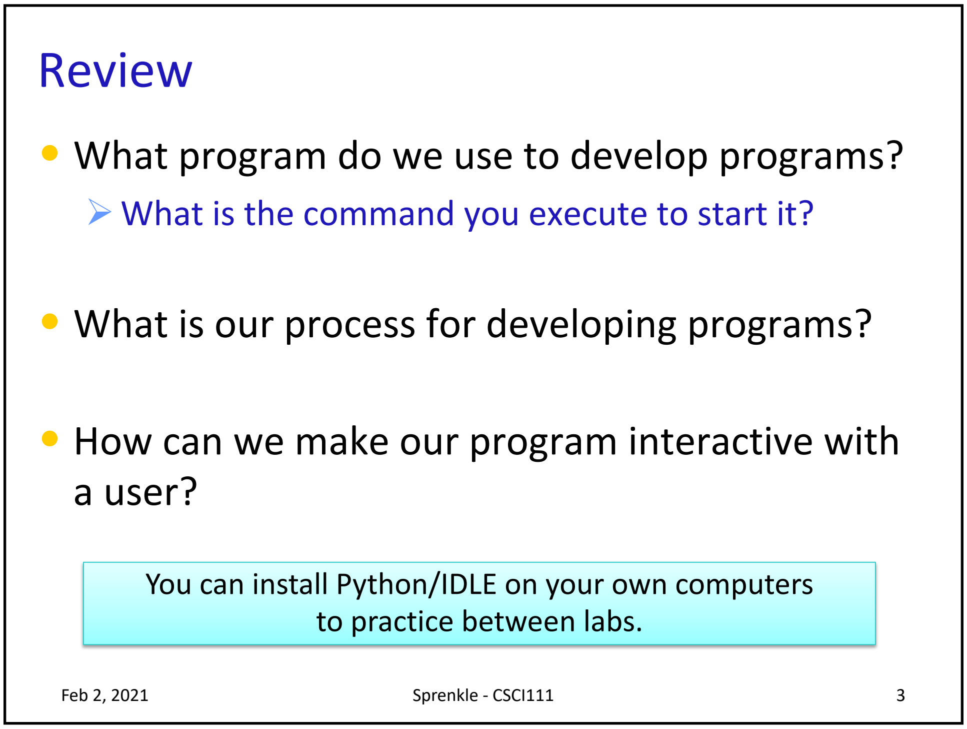
Objectives

- Review
- Lab 2
  - Programming practice

Feedback on Lab 1

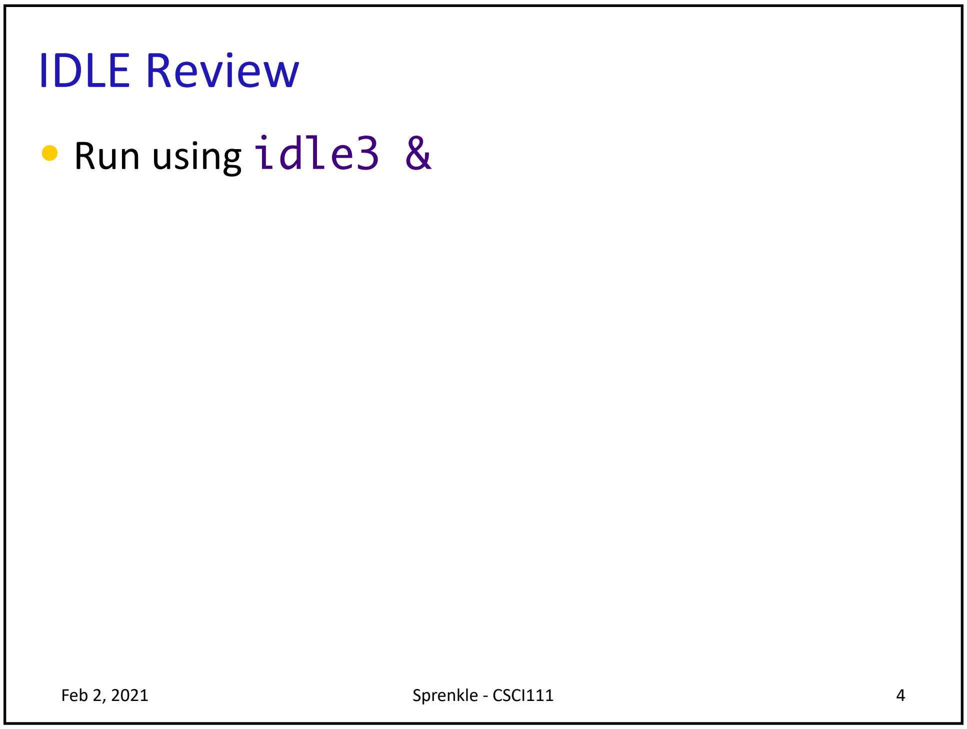
- Overall good
- Notes
  - Saved output from each program
    - With user input, try several different good test cases
  - Want *good* output
    - think about what the user wants to see
  - High-level comments
    - Describes what the program does
      - Helps for quick overview when reviewing
  - Electronic submission
    - In directory – looked good!



