Anaconda and conda notes

In Windows run *Anaconda Prompt*. Run it as administrator!

In Linux run `conda` from terminal.

Current environment is displayed in parenthesis at the prompt, like

```
(base) D:\current\path>     # Win
(base) ~/current/path$      # Lin
```

Common options

```
-c, --channel            # ~= package repository address, see below;
-n, --name               # of the environment
-p, --prefix             # = PATH  "  (so it should be --path)
```

It's `prefix` as is in Python's `sys` module

```
import sys
sys.prefix
```

Remarks

By the way, notice also

```
import site
site.getsitepackages()
```

Remember that `conda` is alternative for `pip`, `venv`, etc. and it's strongly discouraged to mix them, see below.

Nevertheess it's good to know something about Python's virtual environments (which are NOT Conda's virtual environments!) and suchlikes.

Managing conda

Version and info

```
conda --version
conda -V
conda info
```

Default system Python

```
where python    # Win
which python    # Lin
```

Remove unused cached files including unused packages (in current environment)

```
conda clean --all
```

Keeping anaconda up to date

In current environment or in a given environment via its name or path:

```
conda update --all
conda update -n myenv --all
conda update -p /path/to/myenv --all
```

E.g. update of conda

```
conda update conda
conda update -n myenvir conda
```

Be careful with

`conda update anaconda`

It updates only the Anaconda **metapackage** what may actually result in *downgrading* some packages! See:

`conda install --file anaconda_latest_pkgs_no_versions.txt`

file `anaconda_latest_pkgs_no_versions.txt` may be created as first column of

`conda list > anaconda_pckgs.txt`

from `base` environment, or take it from source above.

Packages

List of packages in an environment

```
conda list                 # in active environment
conda list -n nywnv        # in other nvironment specified via name

conda list --revisions     # all revisions made within the environment
conda list -n myenv --revisions     #  "
```

To see if a specific package is installed in an environment

`conda list -n myenv scipy`

Search for packages

To see if a specific package, such as SciPy, is available for installation (from conda repositories)

```
conda search scipy
conda search scipy --info
```

To list only the packages whose full name is exactly `python`, add the `--full-name` option

`conda search --full-name python`

Packages are downloaded from places in the net called **channels**. They are configured in the configuration file `.condarc` Sometimes these settings need to be overwritten with `--override-channels`.

To see if a specific packag is available for installation from Anaconda.org:

`conda search --override-channels --channel defaults scipy`

To see if a specific package exists in a specific channel and is available for installation:

`conda search --override-channels --channel http://conda.anaconda.org/mutirri iminuit`

Install and update of packages

`conda update` is used to update to the latest compatible version.
`conda update` always installs the highest version with *the same major version* number
`conda install` can be used to install any version.
`conda install` always installs the highest version.

In a current environment

```
conda update pck1 ... pcgkN
conda install pck1 ... pckgN
```

In a given environment via its name or path

```
conda update -n envir pckg1 ... pckgN
conda update -p /path/to/envir pckg1 ... pckgN
conda install -n envir pckg1 ... pckgN
conda install -p /path/to/envir pckg1 ... pckgN
```

Supressing user prompt (useful for scripts)

```
conda install --yes pckg1 ... pckgN
```

Installing conda packages with a specific build number If you want to install conda packages with the correct package specification (version and build), try `pkg_name=version=build_string`

```
conda install beautifulsoup4=4.8.0=py37_0
```

or from a specific channel

```
conda install  -c numba/label/dev llvmlite=0.31.0dev0=py37_8
```

Read more about build strings and package naming conventions.
Learn more about package specifications and metadata.

Packages specification

```
conda search PKGNAME=3.1 "PKGNAME[version='>=3.1.0,<3.2']"
conda install "PKGNAME[version='3.1.2|3.1.4']"
conda install "PKGNAME>2.5,<3.2"

conda install numpy=1.11         # with the newest minor version
conda install numpy==1.11        # install exact version
conda install "numpy>1.11"
conda install "numpy=1.11.1|1.11.3"
conda install "numpy>=1.8,<2"
```

more exmples with description of versioning on the page.

