

## Objectives

- Lab 10 Review
- Overriding methods
- Helper methods
- Search strategies

Mar 29, 2023

Sprenkle - CSCI111

1

1

## Progression

- Lab 2: tricycle
  - Used API of objects/classes that were defined elsewhere
- Lab 9: training wheels
  - You were given most of a class definition
  - Had to test it, find common bugs
  - Fill in some methods
- Lab 10: training wheels came off
  - Given a “stub” of a class definition
  - Need to implement, test
  - Practice, practice, practice!

Mar 29, 2023

Sprenkle - CSCI111

2

2

## Lab 10 Review

- Solving a real problem
- Started with designing the solution from a vague specification
- Broke into smaller problems (different classes, different responsibilities)
- Implementing smaller components
  - Following the specification
- Building to large component

Mar 29, 2023

Sprenkle - CSCI111

3

3

## Lab 10 Discussion

- How can we call *other* methods of that data type when we're in one method of the data type?
  - Example: If I'm in the `__str__(self)` method of the `Person` class, how can I call the `getNumFriends()` method?
- How do the `SocialNetwork` class and `Person` class work together?

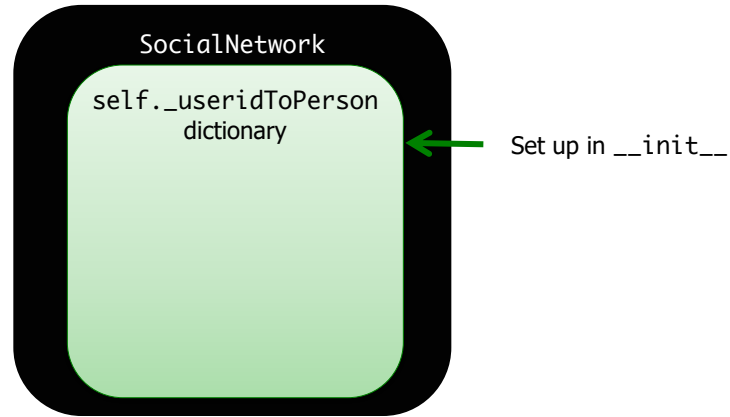
Mar 29, 2023

Sprenkle - CSCI111

4

4

## Lab 10: SocialNetwork



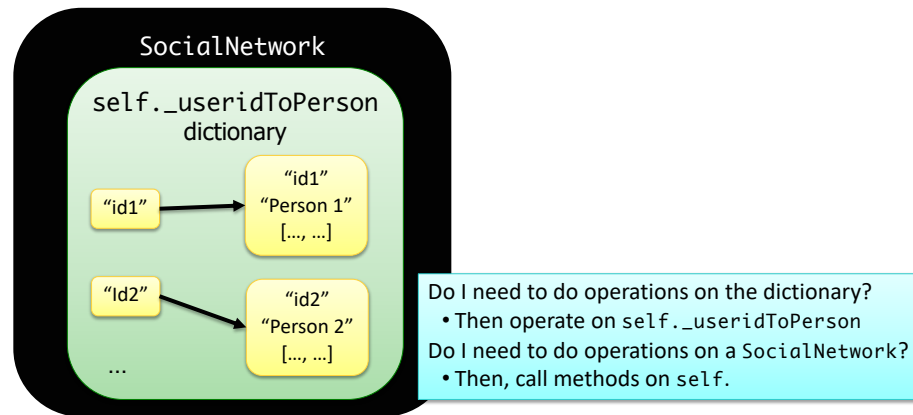
Mar 29, 2023

Sprenkle - CSCI111

5

5

## Lab 10: SocialNetwork



Mar 29, 2023

Sprenkle - CSCI111

6

6

## Notice How Problems Broke Down...

- In Person class
  - Concatenating strings was probably the hardest part
- In SocialNetwork class
  - What can I do with a dictionary? How do I do this on a dictionary?
  - What can I do with a file?
- Big problems break down into problems that you can easily solve, if you are comfortable with strings, dictionaries, files, ...

Mar 29, 2023

Sprenkle - CSCI111

7

7

## The Common Conundrum

- You have a large tool box.
- Keep track of all the tools you have in your box
  - You will be combining a variety of tools in different ways

*This is **Problem Solving!***

Mar 29, 2023

Sprenkle - CSCI111

8

8

## The Common Conundrum

- You have a large tool box.
  - Keep track of all the tools you have in your box
    - You will be combining a variety of tools in different ways
- This is **Problem Solving!***
- How can you figure out what tool to use?
    - What information do I have? What do I need?
    - How is the information represented? What is its type?
    - What operations/methods/functions are available?
    - When I ran into this situation before, how did I solve it?
    - How can I make it clearer what is going on?

