ALTIS
INVESTMENT MANAGEMENT

Developing financial software

Numerical reliability, lazy initialization, digital signatures

Stefano Taschini, Ph.D.
Sr. Engineer, Investment Analytics

Pycon Italia 3
Firenze, May 8th-10th
http://www.pycon.it/

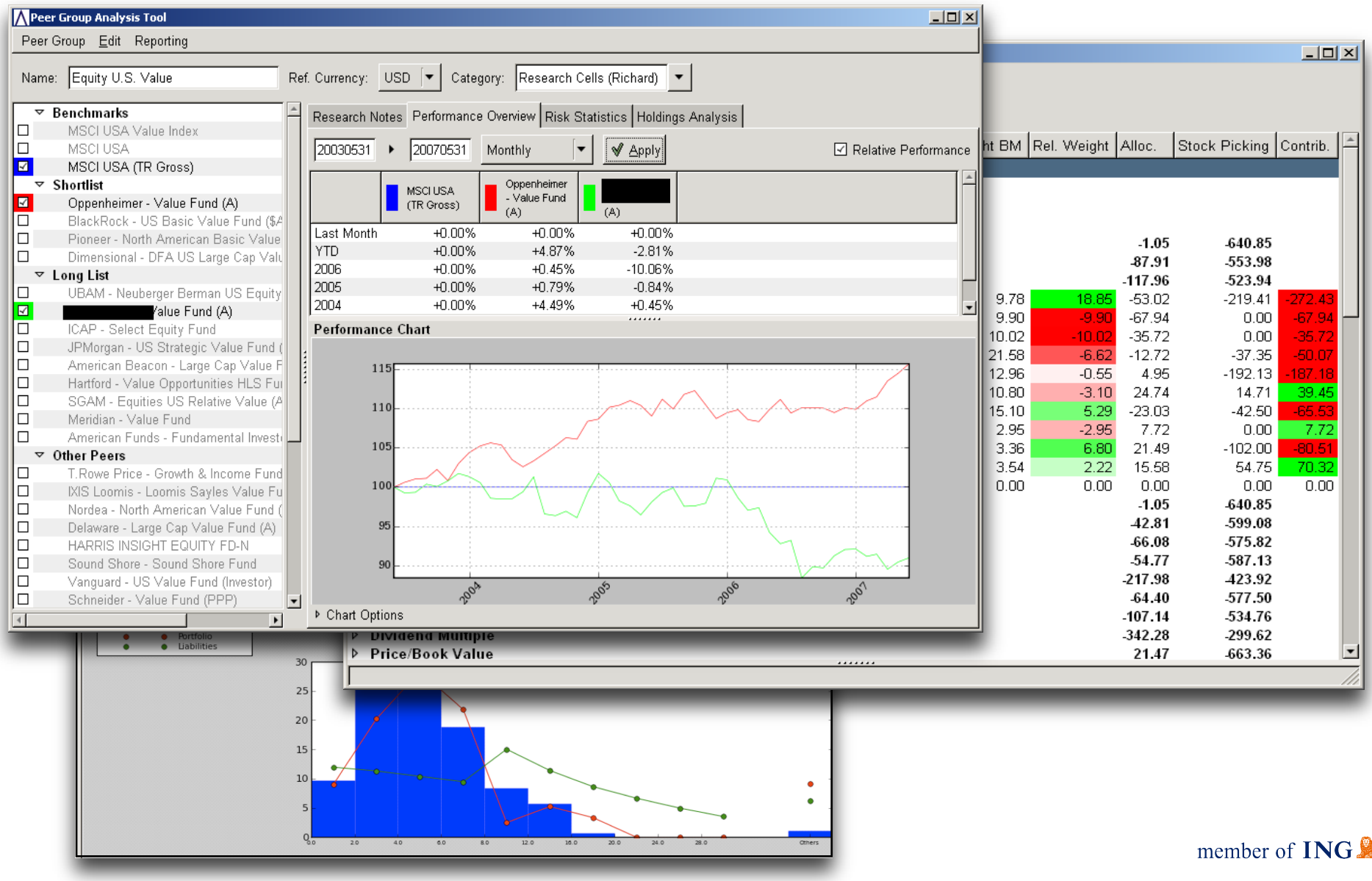
A Tuscan invention

Fibonacci
Liber Abaci
1202

Filippo Calandri
De arimetricha opusculum
Firenze 1491

Fast fwd to today

ALTIS
INVESTMENT MANAGEMENT



member of ING

The company

ALTIS
INVESTMENT MANAGEMENT

A specialist 'Boutique' which is part of ING Investment Management

- Track record since 1999, when team was formed at Morgan Stanley
- The first to develop 'Holdings-Based Analysis' driven by 'Aggregated Risk Management'
- 2004: Independent Specialist, acquired by ING in 2008 as its specialist multi-manager resource
- Manages in excess of €5bn, advises on in excess of €11bn

