Lab 10

- Classes
- Lists
- Dictionaries
- Files

1

Review

- How can you get all the values from a dictionary?
  - How can you turn it into a list?
- What are other operations/methods you can perform on dictionaries?

- Why do we create classes?
- What is our process for creating a class?
  - What are important methods to implement?
  - How do we implement them?
  - What does self represent?
- What is the design of our social network application?

2

Implementing a Card Method

```
def getCardColor(self):
    if self._suit == "hearts" or self._suit == "diamonds":
        color = "red"
    else:
        color = "black"
    return color
```

3

Implementing a Card Method

```
def getCardColor(self):
    if self._suit == "hearts" or self._suit == "diamonds":
        color = "red"
    else:
        color = "black"
    return color
```

Alternative:

Calling a method on the self object

```
def getCardColor(self):
    if self.getSuit() == "hearts" or self.getSuit() == "diamonds":
        color = "red"
    else:
        color = "black"
    return color
```

Recall: what data type is self?

4

Implementing a Card Method

```
def getCardColor(self):
    if self._suit == "hearts" or self._suit == "diamonds":
        color = "red"
    else:
        color = "black"
    return color
```

Alternative:

```
def getCardColor(self):          Calling a method on the self object
    if self.getSuit() == "hearts" or self.getSuit() == "diamonds":
        color = "red"
    else:
        color = "black"
    return color
```

self is automatically passed in by Python and represents the object being acted upon.

5

Lab 10 Social Network Design

- 2 classes: Person and SocialNetwork
- 3 files

`social.py`

SocialNetwork (test functions)

`instaface.py`

User Interface

Person (test functions)

`person.py`

- User interface (UI)/driver
- Uses SocialNetwork object
- Gets command-line arguments
- Calls methods on the SN object
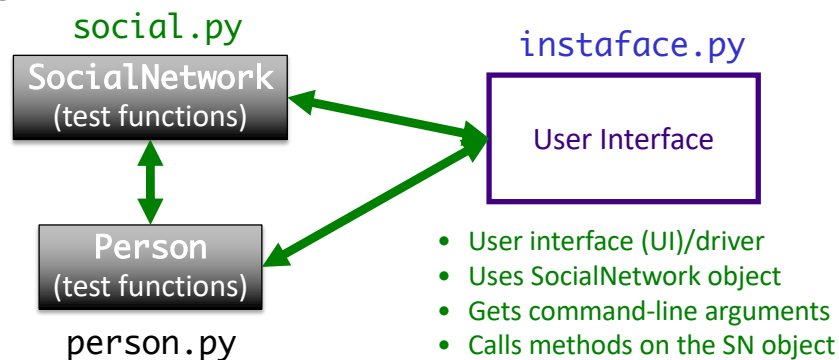
6

Review: Social Network Classes/UI Data

- Person
  - User id
  - Name
  - Friends

- Social Network
  - People in network

- UI
  - Social network

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

7

Review: Social Network Classes/UI Data

- Person
  - User id – str
  - Name - str
  - Friends – list of Person objects

- Social Network
  - People in network – dictionary, mapping userid to Persons

- UI
  - Social network

8

SN Classes/Driver Functionality

- Person
  - ➢ Getters (accessors)
  - ➢ String rep
  - ➢ Setters

- Social Network
  - ➢ Getters
  - ➢ String rep
  - ➢ Add people to network
  - ➢ Add connections
  - ➢ Writing to a file

- UI
  - ➢ Getting 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
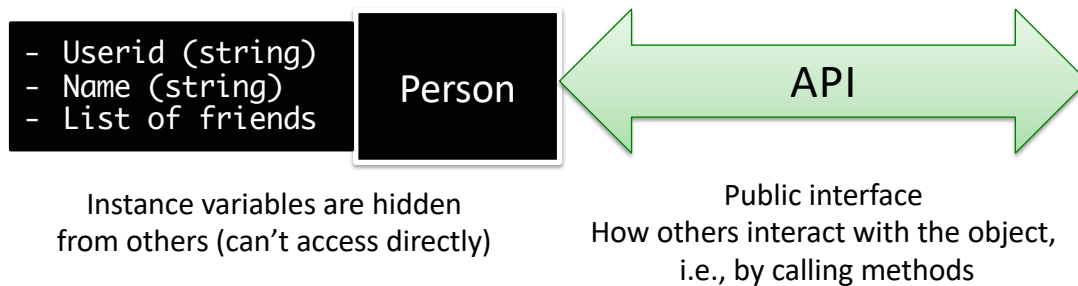
Review API Handout

Social Network Classes

```
- Userid (string)
- Name (string)
- List of friends
```

Person

API

Instance variables are hidden
from others (can't access directly)

Public interface
How others interact with the object,
i.e., by calling methods

Social Network Classes

```
- Userid (string)
- Name (string)
- List of friends
```
Person

← **API** →

Instance variables are hidden
from others (can't access directly)

Public interface
How others interact with the object,
i.e., by calling methods

```
- Dictionary of
  userids mapped
  to Person object
```
**Social
Network**

