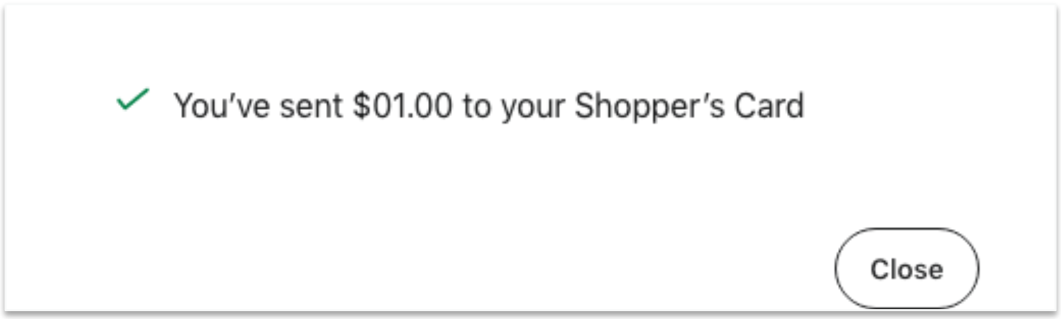
Objectives

- Reviewing creating our own classes
- Designing a Social Network
- Prep for Lab 10



✓ You've sent $01.00 to your Shopper's Card

Where We Are: 10 weeks in

- With what you now know, opens up the possibilities for the programs you can write
  - Just about anything computational is possible

- Just need to
  - Break it up into smaller pieces
  - Iterate!

Parts of an Algorithm

- Input, Output
- Primitive operations
  - What data you have, what you can do to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques

Going beyond the primitive data to making our own structures

Review: Classes

- Defining Classes
  - Why do we want to define classes/new data types?
  - How do you define a method?
    - What parameter needs to be the first parameter in every method? What does that parameter represent?
  - Define ***instance variable***
    - How do we create instance variables? Access them?
  - What are the rules for defining the __str__ method?

- Using classes
  - How do you create a new object of a given type?
    - What method does that call?
  - How do we call a method on an object?
  - What method is automatically called when we print an object?
  - What is the API for a class, in general? What is the API for the Card class?

Towards The Grand Finale!

DESIGNING CLASSES

Summary: Designing Classes

- What does the object/class represent?
- How to model/represent the class's *data*?
  - Instance variable
  - Data type
- What *functionality* should objects of the class have?
  - How will others want to use the class?
  - Put into methods for others to call (API)

General Class Design:

- **nouns** in a problem are **classes/objects** or **data**
- **verbs** are **methods**

Top-Down Design

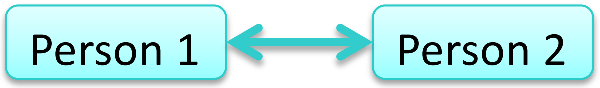
Break down larger problems into pieces that you can solve

- Smaller pieces: classes, methods, functions
- Implement smallest pieces and build up

- We’ve been doing this most of the semester
  - Typically, program was 1) read input, 2) process input, 3) print result
    - Started putting Step 2 into >= 1 functions
    - Steps 1 and 3 were sometimes functions
- Now: on larger scale

Requirements for a Social Network Application

- Reads social network from two files
  - One file contains *people*
    - Their id/username, first and last names
  - One file contains *connections* between people
- Adds connections between people (makes them friends)
  - Symmetric relationship



- Provides a user interface to access/update a social network

Designing a Social Network Application

- Break down into pieces
- What classes do we need?
  - What data needed to model those classes?
  - What functionality do each of those classes need?
- What does our user interface do?
- How should we implement those classes/program?

Recall: General Class Design:
- **nouns** in a problem are **classes/objects** or data
- **verbs** are **methods**

Designs

- For each of your classes
  - Data
  - API

Social Network Classes & UI: Data

- Person
  - User id
  - Name
  - Friends
- Social Network
  - People in network
- User Interface (UI)
  - Social network

