

Objectives

- Programming Refresher, continued
- Design Discussion

Open Poll Everywhere

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).

Fill in the blank: I found ~80% of the terms to be _____ to define/identify

Easy!

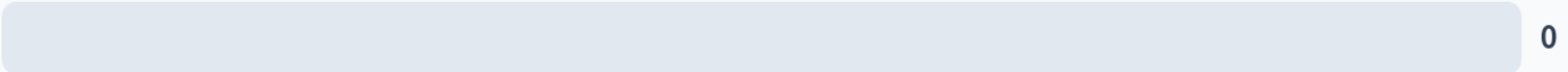
Hard!

Somewhere in the middle

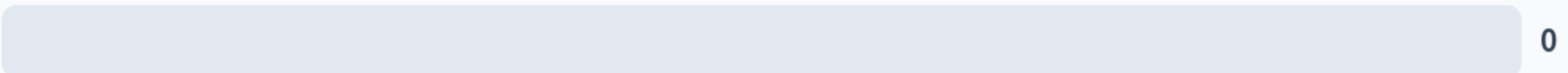
Easy after I heard the definition

Fill in the blank: I found ~80% of the terms to be _____ to define/identify

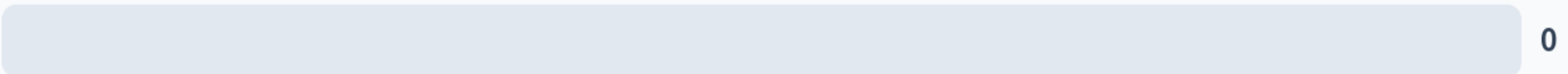
Easy!



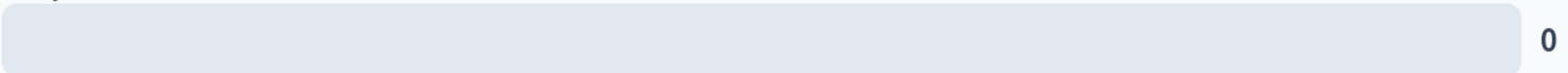
Hard!



Somewhere in the middle

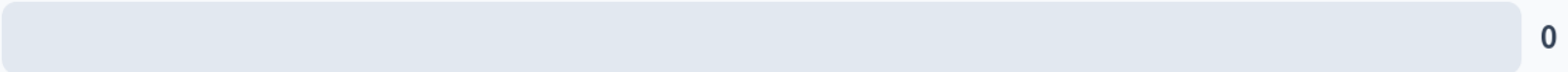


Easy after I heard the definition

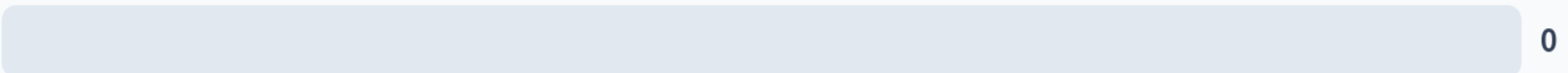


Fill in the blank: I found ~80% of the terms to be _____ to define/identify

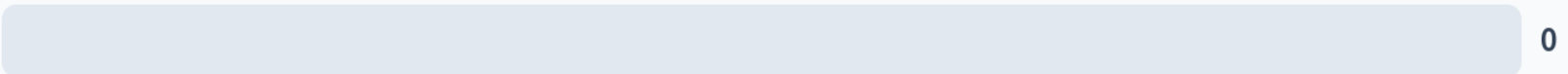
Easy!



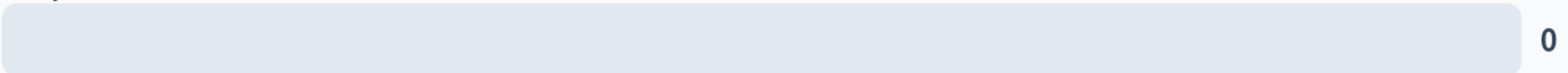
Hard!



Somewhere in the middle



Easy after I heard the definition



Trickiness in Terms

- Synonyms and related terms are often the hardest
- Consider *state, variables, attributes/fields*
 - How are they similar/different?

