

PyCharm

Debugger, profiler, and docker environment

What is PyCharm?

- IDE - integrated development environment
 - Meant for developing programs and/or building software in python
 - Great for projects!
 - GUI based editor
 - Unlike other editors (Atom, Emacs, VIM, Sublime, etc.)

PyCharm Perks

- Code completion/intentions
- Debugger
- Git visualization in editor
- Code coverage
- Project management with interpreter
- Refactoring
- Keyboard-centric

Getting started

- Download PyCharm Professional (available to students)
- Clone or pull the latest from this repo: ``git clone https://github.com/ContextLab/CDL-tutorials.git``
- Also ahead of time, build your CDL docker image

Configure Interpreter

- Choose an interpreter
 - If multiple versions of Python - manage this by configuring your python interpreter
 - Can manage versions of packages by creating a virtual (conda) environment
 - BUT! you can also manage this using DOCKER environment! We'll get to this...

PEP8 and Running

- Toggle your cursor over yellow lines - PEP8 suggestions
 - Fix the errors
- Run the ``example_debug.py`` script
 - Edit Run Configuration if necessary
- Show output from the run

Debugging

- Debugging allows you to step through the code
 - Add breakpoints to the script and click `Debug`
 - You can also pull up a console in the context of your environment
 - Walk through `add_it` function in `helpers.py``
 - Place cursor further down and skip to cursor
 - Walk through `hypertools.py``
 - Show plot output

Profiler

- To optimize your code you can profile it
 - Open `example_profile.py` in a separate tab
 - Right click and select `Profile`

Docker as Remote Interpreter

- So far we've been using Python 2.7... But what if we want to run this on Python 3.x?
- PyCharm integration with Docker
- Run applications in the variously configured development environments deployed in Docker containers


Docker Plugins

- Docker Integration and Python Docker plugins should be enabled
- The plugins are bundled with PyCharm and activated by default.
- If the plugins are not activated, enable them on the Plugins settings page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins.


Docker set up

- Follow the instructions in for the Docker tutorial. We will be using the CDL docker as our environment
(note: this feature is only supported in the professional edition of PyCharm, so be sure to download that version).
- Build the docker image. Make sure you have docker running, navigate to your local copy of the Docker repo and execute the line: ``docker build -t cdl .``. This will create a docker image from the instructions specified in Dockerfile.
- To launch the docker image (for the first time) run the following line: ``docker run -it -p 9999:9999 --name CDL -v <absolute-path-to-your-docker-repo>:/cdl/ cdl``

More Docker

- Open the `Settings / Preferences` dialog (,), click `Build, Execution, Deployment` and then `Docker`.
- Add a Docker configuration (The Add `+` button) and specify how to connect to the Docker daemon.
- In the Docker tool window `(View | Tool Windows | Docker)`, select the Docker icon, and then either click the Connect button or select `Connect` from the context menu.

Define Docker-based remote interpreter

- Open the Settings dialog (press , or click settings on the main toolbar).
- Click the Project Interpreter page, on this page click the gear icon next to the Project Interpreter field, and choose Add from the drop-down list.
- In the dialog box that opens, select the Docker option, from the drop-down lists select the Docker server (if the server is missing, click New...) and image name.
- For more information, check this out: <https://www.jetbrains.com/help/pycharm/using-docker-as-a-remote-interpreter.html>
- Now you can go ahead and run your script in a docker container!

