UNIVERSITY OF NOTRE DAME

MENDOZA COLLEGE OF BUSINESS

Software Engineering
ITAO 30210

File Structure, Modules and Packages

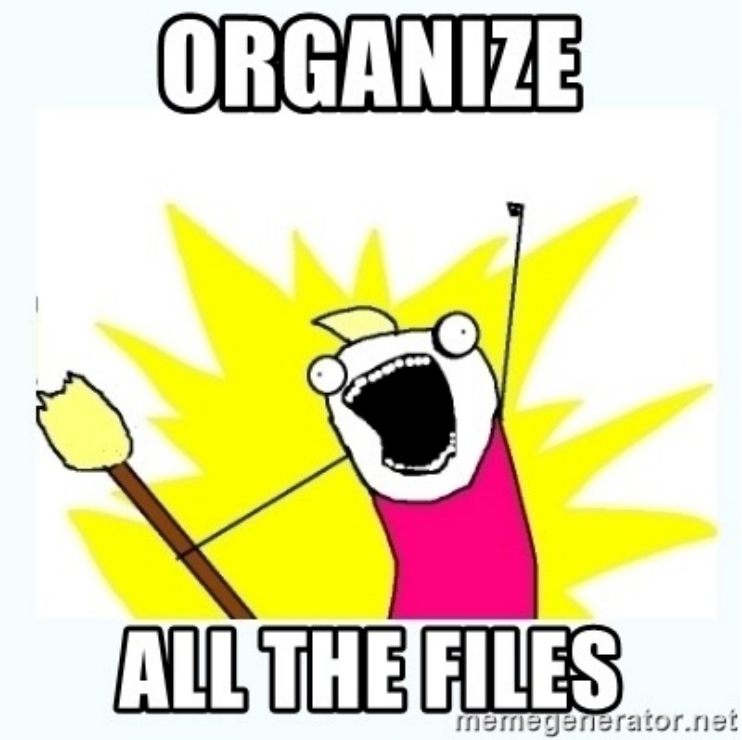
My favorit color is Green!

My favorite color is Green! My favorite color is Red! My favorite color is Red! My favorite color is Red! My favorite color is Purple! My favorite color is Red! My favorite color is White! My favorite color is Blue! My favorite color is Red! My favorit color is Red! My favorite color is Red! My favorite color is Red! My favorite color is Red! My favorite color is Red! My favorite color is Red! My favorite color is Burgandy! My favorite color is Yellow! My favorite color is Red!

What to do with all the code?

- Sometimes it takes a lot of code to accomplish tasks
- OOP leads to breaking up code into smaller self contained bits (Objects)
- Within objects, you can break code into smaller methods
- But there can be a lot of Objects
- Modules and Packages can help organize code

What to do with all the files?



Modules

- Built in Python Modules
  - Lists, Dictionaries, Strings…
  - Array, Math, OS

Modules

- Third Party Python Modules
  - Numpy, Pandas
  - Psychopg, SQLAlchemy, Art

```
In [2]: 1 import numpy as np

In [5]: 1 np.__version__
Out[5]: '1.17.2'

In [6]: 1 np.version
Out[6]: <module 'numpy.version' from '/Users
        /numpy/version.py'>
```

Modules

- Your Custom Python Modules
  - Dice
  - Game
  - Player
  - …

Modules

- **Modular programming** refers to the process of breaking a large, unwieldy programming task into separate, smaller, more manageable subtasks or **modules**. Individual modules can then be cobbled together like building blocks to create a larger application.

Modules

- What are Modules?
  - Python code in a file with a .py extention
    - Contains:
      - Classes
      - Functions
      - Variables
      - Scripts

Modules

- How do you use Modules?
  - Create a file: my_module.py
  - Import file: `import my_module`

What about Packages?

- Packages are a collection of Modules
- One your computer: directories of python files
- Another way to organize and separate code
- `__init__py` file in package directory:
  - Used to be required, is not anymore
  - Can be used to execute code when package is imported
  - Can register Package level variables
  - Beyond the scope of this class

What about Packages?

