Python, Day 2: Numbering Systems

Andrew Bydlon

January 6, 2019

Various numbering systems

As stated, there are several classes of number data in Python.

- '**int**': An integer.
- '**float**': A real number.
- '**complex**': A complex number.
- '**bool**': A boolean variable, taking values True or False.

Common functions

The absolute value can be taken with the abs() command.

Example

```
> > > abs(-3)
      3
> > > abs(-1.42)
      1.42
> > > abs(1+1j)   # This returns the norm of a complex number.
      1.4142135623730951
```

We can take the power of 2 numbers by taking the pow() command. pow(a,b) returns a^b.

Example

```
> > > pow(2,3)
      8
```

Common functions cont'd

round() rounds a number to the nearest integer. You can add an integer to round to a specific number of decimal digits.

Example

```
> > > round(3.5)
      4
> > > round(3.1415926,4)
      3.1416
```

The max() and min() functions return the largest the smallest number in a given collection of *int*s or *float*s.

Example

```
> > > max(1.3, -3.4, 3.14, 2.71)
      3.14
> > > min(1.3, -3.4, 3.14, 2.71)
      -3.4
```

Quick Aside: Importing packages

One of the more important aspects of python is to be able to use the packages that others have written and compiled to make your life easier. This is done with the import command:

```
import math
```

You can get a list of available modules with the help command:

```
help('modules')
```

To get access to many more mathematical functions, we will use import math.

We will return to this topic of **modules** later on in the course.

Functions in the *math* module

Here is a list of some important functions of the *math* module.

- **math.pi** the constant $\pi = 3.1415926\ldots$
- **math.e** the constant $e = 2.71828\ldots\ldots$
- **math.ceil(n)** the ceiling, or round-up, operation: $\lceil 3.01 \rceil = 4$
- **math.floor(n)** the floor, or round-down, operation: $\lfloor 3.99 \rfloor = 3$
- **math.sqrt(n)** the square-root: $\sqrt{2}$
- **math.log(n,b)** $\log_b(n)$. If no b is given, it defaults to $\ln(n)$: $\log(100, 10) = 2.0\ldots$ $\log(3) = 1.098612289\ldots$
- **math.sin(n)** the sine function: $\sin(\frac{\pi}{3}) = 0.8660254\ldots$
- **math.cos(n)** the cosine function: $\cos(\frac{\pi}{3}) = .50$
- **math.tan(n)** the tangent function: $\tan(\frac{\pi}{3}) = 1.7320508\ldots$
- **math.degrees** converts radians to degrees.
- **math.radians** converts degrees to radians.

Formatting numbers

Occasionally you want to print a number to the user in a more reasonable format, e.g. with money or counting objects. This uses the format() function.

format(YourNumber, Specifier)

Again, to see possibilities for formatting (beyond what is specified in these slides), check help("format") or help("FORMATTING").

width.precisionf formatting

This method of formatting allows you to specify the *width* of a character and the *precision*.

Example

```
> > > format(math.pi,"9.2f")    # Note that it is right aligned
      '     3.14'
> > > format(math.pi,"3.5f")
      '3.14159'
```

- The **width** is the minimum number of characters reserved for a value. It increases dynamically.
- The **.precision** is the number of decimal places to store.
- The 'f' specifies floating point.

The general syntax is format(n,"**w.p**f").

Other formatting options

Example (Scientific Notation)

```
> > > format(123456789,"9.2E")
'  1.23E+08'
> > > format(123456789,"3.5e")
'1.23457e+08'
```

Example (Commas and Percentages)

```
> > > format(123456789.0123,"9,.2f")
'123,456,789.01'
> > > format(.05385,"%")
'5.385000%'
```

Example (Left Align)

```
> > > format(math.pi,"<10.2f")
'3.14      '
```

Formatting options for integers

You can format integers using decimal (base 10), binary (base 2), hexidecimal (base 16), octal (base 8):

Example

```
> > > format(68,"4d")
    '  68'
> > > format(12,"b")
    '1100'
> > > format(248,"x")
    'f8'
> > > format(100,"o")
    '144'
```

Assignment 2

Let's make a program to calculate the Annuity A on compounded interest. Recall the formula

$$A = P \cdot \left(1 + \frac{r}{n}\right)^{nt}$$

Take the users

- **principal investment** P
- **annual interest rate** r
- **compounding frequency** (times compounded per year) n
- **time in years** t

and display (with formatting) their Annuity at time t. It should print something like the following:

"Your Annuity at time (t) is $ (A). You earned $ (A-P)."