Python Development

Now What?

- More Python
- Making your programs for others
- Releasing your code
- Licensing
- Collaborating
- Testing / Building

Structuring your Python projects

- Modules* – individual python files
- Packages* – like folders of files, often structured that way too
- Projects* – Multiple packages, documentation, examples, unit tests, and distribution code all put together

* can be much more complicated than this

Modules

- Typical structure:
  - Import packages & modules
  - Define functions
  - Code which always gets run, on import or main
  - Test for main
    - Run local script

Packages

- Typically a folder with code grouped together
- /folder – this name is the name of the package
  - __init__.py – required to be a package
  - module1.py - submodules
  - module2.py
- If you want to load a package it should be in your python path.
- __init__.py gets run when you import a package. Can use this to pre-load submodules or define constants, etc.

Python Path

- The list of directories python searches for your code in
  - Looks first in the folder you are running the script from
  - Secondarily, looks in all the places you have added in your $PYTHONPATH environment variable
  - Third, looks in Python system directories (many of them, different by os)
- This list can be viewed and edited in the sys module under the path variable

Projects

- /Project
  - /documentation
  - /examples
  - /python
    - /package1
      - /subpackage1
      - /subpackage2
    - /package2
  - /licenses – where you put the licenses of the code you are reusing with permission
  - setup.py
  - LICENSE
  - README

Setup script

- This is the basic way to release and install python code.
- Installing moves project code from local folder to python system directory.
- Download source, open command terminal, type
  - “python setup.py install”
  - Can also build, build_ext, etc.
- Being phased out in favor of pip
- Setup script still important

Example: foldable_robotics

Speeding up Python

- Python is interpreted, not compiled.
- How do you work with compiled code?
- Simplest way: load_dll
  - Requires that dll to be in your system path, machine dependent.
- Try cython.
  - Convert your python code into c
  - Compile your c code into a dll
    - doesn't have to be from python originally
    - Now supports c++! (with some glue code)
    - Requires a compiler to be installed
  - Import the dll just like a normal python module
  - However, machine dependent.

Example: pypoly2tri

Making Windows and applications

- PyQt4 / PyQt5 / PySide – Qt
  - Many others, but this is very cross-platform, used a lot, etc
- Already have PyQt4 installed in your anaconda distribution
- Way to create windows, menus, etc
- Turn your script into a user application

Example: putting matplotlib in a window

- Matplotlib can work with

Collaborative projects

- Git – versioning tool
  - Command-line tool for tracking and committing changes
  - many guis available
  - I suggest “git-extensions” for windows
- Github
  - website built around git: https://github.com/
  - Other collaboration tools
    - Documentation, bugtracking, code releases
  - Integration with other services

Concept of GIT

- Work simultaneously, merge your code when ready
- There is no central administration, repositories can be local or on the web
- Create branches to manage

Example

Licensing – really important

- Strongly encourage you to learn about licensing. The future of your work depends on it.
- Copyright and who owns it.
- “What is Open source?”
  - Ability to see/use/change/release others’ code?
  - Ability to use others’ code with your code
- Copyright vs copyleft
  - GPL, LGPL, MIT, BSD, Apache

PyPi – getting pip working

- https://packaging.python.org/distributing/#uploading-your-project-to-pypi

Building a setup script

- Freezing your code
  - Py2exe,pyinstaller, cx_freeze,

Future

- This class
  - One of a kind
  - Still emerging
  - Chance to define it
- Research
  - Open field
  - Can move faster
  - New manufacturing processes need new design tools

Wrapup

- You've built many parts you find in CAD
  - Motion analysis – nonlinear constraint solving – how sketches are dimensioned, how assemblies are moved around.
  - Python code for
  - Constructive Solid Geometry
    - Union, difference,intersection
  - Manufacturing
    - Accessing the functions for understanding the manufacturing process
  - Dynamics? Some
  - Stiffness? Some
  - Kinematics? Some

Wrapup

- You've built devices from practically *nothing*.