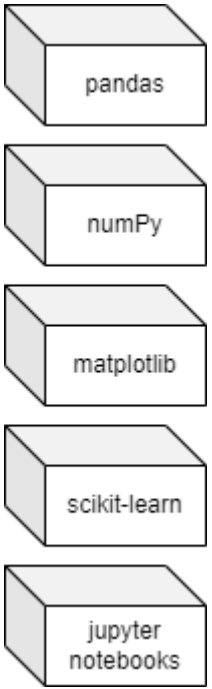
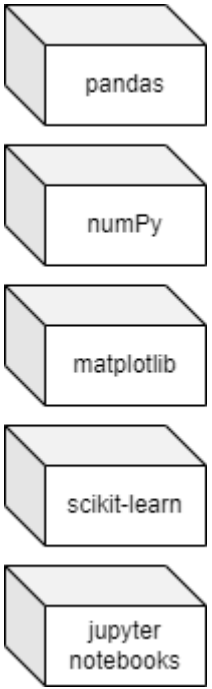
CS4650 Topic 5: Packages and Managers

Packages



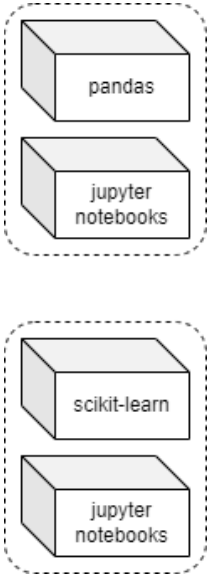
- A *package* is a piece of code written by someone to perform a specific task.
- This code has been shared with the community.
- There are a bunch (150+) of these.
- Some prepare data, some process data, some perform calculations, and some display results.
- To solve some data science problems, either one of these packages, or a string of several are used together.

Packages



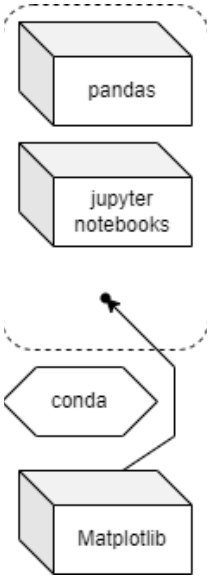
- pandas -- for exploring and manipulating data.
- NumPy -- for performing numerical operations on data.
- Matplotlib -- for creating visualizations of data.
- scikit-learn -- for building and analysing machine learning models.
- Jupyter Notebooks -- for writing Python code, running experiments and communicating the work to others.

Environments



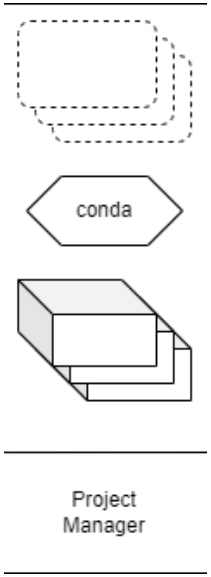
- An *environment* is the collection of packages used to solve a problem.
- A person typically has multiple environments, each customized to solve one type of problem.
- Each environment can have a different collection of packages, and packages can be added or removed easily.
- A particular package might be in several environments.
- An environment can be shared, so if one person develops a useful environment, then other people may be given a copy.

Package Manager



- A *package manager* is a tool that is used to insert, update, or remove packages from an environment.
- Each package has a *version number*. If the package is improved, a new version of the package becomes available.
- The package manager reviews the packages that are in an environment, and if it sees that there are new versions, can inform the user.
- *conda* is one package manager, *pip* is another.

Project Manager



- A *project manager* is a complete toolset: It has
  - A collection of environments
  - A collection of pre-installed packages
  - *conda*, to fetch other packages, but also to link pre-installed packages to the environments.
- *Anaconda* is one project manager, with a large set of pre-installed packages.
- *Miniconda* is a package manager 'at the other end', with a minimal set of pre-installed packages.
- Either of these managers will work for this class, as any needed packages can be easily installed.

Data Science 'Flow'

- A lot of data science is experimental. The correct approach (if any) for solving a problem is usually not known at the start.
- You might typically find a similar problem, reasoning that a similar environment might work to solve this problem.
- As you work with this environment, you might discover that it's not quite right, that some changes are needed. No problem, add another package, or delete an unnecessary one.
- There might be a lot more experimentation when trying to solve a novel problem.

