Announcements

- Homework
  - No homework due this week!
  - I'm trying to get caught up on the HW grading...
- Midterm on Friday!
  - Learning Objectives have been posted. I write the test from these.
  - Last semester's test posted.
  - Some sample questions have also been posted
  - Format and info on test on next slides
- Polling: `rembold-class.ddns.net`

Midterm Guidelines

- Test on Chapters 1–5. No new content that we talk about today or Wednesday will be on the test.
- Test will be written to be completed in an hour, but you can take another hour if needed.
  - We will of course not otherwise have a lab that day.
- Just a pen(cil) and paper test. No notes, computers, calculators, books, etc.
- Nothing will require knowledge of less common methods (say like `.count()` for strings or lists).
  - If its not covered in the Learning Objectives, I won't be asking you to know it come test day.
- Most likely will distribute it digitally and then ask you to have your camera from your laptop or phone pointing at you from the side or back and connected to a Zoom call, but still finalizing details

Format of Midterm

- $\approx$ 2 problems of the definition/true-false/matching type
- $\approx$ 2 multiple choice type problems, similar to polling questions
- $\approx$ 1 question asking you to identify what a program/function does and write a doc-string for it
- $\approx$ 1 question about deciding if a certain piece of code will meet an objective, and if not how to fix it (like hw)
- $\approx$ 1 question about reordering all the provided pieces of a program to get a desired effect.
- $\approx$ 1 question asking you to write out some code to solve a simple problem.

Review Question

Three of the below expressions are valid, one is not. Which one would return an error?

A) `{'A': {'B': (1,2)}, 'C': 3}`

B) `{1,2,(3,4),5}`

C) `[{'Alpha': 1, 'Omega': 26}, {2,3,4,5}]`

D) `{['A','B']: {1,2}}`

Formatting Detour!

- Constructing text or a sentence by interleaving strings and other objects comes up a lot in communicating code results to a user.

```
A = 10
print('The value of A is: ' + str(A))
```

Formatting Detour!

- Constructing text or a sentence by interleaving strings and other objects comes up a lot in communicating code results to a user.

```
A = 10
print('The value of A is: ' + str(A))
```

- Having to convert everything to strings and append together is clunky and easy to mess up
  - A popular alternative is to use the `.format` method or its cousin: f-strings

The .format method

- The .format method operates on strings
- Acts on a substitution system
  - Locations in the string indicated by { } get swapped out
  - Objects in the function call get converted to a string automatically and swapped in
- Simple Examples
  - `'The value of A is {}'.format(10)`
  - `'{} + {} is {}'.format(2,5,2+5)`

Labels can be key

- By default, all arguments to `.format()` are positional
  - Order of parameters is matched with order of `{}`
- Can also provide order index inside `{ }`
  - `'{1} + {0} is {2}'.format(2,5,2+5)`
- Can also provide keyword labels, but then must call with a keyword
  - `'{name} is {age} years old.'.format(age=34, name='Amy')`
  - `'{} + {B} + {C} = {}'.format(3,10,C=5,B=2)`

The Format of Format

- Often times when displaying a value in a string, we may want a certain format specification:
  - Certain number of decimals
  - Pad with a certain amount of zeros or spaces
  - Justify left or right in that space
  - Use scientific notation
  - etc
- We can do all this by specifying a sort of code inside the `{ }` *in the string itself*!
  - Format spec is separated from the label with a `:`
  - `'The value of A is {0:.5f}'.format(A)`

Shaping your format

- Lots we can do, rarely need all at once
- Commonly used spec parts:
  - `[[fill]align][width][,][.precision][type]`

Shaping your format

- Lots we can do, rarely need all at once
- Commonly used spec parts:
  - `[[fill]align][width][,][.precision][type]`

  type: What type of object you want to represent as a string
    - 'd' - base-10 integer (default)
    - 'f' - float or decimal
    - 'e' - scientific notation

Shaping your format

- Lots we can do, rarely need all at once
- Commonly used spec parts:
  - `[[fill]align][width][,][.precision][type]`

.precision: Number of numbers after decimal

- Only makes sense for floating values

Shaping your format

- Lots we can do, rarely need all at once
- Commonly used spec parts:
  - `[[fill]align][width][,][.precision][type]`

comma: If you want a comma separating thousands

Shaping your format

- Lots we can do, rarely need all at once
- Commonly used spec parts:
  - `[[fill]align][width][,][.precision][type]`

width: How many characters you want the formatted value to be

- If smaller than needed, will expand to fit number
- If larger, right justified by default

Shaping your format

- Lots we can do, rarely need all at once
- Commonly used spec parts:
  - `[[fill]align][width][,][.precision][type]`

fill-align: What alignment and fill you want

- Alignment is <,>, or ^ for left, right, center justified
- Fill can be any string character (default is space)

Not to be confused with a g-string

- Short for format string, an f-string achieves the same thing as `.format` but with less syntax
- Introduced in Python 3.6
- Need an 'f' right before the string to let Python know it needs to do more with the string
  - This is important! Otherwise it is not an f-string and Python will just print out the squiggle brackets and their contents!
- Place the desired variables (or values) directly into the `{ }`!

```
A = 10
B = 15.123234
print(f'A is {A} and B is {B:.2f}')
```

- All other syntax and format specs like `.format`!