Announcements

- Homework
  - Homework 6 posted!
  - Should already be able to do Prob 1 and 2
  - Prob 3 should be doable by Wednesday
- Grade reports actually got posted!
  - Includes everything that I currently have gotten graded.
  - Today is last day to choose credit/no credit
- Polling: `rembold-class.ddns.net`

Review Question

To the right is a text file that was written to. Which of the below sample codes could have resulted in the text file to the right?

```
Number: 34
Number: 39
Number: 44
```

A

```
f = open('File.txt', 'w')
for i in range(34,45,5):
    f.write('Number:', i, '\n')
f.close()
```

B

```
f = open('File.txt', 'w')
for i in range(34,45,5):
    f.write('Number: ' + i )
f.close()
```

C

```
f = open('File.txt', 'w')
for i in range(34,45,5):
    f.write('Number: '+str(i)+'\n')
f.close()
```

D

```
f = open('File.txt', 'w')
for i in range(34,45,5):
    f.write('Number:' + i + '\n')
f.close()
```

Important Ideas

- At some point, programs will reach a complexity where it would be nice to separate them over multiple files
  - This helps with teamwork as well, since different people can easily work on different files then
- Need some way to bring all those files together in a master program when the program is run
- Python does this through the use of `import`

Modules

- A module is a `.py` file which is imported into another program
- Can consist of both executable statements as well as function definitions
  - Usually mostly function definitions
  - Statements usually just for initialization
- Can think of as a library, from which you can checkout a particular function that you want to use
- `import`ing the module gives you access to all or some subset of those functions

Usage example

Example

Importing and using the module `circles.py`

Namespaces

- If just imported, functions and constants exist only in that modules namespace
  - Need to refer to them with module name first, ie. `circles.area()`
- This prevents conflicts with any other functions you might have also called `area()`!

I'm Lazy

- You can import in such a way that you don't have to use the module name
  - Import only the function you want
    - `from circles import area`
    - Could then call normal as just `area(3)`
  - Import everything into global namespace
    - `from circles import *`
    - Then *all* functions can be called without the module in front
- Be very careful doing this!! Especially if you are importing a bunch of modules.
  - Easy to accidentally override mission critical functions!
  - Can make debugging a nightmare
- I recommend just sticking to typing out the module name

The name is main

- Sometimes you might want to write a script that serves multiple purposes
  - Can be imported to give access to the defined functions
  - Can be run directly to give some output
- In these situations you can use

```
if __name__ == '__main__':
```

  - Code inside that `if` statement will be run only if the program is run directly, not if it is imported

What's your vector?

- So far we've looked mostly at scalar variable types
  - `int`
  - `float`
  - `bool`
  - `str` ← the only non-scalar type!
- Chapter 5 is all about non-scalar variable types
  - `tuple`
  - `list`
  - `range`
  - `dict`
  - `set`

Introducing Tuples

- Recall basic properties of a string
  - Comprised of ordered smaller elements (characters)
  - Immutable (can not be changed in place)
  - Delimited by quotes

Introducing Tuples

- Recall basic properties of a string
  - Comprised of ordered smaller elements (characters)
  - Immutable (can not be changed in place)
  - Delimited by quotes
- Pythonic tuples: generalized strings
  - Comprised of ordered smaller elements (of *any* type!)
    - Element variable types don't even need to be consistent!
  - Still immutable
  - Delimited by parentheses

Assigning Tuples

- Place any sequence of variable types between parentheses, separated by commas
  - `t_one = (1, 2, 3, 4)`
  - `t_two = ('a', 'b', 'c')`
  - `t_three = (1, 'a', 2, 'fish', True)`

Assigning Tuples

- Place any sequence of variable types between parentheses, separated by commas
  - `t_one = (1, 2, 3, 4)`
  - `t_two = ('a', 'b', 'c')`
  - `t_three = (1, 'a', 2, 'fish', True)`
- Empty tuple just empty parentheses
  - `t_empty = ()`

Assigning Tuples

- Place any sequence of variable types between parentheses, separated by commas
  - `t_one = (1, 2, 3, 4)`
  - `t_two = ('a', 'b', 'c')`
  - `t_three = (1, 'a', 2, 'fish', True)`
- Empty tuple just empty parentheses
  - `t_empty = ()`
- Tuple of 1 is the tricky one
  - We already use parentheses to group together order of operation terms
  - Even if you have only a single element, you need the trailing comma to make it a tuple
  - `t_single = ('a',)`