Offices in Switzerland (9 staff) and the Netherlands (4 staff)

- 12 out of 13 staff are full-time professionals – no marketing or other distractions

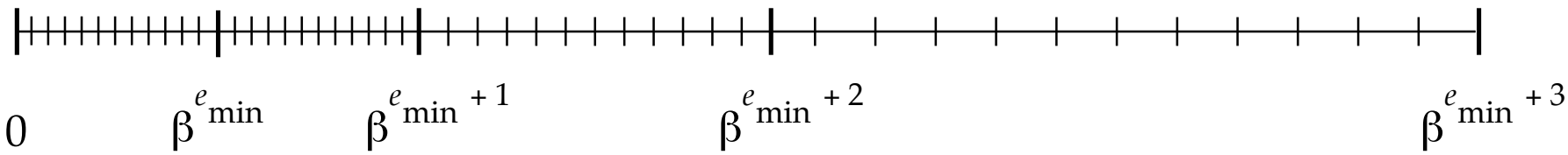
An integrated part of ING's fiduciary offering as part of 'Implemented Client Solutions'

- But operates at arm's length without conflicts of interest as manager selection boutique

Floating-Point numbers

```
>>> 1 + 1.1
2.1000000000000001
```

$$\mathbb{F} = \{-\infty, +\infty\} \cup \{x \in \mathbb{R} \,|\, x = m\beta^e\}$$



Do you really mean it?

ALTIS
INVESTMENT MANAGEMENT

```
>>> 1 + 1.1
2.1000000000000001

>>> 100 + 110
210

>>> def format(x):
...     import re
...     return re.sub('\d\d$', '.\g<0>', repr(x))
>>> format(100 + 110)
'2.10'
```

member of ING

Decimal FP

ALTIS
INVESTMENT MANAGEMENT

```
>>> from decimal import Decimal as D
>>> D('1') + D('1.10')
Decimal('2.10')

>>> from decimal import localcontext
>>> with localcontext() as ctx:
...     ctx.prec = 16
...     D(2 ** 53).log10()
Decimal('15.954589770191100')

>>> import math
>>> math.log10(2 ** 53)
15.954589770191003
```

member of ING

An example

ALTIS
INVESTMENT MANAGEMENT

$$f(x,y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

```
>>> def f(x,y):
...     return (
...         (333.75 - x**2)* y**6 + x**2 *
...             (11* x**2 * y**2 - 121 * y**4 - 2)
...         + 5.5 * y**8 + x/(2*y))

>>> f(77617.0, 33096.0)
1.172603940053178 7
```

member of ING

With decimal FP

ALTIS
INVESTMENT MANAGEMENT

```
>>> def fd(x,y):
...     return (
...         (D('333.75')-x**2)* y**6 + x**2 *
...             (11* x**2 * y**2 - 121*y**4 - 2)
...         + D('5.5') * y**8 + x/(2*y))

>>> fd(D(77617), D(33096))
Decimal('-999999998.827396059946821368 1')

>>> with localcontext() as ctx:
...     for p in [29, 30, 31]:
...         ctx.prec = p
...         print fd(D(77617), D(33096))
100000001.17260394005317863186
20000001.172603940053178631858 8
-999998.827396059946821368141165 1
```

member of ING

Using GMPy

ALTIS
INVESTMENT MANAGEMENT

```
>>> from gmpy import mpf
>>> f(mpf(77617), mpf(33096))
mpf('3.285436508427236398 77e26')

>>> pprint([f(mpf(77617, n), mpf(33096, n)) for n in (53, 100, 150)])
[mpf('3.285436508427236398 77e26'),
 mpf('1.172603940053178631823874167326495690282e0',100),
 mpf('1.172603940053178631823874167326495690281887942030 2e0',150)]
```

member of ING

Rationals

ALTIS
INVESTMENT MANAGEMENT

```
>>> from fractions import Fraction as F
>>> def ff(x,y):
...     return (
...         (F(33375, 100)-x**2)* y**6 + x**2 *
...             (11* x**2 * y**2 - 121*y**4 - 2)
...         + F(55, 10) * y**8 + x/(2*y))
```