Review

- What program do we use to develop programs?
  - What is the command you execute to start it?
- What is our process for developing programs?
- How can we make our program interactive with a user?

You can install Python/IDLE on your own computers
to practice between labs.



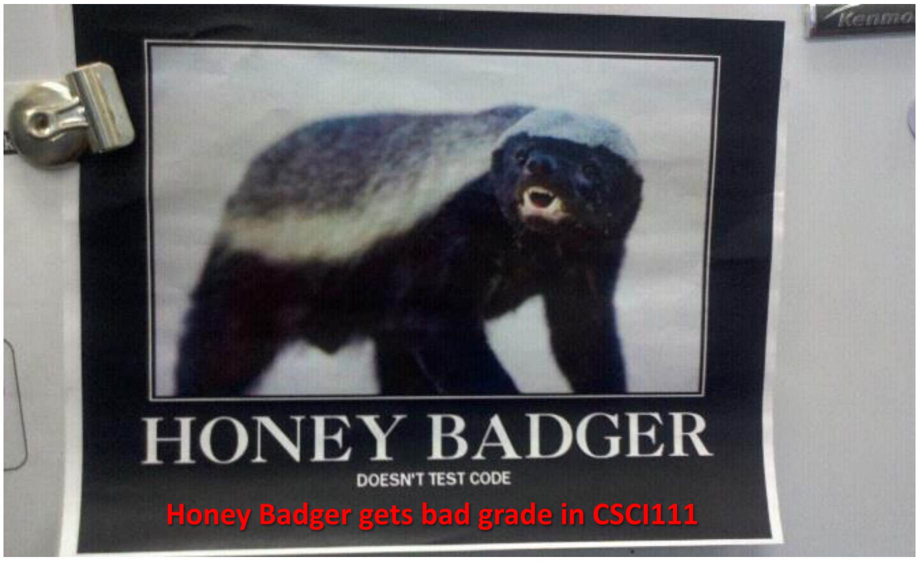
IDLE Review

- Run using `idle3 &`

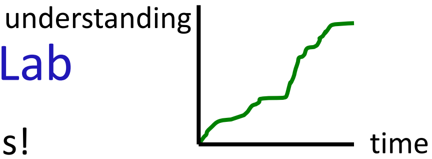
Review: Development Process

1. Create a sketch of how to solve the problem (the algorithm)
2. Fill in the details in Python
3. Test code using *good, varied* test cases to try to find errors in code
4. If program's output does not match the expected output, debug to find the problem and fix it
   - Repeat testing and debugging until no more faults
5. Make code "better", test again
   - Fix variable names, better comments

Testing



Lessons from Lab



- Look at examples!
    - “We were able to do this in that other program. How did we do that?”
    - On the course schedule page
- Explore!
    - Try things out in interactive mode
    - Then, put the ones that work into a script/program
- Testing!
    - Start with smaller and easy-to-verify tests
    - Test a variety of inputs
- Follow all of the directions!

Recommendation

- Get user input last – this is a fairly routine step
- Develop/test without getting input first
    - Speeds up process
- Then, add user input

Review: Linux Commands

- What is the command to…
  - Determine which directory you’re in?
  - View the contents of a directory?
  - Create a directory?
  - Copy a file?
  - Delete a file?
- How do you refer to … your home directory? The current directory? The parent directory?

Linux Command: mv

- Used to *move* or *rename* a file
- `mv <sourcefile> <destination>`
- Example usage:

```
mv file.py newfile.py
```

