Lists

CS106AP Lecture 10
Today’s questions

How can we store and organize data in our code?
Today’s topics

1. Review
2. Introduction to Lists
3. Advanced Lists
4. What’s next?
Review
Image coordinate system

width: 100  height: 50

0 1 2 3 96 97 98 99

0
1
2
48
49

y (height)
x (width)
Nested for loops

```python
image = SimpleImage(filename)
for y in range(image.height):
    for x in range(image.width):
        pixel = image.get_pixel(x, y)
        # Do something with pixel
```

- Iterate over the rows
- Iterate over the columns
- Gets the pixel at x, y
flip_horizontal()
Summary

- Use nested for range() loops to manipulate pixels when we care about x,y
  - Use `image.get_pixel(x, y)` to get the pixel at the specific coordinates
Summary

- Use nested for range() loops to manipulate pixels when we care about \( x, y \)

- Common pattern: Swapping two variables
  - Use a temporary ("temp") variable to store the old value
    
    \[
    \begin{align*}
    x_1 &= 3 \\
    x_2 &= 4 \\
    \text{temp} &= x_1 \\
    x_1 &= x_2 \\
    x_2 &= \text{temp}
    \end{align*}
    \]
Summary

- Use nested for range() loops to manipulate pixels when we care about \( x, y \)

- Common pattern: Swapping two variables

- `SimpleImage.blank(new_width, new_height)` allows us to create a new, empty image of a specific size
  - Then we can loop over its pixels to set their RGB
Advanced For Range Loops

```python
>>> for i in range(4, 0, -1):
```
Advanced For Range Loops

```python
>>> for i in range(4, 0, -1):
    (start_index, end_index, step)
```
Advanced For Range Loops

```python
>>> for i in range(4, 0, -1):
...     print(i)
```

(start_index, end_index, step)
Advanced For Range Loops

```python
>>> for i in range(4, 0, -1):
...     print(i)
...
4
3
2
1
```

(start_index, end_index, step)
Advanced For Range Loops

```python
>>> for i in range(0, 8, 2):
```
Advanced For Range Loops

```python
>>> for i in range(0, 8, 2):
    print(i)
```

Advanced For Range Loops

```python
>>> for i in range(0, 8, 2):
    print(i)
0
2
4
6
```
How can we store and organize data in code?
Lists!
What is a list?

**Definition**

**List**
A data type for storing values in a linear collection.
What is a list?

[1, 2, 3, 4, 5]
What is a list?

Use [] to write a list in code!

**Definition**

**List**

A data type for storing values in a linear collection.
What is a list?

[1, 2, 3, 4, 5]

Lists contain elements!
(separate with commas)

**Definition**

**List**
A data type for storing values in a linear collection.
What is a list?

[1, 2, 3, 4, 5]

[‘a’, ‘b’, ‘b’, ‘d’]

**Definition**

**List**

A data type for storing values in a linear collection.
What is a list?

[1, 2, 3, 4, 5]
[‘a’, ‘b’, ‘b’, ‘d’]
[True]

Definition

List
A data type for storing values in a linear collection.
What is a list?

[1, 2, 3, 4, 5]

[‘a’, ‘b’, ‘b’, ‘d’]

[True]

**Definition**

**List**

A data type for storing values in a linear collection.

**Lists can have 1 element!**
What is a list?

[1, 2, 3, 4, 5]

[‘a’, ‘b’, ‘b’, ‘d’]

[True]

[1, ‘a’, 2, ‘b’, True]

**Definition**

**List**

A data type for storing values in a linear collection.
What is a list?

[1, 2, 3, 4, 5]

[‘a’, ‘b’, ‘b’, ‘d’]

[True]

[1, ‘a’, 2, ‘b’, True]

Definition

List

A data type for storing values in a linear collection.

Lists can contain different types!
What is a list?

- [1, 2, 3, 4, 5]
- ['a', 'b', 'b', 'd']
- [True]
- [1, 'a', 2, 'b', True]
- []

**Definition**

List

A data type for storing values in a linear collection.
What is a list?

[1, 2, 3, 4, 5]
[‘a’, ‘b’, ‘b’, ‘d’]
[True]
[1, ‘a’, 2, ‘b’, True]
[]

compare to the empty string ("")!