member of ING

Rationals

ALTIS
INVESTMENT MANAGEMENT

```
>>> from fractions import Fraction as F
>>> def ff(x,y):
...     return (
...         (F(33375, 100)-x**2)* y**6 + x**2 *
...             (11* x**2 * y**2 - 121*y**4 - 2)
...         + F(55, 10) * y**8 + x/(2*y))

>>> ff(F(77617), F(33096))
Fraction(-54767, 66192)

>>> float(_)
-0.8273960599468214
```

$$f(77617, 33096) = -\frac{54767}{66192} = -0.827396\ldots$$

member of ING

Intervals

ALTIS
INVESTMENT MANAGEMENT

```
>>> from interval import interval
>>> k = interval([0, 1], [2, 3], [10, 15])
```

$$k = [0, 1] \cup [2, 3] \cup [10, 15]$$

```
>>> interval[1, 2]
interval([1.0, 2.0])

>>> interval(1, 2)
interval([1.0], [2.0])

>>> interval(1), interval[1]
(interval([1.0]), interval([1.0]))
```

member of ING

Rounding errors

ALTIS
INVESTMENT MANAGEMENT

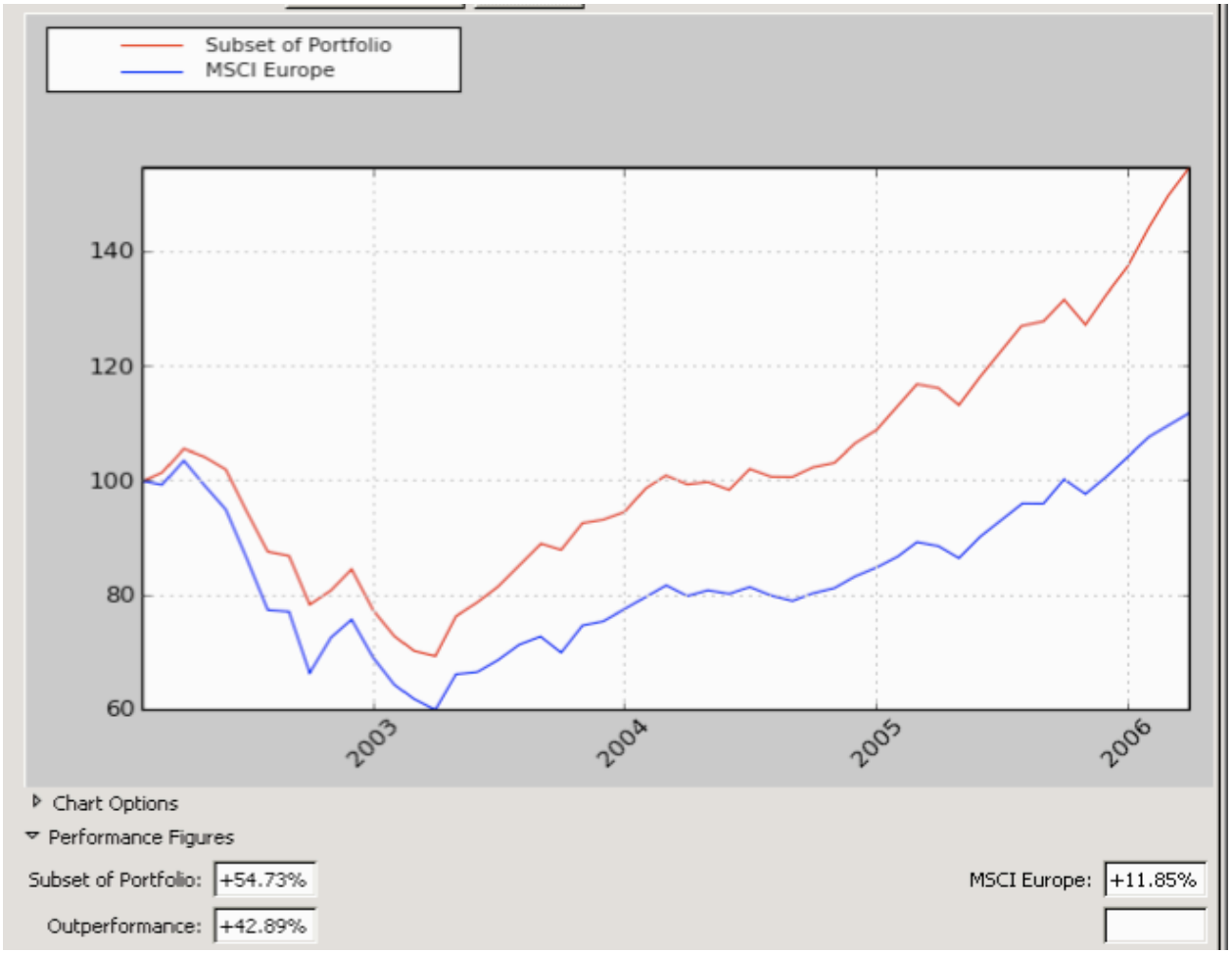
```
>>> interval[10] / interval[3]
interval([3.3333333333333, 3.3333333333333339])

>>> from interval import imath
>>> imath.exp(1)
interval([2.7182818284590451, 2.7182818284590455])

>>> print f(interval(77617.0), interval(33096.0))
interval([-3.54177486215e+21, 3.54177486215e+21])
```

