Objectives

- Python review, continued
- Design Discussion

Review && Submission

- What is the difference between comments and doc strings?
- What is the difference between functions and methods?
- What is the difference between *inheriting* and *importing*?
- What is the difference between instance, class, and local variables?
  - What questions should you ask to determine which you should use?
- Why is exception handling not necessarily the same as error handling?
- Discuss the trickiest terms
  - Each pod: Put top 3 on white board

In parallel, show Professor Sprenkle terminology table

Answers, in Brief

- All doc strings are comments; not all comments are doc strings
  - Doc strings are specifically to describe interface for functions/methods and for classes
- Methods: specific to objects of a certain class
- Inherits: get all properties/methods of parent
  - import: just *uses* code from other
- Instance: one for each object of class
  - Class: one for all objects of class
  - Local: short-lived variable for a specific piece of code
- Error handling: broader

Review Terms

Definition, Example, Indicator/Cues

Variable	Object
[code] block	Encapsulation
Comment	Parameter (formal and actual)
Doc String	Argument
Function, both calling and defining	Interface
Data Type	Abstract Class
Class	Inheritance
self	Condition
Constructor, both defining and calling	Loop
Method, both defining and calling	Exception
Attribute/Field	Exception handling
State	Error handling
Instance variable	Import
Class variable	
Local variable	

Trickiness in Terms

- Synonyms and related terms are often the hardest
  - State >> Variables >> Attributes, fields
    - State is a more general term (and can refer to variables collectively)
    - All attributes/fields are variables but not all variables are attributes/fields

Assignment Goals Recap

- To review how to program in Python
- To review the terminology of programming—with *precision*—so that we are better able to communicate throughout the semester (and beyond).

Design Questions

1. `turn` is an *instance* variable of the `Game` class.
   - Is it better design for `turn` to be a local, instance, or class variable? Justify your answer.
2. `user_input` is a *local* variable in the `getInput` method of the `ConnectFour` class.
   - Is it better design for `user_input` to be a local, instance, or class variable? Justify your answer.
3. `RANKS` is a *class* variable of the `Card` class.
   - Is it better design for `RANKS` to be a local, instance, or class variable? Justify your answer.
4. `tokens` is an *instance* variable of the `ConnectFour` class.
   - Is it better design for `tokens` to be a local, instance, or class variable? Justify your answer.
5. `Player` is a class in `war.py`.
   - Is it better design for the `Player` class to be defined in `war.py` or in `game.py`? Justify your answer.
6. `War`'s `step` method takes as a parameter `dummyInput`. What purpose does it serve?

Text Editors

- For editing (plain) text!
  - Only text/characters
    - Example: no font, size changes
  - As opposed to *rich* text
- Examples?
  - Basic: Notepad
  - No frills, all terminal: nano
  - For the nerdier: emacs, vi/vim
  - GUI, in increasing order of fancy: jEdit, gedit; heading toward IDE: Sublime, Pulsar, VSCode

Text Editors: What was that about?

- We'll use text editors to start in a couple of ways (git and programming)
- Want to stick with the basics/fundamental for now
  - Build on them!

Looking Ahead

- Design discussion
- Decide on your favorite text editor to use for development and for commit messages
  - Emacs, vim, jEdit, Pulsar, Sublime, Notepad++, VSCode, nano, …
  - We want to stick with the basics for now
- Complete the set up assignment