CISS450: Artificial Intelligence
Lecture 2: Built-in Numeric Types

Yihsiang Liow

Built-in Numeric Types

- There are three built-in numeric types:
  - Integers (actually integers and long integers)
  - Floats
  - Complex numbers
  - Boolean
- We will only talk about integers and floats
- Refer to the online material for complex numbers in Python

Integer

- There are three kinds of integers in Python: integer, long integer, boolean
- Comparison:

Python	**C/C++**
integer	long int
long integer	

- The programmer need not worry about converting integer to long integer when there is an overflow. Python does this for you automatically.

Integer

- The integer is at least 32 bits
- The long integer is limited only by the computer's memory resources
- The boolean has two values representing true and false; they correspond to 1 and 0 respectively.

Integer Literals

- Examples:
  - Decimals: 0, 3, -25, 99999999999999999999999
  - Octals: 03, 015 (begin with zero)
  - Hexadecimals: 0x2, 0x54 (begin with zero-x)
- Octals = numbers in base 8 (digits 0,1,2,...7)
- Hexadecimals = numbers in base 16 (digits 0,1,...,9,A,B,C,D,E,F)
- Before Python 2.2, you have to write 99999999999999999999999L. The “L” or “l” is now not necessary.

Floats

- Python's floating point type uses the IEEE 754 double-precision standard
- C/C++ has two floating point types:
  - IEEE 754 single-precision standard, and
  - IEEE 754 double-precision standard

  The name of the types are `float` and `double`.
- Remember that floating point values are approximations. Try `0.0000001 * 1000000`.

Float Literals

- Same as C/C++
- Examples:
  - 0.0, 1.23, -3.14
  - 12e45, -12e45, 12e-45
  - 1.2e45, -1.2e45, 1.2e-45, -1.2e-45
  - 12e4.5, -123e4.5, 12e-4.5, -12e-4.5
  - 1.2e4.5, -1.2e4.5, 1.2e-4.5, -1.2e-4.5
  - You can use E instead of e

Float Literals

- Same as C/C++
- Examples:

```
x = 2.0 / 3
print("---%i---%5.2f---" % (x, x))
```

Built-in Operators

- `+`, `-`, `*`, `/`, `**`, `%`
- `+=`, `-=`, `*=`, `/=`, `**=`, `%=`
- `==`, `!=`, `<`, `<=`, `>`, `>=`
- Review PEMDAS
- WARNING:

C/C++	**Python**
N.A.	`**` (exponentiation)
`(x < y) && (y < z)`	`x < y < z`

- In general, you can chain up comparisons

Built-in Operators

- For mixed expression, integers are coerced to floats
- For explicit type conversion, use `int()` and `float()`
- Operations on floats are much slower. So avoid if possible.

Built-in Functions

- pow pow(a,b) returns a**b
  pow(a,b,c) return a**b % c
- abs abs(x) returns absolute value of x
  abs(5) returns 5, abs(-3) return 3
- divmod divmod(a,b) returns (a/b, a%b). This is a tuple of two values (see later).
- round round(x,n) returns x rounded up to nth digit after the decimal point. Default value of n is 0.
  round(1.2345) return 1.0
  round(1.2345,3) returns 1.2350000...

Booleans

- Literals: `True`, `False`
- WARNING:

C/C++	Python
`&&`	`and`
`||`	`or`
`!`	`not`, `!`

- Non-zero numeric values are considered true; zero numeric values are considered false

and, or, not

- and and or are also operators of numeric values, not just boolean values
- Given x and y:

  x and y = x if x can be coerced to False, otherwise y

  x or y = x if x can be coerced to True, otherwise y

  not x= True if x can be coerced to True, otherwise False
- Note that if x and y are 0 or 1 (i.e., True or False), then the above is just like what you would expect from the truth tables

Bitwise Operations

- Let x,y be integers. Let n be a positive integer. The following are the bitwise operations:

x & y	bitwise AND
x | y	bitwise OR
x ^ y	bitwise XOR
x << n	left-shift on x by n bits
x >> n	right-shift on x by n bits
~x	bitwise inversion

- Check documentation for further details