Install package from a specific channel, e.g.

```
conda install conda-forge::pckg
```

Installing packages from Anaconda.org Try hard to install packages first via `conda` then from Anaconda.org then maybe `conda-forge`... Use `pip` only after checking all the mentioned possibilities as `pip` packages do not have all the features of conda packages

Read it!!!

Find a package on all channels using the Anaconda Client

```
anaconda search pckg
```

Default packages

```
conda config --add create_default_packages pckg1 ... pckgN
```

See also here.

Removing packages

```
conda remove pckg1 ... pckgN
conda uninstall pckg1 ... pckgN
conda remove -n myenv pckg1 ... pckgN
conda remove -p /path/to/myenv pckg1 ... pckgN
```

Remove unused cached files including unused packages

```
conda clean --all
```

Preventing packages from updating (pinning)

Listing package dependencies

Installing similar packages

Installing packages that have similar filenames and serve similar purposes may return unexpected results. The package last installed will likely determine the outcome, which may be undesirable. If the two packages have different names, or if you're building variants of packages and need to line up other software in the stack, we recommend using Mutex metapackages.

Environments

A conda environment is a directory that contains a specific collection of conda packages that you have installed. ... More information.

Determining your current environment

List of all environments

```
conda info -envs
conda env list
```

The asterisk `*` is displayed at the side of the current environment.

By default, the command prompt is set to show the name of the active environment. To disable this option:

```
conda config --set changeps1 false
```

To re-enable this option:

```
conda config --set changeps1 true
```

Creating

```
conda create --name newenv pckg1 ... pckgN
conda create --name newenv python=3.7
conda create -n newenv python=3.7
```

The above will create environments in a anaconda directory, sth. like `/path/to/anaconda/envs/`.
One may also create in a specified path, e.g.

```
conda create --prefix /path/to/project/envs
conda create --prefix ./envs
```

Activating, deactivating (from anaconda directory)

```
conda activate myenv       # conda 4.6 +
conda deactivate myenv     #  "

activate myenv             # Windows  conda 4.6 -
source activate myenv      # Lin/Mac  conda 4.6 -

conda activate             # default is  base  environment
activate                   #  "
source activate            #  "
```

Removing entire environment

```
conda remove --name envir --all

conda env remove --name envir
```

Creating envir with special config (set of packages)

Install all the programs that you want in this environment at the same time. Installing 1 program at a time can lead to dependency conflicts.

```
conda create -n myspecialenv -c bioconda -c conda-forge python=3.5 pandas \
beautifulsoup seaborn nltk

conda create -n newenv --clone root       # !!!
conda update -n newenv --all

conda create -n myenv python=3.6
conda create -n myenv scipy
```

The latter in two steps

```
conda create -n myenv python
conda install -n myenv scipy
```

Other examples

```
conda create -n myenv scipy=0.15.0
conda create -n myenv python=3.6 scipy=0.15.0 astroid babel
```

Specifying a location for an environment

In a current directory (i.e. working directiory of your project)

```
conda create --prefix ./env jupyterlab=0.35 matplotlib=3.1 numpy=1.16
```

The full path of such environment (if active) will be displayed in the conda prompt. To have displayed only the root directory of the environment run

```
conda config --set env_prompt '({name})'
```

what creates or modifies `.condarc` file; then

```
cd /path/to/project/
conda activate ./env
```

The prompt shall look like `(env) /path/to/project $`.

Cloning an environment

You can make an exact copy of an environment by creating a clone of it:

```
conda create --name myclonedenv --clone myenv

conda info --envs   # to verify that the copy was made
```

Building identical environments on the same system

```
conda list --explicit
conda list --explicit > spec-file.txt

conda create --name myenv --file spec-file.txt
```

```
conda install --name myenv --file spec-file.txt
```

Conda does not check architecture or dependencies when installing from a `spec file`. To ensure that the packages work correctly, make sure that the file was created from a working environment, and use it on the same architecture, operating system, and platform, such as linux-64 or osx-64.

Activating an environment

Nested activation

```
conda activate --stack myenv
```

It is sensible then to deactivate nested environments (when no longer needed)

```
conda deactivate
```

Restoring environment

The history of changes of the environment

```
conda list --revisions
```

To restore environment to a previous revision:

```
conda install --revision=REVNUM
conda install --rev REVNUM.
```

Automation of creation

Via `environment.yml`

```
conda env create -f envir.yml
```

You may write `envir.yml` by hand or create it from existing environment.

```
conda env create
```

will create environment from `environment.yml` in the current directory (if exists?).

