AMS 595/691: Fundamentals of Computing: Part II

Lecture 5: Scripts and Modules; Debugging and Testing

Xiangmin Jiao

Stony Brook University

Python Scripts

- Python scripts have suffix ".py"
- Scripts can use any control flow, contain function definitions, and its own docstring
- You can create and edit Python scripts using
  - any text editor with syntax highlighting (e.g., vi or emacs on UNIX),
  - IDE environment (such as Spyder or PyCharm), or
  - an online editor (e.g., Jupyter notebook)
- You can run Python script in different ways
  - In IPython: `%run test.py`
  - In UNIX shell or Windows command prompt: `python test.py`
  - To run script as standalone program:
    - ★ set the file's executable bit using "`chmod a+x test.py`"
    - ★ make the first line

```
#!/usr/bin/env python
```

- Practice
  - Copy and paste this code into editor, save it, and run it in Linux shell
  - For quick start with editors, see these cheat sheets: vi and emacs

Interaction with UNIX Shell

- Standalone scripts may take command-line arguments via `sys.argv`
  - `sys.argv` is a list of strings
  - `sys.argv[0]` is command name
- To prompt user to type at runtime, use

```
input('message')
```

- Use `sys.exit(code)` to return error code back to unix system
- Practice
  - Copy and paste the code into an editor and save it as test_argv.py
  - Set its executable bit, run it in Linux shell using command

```
./test_argv.py 123 abc def "abc def"
```

  - Run UNIX command "echo $?" to see error code
- For quick start with Linux, see this cheat sheet

Python Modules

- Objects can be organized into modules
  - Objects can be variables, functions, and classes (later)
  - Modules improves *modularity* for better organization of large programs
  - Modules improves *reusability* of codes
- Python modules are similarly to scripts, but can be imported by other codes
- A module has its own namespace, and needs to be imported
- Note: Modules are cached
  - If you modify a module and re-import it in the old session, you will get the old one
  - To reload module in Python3, run
    - `import importlib`
    - `importlib.reload(demo)`
- Practice:
  - Copy and paste this code into an editor and save it as `demo.py`
  - In IPython, try commands `import demo; demo?; dir(demo); demo.print_a(); demo.print_b()`

Scripts or Modules?

- Rule of thumb
  - Sets of instructions that are called several times should be written inside functions for better code reusability
  - Functions (or other bits of code) that are called from several scripts should be written inside a module, so that only the module is imported in the different scripts (do not copy-and-paste your functions in the different scripts!)
- A module can also be a script itself
  - Put statements other than variable/function/class definitions within block enclosed with

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

  - It is particularly useful to embed test scripts for the module in the file
- Practice:
  - Copy and paste this code into an editor and save it as demo2.py
  - Try `import demo` and `%run demo2` in IPython

Interactive Debugger

- Command-line debugger: pdb, similar to gdb for C/C++
  - Start up in command-line:

    ```
    python -m pdb test.py <args>
    ```

  - Useful commands:
    - `run`, `next`, `step`, `break`
    - `where`, `up`, `down`
    - `p <expression>`, `display <expression>`
  - Another common use is to insert the statements

    ```
    import pdb; pdb.set_trace()
    ```

    to Python code where you want to break into pdb
- Use Spyder to debug, which has integrated GUI support for `pdb`
  - Use menu items to specify command-line arguments
  - Double-click on line number to set breakpoints
  - Use icons to use run/debug, next, step, etc.
  - Use variable explorer to inspect or change variables

Search Path and Directories of Modules

- When import statement is executed, modules are searched in:
  - installation-dependent default path (e.g., /usr/lib/python)
  - list of directories specified by environment variable `PYTHONPATH`
- List of directories searched by Python is given by `sys.path` variable
- To modify search path:
  - In UNIX, add following to `~/.bashrc` (assuming you use bash)

    ```
    export PYTHONPATH=$PYTHONPATH:/user/defined/path
    ```

  - or append path to `sys.path` variable within Python script

    ```
    import sys
    new_path = '/home/username/user_defined_modules'
    if new_path not in sys.path:
        sys.path.append(new_path)
    ```

- Modules can also be organized into a *package*
  - Package is a directory containing `__init__.py` file
  - Package can have subpackages (subdirectories with `__init__.py` file)
  - For example, `numpy`, `scipy`, `scipy.ndimage` are all packages