**Definition**

**List**

A data type for storing values in a linear collection.
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> letters[0]
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> letters[0]
'a'
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> letters[0]
'a'

>>> letters[3]
'd'
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> letters[0]
'a'

>>> letters[3]
'd'
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
>>> letters[0]
'a'
>>> letters[3]
'd'
```

Indexing lists is similar to indexing strings, except with elements instead of characters!
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
>>> letters[:2]
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
>>> letters[:2]
['a', 'b']
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> letters[:2]
['a', 'b']

>>> letters[1:]
['b', 'c', 'd']
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> letters[:2]
['a', 'b']

>>> letters[1:]
['b', 'c', 'd']
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
>>> letters[:2]
['a', 'b']
Slicing is similar to strings, too!
>>> letters[1:]
['b', 'c', 'd']
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> len(letters)
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']

>>> len(letters)
4
```
How to inspect a list

```python
>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
4
```

len() works for lists as well!
Printing lists

```python
>>> fruits = ['apple', 'banana', 'mango']
```
Printing lists

>>> fruits = ['apple', 'banana', 'mango']

>>> print(fruits)
Printing lists

```python
>>> fruits = ['apple', 'banana', 'mango']

>>> print(fruits)
['apple', 'banana', 'mango']
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst += [6, 7]
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst += [6, 7]
>>> lst
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst += [6, 7]
>>> lst
[1, 2, 3, 4, 5, 6, 7]
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst += [6, 7]
>>> lst
[1, 2, 3, 4, 5, 6, 7]

>>> lst += 8
```

How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst += [6, 7]
>>> lst
[1, 2, 3, 4, 5, 6, 7]
>>> lst += 8
TypeError
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst += [6, 7]
>>> lst
[1, 2, 3, 4, 5, 6, 7]
>>> lst += 8
```

`+=` can only be used for concatenating other lists!
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
```

```python
>>> lst.append(6)
```
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]

>>> lst.append(6)
```

append adds a single element to the end of a list!
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst.append(6)
>>> lst
```

append adds a single element to the end of a list!
How can I change what’s in a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> lst.append(6)
>>> lst
[1, 2, 3, 4, 5, 6]
```

append adds a single element to the end of a list!
How can I remove something from a list?

```python
>>> lst = [1, 2, 3, 4, 5]
```
How can I remove something from a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> last_elem = lst.pop()
```
How can I remove something from a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> last_elem = lst.pop()
```

`pop()` removes the last element in a list and returns it.
How can I remove something from a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> last_elem = lst.pop()
>>> last_elem
```

`pop()` removes the last element in a list and returns it.
How can I remove something from a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> last_elem = lst.pop()
>>> last_elem
5
```

`pop()` removes the last element in a list and returns it.
How can I remove something from a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> last_elem = lst.pop()
>>> last_elem
5
>>> lst
```

`pop()` removes the last element in a list and returns it.
How can I remove something from a list?

```python
>>> lst = [1, 2, 3, 4, 5]
>>> last Elem = lst.pop()
>>> last Elem
5
>>> lst
[1, 2, 3, 4]
```

`pop()` removes the last element in a list and returns it.
How can I check if something’s in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']
```
How can I check if something’s in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']
>>> 'mango' in fruits
```
How can I check if something’s in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']

>>> 'mango' in fruits  # returns True if element in list, else False!
```
How can I check if something’s in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']
>>> 'mango' in fruits
True
```

returns True if element in list, else False!
How can I check if something’s in a list?

>>> fruits = ['apple', 'banana', 'mango', 'kiwi']

>>> 'mango' in fruits

returns True if element in list, else False!

True

>>> 'broccoli' in fruits
How can I check if something's in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']
>>> 'mango' in fruits
True
```

returns True if element in list, else False!

```python
>>> 'broccoli' in fruits
False
```
How can I check if something’s in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']

>>> 'mango' in fruits
True

>>> 'broccoli' in fruits
False

>>> 'broccoli' not in fruits
returns True if element in list, else False!
```
How can I check if something’s in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']
>>> 'mango' in fruits
True
```

you can use **not** with **in**

```python
>>> 'broccoli' not in fruits
False
```
How can I check if something’s in a list?

```python
>>> fruits = ['apple', 'banana', 'mango', 'kiwi']
>>> 'mango' in fruits
True
```

returns True if element in list, else False!

```python
>>> 'broccoli' in fruits
False
```

you can use not with in

```python
>>> 'broccoli' not in fruits
True
```
How can I loop over a list?

```python
>>> fruits = ['apple', 'banana', 'mango']
```
How can I loop over a list?

```python
>>> fruits = ['apple', 'banana', 'mango']

>>> for fruit in fruits:
```
How can I loop over a list?

```python
>>> fruits = ['apple', 'banana', 'mango']
>>> for fruit in fruits:
```
How can I loop over a list?

```python
>>> fruits = ['apple', 'banana', 'mango']

>>> for fruit in fruits:
    like a foreach loop over a string!
```
How can I loop over a list?

