Release Mechanics

- Python 2.2 is resting ("pining for the fjords")4maybe2137

Generator Expressions

- Consider: print sum(x**2 for x in range(10))
- Compare:
  - total = 0
    for in *range(10)*:
      total +=
    print total
- Factor out summing algorithm for reuse
- Concentrate on providing input without distractions
- Easily switch between different data processors
  - "accumulator functions"
  - "iterator algebra" a.k.a. pipele932
- Don't build whole input sequence in memory first

Decorators

- Consider (introduced in python 2.2):
  - class C:
        def func(args):
            ...100 lines of body text...
        func = staticmethod(func)
- Problem with this syntax:
  - programmer may forget to add the staticmethod() call
  - reader may miss the staticmethod() call
- Proposed solution:
  - class C:
        DECORATOR SYNTAX func(args

What Is a Decorator?

- A decorator is a (meta-)function:
  - input is a callable or descriptor (implements __get__)
  - output is a callable or descriptor
  - examples: classmethod, staticmethod, (property)
-

Possible Decorator Syntaxes

- C#:
  - prefix object with [decorator, decorator, ...]
- Juneava 1.5 (Tiger):
  - @decorator and/or @decorator(key=value, ...)
- Other proposals from Python developers:
  - *[decorators]*
  - <decorators>
  - [as decorators]
  - other keywords or symbols

Decorator (Ab)uses

- def funcattrs(**kwds):
    def helper(func):

What Now?

- Maybe the ambiguity is acceptable
- Maybe I should learn to accept the community form
- Maybe I should consider the Java 1.5 syntax
- Maybe it's time for another community vote???
- Let's at least have a show of hands here
- Come talk to me afterwards

Python 3.0

- Incompatible changes
  - but no radically different syntax
    d.values
- Library reorganization (less
  ();

Question Time