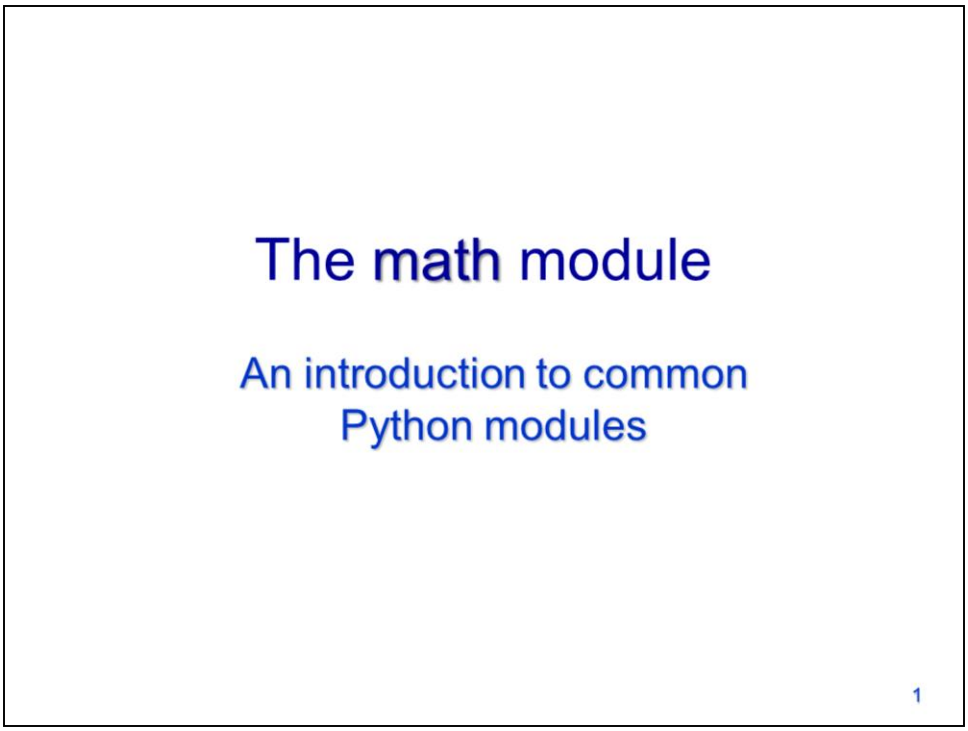


This video will provide an introduction to using Python modules. It will also discuss some features of the **math** module.

Modules and Functions

- Modules are toolboxes…
  - contain tools (functions) that perform related tasks.
- Code packaged for easy reuse and sharing.
- Many built-in modules for Python
- Some modules load automatically for you, for others you must use an import statement.
- Custom functions and modules can package your code for reuse.

