File I/O

Announcements

- Lab 7 is posted
  - Email part 1, the top-down design, to me by Saturday, March 18 (ideally sooner)
  - mauskop@cs.swarthmore.edu
- Quiz 3 should be handed back Wednesday
- Lab 6 should be handed back Friday

Today's plan

- Review top-down design
- Lists of lists
- File input/output
- cs21-quiz.py

Top-down design

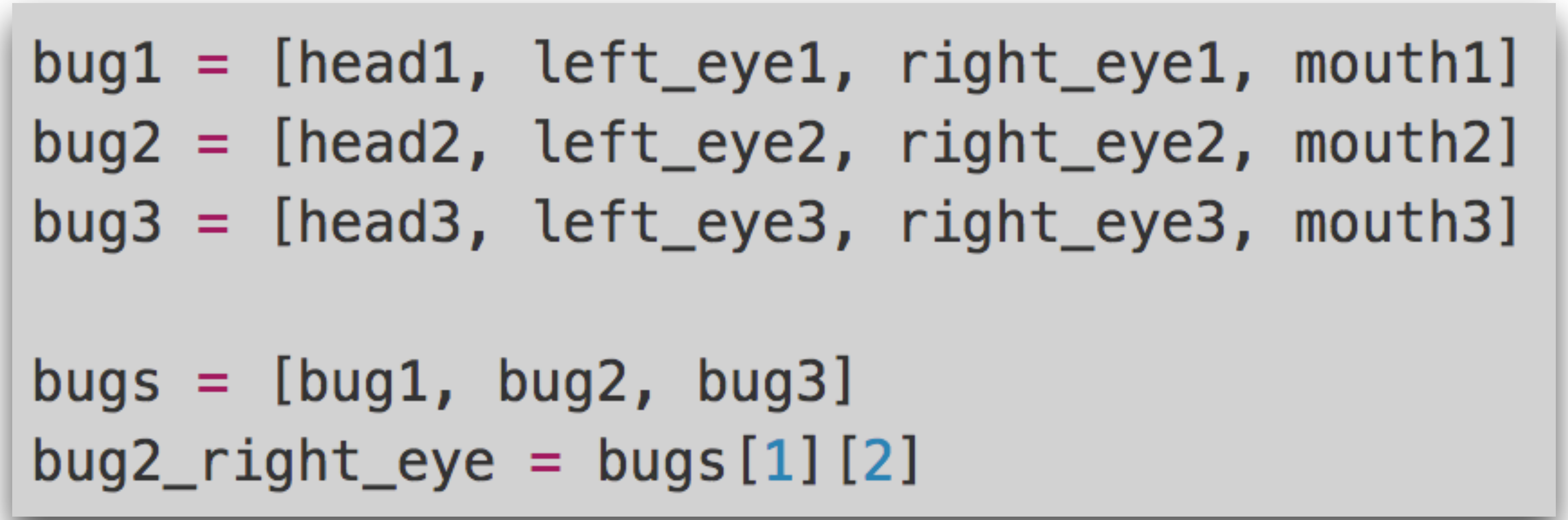
- Write `main()` first, creating a wish list of functions
- For each wished-for function, write a **function stub**
- Get the program running without errors
- Incrementally write full definition for each function

Top-down design

- `main()` manages your program, orchestrating high-level steps and managing data
- In `main()` you will likely initialize data, save data returned by functions, pass data as arguments to functions, etc.
- When working on the design, think about what data your program needs and how it should be stored for most convenience.

Lists of lists

- Sometimes a program’s data calls for a list, or parallel lists, or even a list of lists.

