Math
The math module

• **import math** will let you use quite a lot of built-in mathematical functions
  – **math.pi** and **math.e** are the values $\pi$ and $e$, respectively
  – **math.log(x, base)**, **math.log2(x)**, and **math.log10(x)** return logarithms of $x$
  – **math.pow(x, y)** returns $x$ raised to the power $y$
  – **math.sqrt(x)** returns the square root of $x$
  – All the standard trigonometric functions (**math.sin(x)**, **math.cos(x)**, etc.) are provided
  – ...and many more

• You don’t have to learn all these, you just need to know where to find them when you need them
Complex Numbers

• Python has complex numbers, written as \( re + im \ j \) (or as \( re – im \ j \))
  – Python uses \( j \) rather than \( i \) because electrical engineers like to use \( i \) for something else
  – Example: \( c = 3 + 5.5j \)

• Python also has a \texttt{cmath} module which is mostly for trigonometric functions on complex numbers
The statistics module

• Python has a statistics module which provides some basic descriptive statistics methods
  – It has mean, median, mode, stdev, and variance, and not much else
The numpy module

• Very popular with engineers and machine learning users
• Has a large variety of linear algebra and signal processing libraries
  – Matrix inversion, factorization, multiplication
  – Fourier transforms
  – Optimization
  – ...
• Also has tools for integrating with C/C++
• There is also a scipy module with similar (more) functionality
The random module

- There are no random numbers on a computer. Not in Python, not in any language.
- What we do have are pseudorandom numbers, which are generated by a simple formula designed to give numbers that look random.
  - Pseudorandom numbers are fine for most games and minor applications, but should never be used in a supposedly secure application!
- Some of the more useful methods are:
  - `random.random()` returns a float in the range [0, 1)
  - `random.randint(a, b)` returns a value between `a` and `b`, inclusive
  - `random.shuffle(seq)` randomizes the sequence `seq` in place
  - `random.choice(seq)` returns a randomly chosen element of `seq`
  - `random.seed(i)` initializes the random number generator with the integer `i` -- this allows you to use the same “random” sequence each time
datetime and calendar

• The `datetime` module provides objects representing dates and times of day
• The `calendar` module provides calendar-related functions
• Some uses of these are not too difficult
• IDLE example: `date.py`
• However, dates and times are too complicated (time zones, daylight savings, leap years, etc.) for a general programming course
  – This course should give you the tools you need to understand the documentation, if you ever need to
  – Can use them for basic things such as recording elapsed time