member of ING

References

- http://pypi.python.org/pypi/pyinterval/
- http://conference.scipy.org/slides/
- http://gmpy.googlecode.com/
- D. Goldberg, "What every computer scientist should know about floating-point arithmetic", *ACM Computing Surveys*, vol. 23 (1991), pp. 5–48
  - Republished many times and available on the web.

Before & After

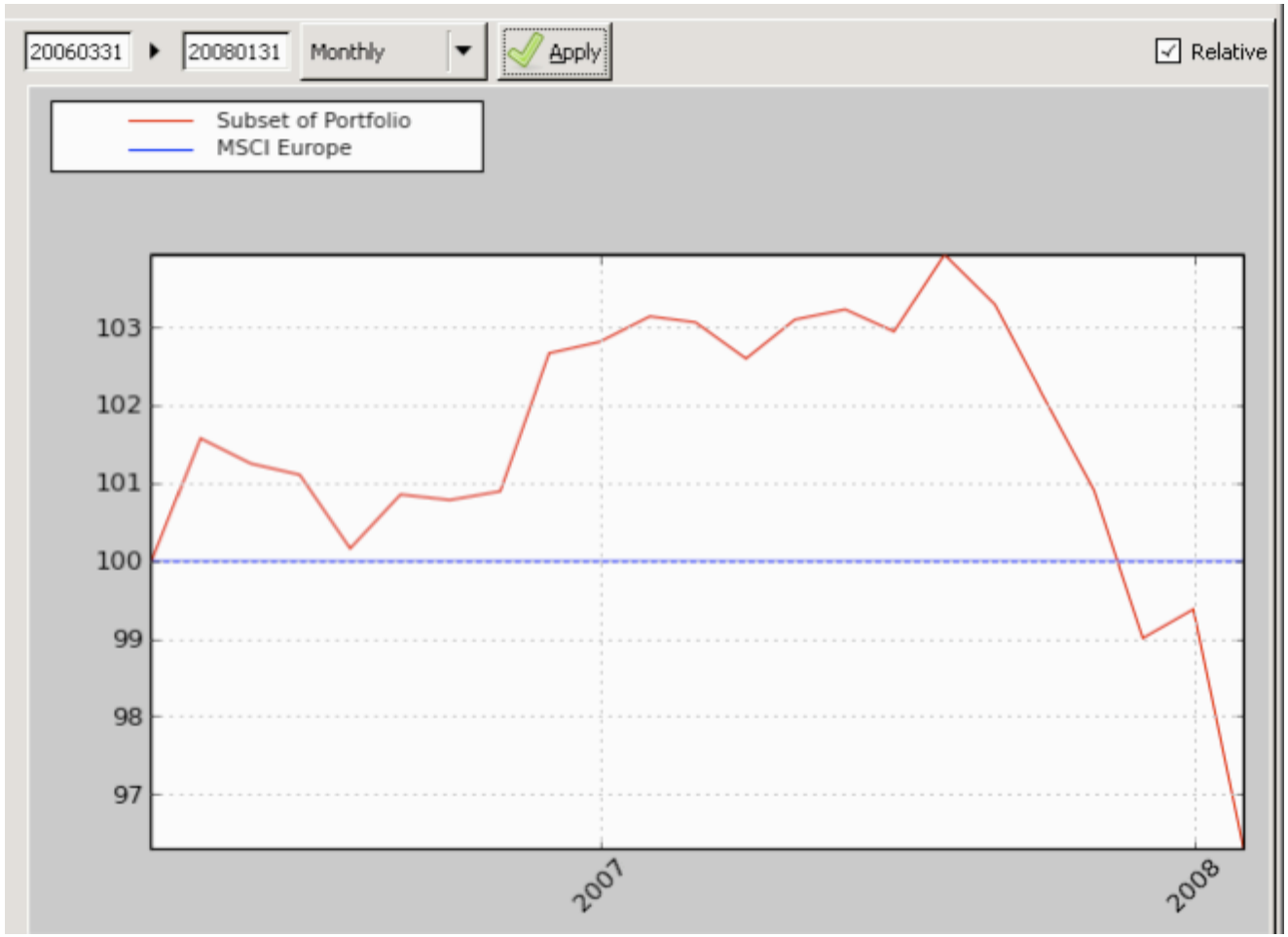


Before & After

ALTIS
INVESTMENT MANAGEMENT

member of ING

Before & After



Lazy Initialization

ALTIS
INVESTMENT MANAGEMENT

- Complexity management
  - 60k sloc in Python
- Lazy Evaluation
  - Evaluate an expression only when it’s needed
- Functional-style programming
  - Immutable state
  - A variable is set only at initialization
- Lazy initialization
  - State is still mutable
  - A variable is initialized only when it’s needed
  - Traversal of dependency DAG.

Goal

ALTIS
INVESTMENT MANAGEMENT

```
>>> class A(object):
...     "A fantastic class"
...     @initializer
...     def x(self):
...         "A very delicate property."
...         return 1

>>> a = A()
>>> a.x
1

>>> a.x = 2
>>> a.x
2

>>> del a.x
>>> a.x
1
```

member of ING

Implementation

ALTIS
INVESTMENT MANAGEMENT

```
>>> class initializer(object):
...     def __init__(self, f):
...         self.f = f
...         self.__doc__ = f.__doc__
...     def __get__(self, obj, type=None):
...         if obj is None:
...             return self
...         v = self.f(obj)
...         setattr(obj, self.f.__name__, v)
...         return v
...     def __repr__(self):
...         return self.__doc__ or 'Undocumented'
```

member of ING

Added bonus

ALTIS
INVESTMENT MANAGEMENT

```
>>> help(A)
Help on class A in module __builtin__:
<BLANKLINE>
class A(object)
 |  A fantastic class
 |
 |  Methods defined here:
 |
 |  x = A very delicate property.
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
<BLANKLINE>
```

member of ING

Deploying a Python Application

ALTIS
INVESTMENT MANAGEMENT

- Components
  - Python interpreter stub + bootstrapping script
  - Python packages and modules
  - Dynamically-linked binary libraries (DLLs)
    - Including Python extensions
  - Data resources (XMLs, pickles, icons...)
- Code signing
  - Guaranteeing the source of the application
  - “Authenticity”, no forgery
