

Lab 10

- Social Network

1

Review Lab 9

- How can you get all the values from a dictionary?
 - How can you turn it into a list?

More work with dictionaries in lab10.
Make sure you understand how to use dictionaries.

2

Review

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

3

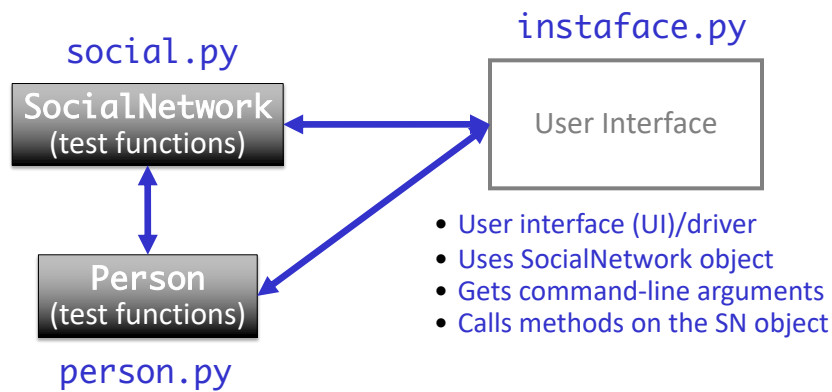
LAB 10: THE CAPSTONE LAB

Bringing together [almost] everything from the semester

4

Lab 10 Social Network Design

- 2 classes: Person and SocialNetwork



March 30, 2021

Sprenkle - CSCI111

5

5

Social Network Classes/Driver Data

- Person
 - User id
 - Name
 - Friends
- Social Network
 - People in network
- UI/Driver
 - Social network

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

March 30, 2021

Sprenkle - CSCI111

6

6

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/Driver
 - 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

How should we test these?

March 30, 2021

Sprenkle - CSCI111

7

7

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?
 - SocialNetwork API handout
 - Update your Person class's API on the handout

March 30, 2021

Sprenkle - CSCI111

8

8

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 Johanssen
warmachine
Don Cheadle
```

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

March 30, 2021

Sprenkle - CSCI111

9

9

Algorithm: People Files

- 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 dictionary
 - Close the file

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

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

March 30, 2021

Sprenkle - CSCI111

10

10

Problem: Connection Files

- Given a connection file that has the format

```
<user_id> <user_id>  
<user_id> <user_id>  
...  
<user_id> <user_id>
```

- 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>
```

- For each line in the file
 - Split the line into the two ids
 - Add connection between the two people
 - Look up the two Persons by their ids
 - Make the two Persons friends

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

March 30, 2021

Sprenkle - CSCI111

13

13

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?

March 30, 2021

Sprenkle - CSCI111

14

14

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

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

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

March 30, 2021

Sprenkle - CSCI111

17

17

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

March 30, 2021

Sprenkle - CSCI111

18

18

COMMAND-LINE ARGUMENTS

19

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: `python3 myprog.py 3`


- Makes input easier
 - Don't have to retype each time executed

20

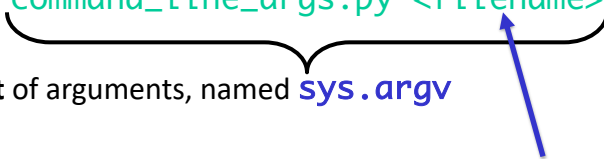
Command-line Arguments

- Using the **sys** module
 - What else did we use from the **sys** module?

```
python3 myprogram.py 3
```



```
python3 command_line_args.py <filename>
```



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

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

March 30, 2021

Sprenkle - CSCI111

21

21

Command-line Arguments

- Using the **sys** module

```
python3 command_line_args.py <filename>
```

sys.argv →

command_line_args.py	<filename>
0	1

- 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

March 30, 2021

Sprenkle - CSCI111 `command_line_args.py`

22

22

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 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

March 30, 2021

Sprenkle - CSCI111

23

23

Use in Lab 10

- Ease executing Instaface
- Examples:
 - `python3 instaface.py <peopleFile.txt>`
`<connectionsFile.txt>`
 - `python3 instaface.py data_files/hollywood.txt`
`data_files/hollywood_connections.txt`

March 30, 2021

Sprenkle - CSCI111

24

24

Looking Ahead

- Implementation up through Social Network should be completed by Thursday at midnight
- Lab is due Monday before class