2

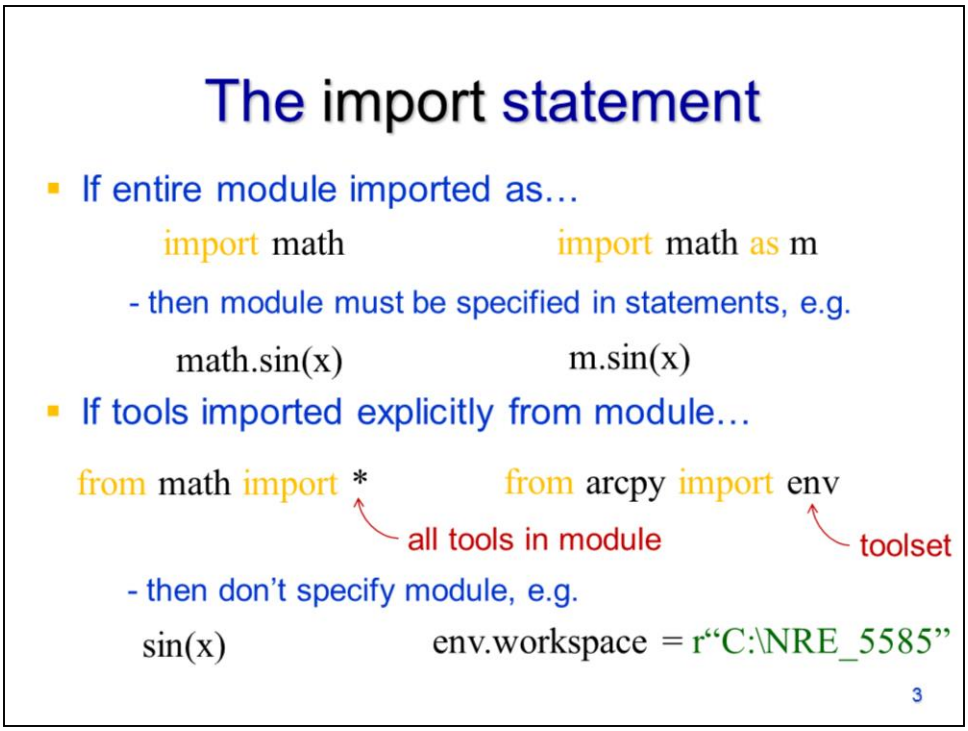
Modules are Python's version of a toolbox – they contain a set of tools that perform related tasks.

Modules are a convenient way to package code so that it can be easily used by many scripts or shared with others.

The Python installed with ArcGIS comes with dozens of modules that can be used in your scripts. Many more modules are available online for free download.

Modules need to be imported into a script before you can access their functionality.

You can also create custom functions and modules that allow you to reuse code more efficiently.



There are a few different ways that we can import modules which will determine the syntax needed to use the module's tools.

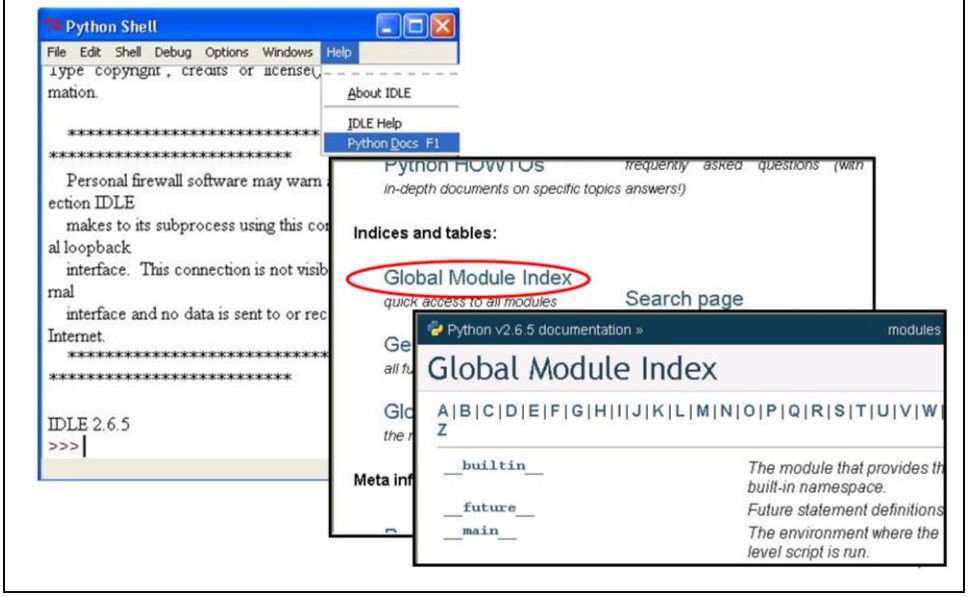
If we simply import the module, then we need to use the full module name to "call" its functions.

When importing a module, we can assign the module a more convenient name that we can use when referring to it in the script.

We can import all tools from a module in which case the module name does not need to be specified when calling its functions. This can make statements more concise but it has the disadvantages of not allowing the use of the auto-completion and it can also make the script more difficult to follow.