  - Renames `file.py` to `newfilename.py`

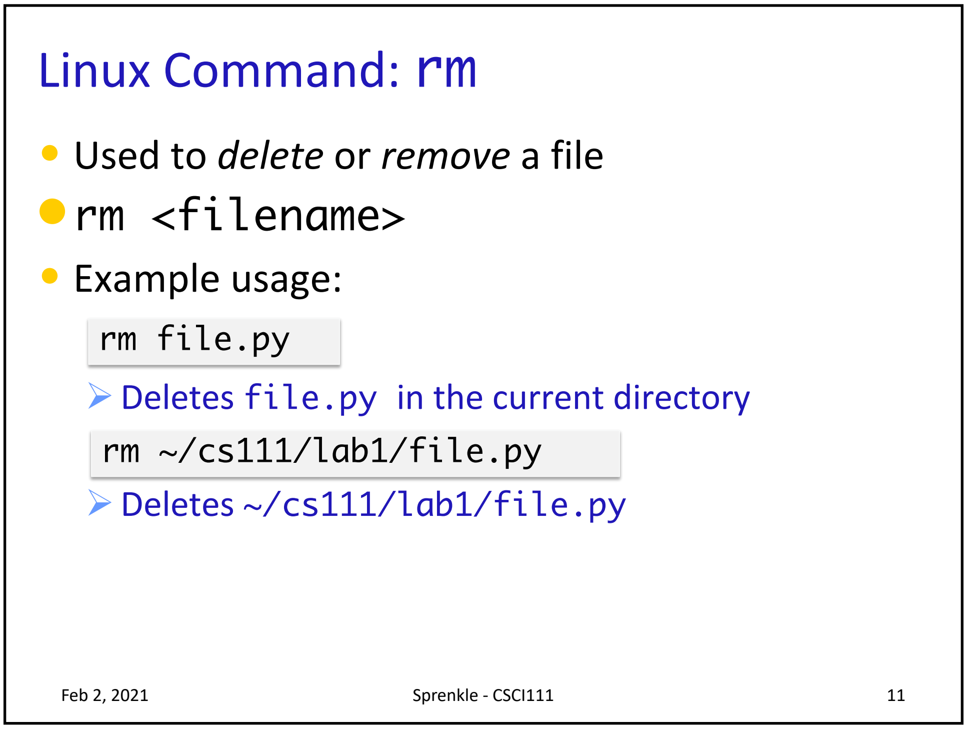
```
mv ~/cs111/file.py newfilename.py
```

  - Moves `~/cs111/file.py` to *current* directory with a new name
  - If `<destination>` is a *directory*, keeps the original source file’s name

```
mv ~/cs111/file.py ~/cs111/lab1/
```

directory

  - File `file.py` will now be in `cs111/lab1` directory



Linux Command: `rm`

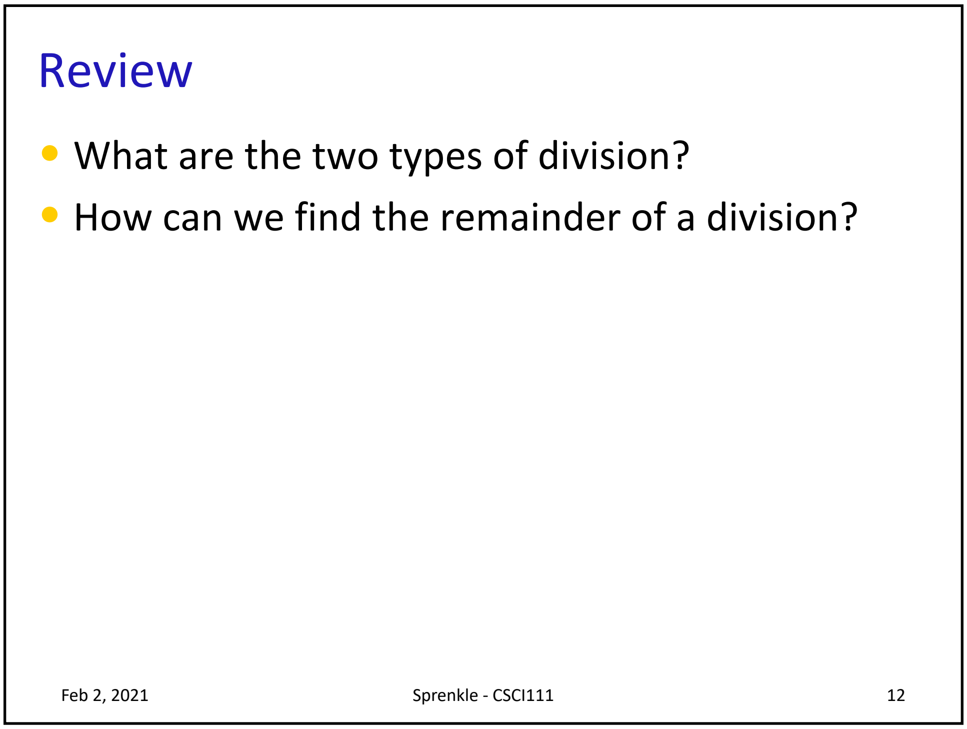
- Used to *delete* or *remove* a file
- `rm <filename>`
- Example usage:

```
rm file.py
```

