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 π 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
>>> c
(3+5.5j)
>>> 2 * c
(6+11j)
```

- Python also has a **cmath** module which is mostly for trigonometric functions on complex numbers

The `statistics` module

- Python has a `statistics` module which provides a surprisingly small number of methods
  - It has `mean`, `median`, `mode`, `stdev`, and `variance`, and not much else

The `random` module

- There are no random numbers on a computer. Not in Python, not in any language. Period.
- 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 *N* in the range 0 <= *N* < 1.0
  - **`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

Decimals and fractions

- The **`decimal`** module provides methods for representing numbers and doing arithmetic in decimal rather than in binary
  - This works more like people expect, and is good for financial applications
  - Decimal arithmetic is more awkward to use, and is much slower than, floating-point arithmetic
- The **`fractions`** module provides methods for representing numbers and doing arithmetic with ***rational*** numbers
  - A fraction, or rational number, is represented by a pair of integers, and integers can be represented exactly
  - Hence, rational arithmetic never loses precision
  - Fractions are simpler to work with than decimals, but are again much slower than floating-point arithmetic

`datetime` and `calendar`

- The `datetime` module provide objects representing dates and times of day
- The `calendar` module provides calendar-related functions
  - Some uses of these are not too difficult

```
>>> x = datetime.date.today()
>>> x
datetime.date(2015, 9, 12)
>>> x.year
2015
```

- 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

The End

Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin.

-- John von Neumann