Data Science 'Flow'

- Once you have solved the problem, you would probably like to archive your results.
- This way you can reproduce your results at a future time.
- You can also use this solution to solve a related problem.
- You could also share your work, so others on your team can benefit from your work.
- The packages, environments, and managers all support this flow.

Which to Choose?

- Which project manager should you use?
- Use Anaconda if you want a robust set of tools to start with, and you have at least 3 GB of space on your computer.
- Use Miniconda if you only want to download what you need, and you don't have 3 GB of space.
- Most "professionals" use Miniconda, only taking what they need.
- I used Miniconda, then downloaded the 5 packages listed above, and the results were 2.1 GB, so the size wasn't that much of a difference.
- If you use Miniconda, it's really easy to install anything you need.
- I will use Miniconda for this lecture, but feel free to use Anaconda.

Installing and Setting Up Anaconda

1. Go to the Anaconda distribution page,
   https://www.anaconda.com/distribution/
2. Pick the download for your computer (you might as well choose the latest version, which is the highest version number).
3. Once downloaded, double click on the download file to go through the setup, keeping the default selections.
4. Open the Terminal (Mac) or Command Line (Windows). If the installation was successful, the terminal prompt will have "(base)" prefixed:
   (base) dave@Daves-MBP ~ %

Installing and Setting Up Anaconda

5. The '(base)' indicates that the selected environment is the default environment, 'base'.
6. You can see all of the packages that are installed by typing 'conda list⏎'.
7. You can see which version of Python is installed by typing 'python⏎'. This will list the current version of Python (and Anaconda). However, you will then be in the Python interpreter (the command prompt will be '>>>'). To exit Python, type 'exit()⏎'
8. You can see all of the environments on your machine by typing 'conda env list⏎'

Installing and Setting Up Miniconda

- With Anaconda, you get all of the most common tools pre-installed, many of which you will never use.
- For longer-term projects, you will probably want to use Miniconda, so that the environments will be a lot smaller (only containing the tools you need for that environment).
- With Miniconda, we will also create *directories* or *folders*. This is in fact a directory in the computer's file system.
- The usual practice is to keep all of the data, code, and tools that you use for a project in this same folder.
- Again, typical practice is to make a subdirectory, *env*, that contains the environment.

Installing and Setting Up Miniconda

1. Download Miniconda from the Conda documentation website:
   https://docs.conda.io/en/latest/miniconda.html
2. Go through the setup steps. This involves running the Terminal, then typing the Bash command to install the application. (The download site shows the Bash command to perform the install.)
3. You can check where the installation occurred by entering 'which conda'
4. Create a project folder on the desktop, for instance 'project_1' by typing 'mkdir desktop/project_1'. 'mkdir' is the Unix/Linux command to make a directory.
5. Change to that directory by entering 'cd desktop/project_1'.

Installing and Setting Up Miniconda

6. Once 'in' that directory, create a new environment, and load an initial set of packages. For example:

   conda create --prefix ./env pandas numpy matplotlib scikit-learn

   The '--prefix' essentially says where to create this environment, the ./ means in the current directory, and the 'env' gives the name of the environment we are creating.
7. You can see the list of packages in the folder by typing the command 'ls', which is short for 'list'.

Installing and Setting Up Miniconda

8. Once the environment is set up, we need to tell the system which environment to use (recall that the default environment 'base' is currently selected). To select the new environment, type
   conda activate Users/dave/desktop/project_1
   This is the path on a Mac, if your user name happens to be 'dave'.
9. Once this is activated, rather than each command prompt being prefixed by (base), they will now be prefixed by (Users/dave/desktop/project_1).
10. Now we realise that we should have also installed the Jupyter notebook package. To install a package, simply use 'conda install jupyter'.

Useful Conda Commands

- Recall that *conda* is the package manager, one of the tools that is part of both the Anaconda and Miniconda project managers.
- Here are some helpful conda commands:
- List the environments:

  conda env list
- List the packages in current environment:

  conda list
- Create an environment called [ENV_NAME]:

  conta create --name [ENV_NAME]

Useful Conda Commands

- Delete an environment named [ENV_NAME]:
  conda env remove --name [ENV_NAME]
- Activate an environment named [ENV_NAME]:
  conda activate [ENV_NAME]
- Deactivate the current environment:
  conda deactivate
- Install a new package named [PACKAGE_NAME] in the current environment:
  conda install [PACKAGE_NAME]