  - Deletes `file.py` in the current directory

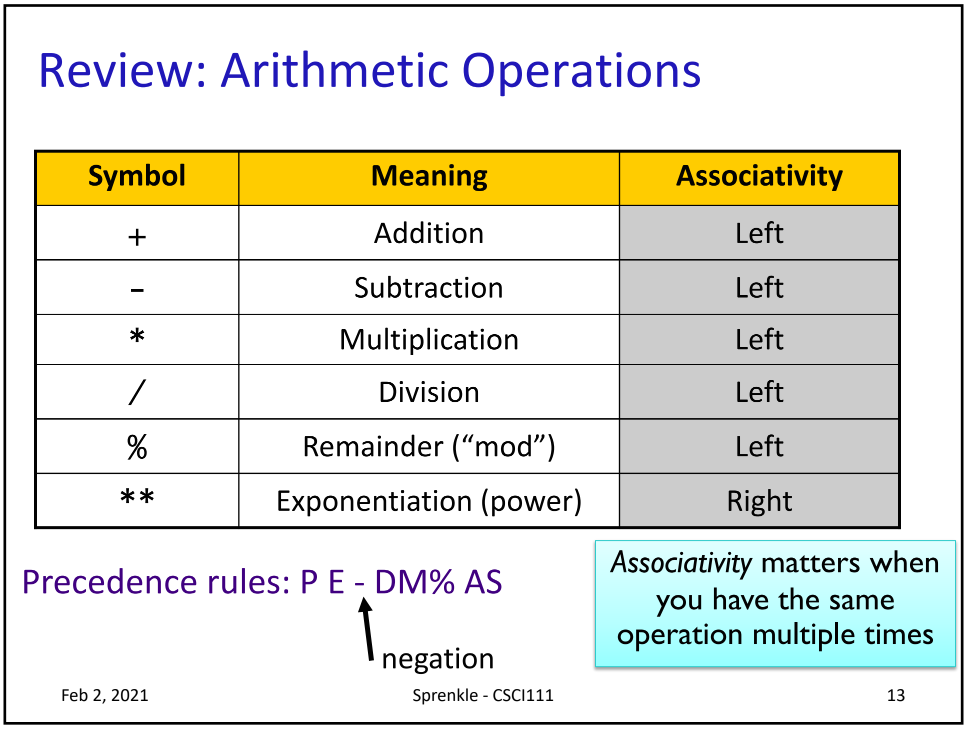
```
rm ~/cs111/lab1/file.py
```

  - Deletes `~/cs111/lab1/file.py`



Review

- What are the two types of division?
- How can we find the remainder of a division?