Some modules contain toolsets which can be imported without the rest of the module. In this case, only the toolset name needs to be specified when calling the toolset's functions.

Documentation on built-in modules



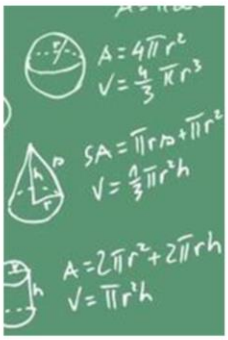
Documentation for Python's built-in modules can be found through the Python Shell's help documentation.

These modules are listed under the Global Module Index.

The math module

Import math as m

- Constants and math operators. For complex numbers, use cmath module.
- Constants…
  - m.pi (3.14159…)
  - m.e (2.71828…)
- Powers and logs…
  - m.exp(x) ← e^x
  - m.log(x, b) ← log of x, base b
  - m.log(x) ← natural log of x



5

The math module contains tools and constants useful for higher-level math operations.

The module includes constants such as pi and e. Note that I assigned the name “m” to the math module when I imported it so I need to refer to this name when using the math tools.

The math module includes exponential and log functions.

The math module continued

- Angular conversions…

  m.degrees(x) - radians to degrees

  m.radians(x) - degrees to radians

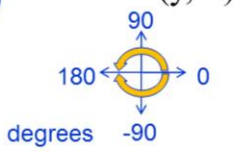
- Trigonometry (results in radians)…

  m.sin(x) m.cos(x) m.tan (x)

  m.asin(x) m.acos(x) m.atan (x)

  m.atan2 (y, x)

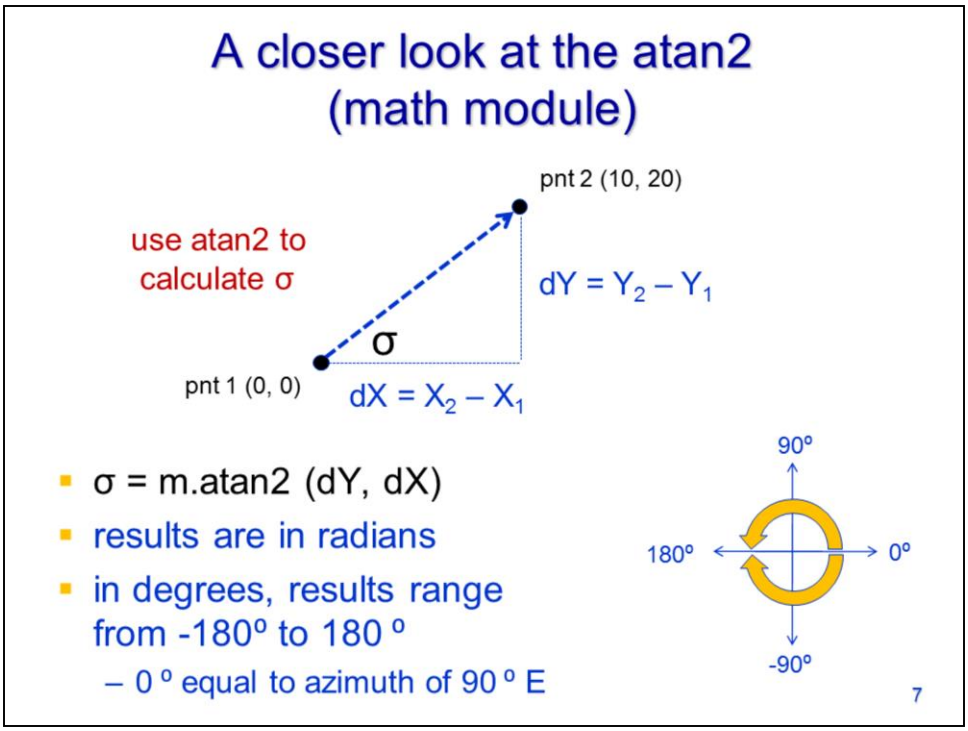
  - determines quadrant vector is located in
  - useful for azimuth calculations



6

The module also contains functions for converting angles from degrees to radians and vice versa.

