Encapsulation

CS106AP Lecture 20
Day 1!

Programming Basics

The Console

Images

Graphics

Midterm

Data structures

Object-Oriented Programming

Everyday Python

Life after CS106AP!
Life after CS106AP!

Day 1!

Graphics

Object-Oriented Programming

Part 1

Part 2

Part 3

Midterm

Programming Basics

The Console

Data structures

Images

Everyday Python

Life after CS106AP!
Today’s questions

Why do we use classes in our own code?

How can we write programs that respond to user actions?
Today’s topics

1. Review
2. Classes and encapsulation
3. Event-driven programming (if time)
4. What’s next?
Animation loops
Use an animation loop

```python
while True:
    if stop_condition:
        break
    # 'Animate' object
    pause(timestep)
```
Use an animation loop

```python
while True:
    if stop_condition:
        break

    # 'Animate' object
    pause(timestep)
```

Small movements help make the animation appear continuous!
Use an animation loop

```python
while True:
    if stop_condition:
        break
    # 'Animate' object
    pause(timestep)
```

*How long to wait in milliseconds*
Classes
Definition

class
A Python class defines a new data type for our programs to use.
Definition

class
A Python class defines a new **data type** for our programs to use.

ints, strings, booleans, lists, floats, dictionaries, etc. are all **built-in** Python data types
Definition

class
A Python class defines a new data type for our programs to use.

Classes allow us to define our own data types!
What is a class?

- A blueprint for a new type of Python object!
  - The blueprint describes a general structure, and we can create specific instances of our class using this structure.

**Definition**

**instance**
When we create an object that is our new type, we call this creating an instance of our class.
What is a class?

- A blueprint for a new type of Python **object**!
  - The blueprint describes a general structure, and we can create specific **instances** of our class using this structure.

```
image = SimpleImage(width, height)
```

*Creates an **instance** of the SimpleImage class (i.e. an object of the type SimpleImage)*
What is a class?

- A blueprint for a new type of Python **object**!
  - The blueprint describes a general structure, and we can create specific **instances** of our class using this structure.

- 3 main parts
  - Attributes
  - Methods
  - Constructor
What is a class?

- A blueprint for a new type of Python object!
  - The blueprint describes a general structure, and we can create specific instances of our class.

- 3 main parts
  - Attributes (e.g. `oval.fill_color`, `oval.width`, etc.)
  - Methods
  - Constructor

Variables stored inside the class
What is a class?

- A blueprint for a new type of Python **object**!
  - The blueprint describes a general structure, and we can create specific **instances** of our class using this structure.

- 3 main parts
  - Attributes (e.g. `oval.fill_color`, `oval.width`, etc.)
  - Methods (e.g. `oval.move()`)
  - Constructor

**Functions you can call on the object**
What is a class?

- A blueprint for a new type of Python **object**!
  - The blueprint describes a general structure, and we can create specific **instances** of our class using this structure.

- 3 main parts
  - Attributes (e.g. `oval.fill_color`, `oval.width`, etc.)
  - Methods (e.g. `oval.move()`)
  - Constructor (e.g. `GOval(width, height)`)
How do we design a class?

We must specify the 3 parts:

1. Attributes: *What subvariables make up this new variable type?*

2. Methods: *What functions can you call on a variable of this type?*

3. Constructor: *What happens when you make a new instance of this type?*
Let’s create a social network for Python users!

Pynstagram.py
Let’s create a class to define a PynstaUser!
How do we design a class?

We must specify the 3 parts:

1. **Attributes:** *What subvariables make up this new variable type?*
2. **Methods:** *What functions can you call on a variable of this type?*
3. **Constructor:** *What happens when you make a new instance of this type?*
**PynstaUser**: We must specify our 3 parts

- Attributes
  - Name (string)
  - Posts (list of strings)
  - Friends (list of other PynstaUsers)

- Methods
  - Post a status
  - Add a friend

- Constructor: User should provide a username
**PynstaUser**: We must specify our 3 parts

- **Attributes**
  - Name (string)
  - Posts (list of strings)
  - Friends (list of other PynstaUsers)

- **Methods**
  - Post a status
  - Add a friend

- **Constructor**: `PynstaUser(name)`
class PynstaUser:

    def __init__(self, username):
        self.name = username
        self.friends = []
        self.posts = []

    def add_friend(self, user):
        ...

    def post(self, message):
        ...
Think/Pair/Share:
How would you design classes for the following?
How would you design a class for...

- A summer student at Stanford who is enrolling in classes
- A book in a library that can be checked in/out
- A zookeeper who will only be responsible for one specific animal at a time

We must specify the 3 parts:

1. Attributes: What subvariables make up this new variable type?
2. Methods: What functions can you call on a variable of this type?
3. Constructor: What happens when you make a new instance of this type?
Why do we use classes?

● For ourselves
  ○ Grouping related data and the functions that act on it
  ○ Modular code development (isolation of particular tasks)

● For others
  ○ We hide the implementation details of our code so others don’t need to worry about them.
  ○ They can just use the class, like we do for SimpleImage.
Why do we use classes?

- For ourselves
  - Grouping related data and the functions that act on it
  - Modular code development (isolation of particular tasks)

- For others
  - We hide the implementation details of our code so others don’t need to worry about them.
  - They can just use the class, like we do for SimpleImage.
Why do we use classes in our own code?
Why do we use classes in our own code?

Encapsulation!
**Definition**

**encapsulation**
The process of grouping related information and relevant functions into one unit
**Definition**

**encapsulation**

The process of grouping related information and relevant functions into one unit. a class!
An analogy...