Mar 29, 2023

Sprenkle - CSCI111

Lab 10 FAQ for common issues

9

9

## Testing Mutators

(Assuming object was already created)

1. Run mutator method
2. Use getter to test that mutator worked

Mar 29, 2023

Sprenkle - CSCI111

10

10

## References

- Check out the slides for lab10
  - [Hints on reading in files](#)
- Lab 10 FAQ
- When did I solve a similar problem? – Refer back to that problem

Mar 29, 2023

Sprenkle - CSC111

11

11

**`__LT__` and `__EQ__` METHODS**

Mar 29, 2023

Sprenkle - CSC111

12

12

## `__eq__`: Compare Objects of Same Type

- Header: `def __eq__(self, other)`
  - **Assumption:** `other` is another object of the *same type*
- Returns
  - True if `self` is equivalent to `other`
  - False otherwise
- If *override* the method in your class, can use objects in comparison expressions with `==`

How would you determine if two `Card` objects are equivalent?

Mar 29, 2023

Sprenkle - CSC1111

13

13

## `__lt__`: Compare Objects of Same Type

- Header: `def __lt__(self, other)`
  - **Assumption:** `other` is another object of the *same type*
- Returns
  - True if `self < other`
  - False otherwise
- If *override* method in your class, can use objects in comparison expressions, such as with `<` and `sort`

How do you compare two `Card` objects?

Mar 29, 2023

Sprenkle - CSC1111

14

14

## Comparing Objects of the Same Type

```
def __eq__(self, other):  
    """ Compares Card objects by their ranks and suits """  
    if type(self) != type(other):  
        return False  
  
    return self._rank == other._rank and self._suit == other._suit
```

```
def __lt__(self, other):  
    """ Compares Card objects by their ranks """  
    if type(self) != type(other):  
        return False  
  
    return self._rank < other._rank  
  
# Could compare by black jack or rummy value
```

Mar 29, 2023

Sprenkle - CSC1111

card.py

15

15

## DataFrequency Object

```
def __lt__(self, other):  
    """  
    Compares this object with other, which is also a  
    DataFrequency object.  
    Used by default when using the list's sort method.  
    """  
  
    return self._count < other._count
```

Could then sort the list of DataFrequency objects as

```
myDataFreqList = ... #create list  
myDataFreqList.sort()
```

sort automatically calls the `__lt__` method

The **key** parameter to sort method adds flexibility/customization

Mar 29, 2023

Sprenkle - CSC1111

16

16



## HELPER METHODS

Mar 29, 2023

Sprenkle - CSC1111

17

17

## Helper Methods

- Part of the class
- **Not** part of the API
- Make your code easier to write (internally) but others outside the class shouldn't use
- Convention: method name begins with “\_”

Let's create a method that determines if a Card is a face card!

Mar 29, 2023

Sprenkle - CSC1111

18

18

## Example Helper Method

Helper Method:

```
def _isFaceCard(self):  
    if self._rank > 10 and self._rank < 14:  
        return True  
    return False
```

In use:

```
def rummyValue(self):  
    if self._isFaceCard():  
        return 10  
    elif self._rank == 10:  
        return 10  
    elif self._rank == 14:  
        return 15  
    else:  
        return 5
```

Mar 29, 2023

card.py

Sprenkle - CSC1111

19

19

## Example Helper Methods

- Naming with underscore *loosely* enforces that other can't use
  - Does not show up in `help`
  - Does show up in `dir`
    - Shows all properties, methods of object

Helper Method:

```
def _isFaceCard(self):  
    if self._rank > 10 and self._rank < 14:  
        return True  
    return False
```

Mar 29, 2023

card.py

Sprenkle - CSC1111

20

20

# SEARCHING

21

## Search Using `in`

- Iterates through a list, checking if the element is found
- Known as *linear search*
- **Implementation:**

```
def linearSearch(searchlist, key):  
    for elem in searchlist:  
        if elem == key:  
            return True  
    return False
```

value	8	5	3	7
pos	0	1	2	3

What are the strengths and weaknesses of implementing search this way?