  - Requirement of some deployment environments
- MS Windows support for .EXE and .DLL files
  - PE (Portable Executable) file format
  - `#include <winnt.h>`

Solution

ALTIS
INVESTMENT MANAGEMENT

- One-file packaging:
  - Py2exe's `bundle_files` option
  - No unpackaging to temporary directories:
    - Else, almost as effective as signing just the installer.
  - The four component can then be signed all-together:
    - i.e. you guarantee that they belong all together. (CVS vs SVN)

- Loading from one file:
  - Python interpreter stub + bootstrapping script:
    - Taken care by the OS shell + API
  - Python packages and modules
    - zipimporter (stdlib)
  - Dynamically-linked binary libraries (DLLs)
    - Joachim Bauch's MemoryModule
    - Used in py2exe's interpreter stub and zipextimporter
  - Data resources (XMLs, pickles, icons...)
    - Converted to code resources (à la PyQT)
    - Or, embedded into the PE

Conclusions

ALTIS
INVESTMENT MANAGEMENT

- Beware of the numbers you produce
  - FP are leaky abstractions
  - Try either *a priori* reliability assessments or *a posteriori* or both
- Beware of system complexity
  - Adopting and adapting functional techniques
  - Lazy initialization as an interesting tool
- Use real one-file approach when signing code

Certain of the statements contained herein are statements of future expectations and other forward-looking statements. These expectations are based on management's current views and assumptions and involve known and unknown risks and uncertainties. Actual results, performance or events may differ materially from those in such statements due to, among other things, (i) general economic conditions, in particular economic conditions in Altis core markets, (ii) performance of financial markets, including emerging markets, (iii) the frequency and severity of insured loss events, (iv) mortality and morbidity levels and trends, (v) persistency levels, (vi) interest rate levels, (vii) currency exchange rates (viii) general competitive factors, (ix) changes in laws and regulations, (x) changes in the policies of governments and/or regulatory authorities. Altis assumes no obligation to update any forward-looking information contained in this document.

www.altis.ch