- Dog
- Coffee Maker
- Cat
- Daughter
- Wife
- Oven
- Son
- Gerbil

What about Packages?

- Pets
  - Dog
  - Cat
  - Gerbil
- Family
  - Wife
  - Daughter
  - Son
- Appliances
  - Coffee Maker
  - Oven

Importing Modules

- There are several ways to import modules and packages
  - `import {module|package}`
    - `import farkle`
    - `import pytest`
    - `import numpy`
  - `from {module_path} import {Object|Function|Variable}`
    - `from app.models.game import Game`
  - `from {module_path} import *`
    - `from app.models.dice import *`

What about Packages?

- Dog
- Coffee Maker
- Cat
- Daughter
- Wife
- Oven
- Son
- Gerbil

What about Packages?

```
import dog
import coffee_maker
import cat
import daughter
import wife
…
```

What about Packages?

- Pets
  - Dog
  - Cat
  - Gerbil
- Family
  - Wife
  - Daughter
  - Son
- Appliances
  - Coffee Maker
  - Oven

What about Packages?

```
from pets.dog import *
from pets.cat import Cat
from family.wife import *
from family.daughter import Daughter
from appliances.oven import Oven
```

What about Packages?

```
import pets
import family
import appliances
```

File Names - Recommendations

- Classes:
  - One file per Class
  - Lower case, singular
  - Separate words with underscores (along with variables)
  - Group classes into directories (packages) if appropriate
  - Class definitions should be capitalized and use CamelCase, file names should be lower case and use underscores

File Structure

File Structure

```
(base) jarp@jarp-mbp16 itao % mkdir file_structure
(base) jarp@jarp-mbp16 itao % cd file_structure
(base) jarp@jarp-mbp16 file_structure % touch main.py
(base) jarp@jarp-mbp16 file_structure % mkdir app
(base) jarp@jarp-mbp16 file_structure % mkdir app/models
(base) jarp@jarp-mbp16 file_structure % mkdir app/runners
(base) jarp@jarp-mbp16 file_structure % mkdir app/datastore
(base) jarp@jarp-mbp16 file_structure % mkdir app/io
(base) jarp@jarp-mbp16 file_structure % mkdir tests
(base) jarp@jarp-mbp16 file_structure % mkdir tests/models
(base) jarp@jarp-mbp16 file_structure % mkdir tests/runners
(base) jarp@jarp-mbp16 file_structure % mkdir db
(base) jarp@jarp-mbp16 file_structure % touch app/models/game.py
(base) jarp@jarp-mbp16 file_structure % touch app/models/dice.py
(base) jarp@jarp-mbp16 file_structure % touch app/models/player.py
(base) jarp@jarp-mbp16 file_structure % touch app/models/roll.py
(base) jarp@jarp-mbp16 file_structure % touch app/models/turn.py
(base) jarp@jarp-mbp16 file_structure % touch app/runners/game_runner.py
(base) jarp@jarp-mbp16 file_structure % touch tests/models/game_test.py
(base) jarp@jarp-mbp16 file_structure % touch tests/models/player_test.py
(base) jarp@jarp-mbp16 file_structure % touch tests/runners/game_runner_test.py
(base) jarp@jarp-mbp16 file_structure % open .
(base) jarp@jarp-mbp16 file_structure % touch app/io/game_dialog.py
(base) jarp@jarp-mbp16 file_structure % touch test/io/game_dialog_test.py
touch: test/io/game_dialog_test.py: No such file or directory
(base) jarp@jarp-mbp16 file_structure % mkdir tests/io
(base) jarp@jarp-mbp16 file_structure % touch test/io/game_dialog_test.py
touch: test/io/game_dialog_test.py: No such file or directory
(base) jarp@jarp-mbp16 file_structure % touch tests/io/game_dialog_test.py
(base) jarp@jarp-mbp16 file_structure % touch db/farble.sqlite
```