A Second Look:
constants, data types, variables, expressions,….

FEB 3RD 2012

More in-depth discussion

- Data types
- Variables
- Expressions
- Key words
- Built-in functions
- Modules
- Control flow statements

Data types

- We have seen four data types thus far:
  - int: -90, 8987
  - float: 9.98, -3.54
  - str: “hello”, “a”
  - bool: True, False

Numeric data types

- Python supports four numeric data types:
  - *plain integers,*
  - *long integers,*
  - *floating point numbers*, and
  - *complex numbers.*

- Plain integers, i.e., objects of type `int`, are those that fit in 32 bits.

Bits, bytes, words

- A *bit* (short for binary digit) is the smallest unit in a computer.
- A *byte* is 8 bits; a *word* is 2 bytes (16 bits).
- Any integer that can be represented in binary using 4 bytes (or 2 words or 32 bits) is an `int` type object in Python.
- The largest `int` object is

$$2^{31} - 1 = 2147483647$$

And the smallest is -2147483648

Playing with these notions

- Try

```
import sys
sys.maxint
```

- Also try this

```
n = -37
bin(n)
n.bit_length()
```

- Try this also

```
type(sys.maxint+1)
```

A few words on long type

- Integers of type long can be arbitrarily large (or small). In other words, the type long provides *infinite precision*.
- A long constant can be explicitly specified by appending an L at the end of the integer. Try

```
x = 875L
type(x)
```

- Operations can be performed on a mix of long and int objects; the type of the answer will be the larger type, i.e., long.

The `float` type

- Numbers with decimal points are easily represented in binary:
  - 0.56 (in decimal) = 5/10 + 6/100
  - 0.1011 (in binary) = ½+0/4 + 1/8 +1/16

- The i^{th} bit after the decimal point has place value $1/2^i$.

- **Example:** 0.1101 = ½ + ¼ + 1/16 = 13/16 = 0.8125

- However, not all real numbers (even rational numbers) can be represented *exactly* by finite sums of these fractions.

Be wary of floating point errors

- Try 0.1 + 0.2
- Try adding 0.1 ten times.
- Try 0.1 + 0.1 + 0.1 – 0.3

- In general, *never* test for equality with floating point numbers.
- This is an infinite loop! Try it.

```
sum = 0.1
while sum != 1:
        sum = sum + 0.1
```

Some functions for floating point numbers

- The math module contains functions (e.g., `math.sqrt(x)`) for floating point numbers.

Function	What it does
`math.ceil(x)`	Returns the ceiling of x as a float
`math.floor(x)`	Returns the floor of x as a float
`math.trunc(x)`	Returns x truncated to an int
`math.exp(x)`	Returns e^x
`math.log(x)`	Returns logarithm of x to the base e
`math.log(x, b)`	Returns logarithm of x to the base b

There are many other functions in the math module: trignometric, hyperbolic, etc. There are also constants: math.pi and math.e.

Try solving these problems

- Given the radius of a circle, find its area.
- Given a positive integer, find the number of digits it has.

  Example: `int(math.ceil(math.log(565656, 10)))`
- There are also some built-in Python functions that are useful for math:
  - `round(x, n)`: returns the floating point value *x* rounded to *n* digits after the decimal point. If *n* is omitted, it defaults to zero.
  - `abs(x)`: returns the absolute value of *x*

Range of floating point numbers

- What is the largest floating point number in Python? Unfortunately, there is no `sys.maxfloat`. Here is an interesting way to find out:

```
prod = 1.0
while prod*1.1 != prod:
        prev = prod
        prod = prod*1.1
print prev, prod
```

- The output is `1.78371873262e+308  inf`

What does this output mean?

- Python uses an object called `inf` to represent positive infinity.
- When `1.78371873262e+308` was multiplied by 1.1 (i.e., increased by 10%), we went beyond the upper limits of type `float`.
- This means that the largest floating point number in Python has 308 digits.
- Notice that the `while`-loop terminated because

  `inf * 1.1` equals `inf`.

A better version of this program

```
import math
prod = 1.0
while not math.isinf(prod):
        prev = prod
        prod = prod*1.1
print prev, prod
```

- There is a function called `isinf(x)` in the `math` module that tells us if `x` equals `inf`.