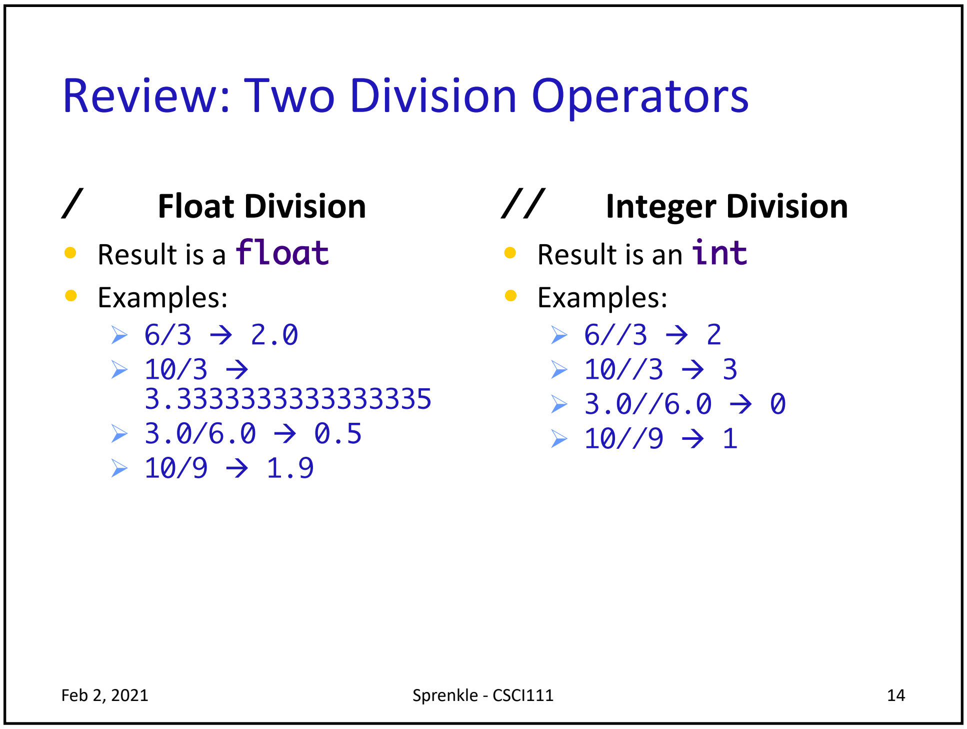
Review: Arithmetic Operations

Symbol	Meaning	Associativity
+	Addition	Left
-	Subtraction	Left
*	Multiplication	Left
/	Division	Left
%	Remainder ("mod")	Left
**	Exponentiation (power)	Right

Precedence rules: P E - DM% AS

negation

Associativity matters when you have the same operation multiple times



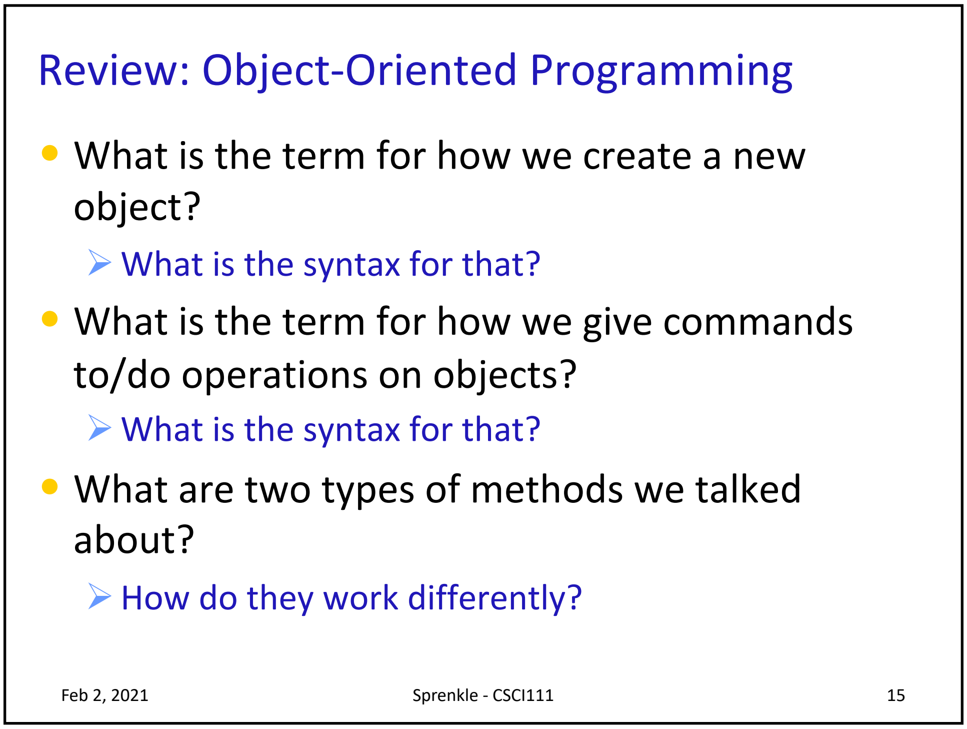
Review: Two Division Operators

/ Float Division

- Result is a `float`
- Examples:
  - `6/3 → 2.0`
  - `10/3 → 3.3333333333333335`
  - `3.0/6.0 → 0.5`
  - `10/9 → 1.9`

