Control Flow

• Control flow refers to the order in which a program executes

• This is fundamental in the imperative programming paradigm
  – E.g. Java or Python

• In other programming paradigms, the compilers or the interpreters take care of the ordering
  – E.g. functional and logic programming
Control Flow Mechanisms

- Sequencing
  - Textual order, precedence and associativity in expression
- Selection
- Iteration
- Procedural abstraction
- Recursion
- Concurrency
- Nondeterminacy

Expression Evaluation

- Expressions consist of operands (e.g. a variable) and operators or functions (e.g. +, abs())
  - By definition, operators and functions return a value
- Operators are also functions
  - Infix notation is just syntactic sugar
  - In C++, a + b means a.operator+ (b)
Overloading Operators

- Python example

```python
import time

class Time:
    def __init__(self, seconds):
        self.seconds = seconds
    def __repr__(self):
        return time.ctime(self.seconds)
    def __add__(self, x):
        return Time(self.seconds + x)
    __radd__ = __add__ # support for x+t
    def __sub__(self, x):
        if hasattr(x, 'seconds'):
            # test if x could be a Time
            return self.seconds - x.seconds
        else:
            return self.seconds - x
```

Overloaded Operator

\[\begin{array}{c}
\text{++} \\
\text{--}
\end{array}\]

Operators

- Operators are used in
  - Prefix notation
    - E.g. Expression \((* (+ 1 3) 2)\) in Lisp
  - Infix notation
    - E.g. Expression \((1 + 3) * 2\) in Java
  - Postfix notation
    - E.g. Increment \(a++\) in C

- Operators can have 1 or more operands
  - Increment in C is a one-operand operator: \(a++\)
  - Subtraction in C is a two-operand operator: \(a-b\)
  - Conditional expression in C is a three-operand operators:
    \((a == 3 ? 0 : 1)\)
Operators

Precedence and Associativity

- Precedence and associativity deal with the evaluation order within expressions.
- *Precedence* rules specify the order in which operators of different precedence level are evaluated. *- \* usually groups more tightly than +.*
- What is the result of $4 \times 5 ** 6$?

### Operator Precedence Table

<table>
<thead>
<tr>
<th>Fortran</th>
<th>Precal</th>
<th>C</th>
<th>Ada</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=, -= (post-inc., dec.)</td>
<td><strong>not</strong></td>
<td>+=, -= (pre-inc., dec.), abs (absolute value), +, -(memory), &amp; (address of), * (contents of), ! (logical not), not, **</td>
<td>*/**, /, div, mod, **(binary), /, % (modulo division), **(binary), +, - (binary), +, -(memory)</td>
</tr>
<tr>
<td>*/**, /, div, mod, **(binary), /, % (modulo division)</td>
<td>*/**, /, div, mod, **(binary), /, % (modulo division)</td>
<td>*/**, /, div, mod, **(binary), /, % (modulo division)</td>
<td>*/**, /, div, mod, **(binary), /, % (modulo division)</td>
</tr>
<tr>
<td>+=, -= (post-inc., dec.)</td>
<td><strong>not</strong></td>
<td>+=, -= (pre-inc., dec.), abs (absolute value), +, -(memory), &amp; (address of), * (contents of), ! (logical not), not, **</td>
<td>*/**, /, div, mod, **(binary), /, % (modulo division), **(binary), +, - (binary), +, -(memory)</td>
</tr>
<tr>
<td><strong>not</strong></td>
<td><strong>not</strong></td>
<td><strong>not</strong></td>
<td><strong>not</strong></td>
</tr>
</tbody>
</table>

- \* usually groups more tightly than +.*
Operator Precedence

Precedence Table

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>(bit-wise and)</td>
</tr>
<tr>
<td>^</td>
<td>(bit-wise exclusive or)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>.and.</td>
<td>(logical and)</td>
</tr>
<tr>
<td>.or.</td>
<td>(logical or)</td>
</tr>
<tr>
<td>.equiv., .nequiv.</td>
<td>(logical comparisons)</td>
</tr>
<tr>
<td>.eqv., .neqv.</td>
<td></td>
</tr>
<tr>
<td>.neq., .eq</td>
<td></td>
</tr>
<tr>
<td>,</td>
<td>(sequencing)</td>
</tr>
</tbody>
</table>

Operators Precedence

- **Precedence** rules specify the order in which operators of different precedence level are evaluated
  - * usually groups more tightly than +
- What is the results of $4 * 5 ** 6$?
- Precedence in Python
  - [http://www.python.org/doc/current/ref/power.html](http://www.python.org/doc/current/ref/power.html)
- Precedence in boolean expression is also very important
  - Pascal's `if A < B and C < D then (*ouch*)`
Operators

**Associativity**

- **Associativity** rules specify the order in which operators of the same precedence level are evaluated
  - + is usually evaluated from left-to-right
- What is the result of $4 \times 5 \times 6$?
- In Fortran, ** associates from right-to-left, as in Math
- In Ada, ** does not associate, so you have to write the previous expression as $4 \times (5 \times 6)$ to obtain the expected answer

Assignment

- The basic operation in imperative language is assignment
  - The side effect of this operation is a change in memory
  - Assignments affect the whole state of the program
- Purely functional language do not have assignment
  - Side effects are not possible
  - Expression in purely functional languages depend only in their referencing environment
- **Expressions** produce values
- **Statements** do not return values, but they have side effects
Reading Assignment

- Scott’s chapter 6
  - Intro
  - Section 6.1 Intro
  - Subsection 6.1.1