← **API** →
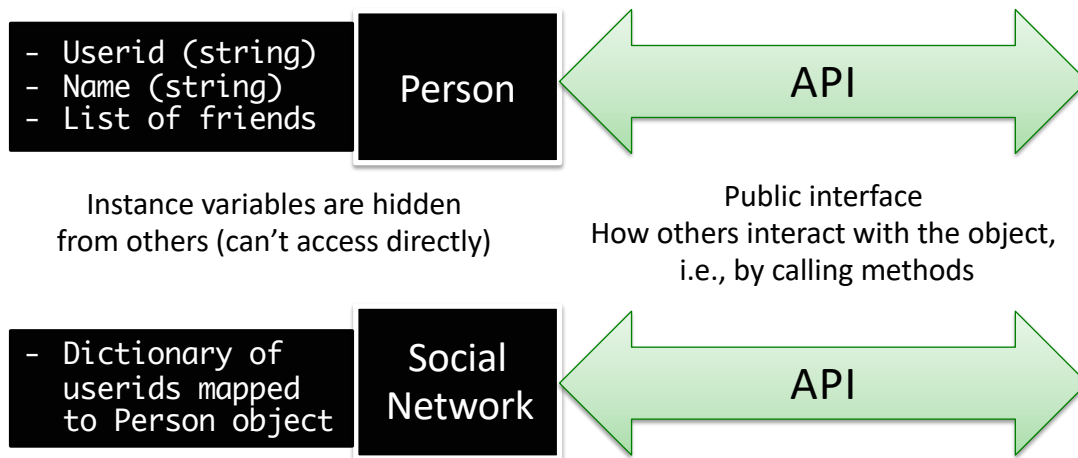
Towards a Solution and Hints

- Given "stubs" for both of the class files

- `social.py` is the most filled out

  ➤ Has the methods and docstrings defined

  ➤ **BUT** still refer to the description on the lab web page
    for all information

  > For every problem you're trying to solve, think about the variables you're
  > dealing with and their data types
  > - Is there a method/operation/function I can use to solve this problem?
  >   - Consult the SocialNetwork API handout as appropriate
  >   - Update your Person class's API on the handout

Problem: People Files

- Given the file name of a 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

13

Algorithm: People Files

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

- Algorithm:
  - ➤ Open file
  - ➤ Read the [first] line in the file
    - that represents the number of users in the file
  - ➤ Repeat <number of users> times
    - Read the line → that's the userid of the person
    - Read the line → that's the name of the person
    - Create a Person object
      - ➤ Update the Person's name
    - Add the Person object to the social network
  - ➤ Close the file

> File's `readline()` always reads in the **next** line of the file

14

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

Algorithm: 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
```

- For each line in the file

  ➢ Split the line into the two ids

  ➢ Add connection between the two people

UI Specification

- Checks if user entered command-line argument
  - ➢ 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

Demo

17

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
```

Why not a GUI?

18

Implementation Plan

1. Implement `Person` class
   ➢ Test (write test function, e.g., `testPerson()`)
2. Implement `SocialNetwork` class
   ➢ Example runs in lab write up
   ➢ Note: Methods for classes will **not** prompt for input; Use **input parameters**
   ➢ Test
3. Implement user interface program

19

Plan for Implementing a Class

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

20

Export SocialNetwork to Files

- I provide method to write connections to a file
  - Because only want connection once
- You handle writing to people file
  - Must be in **same format** that you read in
  - Just "undoing" the read
- Good test: if you read in a people file, export it to another file → original and exported file should look similar
  - If you read in that exported file, should see same social network
  - Files themselves may not be exactly the same because of order printed out

Test Data

- SocialNetwork requires: People file, Connections file
- Social Networks:
  - Simple
  - Hollywood
  - Randomly generated files
    - From W&L first and last names, randomly combined, connected
- Can combine multiple files (with unique usernames) to create larger social networks

COMMAND-LINE ARGUMENTS

23

Motivating Command-Line Arguments

- To make it easier to use the social network with different files, we're going to use command-line arguments

- How?

  ➤ Rather than typing out file names, we can use tab-completion on the command line

  ➤ To run the program with the same files, hit up to run the command again

24

Motivating Command-Line Arguments

- Ease executing Instaface
- Examples:

```
python instaface.py
```
Uses the default files

Template:
```
python instaface.py <peoplefile> <connectionsfile>
```
Uses the files specified

Example:
```
python instaface.py data/hollywood.txt data/hollywood_connections.txt
```

25

Command-line Arguments

- We can run programs from terminal (i.e., the "command-line") and from IDLE
- From the command-line, can pass in arguments, similar to how we use Unix commands
  - Ex: cp `<source>` `<dest>`

    Command-line arguments

  - Ex: python `myprog.py 3`
- Makes input easier
  - Don't have to retype each time executed

26

Command-line Arguments

- Examples:

python myprogram.py 3

python command_line_args.py <filename>

List of arguments, named **sys.argv**

- How can we access <filename>?
  - Then we can use in our program!

Using Command-line Arguments

- Using the **sys** module

python command_line_args.py <filename>

command_line_args.py	<filename>
0	1

sys.argv →

- How can we access <filename>?
  - sys.argv is a *list* of the arguments
  - sys.argv[0] is the name of the program
  - sys.argv[1] is the filename

Using Command-line Arguments

- In general in Python:
  - ➢ `sys.argv[0]` is the Python program's name
- Have to run program from terminal (not from IDLE)
  - ➢ Can still *edit* program in IDLE though
- ➡ Useful trick:
  - ➢ If can't figure out bug in IDLE, try running from command-line
    - May get different error message

29

Lab 10 Notes

- READ the instructions and given code carefully!

- SocialNetwork: given a bunch of code, so development process changes a bit
  - ➢ See class's doc string for more info

- Pause, deep breaths, read, review

30

Looking Ahead

- Lab 10 due on Friday

31