// Integer Division

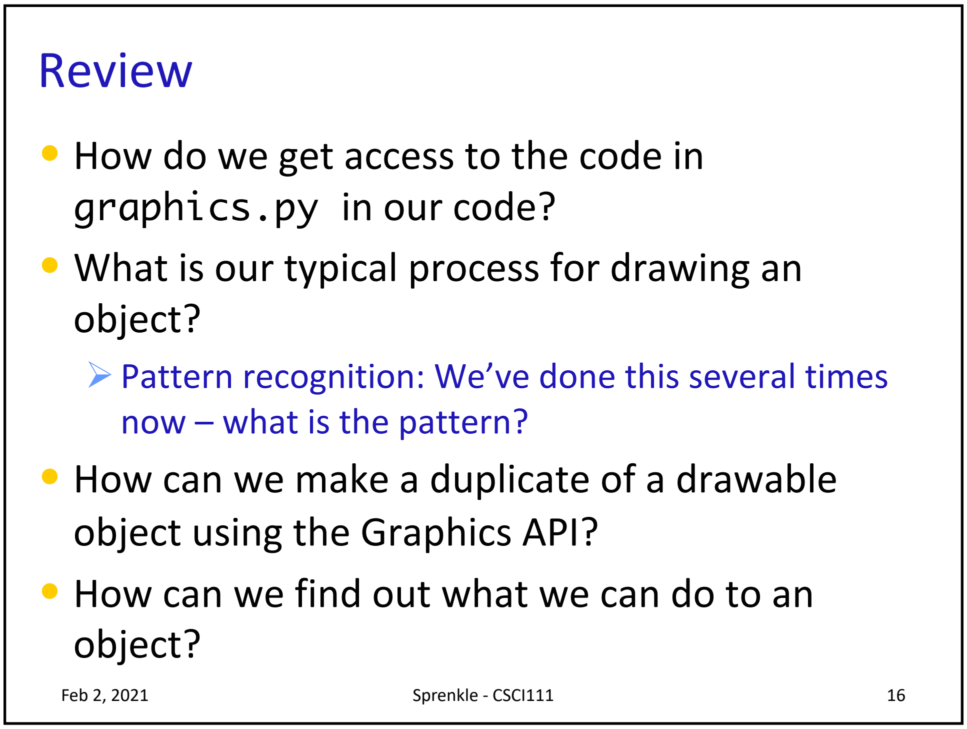
- Result is an `int`
- Examples:
  - `6//3 → 2`
  - `10//3 → 3`
  - `3.0//6.0 → 0`
  - `10//9 → 1`



Review: Object-Oriented Programming

- What is the term for how we create a new object?
  - What is the syntax for that?
- What is the term for how we give commands to/do operations on objects?
  - What is the syntax for that?
- What are two types of methods we talked about?
  - How do they work differently?