Suppose you’re a store owner looking to hire help to run your business. Here are some of the tasks that need to be covered:

- Inventory management
- Cashier
- Advertising
- Customer service
An analogy...

Suppose you’re a store owner looking to hire help to run your business. You can either hire one person to do everything, or you can hire multiple people to split the work. Which do you choose?
An analogy...

Suppose you’re a store owner looking to hire help to run your business. You can either hire one person to do everything, or you can hire multiple people to split the work. Which do you choose?

- Easier for you to hire one person instead of multiple!
- Then you only need to deal with one person later on!
An analogy...

Suppose you’re a store owner looking to hire help to run your business. You can either hire one person to do everything, or you can hire multiple people to split the work. Which do you choose?

- Easier for you to hire one person instead of multiple.
- Then you only need to deal with one person later on.
An analogy...

Suppose you’re a store owner looking to hire help to run your business. You can either hire one person to do everything, or you can hire multiple people to split the work. Which do you choose?

- If something goes wrong, it’s harder to know where the issue happened.
- You have to rely on that person for everything.
An analogy...

Suppose you’re a store owner looking to hire help to run your business. You can either hire one person to do everything, or you can hire multiple people to split the work. Which do you choose?

- Splitting the work across people helps you know who’s responsible for what.
- The individual people only need to know the information for their specific job.
Encapsulation via classes is like hiring multiple people!
Encapsulation via classes is like hiring multiple people!

- Integration
  - All the smaller parts add up to create the entire functionality
  - Similar to top-down decomposition
Encapsulation via classes is like hiring multiple people!

- Integration

- Modular development
  - You can separate different types of tasks and know where different information/functionality should be.
  - Easier for testing and debugging!
Encapsulation via classes is like hiring multiple people!

- Integration

- Modular development

- Instance variables (attributes)
  - Knowledge (data) for a specific class stays inside that class.
  - That information is easier to access across methods within that class.
  - If you need to access the information outside the class, there’s a predefined structure for doing so.
Encapsulation via classes is like hiring multiple people!

- Integration

- Modular development

- Instance variables (attributes)
  - Knowledge (data) for a specific class stays inside that class.
  - That information is easier to access across methods within that class.
  - If you need to access the information outside the class, there’s a predefined structure for doing so.

Getters and setters!
Encapsulation via classes is like hiring multiple people!

- Integration

- Modular development

- Instance variables (attributes)
  - Knowledge (data) for a specific class stays inside that class.
  - That information is easier to access across methods within that class.
  - If you need to access the information outside the class, there’s a predefined structure for doing so. (more on this tomorrow)
Encapsulation and graphics
Bubbles.py
[extended demo]
What’s next?
Encapsulation enables **abstraction**!

- How classes help other people who use our code

- Classes as providing specific ways to interact with our code
Extra slides
How do we write programs that respond to user actions?
How do we write programs that respond to user actions?

Event-driven programming!
The event listener model

Your code

```python
def main():
    ...
    ...
    ...

def your_mouse_listener():
    ...
    ...
```
The event listener model

Your code

def main():
    ...
    ...

def your_mouse_listener():
    ...

Definition:

**mouse listener function**
A function that occurs immediately when a user triggers a particular mouse event
The event listener model

Your code

```python
def main():
    ...
    ...
    
def your_mouse_listener():
    ...
```

**Definition**

*mouse listener function*

A function that occurs immediately when a user triggers a particular *mouse event* clicking, moving, dragging
The event listener model

Your code

```python
def main():
    ...
    ...
    ...
def your_mouse_listener():
    ...
```
The event listener model

Your code

```python
def main():
    ...
    ...
    ...

def your_mouse_listener():
    ...
```
def main():
    ...
    ...
    ...

def your_mouse_listener():
    ...
    ...
The event listener model

Your code

def main():
    ...
    ...
    ...

def your_mouse_listener():
    ...
    ...
    MOUSE CLICK
The event listener model

Your code

```python
def main():
    ...
    ...
    def your_mouse_listener():
        ...
```

The function happens immediately, no matter where you are in your program!
Creating a mouse listener

1. Write a mouse listener function

```python
def mouse_listener_handler(event):
    ...
```
Creating a mouse listener

1. Write a mouse listener function

   ```python
   def mouse_listener_handler(event):
       ...
   ```

   It must take in an `event` for campy to recognize it as a valid mouse listener.
Creating a mouse listener

1. Write a mouse listener function

```python
def mouse_listener_handler(event):
    ...
    event gives us access to information about the mouse event (e.g. x, y coordinates of the click).
```
Creating a mouse listener

1. Write a mouse listener function

   ```python
   def mouse_listener_handler(event):
       ...
   ```

2. Use the corresponding campy `onmouseevent()` function to set up your mouse listener

   ```python
   onmouseclicked(mouse_listener_handler)
   ```
Creating a mouse listener

1. Write a mouse listener function

   ```python
def mouse_listener_handler(event):
   ...
   ```

2. Use the corresponding campy `onmouseevent()` function to set up your mouse listener

   ```python
   onmouseclicked(mouse_listener_handler)
   ```

   *Pass in your mouse listener function as the argument*
Creating a mouse listener

1. Write a mouse listener function

   ```python
def mouse_listener_handler(event):
    ...
```

2. Use the corresponding campy `onmouseevent()` function to set up your mouse listener

   ```python
   onmouseclicked(mouse_listener_handler)
   ```

   Don’t include parentheses after the function name!
Bubbles.py

[mouse listener demo]