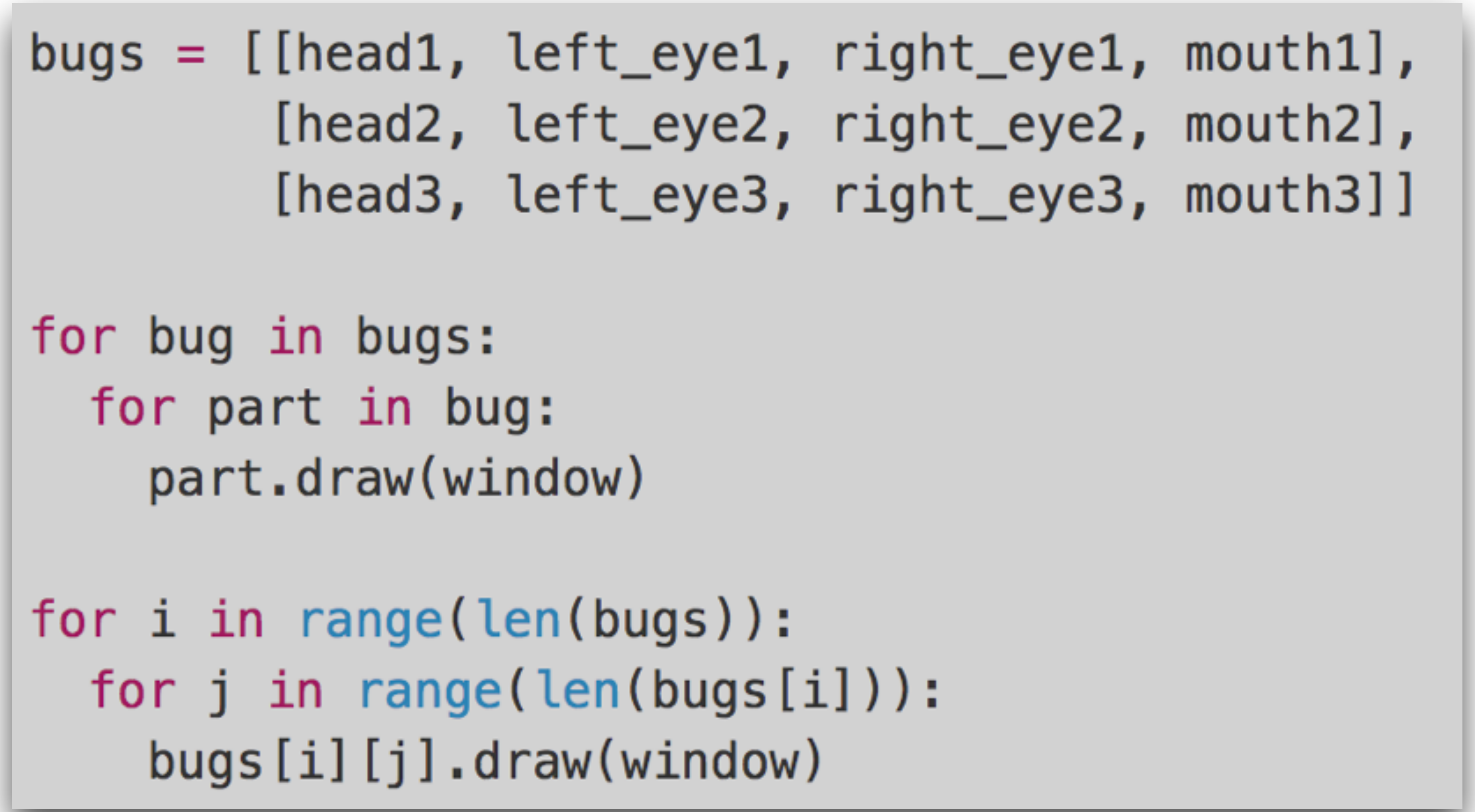


```
bug1 = [head1, left_eye1, right_eye1, mouth1]
bug2 = [head2, left_eye2, right_eye2, mouth2]
bug3 = [head3, left_eye3, right_eye3, mouth3]

bugs = [bug1, bug2, bug3]
bug2_right_eye = bugs[1][2]
```

Lists of lists

- Often we'll use nested for loops to traverse the items in a list of lists:



```python
bugs = [[head1, left_eye1, right_eye1, mouth1],
        [head2, left_eye2, right_eye2, mouth2],
        [head3, left_eye3, right_eye3, mouth3]]

for bug in bugs:
  for part in bug:
    part.draw(window)

for i in range(len(bugs)):
  for j in range(len(bugs[i])):
    bugs[i][j].draw(window)
```

File input/output

- A new way for programs to get input and produce output
    - Often without the user's knowledge
- Writing to a file is a way for a program to save information between runs
- Reading from a file is useful if our program needs information from a previous run or just needs a lot of data

File input/output

- Use `open` function
- `open` takes the name of a file and the mode of interaction: ‘r’ for ‘read’, ‘w’ for ‘write’, ‘a’ for ‘append’
- `open` returns a file object which has methods like `.readline()`, `.readlines()`, and `.write()`

File input/output

- File objects are sequences so we can put them in `for` loops
    - Each item in the sequence is a line of text represented as a string, with a `"\n"` on the end
- The string methods `.strip()` and `.split()` are useful in **parsing** data from an input file
- Reading/writing a file is ***much slower*** than accessing/assigning a Python variable

```
grades.txt

...
David Mauskop,8.5,9,9.5
Tia Newhall,10,10,7.75
Jeff Knerr,9,10,9
...
```

```python
gradeFile = open("grades.txt", 'r')
data = []
for text in gradeFile:
  text = text.strip()
  values = text.split(",")
  values[1] = float(values[1])
  values[2] = float(values[2])
  values[3] = float(values[3])
  data.append(values)
```

cs21-quiz.py