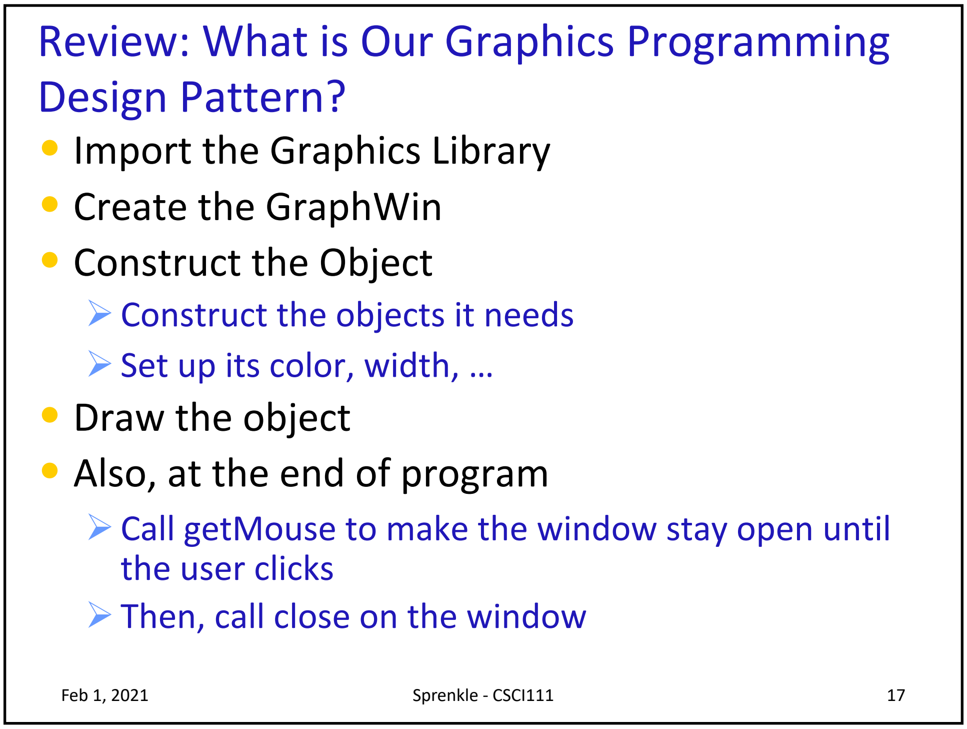
15



Review

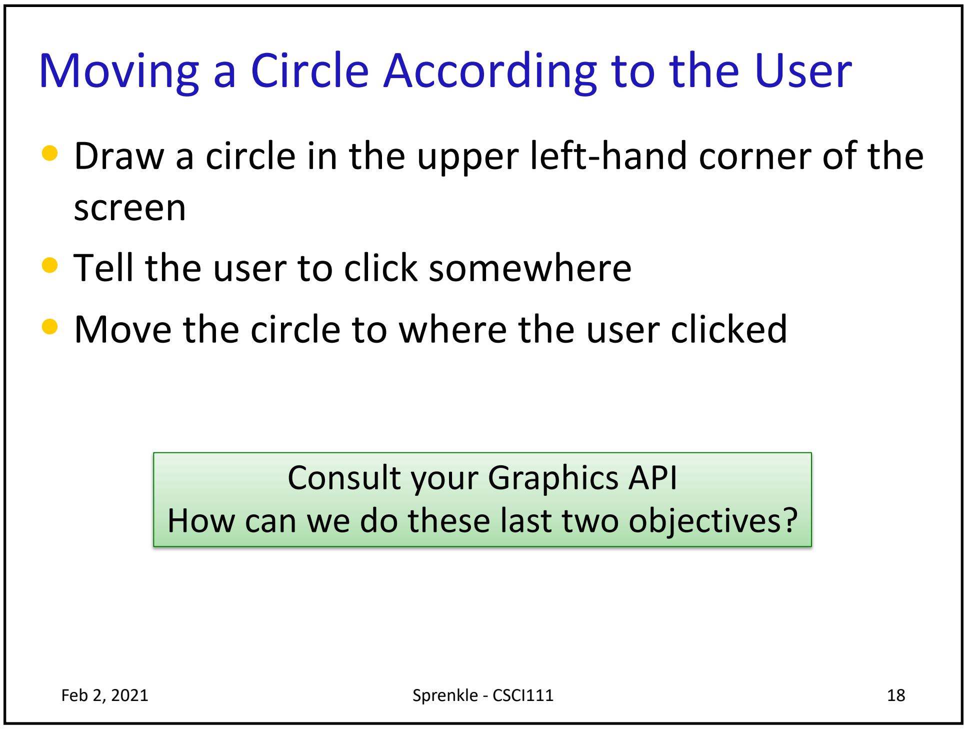
- How do we get access to the code in `graphics.py` in our code?
- What is our typical process for drawing an object?
  - Pattern recognition: We’ve done this several times now – what is the pattern?
- How can we make a duplicate of a drawable object using the Graphics API?
- How can we find out what we can do to an object?

16



Review: What is Our Graphics Programming Design Pattern?

- Import the Graphics Library
- Create the GraphWin
- Construct the Object
  - Construct the objects it needs
  - Set up its color, width, …
- Draw the object
- Also, at the end of program
  - Call getMouse to make the window stay open until the user clicks
  - Then, call close on the window



Moving a Circle According to the User

- Draw a circle in the upper left-hand corner of the screen
- Tell the user to click somewhere
- Move the circle to where the user clicked

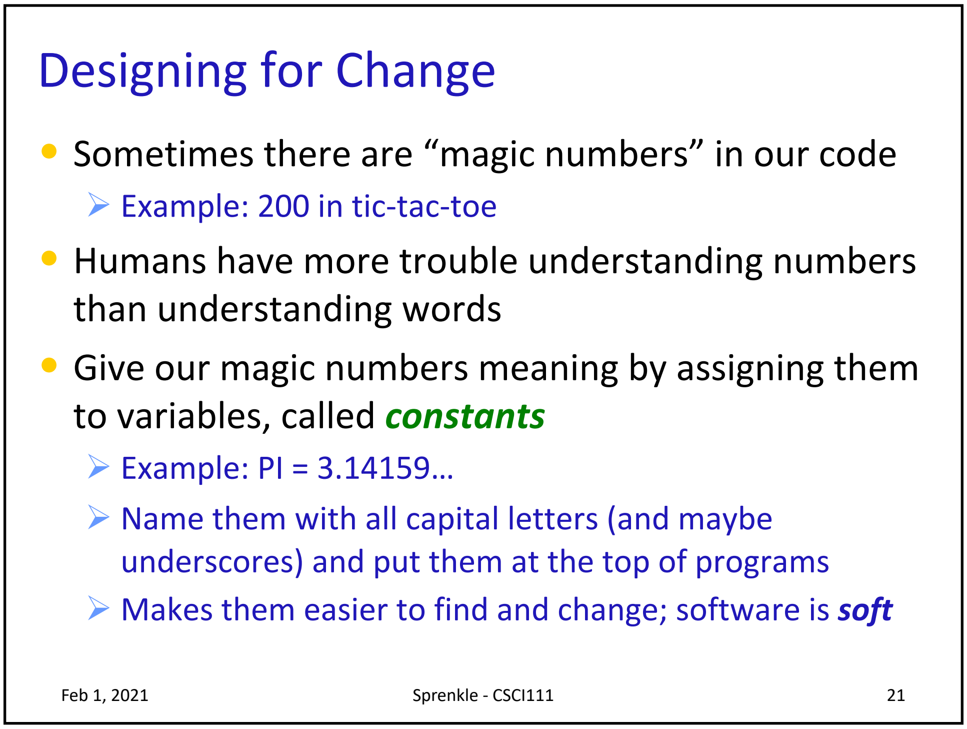
Consult your Graphics API
How can we do these last two objectives?