Trigonometric functions are also included. Note that the trig functions require the input angle to be in radians. **atan2** is a useful function for calculating azimuths but it can be a bit non-intuitive – so we'll take a closer look at it.

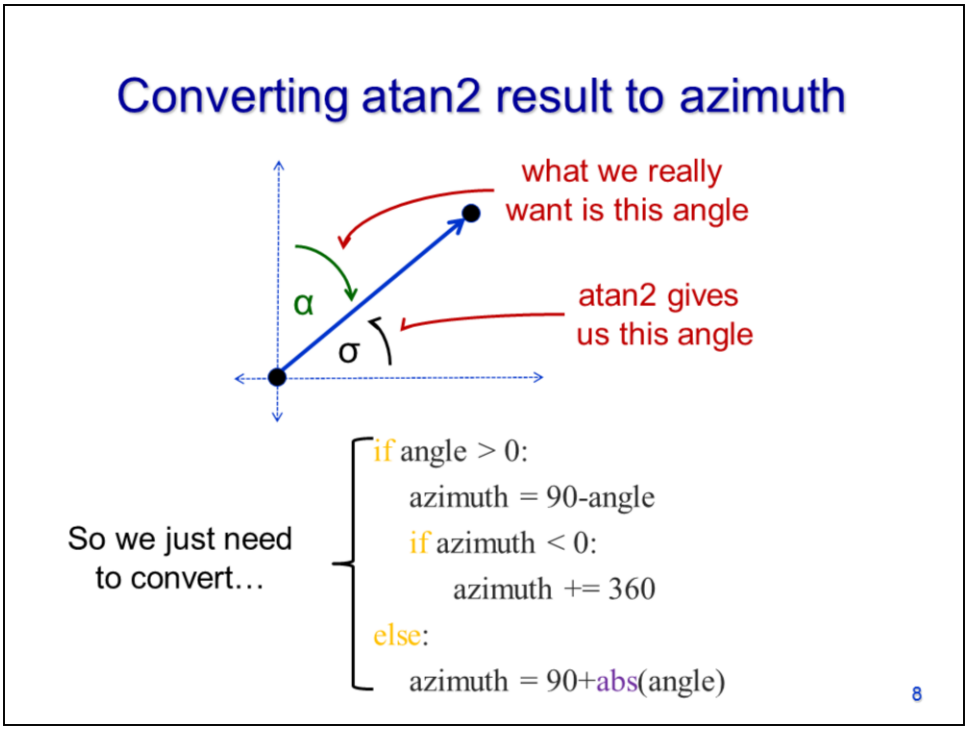


The atan2 function calculates the angle formed between a line and the x-axis.

The inputs for atan2 are the change in the Y direction and the change in the X direction.

The function gives results in radians.

Note that the function gives results that range from –pi to pi (-180 to 180 degrees); 0 degrees is along the positive x-axis. The angle increases positively in the counter-clockwise direction and negatively in the clockwise direction.



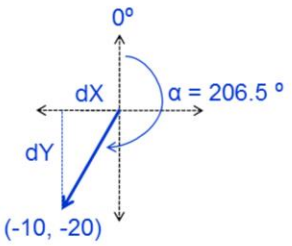
Azimuth is typically a more useful measure of direction for geospatial work than the output of atan2. Azimuth is a measure of direction with 0 degrees at North and increasing in the clockwise direction.

The code here will convert an atan2 angle into an azimuth.

Script example: math module

```
import math as m
dX = -10
dY = -20
angle = m.degrees(m.atan2(dY,dX))
if angle > 0:
    azimuth = 90-angle
    if azimuth < 0: azimuth += 360
else: azimuth = 90+abs(angle)
```

Calculate azimuth from dX and dY



9

The code here shows how to calculate azimuth based on the change in X and Y directions.