Conda Virtual Environments

Please note, this cheat-sheet was made using Conda's official documentation. All attributing goes to Conda.
For more detailed information please review th3e official documentation as: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

Create Virtual Environment

1. Create the environment `conda create --name myenv`
2. Proceed, type `y`
   - Creates the environment in `/envs/`
3. To create an environment with a specific Python version `conda create -n myenv python=3.9`
4. Create an environment with a specific package `conda create -n myenv scipy`
5. Create an environment with a specific version of Python and multiple packages: `conda create -n myenv python=3.9 scipy=0.17.3 astroid babel`

Create an environment from an .yml file

1. Create environment from `environment.yml file` using `conda env create -f environment.yml`. Where environment is the name of the file and the name of your environment. These can be distinct as well
2. Activate the new environment `conda activate environment
3. Verify that the new environment was installed correctly `conda env list`

List environments

- `conda info --envs`

Cloning an environment

Create an exact copy of the environment by cloning it:
`conda create ---name myclone --clone myenv`

To verify the copy was made:
`conda info --envs`

Activate an environment

`conda activate myenv`

Deactivate an environment

`conda deactivate`

List environments

`conda env list`

List packages in environment

- Environment is not activated
  - `conda list -n myenv`
- Environment is activated
  - `conda list`
- Determine if package is installed in environment
  - `conda list -n myenv scipy`

Using pip in an environment

If conda does not have access to a specific package, or if you are having difficulty installing a specific library, use pip to install it within the conda environment.

```
conda install -n myenv pip
conda activate myenv
pip <pip_subcommand>
```

Create .yml file through export

If you want to make your environment file work accross all platforms, use `conda env export --from-history > file.yml` flag. This will only include packages you've explicity asked for, as opposed to including every package in your environment.

If you use `conda env export` it will export all of 5those packages.

Creating a .yml file manually

```
name: stats
dependencies:
  - numpy
  - pandas
```

EXAMPLE: A more complex environment file:

```
name: stats2
channels:
  - javascript
dependencies:
  - python=3.9
  - bokeh=2.4.2
  - conda-forge::numpy=1.21.*
  - nodejs=16.13.*
  - flask
  - pip
  - pip:
    - Flask-Testing
```

Removing an environment

`conda remove --name myenv --all`

To verify the environment was removed run:
`conda info --envs`

Set env as kernel for use in Jupyter Notebook

1. Create a conda environment `conda create --name firstEnv`
2. activate the environment Conda Virtual Environments > Activate an environment
3. Install relevant packages
4. Add conda environment as a kernel for Jupyter notebooks

```
conda install -c anaconda ipykernel
python -m ipykernel install --user --name=firstEnv
```

5. Select firstEnv as your kernel for the notebook