Moving a Circle According to the User

- Draw a circle in the upper left-hand corner of the screen
- Tell the user to click somewhere
  - If you print, user won’t necessarily see what you display
  - Use the Text object
- Move the circle to where the user clicked
  - `<GraphWinObj>.getMouse()`
    - *Returns* the user’s mouse click as a **`Point`** object
  - Save point as a variable, e.g.,
    - `destinationPoint = myWindow.getMouse()`

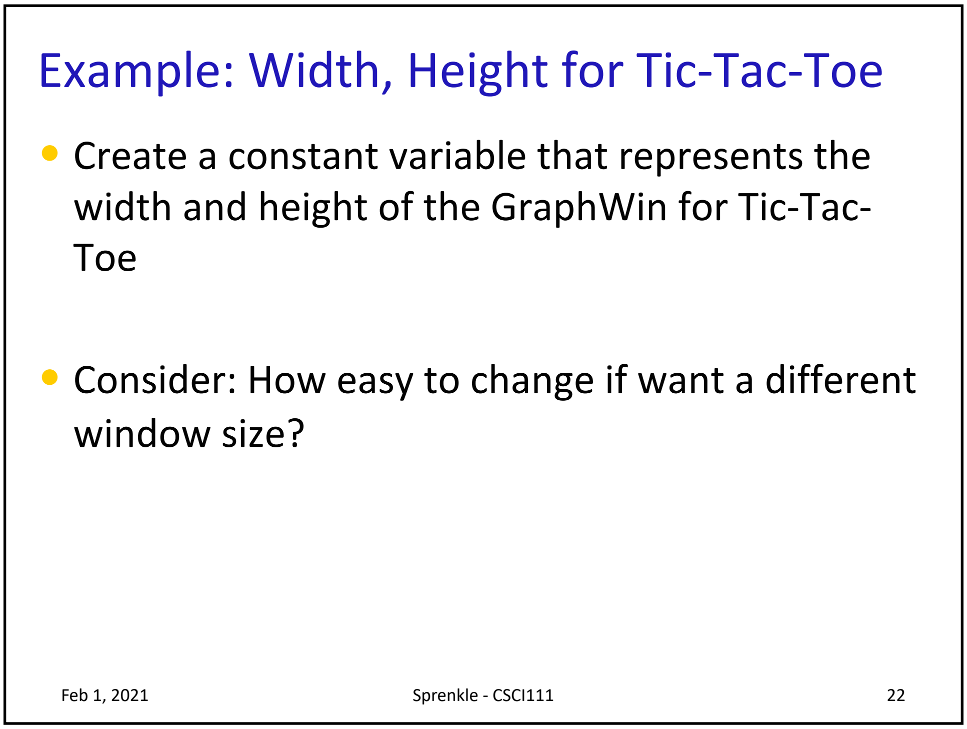
Getting Input from the User

- `<GraphWinObj>.getMouse()`
  - *Returns* the user’s mouse click as a **`Point`** object
- Entry objects
  - Get text from user



Designing for Change

- Sometimes there are “magic numbers” in our code
    - Example: 200 in tic-tac-toe
- Humans have more trouble understanding numbers than understanding words
- Give our magic numbers meaning by assigning them to variables, called ***constants***
    - Example: PI = 3.14159…
    - Name them with all capital letters (and maybe underscores) and put them at the top of programs
    - Makes them easier to find and change; software is ***soft***



Example: Width, Height for Tic-Tac-Toe

- Create a constant variable that represents the width and height of the GraphWin for Tic-Tac-Toe
- Consider: How easy to change if want a different window size?

Example: Width, Height for Tic-Tac-Toe

- Create a constant variable that represents the width and height of the GraphWin for Tic-Tac-Toe
- Consider: How easy to change if want a different window size?
  - Easy!
- Follow a similar process with other data types (strings, colors, …)

Lab Overview

- Arithmetic problems
- Graphics API Problems
  - Update web page