

Characters are numbers

`ord()`

`Get number representation of character`

`chr()`

`Get character representation of number`

More info about strings

- Strings are **immutable**
- **`len()`** returns the length of a string
- You can use loops to iterate across chars in string, either directly or using an index
- **`in`** and **`not in`** test for presence of substring

Lists

Lists

```
[1, 2, 6]
```

- Lists are **collections** of elements
- List with no elements are **empty**

Lists are very similar to strings

- Indexing
- Slicing
- Length
- Membership
- Concatenation
- Repetition
- `for` loop

But lists are mutable

- So this works: `aList[2] = ‘a’`

List methods

.append()

    appends an element to the end (IN PLACE)

.insert()

    inserts an element at a specific index (IN
    PLACE)

.sort()

    sorts a list alphabetically (IN PLACE)

.pop()

    returns and removes the last element

.index()

    get the index of the first occurrence

.count()

    counts the nr of occurrences of a specific
    element in list

List deletion

```
del this_list[index]
```

removes item at given index, also works with range (slicing)

Lists consist of references

- Two **lists** variables can reference the same elements in memory
- If you want one list variable to refer to the same elements as another, you **alias**:

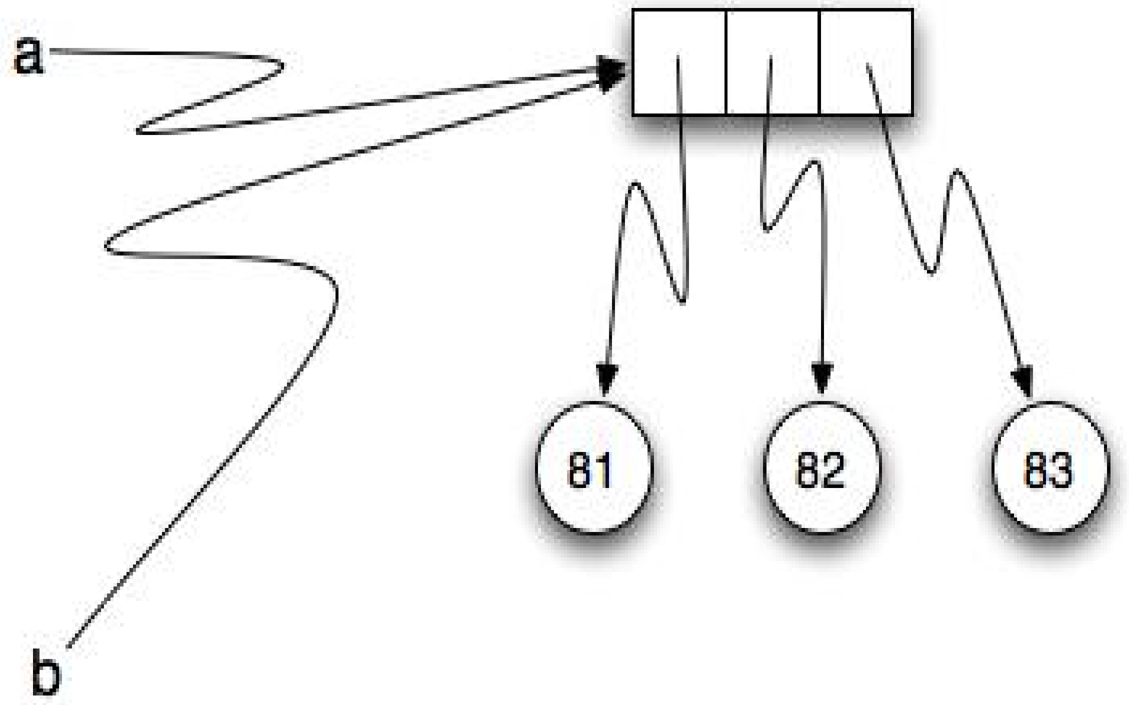
```
list2 = list1
```

- If you want one list variable to refer to a copy of another list, you **clone**:

```
list2 = list1[:]
```

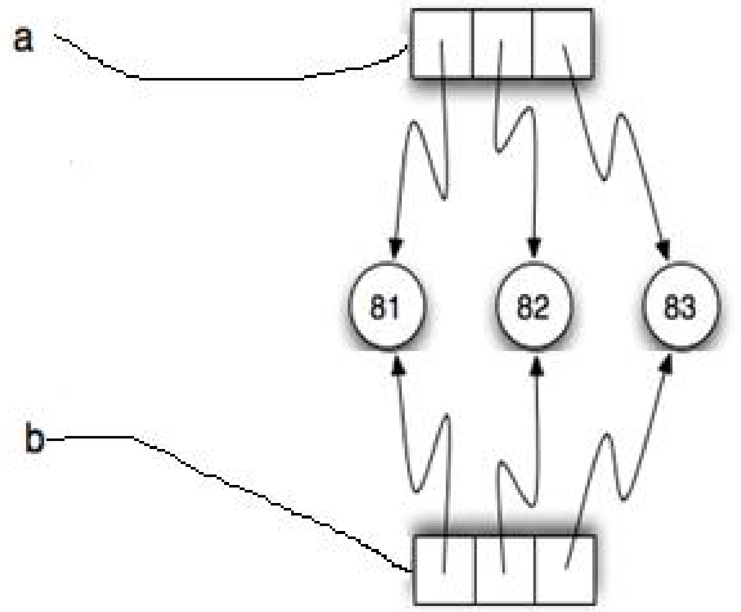
Alias vs clone

b = a



Alias vs clone

```
b = a[:]
```



Lists are passed to functions as references

So changes to a list inside a function affect the list outside the function

(the list in the function is an **alias)**

Nested lists

```
nestedList = [1,[4,4,6],[3,4,5]]

print (nestedList[1])

print (nestedList[1][2])
```

Strings and Lists

`.split()`

String method: split string into list using argument as delimiter

`.join()`

String method: join list into string with current string as delimiter

`for` loops

- Strings and lists
  - By-item
  - By-index
- List comprehension

```
[<expression>  for <item> in <sequence>
  if <condition>]
```

```
[i*2 for i in range(10) if i % 2 == 0]
```

Tuples

- Immutable lists
- `(1, 2, 3)` instead of `[1, 2, 3]`
- Good for returning multiple values in function
- Tuple assignment:

```
(x, y, z) = [4, 5, 6]
(a, b) = (b, a)
x, y, z = [4, 5, 7]
```

Recap

- Strings
- Lists
- Tuples

This week’s homework

- Read e-book Ch Strings, until ‘Turtles and Strings and L-systems’ & Ch Lists, skip ‘The Return of L-systems’
- Solve these problems:
  - Ch Strings: 1-2, 6-8, 10
  - Ch Lists: 3-6, 14 (see next slide!)
- Bring these problems on hard copy:
  - Ch Strings: 2, 6, 8, 10
  - Ch Lists: 4, 6, 14

Addendum to problems

- Ch Lists Problem 3 refers to list in Exercise 1 but means to refer to a list in Exercise 2

- Ch Lists problem 14 uses a `test()` function that checks whether two strings that are provided as input argument are identical. You have to program that test yourself.