Today:
- Python exception handling
- Python plus Fortran: f2py

Next week:
- More Python plus Fortran
- Visualization
- Parallel IPython

Read: Class notes and references
If you try to do something illegal in Python it will generally raise an exception.

```python
>>> x = 0.
>>> y = 1/x
Traceback (most recent call last):
  ZeroDivisionError: float division
```

The exception `ZeroDivisionError` was raised when we tried to divide by 0.

```python
>>> x = "8.0"
>>> y = 1/x
Traceback (most recent call last):
  TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

The exception `TypeError` is raised if we try to divide by a string.
Exceptions in Python

Trying to access a variable not yet set gives:

```python
>>> z = z+1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'z' is not defined
```

If this happens in a longer program, the program will die.

Suppose that if z is not yet defined we want to set z=0 instead of incrementing it. Then we could use:

```python
>>> try:
...     z = z+1
... except:
...     z = 0
... print z
0
```
You can specify different behavior for different exceptions:

```python
x = 0.
try:
    y = 1/x
except ZeroDivisionError:
    print "Can’t divide by 0!"
except TypeError:
    print "Can’t divide by ", type(x)
```

produces:

```
Can’t divide by 0!
```
Exceptions in Python

From `$CLASSHG/codes/python/plotheat2d.py`:

```python
x, y, u = np.loadtxt('heatsoln.txt', unpack=True)

# Solution is plotted on n by n grid so length of
# each vector should be n**2
# Determine n:
try:
    n = int(np.sqrt(len(x)))
except:
    print "Expected len(x) to be a perfect square, 
    "len(x) = ", len(x)
    raise ValueError

X = x.reshape(n,n)
Y = y.reshape(n,n)
U = u.reshape(n,n)
```

R.J. LeVeque, University of Washington
AMath 483/583, Lecture 27, May 27, 2011
Even better:

```python
try:
    fname = 'heatsoln.txt'
    x, y, u = np.loadtxt(fname, unpack=True)
except:
    err_msg = "Could not load data from file %s" % fname
    err_msg += " Did you forget to run the program?"
raise Exception(err_msg)
```
Exceptions in Python

This gives:

In [58]: run plotheat2d.py
---------------------------------------------------------------------------
Exception Traceback (most recent call last)
/Users/rjl/hg/uwamath583s11/codes/python/plotheat2d.py in <module>()
      11 err_msg = "Could not load data from file %s."
      12 + " Did you forget to run the prog
----> 13 raise Exception(err_msg)
      14
      15

Exception: Could not load data from file heatsoln.txt. Did you forget to run the program?
WARNING: Failure executing file: <plotheat2d.py>
NumPy Inf and NaN

Note: Division by zero in NumPy arrays may be allowed, results in special values (NaN = Not a Number):

```python
>>> x = np.linspace(-2., 2., 5)
>>> x
array([-2., -1.,  0.,  1.,  2.])

>>> x/0.
array([-Inf, -Inf,  NaN,  Inf,  Inf])
```

This is useful because often only a few values are “bad”. Note — can set to raise these exceptions:

```python
>>> oldsettings = np.seterr(all = 'raise')
```

To set back to ignoring them:

```python
>>> oldsettings = np.seterr(all = 'ignore')
```
Not-a-Number (NaN)

Some arithmetic operations give undefined results.
The result of such an operation is often replaced by a special value representing NaN.

Examples:

\[ \frac{0}{0} = \text{NaN} \]
\[ 0 \times \text{Infinity} = \text{NaN} \]
Using exceptions in NumPy

Suppose we want to evaluate \( y_i = \sin(x_i)/x_i \) at many points.

If \( x_i = 0 \), must use l’Hôpital’s rule:
\[
\lim_{{x \to 0}} \frac{\sin(x)}{x} = 1
\]

Could program with a loop (very slow in Python!)

\[
x = \text{np.linspace}(-10, 10, 100001)
y = \text{np.ones}(x.\text{shape})
\]

\[
\text{for } i, xi \text{ in enumerate}(x):
\quad \text{if } xi \neq 0:
\quad \quad y[i] = \text{np.sin}(xi) / xi
\]

Takes about 0.94 seconds

Can time using `time.clock()` function.
See `$CLASSHG/codes/python/timing1.py`.

R.J. LeVeque, University of Washington  
AMath 483/583, Lecture 27, May 27, 2011
Better solution: (no loop)

```python
# ignore floating point exceptions:
oldsettings = np.seterr(all='ignore')

z = np.sin(x) / x  # contains a NaN

y = np.where(x != 0, z, 1.)
# sets y[i] = z[j] where x[j] != 0.
# = 1. where x[j] == 0.
```

Takes about 0.011 seconds.

Uses vectorized operation.
Python tools for debugging

- **unittest** module,
- **assert** statements: raise an Exception if what’s asserted is not actually true,
- **pdb**, Python debugger
- Integrated Development Environments (IDEs), e.g. IDLE, Eclipse, Emacs

Demo... See also: Python Debugging section of the notes
Often want to use

Fortran for intensive computations,
Python to provide nice user interface, plot results,
automate a series of runs with different parameters,
do convergence tests as grid size is refined, etc.

Can write data files to disk from Fortran, read into Python,
This is what we’ve done for plotting in homeworks.

Sometimes nice to call Fortran directly from Python.
e.g. LAPACK is used under the hood in NumPy.

f2py provides a wrapper for Fortran code.
Basic idea:

`fortrancode.f90` contains a function or subroutine, e.g. `function f1(x)` that returns a single value.

```
$ f2py -m mymodulename -c fortrancode.f90
```

This creates a binary file `mymodulename.so` that can used as a Python module.

```
>>> from mymodulename import f1
>>> y = f1(3.)
```
f2py — function example

$CLASSHG/codes/f2py/fcn1.f90

function f1(x)
  real(kind=8), intent(in) :: x
  real(kind=8) :: f1
  f1 = exp(x)
end function f1

Then we can do...

$ f2py -m fcn1 -c fcn1.f90
$ python
>>> import fcn1
>>> fcn1.f1(1.)
2.7182818284590451
$CLASSHG/codes/f2py/sub1.f90

subroutine mysub(a,b,c,d)
    real (kind=8), intent(in) :: a,b
    real (kind=8), intent(out) :: c,d
    c = a+b
    d = a-b
end subroutine mysub

Then we can do...

$ f2py -m sub1 -c sub1.f90
$ python
>>> import sub1
>>> y = sub1.mysub(3., 5.)
>>> print y
(8.0, -2.0)

Note: Tuple (c, d) is returned by the Python function.
$CLASSHG/codes/f2py/jacobi1.f90

subroutine iterate(u0,iters,f,u,n)

Takes input array \( u0 \) of length \( n \) and right hand side array \( f \) and produces \( u \) by taking \( \text{iters} \) iterations of Jacobi.

$CLASSHG/codes/f2py/plot_jacobi_iterates.py

# Set \( u = \) initial guess; \( f = \) rhs
for nn in range(nplots):
    u = jacobi1.iterate(u, iters_per_plot, f)
    plt.plot(x, u, 'o-')
    plt.draw()
    time.sleep(.5)
Other wrappers...

- **fwrap**: Improved version of f2py coming soon. Wraps Fortran in C, Cython, Python.
  
  [http://fortrancython.wordpress.com/](http://fortrancython.wordpress.com/)

- **swig**: Connects C and C++ to many other languages
  

- **Cython**: Allows writing C code embedded in Python.
  

- **Jython**: For Java.
  