Trickiness in Terms

- Synonyms and related terms are often the hardest
- Consider state, variables, attributes/fields
 - 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

Review & Submission

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

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 the *interface* for functions/methods and for classes (i.e., how to use them)
- Methods: specific to objects of a certain class
- Classes define new data types
- Inheriting: get all attributes/methods of parent (stronger dependence)
 - `import`: just *uses* code from other
- Error handling: broader than exception handling
- Instance variable: one for each object of class
 - Class: one for all objects of class
 - Local: short-lived variable for a specific piece of code

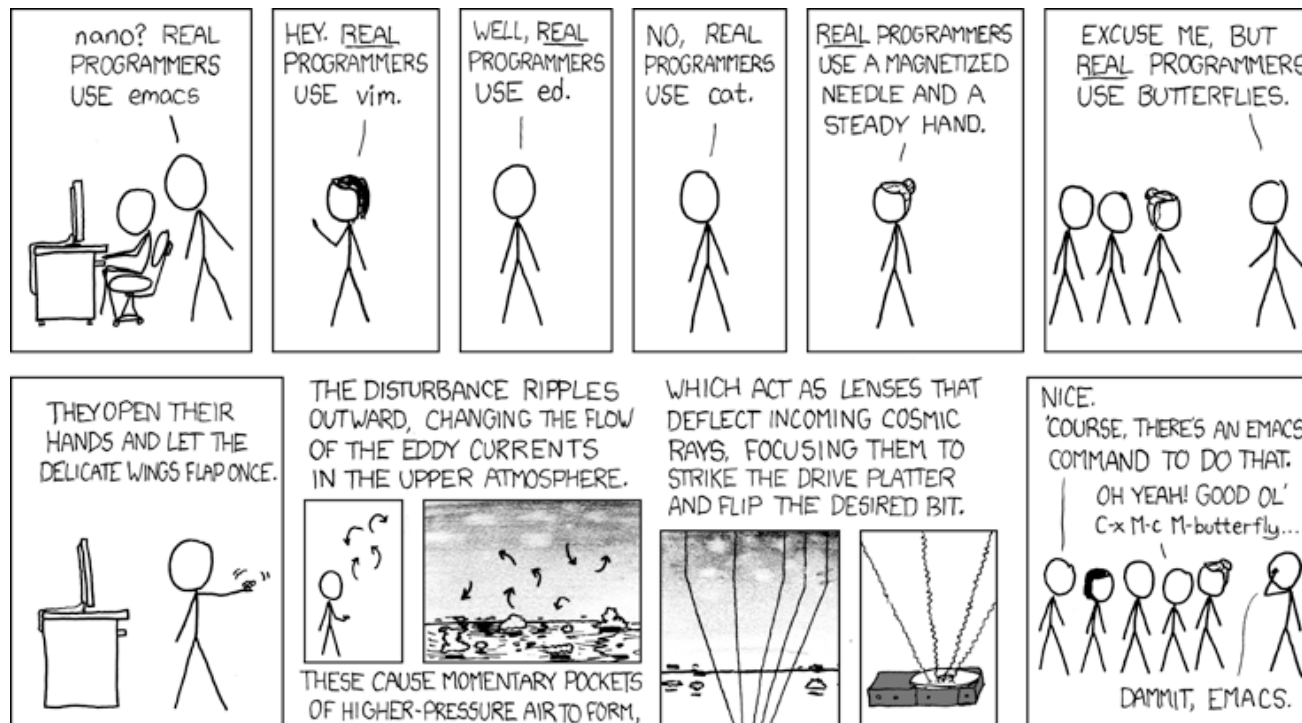
TEXT EDITORS

Text Editors

- For editing (plain) text—as opposed to *rich* text
 - Only text/characters
 - Example: no font, size changes
- We'll use text editors in a couple of ways:
 - Git (version control) commit messages
 - Programming
- Want to stick with the basics/fundamentals for now
 - Build on them later!