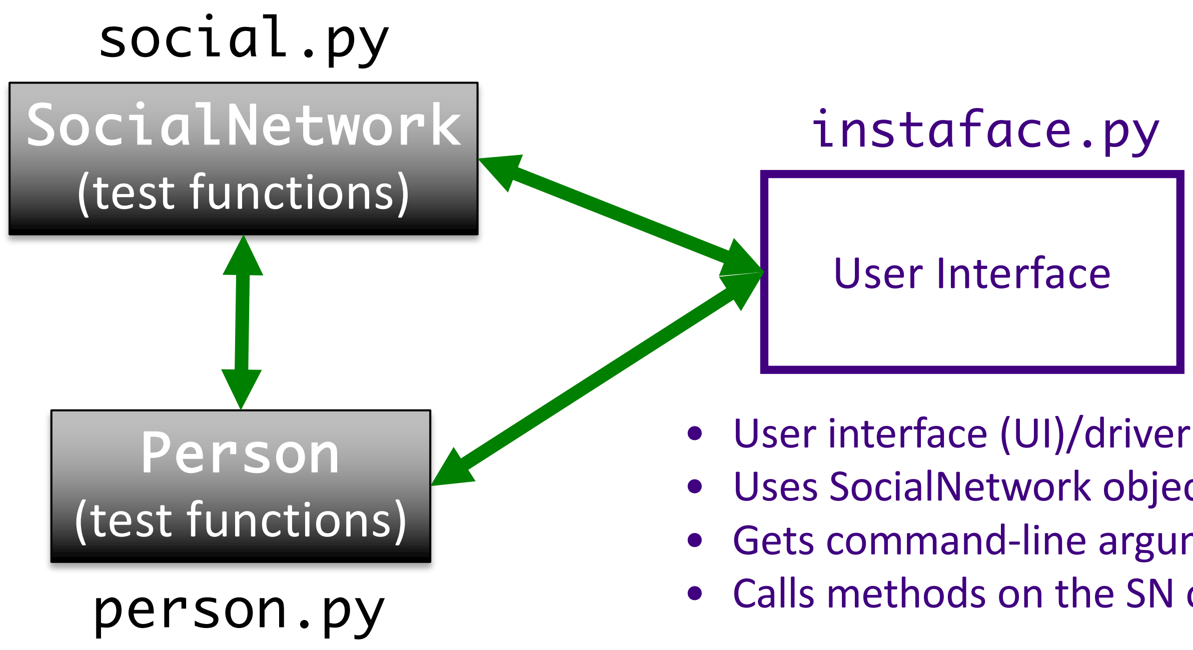
What are the data types for each class's data?

SN Classes & UI Functionality

- Person
  - Getters (accessors)
  - String rep
  - Setters
- Social Network
  - Getters
  - String rep
  - Add people to network
  - Add connections
  - Writing to a file
- User Interface
  - Prompts for user input to
    - Read people, connections files
    - Store social network to file
    - Add a person
    - Add connections
  - Summary: call appropriate methods on classes to do above

Lab 10 Social Network Design

- 2 classes: Person and SocialNetwork



- User interface (UI)/driver
- Uses SocialNetwork object
- Gets command-line arguments
- Calls methods on the SN object

Problem: People Files

- Given the name of people file that has the format

```
<num_users>
<user_id>
<name>
…
<user_id_n>
<name_n>
```

Example:

```
3
captain
Brie Larson
widow
Scarlett Johannsen
warmachine
Don Cheadle
```

- Write algorithm to create `Person` objects to represent each person, add to `SocialNetwork` object

Problem: Connection Files

- Given a connection file that has the format

```
<user_id> <user_id>
<user_id> <user_id>
…
<user_id> <user_id>
```

Example:

```
captain widow
widow warmachine
```

- Each line represents a friend/connection
    - Symmetric relationship
    - Each is a friend of the other
- Update `SocialNetwork` object

InstaFace UI Specification

- Checks if user entered command-line arguments
  - Default files otherwise
- Read people, connections from files
- Repeatedly gets selected options from the user, until user quits
- Repeatedly prompts for new selection if invalid option
- Executes the appropriate code for the selection
- Stops when user quits
- Stores the social network into the file

Note how much of the functionality will be implemented in social network class. Just need to call appropriate method.

InstaFace UI Pseudocode

```
Use default files if only one command-line argument
Read people, connections from files
while True:
    display menu options
    prompt for selection
    while invalid option
        print error message
        prompt for selection
    break if selected quit
    otherwise, do selected option
Store social network to designated file
```

Implementation Plan

1. Implement `Person` class
    - Test (write test functions, e.g., `testPerson()`)
2. Implement `SocialNetwork` class
    - Example runs in lab specification
    - Note: Methods for classes will **not** prompt for input; use input parameters
    - Test
3. Complete implementation of user interface

Plan for Implementing a Class

- Write the constructor and string representation/print methods first
- Write function to test them
  - See card.py for example test functions
- While more methods to implement …
  - Write method
  - Test
  - REMINDER: methods should **not** be using `input` function but getting the input as parameters to the method

This Week

- Pre Lab 10:
  - Reviewing classes, some new stuff
  - Review dictionaries and lists if you’re rusty
- Lab 10
  - Define your own classes
- No broader issue