Debugging
Coursework
hw05 due 3/13
Grade check 3/9–3/17
Question #1

```python
x = np.zeros((3,3))
for i in range(3):
    for j in range(3):
        x[i,j] = i*j + j
```

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 1 2</td>
</tr>
<tr>
<td>1 2 3</td>
<td>0 2 4</td>
<td>0 2 4</td>
</tr>
<tr>
<td>2 4 6</td>
<td>0 4 8</td>
<td>0 3 6</td>
</tr>
</tbody>
</table>
```python
x = np.zeros((3,3))
for i in range(3):
    for j in range(3):
        x[i,j] = i*j + j

A = 
(0 0 0)
(1 2 3)
(2 4 6)

B = 
(0 0 0)
(0 2 4)
(0 4 8)

C = 
(0 1 2)
(0 2 4)
(0 3 6)
```

\[ \star \]
When Things Go Wrong
My code doesn't work.
How do I know it isn’t working?
What do I expect it to do?
What is my code doing instead? Why?
Errors are clues!
A few working definitions:

- **Exceptions**—unusual behavior (although not necessarily unexpected behavior, particularly in Python)
- **Errors**—exceptions which cause the program to be unrunnable (cannot be handled at run time)
- **Traceback**—listing of function calls on the stack at the time the exception arises
- **Bugs**—errors and exceptions, but also miswritten, ambiguous, or incorrect code which in fact runs but does not advertise its miscreancy
Common exceptions

- SyntaxError
- NameError
- TypeError
- ValueError
- IOError
- IndexError
- KeyError
- ZeroDivisionError
- IndentationError
- Exception
Common exceptions

- **SyntaxError**—check missing colons or parentheses
- **NameError**—check for typos, function definitions
- **TypeError**—check variable types (coerce if necessary)
- **ValueError**—check function parameters
- **IOError**—check that files exist
Common exceptions

- **IndexError**—don’t reference nonexistent list elements
- **KeyError**—similar to an IndexError, but for dictionaries
- **ZeroDivisionError**—three guesses...
- **IndentationError**—check that spaces and tabs aren’t mixed
- **Exception**—generic error category
# calculate squares

d = list(range(10))

while i < 10:
    d[i] = d[i] ** 2.0
    i += 1

Which error would this code produce?

A SyntaxError
B IndexError
C ValueError
D NameError
Which of the following would produce TypeError?

A '2' + 2
B 2 / 0
C 2e8 + (1+0j)
D '2' * 2
# in file ‘main.py‘
def do_numerics():
    print(sin(5.0))

do_numerics()
Traceback (most recent call last):
  File "heat_eqn.py", line 87, in <module>
    main()
  File "heat_eqn.py", line 77, in main
    u = set_ic( x,length )
  File "heat_eqn.py", line 38, in set_ic
    u[:, 0] = sin(pi*x/length)**2  # deg C
NameError: name 'u' is not defined

- Read these from end to beginning:
- u in set_ic in main in file heat_eqn.py
Debugging
Flon’s Axiom: “There is not now, nor has there ever been, nor will there ever be, any programming language in which it is the least bit difficult to write bad code.”
This code should find all words whose first two letters are the same:

```python
for line in open( "words.txt" ):
    line = line.strip()
    if len( line ) >= 2:
        a,b = line[1:3]
        if a == b:
            print( line )
```
This code should find all words whose first two letters are the same:

```python
for line in open( "words.txt" ):
    line = line.strip()
    if len( line ) >= 2:
        print( line )
        a,b = line[1:3]
        print( a,b )
        if a == b:
            print( line )
```
This code should find all words whose first two letters are the same:

```python
for line in open( "words.txt" ):
    line = line.strip()
    if len( line ) >= 2:
        a,b = line[0:2]
    if a == b:
        print( line )
```
When do things go wrong?

Three categories of problems:
- before the code runs
- while the code is running
- in the results
Debugging strategies

- Start early.
- Read the problem statement carefully.
- Add print statements.
- Chart the flow of the program.
- Break the program down into functions.
- Document functions before writing them.
- Explain it to someone else.
- Make no assumptions! If your thinking is not precise, your code will not be precise.
- Start over from scratch. Take a fresh look at the problem.
Consider the following series statement for a Bessel function of the first kind,

\[ J_0(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!(m+1)!} \left( \frac{x}{2} \right)^{2m}. \]
What makes a good Python code?

import this
Document your code!

Every function should have a docstring.

def warning(msg):
    '''Display a warning message.'''
    print('Warning: %s' % msg)

Docstrings explain what the function does and what its parameters are.

They always are triple-quoted strings on the first line of the function block.

help(warning)
Use descriptive variable names.

Why do we write comments?
Use descriptive variable names.

Why do we write comments?

For the person who next looks at the code!

```python
x_vals = [0, 0.1, 0.2, 0.3, 0.4]  # meters
faraday = 96485.3328959  # coulombs,
                         # electric charge
```
Use functions to structure code.
This makes code more readable (and debuggable!).
A `main` function lets you organize your program’s logic succinctly.
A `main` function lets you organize your program’s logic succinctly.

We have a special way of writing these so that we can load our code as a module or execute it alone.

```python
def main():
    # your code here

if __name__ == '__main__':
    main()
```
Next steps
Next steps

- hw05 due 3/13
- Grade check 3/9–3/17
- Read for the next class