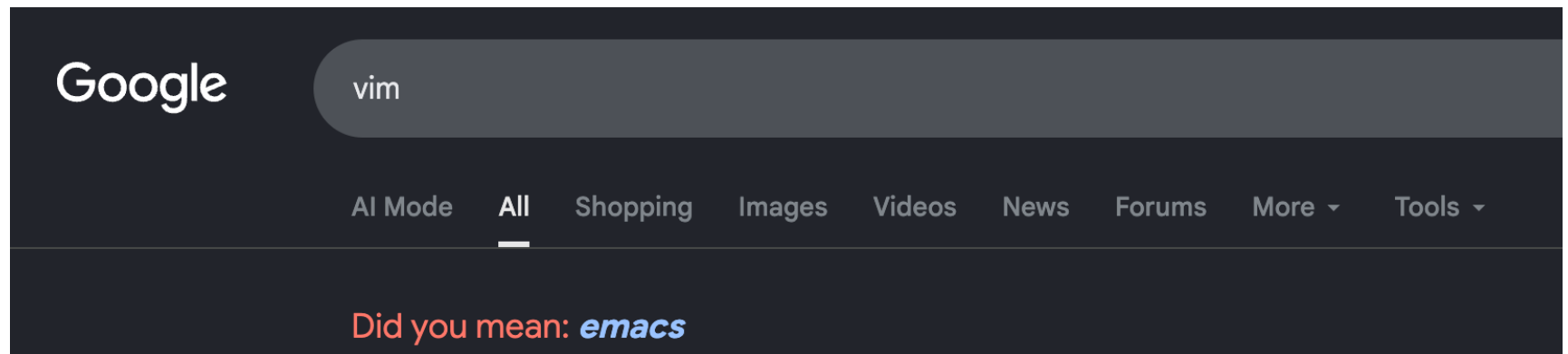
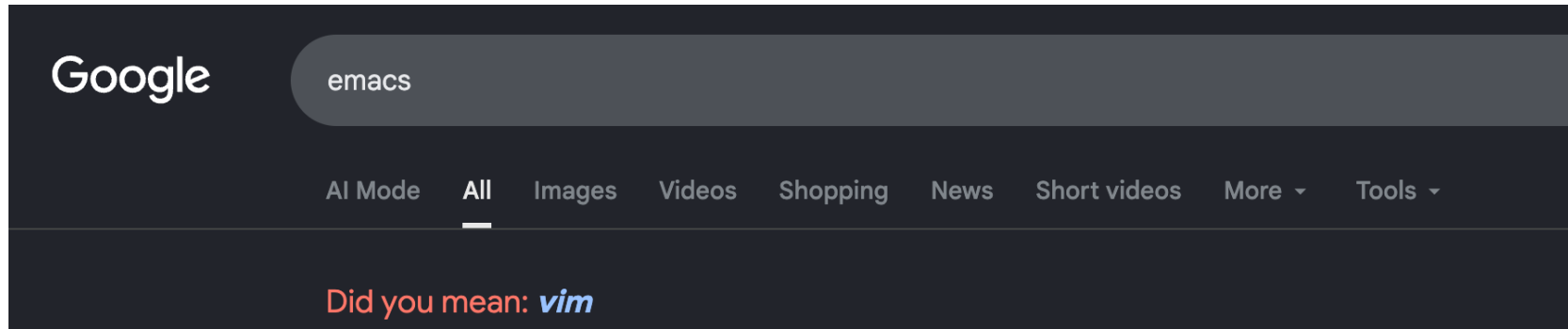
Examples of Text Editors

- Basic: Notepad
- No frills, all terminal: nano
- For the adventurous: emacs, vim
- GUI, in increasing order of fancy:
 - jEdit, gedit
 - heading toward IDE: Sublime (paid), Pulsar, VSCode



<https://xkcd.com/378/>

Text Editor Shade



Set Up Your Development Environment: Git

- Choose a text editor
 - Can change it later, just to get you started
- Set up Git, GitHub
- Due Friday before class

SOFTWARE DESIGN DECISIONS

Software Design Decisions

- We focused our review on terminology and the syntax/semantics of programming
- Let's talk a bit about design!
 - Assignment due Friday

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. `WarGame`'s `step` method takes as a parameter `dummyInput`. What purpose does it serve?

Looking Ahead: Friday

- Complete Design Discussion assignment
- Complete the git set up assignment
 - Decide on your 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
 - Can change later