Via `.condarc` configuration file. To automatically install `pip` or another program every time a new environment is created, add the default programs to the `create_default_packages` section of your `.condarc` configuration file.

The default packages are installed every time you create a new environment.

If you do not want the default packages installed in a particular environment, use the `--no-default-packages` flag:

```
conda create --no-default-packages -n myenv python
```

Sharing an environment

Exporting config to the `.yml` file

```
conda activate myenv
conda env export > envir.yml
conda env export -n myenv > envir.yml
```

See above how to recreate environment using this file.

Exporting an environment file across platforms If you want to make your environment file work across platforms, you can use

```
conda env export --from-history > environment.yml
```

This will only include packages that you've explicitly asked for, as opposed to including every package in your environment. On recreating the environment via `environment.yml` on the different platform all the platfor specific dependencies will be resolved automatically.

Creating an environment file manually A simple environment file

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

A more complex environment file

```
name: stats2
channels:
  - javascript
dependencies:
  - python=3.6   # or 2.7
  - bokeh=0.9.2
  - numpy=1.9.*
  - nodejs=0.10.*
  - flask
  - pip:
    - Flask-Testing
```

Note the use of the wildcard * when defining the patch version number...

You can exclude the default channels by adding nodefaults to the channels list.

```
channels:
  - javascript
  - nodefaults
```

This is equivalent to passing the `--override-channels` option to most conda commands.

Adding `nodefaults` to the channels list in `environment.yml` is similar to removing `defaults` from the channels list in the `.condarc` file. However, changing `environment.yml` affects only one of your conda environments while changing `.condarc` affects them all.

Updating an environment

Edit `environment.yml` and run

```
conda env update --prefix ./env --file environment.yml  --prune
```

The `--prune` option causes conda to remove any dependencies that are no longer required from the environment.

Using `pip` with `conda`

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

Issues may arise when using pip and conda together... [read it !!!]

Setting environment variables

```
conda env config vars list
conda env config vars set my_var=value
conda env config vars set my_var=value -n myenvir
conda env config vars set my_var=value -p /path/to/myenvir
conda env config vars unset my_var -n myenvir
echo my_var        # ?
```

Saving environment variables

Virtual environments

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated spaces for them that contain per-project dependencies for them.

Users can create virtual environments using one of several tools such as Pipenv or Poetry, or a conda virtual environment. Pipenv and Poetry are based around Python's built-in venv library, whereas conda has its own notion of virtual environments that is lower-level (Python itself is a dependency provided in conda environments).

Conda channels

Conda channels are the locations where packages are stored. Packages are automatically downloaded and updated from https://repo.anaconda.com/pkgs/. You can modify what remote channels are automatically searched, see.

```
conda install scipy --channel conda-forge
conda install scipy -c conda-forge
conda install scipy --channel conda-forge --channel bioconda
conda install scipy -c conda-forge -c bioconda

conda install conda-forge::scipy
```

To add a channel to your `conda` configuration i.e. `.condarc`

```
conda config --add channels CHANNELNAME
```

From the command line use `--override-channels` to only search the specified channel(s), rather than any channels configured in `.condarc`. This also ignores conda's default channels.

```
$ conda search scipy --channel file:/<path to>/local-channel --override-channels
```

In `.condarc`, use the key channels to see a list of channels for conda to search for packages.

Configuration

Examine Conda configuration and configuration services

```
conda config --show
conda config --show-sources
```

`.condarc` config file

The `.condarc` file is not included by default, but it is automatically created in your home directory the first time you run the `conda config` command.

```
conda config --add channels conda-forge
```

Information about your `.condarc` file, including where it is located

`conda info`

You can create this file anew, edit if already exists, or download a sample `.condarc` to edit it and save to your user home directory or root directory.

To set configuration options, edit the `.condarc` file directly or use `conda config --set` command

`conda config --set auto_update_conda False`

complete list of conda config commands, see the command reference or

`conda config --help`

For a complete list of all available options for your version of conda, use

`conda config --describe`