22

## Linear Search

- **Overview:** Iterates through a list, checking if the element is found
- **Benefits:**
  - Works on *any* list
- **Drawbacks:**
  - Slow -- needs to check each element of list if the element is not in the list

Mar 29, 2023

Sprenkle - CSC1111

23

23

## High-Low Game/TPIR Clock Game

- I'm thinking of a number between 1-100
- You want to guess the number as quickly as possible, i.e., in fewest guesses
- For every number you guess, I'll tell you if you got it right. If you didn't, I'll tell you whether you're too high or too low

Reminder: write down guesses

Mar 29, 2023

Sprenkle - CSC1111

24

24

## High-Low Game/TPIR Clock Game

- I'm thinking of a number between 1-100
- You want to guess the number as quickly as possible, i.e., in fewest guesses
- For every number you guess, I'll tell you if you got it right. If you didn't, I'll tell you whether you're too high or too low

→ What is your best guessing strategy?

Mar 29, 2023

Sprenkle - CSC1111

25

25

## Strategy: Eliminate Half the Possibilities

- Repeat until find value or looked through all values
  - Guess middle value of possibilities
  - If match, found!
  - Otherwise, find out too high or too low
  - Modify your possibilities
    - Eliminate the possibilities from your number and higher/lower, as appropriate
- Known as **Binary Search**

Mar 29, 2023

Sprenkle - CSC1111

26

26

## Searching...

value	-3	0	0	1	2	7	8	9
pos	0	1	2	3	4	5	6	7

Use algorithm to search for key = 8

Mar 29, 2023

Sprenkle - CSC1111

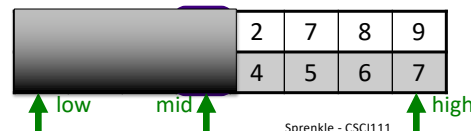
27

27

## Searching for 8

-3	0	0	1	2	7	8	9
0	1	2	3	4	5	6	7

- Find the middle of the list
  - Positions: 0-7, so mid position is  $((7+0)//2) = 3$
- Check if the key equals the value at mid (1)
  - If so, report the location
- Check if the key is higher or lower than value at mid
  - Search the appropriate half of the list



8 > 1, so look  
in upper half

Mar 29, 2023

Sprenkle - CSC1111

28

28

## Searching for 8

- mid is 5  $((7+4)//2)$ , list[5] is 7

...

2	7	8	9
4	5	6	7

↑ low    ↑ mid    ↑ high

8 > 7,  
so look in upper half

Mar 29, 2023

Sprengle - CSC1111

29

29

## Searching for 8

- mid is 5  $((7+4)//2)$ , list[5] is 7

...

2	7	8	9
4	5	6	7

↑

8 > 7,  
so look in upper half

- mid is 6  $((7+6)//2)$ , list[6] is 8

...

8	9
6	7

↑

8 == 8,  
FOUND IT at position 6!

What if searched for 6 instead of 8?

Mar 29, 2023

Sprengle - CSC1111

30

30

# Searching...

value	-3	0	0	1	2	7	8	9
pos	0	1	2	3	4	5	6	7

Use algorithm to search for key = 6

Mar 29, 2023 Sprengle - CSC1111 31

31

# Searching for 6

-3	0	0	1	2	7	8	9
0	1	2	3	4	5	6	7

- Will follow same execution flow, but 6 is not in the list
- mid is 6, list[5] is 7
 

...	2	7	8	9
	4	5	6	7

$6 < 7$ , so will try to look in lower half of the list
- mid is 4, list[4] is 2
 

...	2	...
	4	

$6 > 2$ , so will try to look in upper half of the list, but we've already determined it's not there.

**How do we know to stop looking?**

Mar 29, 2023 Sprengle - CSC1111 32

32



## Implementation Group Work

```
def search(searchlist, key):  
    """Pre: searchlist is a list of integers in  
    sorted order.  
    Returns the position of key (an integer) in  
    the list of integers (searchlist) or -1 if  
    not found"""
```

- Trace through your function using examples
  - Start simple (small lists)
  - Do what the program says *exactly*, not what you *think* the program says

Mar 29, 2023

Sprenkle - CSC1111

33

33

## Looking Ahead

- Lab 10 due Friday
- No broader issue

Mar 29, 2023

Sprenkle - CSC1111

34

34