CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2017

Swarthmore College

Outline Nov 3:

- Quiz 3 (10:30-11am)
- Hand back Lab 5
- Assert
- Slicing
- Graphics stack diagram example (if time)

Notes

- **Lab 7** due **Saturday** night (email me when you finish TDD!)
- Office Hours **TODAY 3-5pm and by appointment**

Lab 5 feedback

Lab 5 (graphics) feedback

- Most people lost points due to hard-coding
  - No line separation variable in lineart.py
  - Moon cover movement not based on the moon's raduis in night.py
- Several cases of under or over commenting
- Commenting formula:
  - **line break**
  - **comment on it's own line**
  - **code block (2-6 lines)**
  - (very short comments can be inline)
  - (indentation should match the level of the code)

Commenting example from word_guesser main

```
# define a hidden word and a list of spaces for the user of the same length
hidden_word = "mystery"
user_lst = len(hidden_word)*["_"]
bad_letters = []

# keep asking the user for letters until they have guessed the word or
# reached 6 letters
guessed = False
while not guessed and len(bad_letters) < 6:

    # print the current progress and get the user's guess
    display_round(user_lst, bad_letters)
    letter = get_guess(user_lst, bad_letters)

    # if the letter was in the hidden word, update the current progress
    if letter in hidden_word:
        process_letter(letter, hidden_word, user_lst)

    # otherwise, add the letter to bad_letters
    else:
        bad_letters.append(letter)

    # if the user has completed the hidden word, what should happen?
    if correct(user_lst, hidden_word):
        guessed = True

# after the game is over, display the end result and print messages to user
display_round(user_lst, bad_letters)
if guessed:
    print("Congratulations, you solved it!")
else:
    print("Sorry, you got 6 letters wrong, game over!")
```

Method vs. Function

- A method is called by a specific instance using “dot” notation

- Both methods and functions can have any number of parameters (including none), and both can return a value:

```
n = get_user_int()
x = pt.getX()
```

- Both methods and functions can return nothing (print or mutate):

```
display(board)
lst.append(item)
```

Mid-semester feedback

In-class options

	LESS	MORE	AS IS
Slides	1	2	22
Board	0	8	17
Handouts	5	7	13
Coding: group	9	8	8
Coding: self	3	12	10
Coding: partner	6	7	12

Office Hours

- Several people said 3-5pm is good, but not on Fridays
- I am usually around 3-5pm Mon & Tues
- Email me for a 30min appointment if these times don’t work for you

What is helping your learning?

- Notes (2)
- Lab (5)
- Office hours (2)
- Experimenting (2)
- Ninja sessions (4)
- Coding in class (5)
- Help from others (2)
- Practice (6)
- Reading the textbook (2)

Other feedback

- Posting slides before class
- Posting practice problem solutions
- Introduce new material more slowly
- More office hours
- Collaboration and talking about code helps

Assert

Assert

- A key word and a way of checking that certain conditions are met before proceeding
- Very helpful for debugging
- **AssertionError** is another type of error (like **ValueError**)
- The expression after **assert** *must* evaluate to a **boolean**

```
>>> x = 7
>>> assert x < 10
>>> assert x > 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError
>>> guess = input("> ")
> gladiator
>>> assert len(guess) == 9
>>>
>>> my_lst = [3,4,5]
>>> assert 3 in my_lst
>>> assert 7 in my_lst
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError
```

Slicing

Slicing (for sequences: list, string, range)

- Slicing produces a subset of the original sequence
- It does not change the original sequence

String example

```
>>> word = "Friday"
>>> word[3:6]
'day'
>>> word[:3]
'Fri'
>>> word[3:]
'day'
>>> word[:]
'Friday'
```

List example

```
>>> my_lst = [0,1,2,3,4,5,6,7,8,9]
>>> my_lst[3:6]
[3, 4, 5]
>>> my_lst[:3]
[0, 1, 2]
>>> my_lst[3:]
[3, 4, 5, 6, 7, 8, 9]
>>> my_lst
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```