IMPORTING MODULES IN PYTHON

Aims of the Lecture

- Learn the purpose of importing modules in Python.
- Install and understand the most commonly imported modules in Python.

Additional Reading and Sources

- Digital Ocean (https://www.digitalocean.com/community/tutorials/how-to-import-modules-in-python-3)
- Decimal Documentation (https://docs.python.org/2/library/decimal.html)
- Numpy Quickstart Tutorial (https://docs.scipy.org/doc/numpy/user/quickstart.html)

Modules

- Python contains a variety of built-in **data types**, **classes** and **functions**.

- As you have seen, sometimes these functions are limited, and thus we resort to importing modules/packages.

- Modules are *.py* files which contain Python code and can be imported to our code.

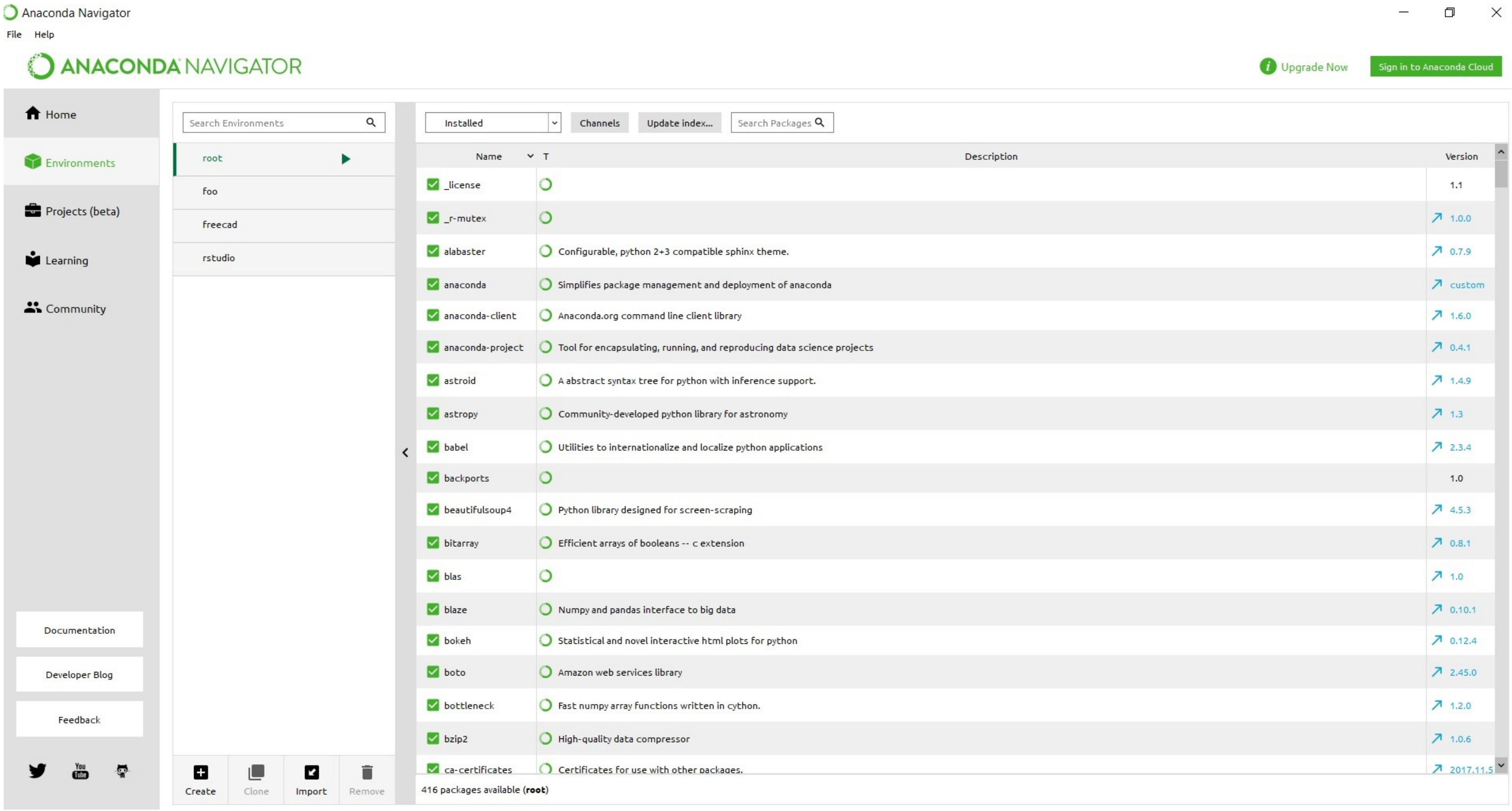
Do we already have modules installed?

- Yes! There are a number of them installed in the **Python Standard Library**
- To check them, you can use the following command:

In [ ]:

```
!pip freeze
```

- If you are using Anaconda, you can also go to the **environments** tab and check the list.