```python
>>> fruits = ['apple', 'banana', 'mango']
```

```python
>>> for fruit in fruits:
...   print(fruit)
```

like a foreach loop over a string!
How can I loop over a list?

```python
>>> fruits = ['apple', 'banana', 'mango']

>>> for fruit in fruits:
...   print(fruit)

apple
banana
mango
```

like a foreach loop over a string!
Think/Pair/Share:
Write a function `find_min()` that returns the minimum float in a list.
How can I loop over a list?

```python
>>> find_min([2.3, 7.1, 10.6])
2.3
```

# syntax reminder:

```python
for elem in lst:
    # do something
```

Think/Pair/Share:

Write a function `find_min()` that returns the minimum float in a list.
Making a list from a string

>>> s = 'I am comprised of words'
Making a list from a string

```python
>>> s = 'I am comprised of words'
>>> words = s.split()
```
Making a list from a string

```python
>>> s = 'I am comprised of words'
```

```python
>>> words = s.split()
splits string where there are spaces
```
Making a list from a string

```python
>>> s = 'I am comprised of words'
>>> words = s.split()
>>> words
```
Making a list from a string

```python
>>> s = 'I am comprised of words'
>>> words = s.split()
>>> words
['I', 'am', 'comprised', 'of', 'words']
```
Making a list from a string

```python
>>> s = 'I am comprised of words'
>>> words = s.split()
>>> words
['I', 'am', 'comprised', 'of', 'words']
>>> s
splits string where there are spaces
```
Making a list from a string

```python
>>> s = 'I am comprised of words'
>>> words = s.split()
>>> words
['I', 'am', 'comprised', 'of', 'words']
>>> s
'I am comprised of words'
```
Making a list from a string

>>> s = 'do, re, mi, fa, sol, la, ti'
Making a list from a string

```python
>>> s = 'do, re, mi, fa, sol, la, ti'
>>> notes = s.split(',\')
```
Making a list from a string

```python
>>> s = 'do,re,mi,fa,sol,la,ti'
>>> notes = s.split(',\')
```

pass in a “delimiter”, which tells us where to split the string
Making a list from a string

```python
>>> s = 'do,re,mi,fa,sol,la,ti'
>>> notes = s.split(',\',)
>>> notes
```

pass in a “delimiter”, which tells us where to split the string
Making a list from a string

```python
>>> s = 'do,re,mi,fa,sol,la,ti'
>>> notes = s.split(',\')
>>> notes
['do', 're', 'mi', 'fa', 'sol', 'la', 'ti']
```

pass in a “delimiter”, which tells us where to split the string
Think/Pair/Share:

Write `words_starting_with()`

Inputs:
- sentence (string)
- char (string)

Returns:
- list of words starting with character
Making a list from a string

```python
>>> words_starting_with('I love ice cream', 'i')
['I', 'ice']

# syntax reminder:

lst = s.split()
```

Think/Pair/Share:

Write a function `words_starting_with()`.

**Inputs:**
- sentence (string)
- char (string)

**Returns:**
- list of words starting with character
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
>>> lst[::-1]
```

```python
>>> lst[::-1]
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
>>> lst[::-1]
['e', 'd', 'c', 'b', 'a']
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
>>> lst[::-1]
['e', 'd', 'c', 'b', 'a']
```

Think/Pair/Share:

What just happened? What is going on in the `[::-1]`?
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']

>>> lst[::-1]
['e', 'd', 'c', 'b', 'a']
```

(start_index, end_index, step)

['e', 'd', 'c', 'b', 'a']
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
>>> lst[::-1]
['e', 'd', 'c', 'b', 'a']
```

```python
for i in range(4, 0, -1):
    # code here
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
>>> lst[::-1]
['e', 'd', 'c', 'b', 'a']
```

(start_index, end_index, step)
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
>>> lst[::-1]  # (start_index, end_index, step)
['e', 'd', 'c', 'b', 'a']
>>> lst
['a', 'b', 'c', 'd', 'e']
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']

>>> lst[::2]
```

Advanced Slicing

```python
>>> lst = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’]
>>> lst[::2]
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']
>>> lst[:2]
[a, c]
```
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']

>>> lst[::2]
['a', 'c', 'e']
```

(start_index, end_index, step)

['a', 'c', 'e']
Advanced Slicing

```python
>>> lst = ['a', 'b', 'c', 'd', 'e']

>>> lst[::2]  # (start_index, end_index, step)
['a', 'c', 'e']
```

```python
for i in range(0, 6, 2):  # for i in range(0, 6, 2):
    # Code block
```
What’s next?