Recall: modules

```
import central

x = central.mean([1.0, 4.2, 2.3, 9.9], geometric=True)
```

Also recall: iteration

```
for c in "hello world":
    print(c > 'a')
```

```
for n in range(10):
    print(n*n)
```

```
for temp in [-2, -20, 6]:
    print(temp)
```

```
for each_thing in a_bunch_of_things:
    # statements that do something with each_thing
```

Today

Let's get visual!

- installing Python packages
- plotting with `matplotlib` and `numpy`
- also, arrays

Packages

How can you share your module with others?

One common answer: PyPI

- Python Package Index
- e.g., https://pypi.org/search/?q=engi1020

Includes software *license* information

Licensing

Copyright

- what does it mean?
- what does it apply to?
- how do you get it?

Licensing

Can be *proprietary* or *open source* (choosealicense.com)

Definition from CIPO:

__________________ is the sole right to produce, reproduce, publish or perform a work (or a substantial part of it) that belongs to one of the following categories:

- **literary** (e.g. books, pamphlets, computer programs and other works consisting of text)
- **dramatic** (e.g. motion picture films, plays, screenplays and scripts)
- **musical** (e.g. musical compositions, with or without words)
- **artistic** (e.g. paintings, drawings, maps, photographs, sculptures and plans)

Copyright also protects performances, sound recordings and communication signals, such as radio waves.

Copyright automatically protects your work as soon as you create it.

It lasts for the life of the creator plus 70 years after their death.

Installing packages

From the command line*:

```
pip install engi1020
Collecting engi1020
  Downloading engi1020-0.1.15-py3-none-any.whl (11 kB)
[...]
Successfully installed engi1020-0.1.15 pyserial-3.5
```

Within a Python interpreter:

```
>>> import pip
>>> pip.main(['install', 'engi1020'])
```

* Windows Command Prompt, macOS Terminal, Linux terminal emulator...

Install some packages

Let's install some packages!

```
$ pip install engi1020 matplotlib numpy
```

Now we're ready for...

Mathematical operations

We have seen:

```
x = 1 + 2
r = 1 % 2
```

```
from math import *
y = sin(x)
z = log(x)
```

... but what about lists?

```
from math import *
sin([0, pi/2, pi])
```

Numerical Python

`numpy` module provides:

- more math functions
- ability to work with *arrays*

What do you see when we run this example?

```
import numpy
from math import *

numpy.sin([0, pi/2, pi, 3*pi/2, 2*pi])
```

`numpy` *arrays*

Like lists, but:

- all elements have the same type
- can do math with them:

```
l1 = [1,2,3]
l2 = [4,5,6]

l1 + l2
l1 * l2
```

```
a1 = array([1,2,3])
a2 = array([4,5,6])

a1 + a2
a1 * a2
```

More numerical Python

`numpy` provides some other useful functions:

- `linspace` — a bit like `range()`, but can do `float`
- `dot` and `cross` — $\boldsymbol{a} \cdot \boldsymbol{b}$ and $\boldsymbol{a} \times \boldsymbol{b}$ (á la MATH 2050)
- `cumsum` (cumulative sum), `gcd`, `lcm`…
- `convolve` — you can worry about that in Term 4 :)

Plotting with matplotlib

Can plot iterable things with matplotlib:

```
plot(x_values, y_values)
```

```
from matplotlib.pyplot import *
from numpy import *

# Our x values will be spread out over the range [0, 2pi]
x = [0, pi/2, pi, 3*pi/2, 2*pi]

# Each y value will be the sin of a corresponding x value
y = sin(x)     # note: numpy.sin, not math.sin

# Plot our y values against our x values
plot(x, y)
```

More detailed plots

```
import matplotlib.pyplot as plt
import numpy as np

x = [0, pi/2, pi, 3*pi/2, 2*pi]
y = np.sin(x)

plt.style.use('seaborn-colorblind')
plt.plot(x, y, 'r--')
plt.title('A simple plot')
plt.xlabel('theta')
plt.xlabel('sin(θ)')
plt.show()
```

- and bar graphs, and histograms, and...
- see the `pyplot` tutorial on the `matplotlib` website

A better `sin` plot?

Summary

Numerical Python

- arrays
- mathematical functions

Plotting with `matplotlib`

- much more to explore in the `pyplot` tutorial!