Name	T	Description	Version
☑ _license	○		1.1
☑ _r-mutex	○		↗ 1.0.0
☑ alabaster	○	Configurable, python 2+3 compatible sphinx theme.	↗ 0.7.9
☑ anaconda	○	Simplifies package management and deployment of anaconda	↗ custom
☑ anaconda-client	○	Anaconda.org command line client library	↗ 1.6.0
☑ anaconda-project	○	Tool for encapsulating, running, and reproducing data science projects	↗ 0.4.1
☑ astroid	○	A abstract syntax tree for python with inference support.	↗ 1.4.9
☑ astropy	○	Community-developed python library for astronomy	↗ 1.3
☑ babel	○	Utilities to internationalize and localize python applications	↗ 2.3.4
☑ backports	○		1.0
☑ beautifulsoup4	○	Python library designed for screen-scraping	↗ 4.5.3
☑ bitarray	○	Efficient arrays of booleans -- c extension	↗ 0.8.1
☑ blas	○		↗ 1.0
☑ blaze	○	Numpy and pandas interface to big data	↗ 0.10.1
☑ bokeh	○	Statistical and novel interactive html plots for python	↗ 0.12.4
☑ boto	○	Amazon web services library	↗ 2.45.0
☑ bottleneck	○	Fast numpy array functions written in cython.	↗ 1.2.0
☑ bzip2	○	High-quality data compressor	↗ 1.0.6
☑ ca-certificates	○	Certificates for use with other packages.	↗ 2017.11.5

416 packages available (**root**)

- It is possible that you get different lists between the !pip freeze and the anaconda list.

How do modules work and what can I do with them?

Remember this example?

```
In [ ]: 2.2+1.1==3.3
```

- The float data type/class is limited to work with decimal numbers.
- That's why someone has created the **decimal** module for us to use freely!

- To use the module, first we have to check that it is installed in our computers:

- To do so, we can simply run the following command expecting that no error message is shown:

In [ ]:

```
import decimal
```

- You should not get an error, as this module is part of the Standard Library!

- Now we can use the module to fix the comparison error:

In [ ]:

```
from decimal import * # the * means "all"
Decimal("1.1")+Decimal("2.2")==Decimal("3.3")
```

- Notice that now we have solved the problem, but we have to write quite a lot!

- I know we don't get charged for using letters, but when using modules it is quite helpful to shorten names!

- We can use the "as" operator to give a new alias to a module/function when imported.

In [ ]:

```
from decimal import Decimal as dec
dec("1.1")+dec("2.2")==dec("3.3")
```

Another module that we can import is the **random** module, which lets us produce random numbers.

In [ ]:

```
import random
```

For instance, we can produce a random integer number between 0 and 10 by running the following command:

In [ ]:

```
random.randint(0,10) # in this case, the 10 is considered!
```

- Small exercise: Implementing a coin toss function:

In [ ]:

```python
# Coin toss
import random # you don't need this if you have imported before

def cointoss():
    coin = random.randint(0,1)
    if coin ==0:
        print('heads')
    else:
        print('tails')
    return

# Call the function
cointoss()
```

- How can we modify the function to do "x" coin tosses based on a user input?

In [ ]:

```
# Multiple coin tosses
```

The numpy array module

- Numpy is one of the most widely used packages in all Python.

- Everything that has to do with matrices/images/data uses **numpy**.

- Just as decimal is an upgrade on the missing features of float, you can think of numpy as an upgrade of lists.

In [ ]:

```
[1,2,3]+[4,5,6]
```

- First, we will check that we have numpy by importing it:

In [ ]:

```
import numpy
```

if you don't have it, install it using the *pip* command:

In [ ]:

```
!pip install numpy
```

- Since we want to save time, we will import numpy using a pseudonym:

In [ ]:

```
import numpy as np
```

- Now we can create a **numpy array**.

```
In [ ]: np.array([1,2,3])
```

```
In [ ]: type(a)
```

This numpy array works like a *vector*, and thus it can now be added to another array.

```
In [ ]: np.array([1,2,3])+np.array([4,5,6])
```

Defining a numpy array

```
In [ ]: # Create an array range
        np.arange(10)
```

```
In [ ]: # Create a 2D numpy array (like a table)
        a = np.array([[1,2,3],[4,5,6]])
        print(a)
```

```
In [ ]: a.ndim
```

```
In [ ]: a.shape
```

```
In [ ]: a.size
```

```
In [ ]: a.dtype
```

```
In [ ]: a.reshape(1,6)
```

Basic operations with numpy arrays

In [ ]:

```
print(a)
a.sum()
```

In [ ]:

```
a.sum(axis=0)
```

In [ ]:

```
a.sum(axis=1)
```

In [ ]:

```
a.min()
```

In [ ]:

```
a.max()
```