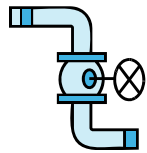


No runs, no drips, & no errors

Error recovery



CS112 Scientific Computation

Department of Computer Science
Wellesley College

Oops!

Even the most meticulous
among us make mistakes

When life's inevitable little
boo boos do occur, *graceful
recovery* is the ticket



Warnings and errors

By now, we are all woefully aware of MATLAB's errors and warnings:

```
>> a = b
```

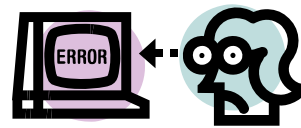
```
??? Undefined function or variable 'b'.
```

```
>> x = 1/0
```

```
Warning: divide by zero.
```

```
x =
```

```
Inf
```



3

Warning!



We can add our own **warning messages** to code that we write:

```
function result = squareRoot(num)
if (num < 0)
    warning(['taking squareroot of negative number ' ...
            num2str(num)]);
end
result = sqrt(num);
```

Calling this function with a negative number:

```
>> n = squareRoot(-3)
```

```
Warning: taking squareroot of negative number -3
```

```
n =
```

```
0 + 1.7321i
```

4

Nicely formatted, useful warnings

Format strings can be used with the `warning` function:

```
function results = analyzeImage (image)
[rows cols] = size(image);
results = zeros(rows, cols, 'uint8');
for row = 2:rows
    for col = 2:cols
        result = <... some computation ...>
        if (result < 0)
            warning('negative result for row %u, col %u: %4.2f',
                    row, col, result);
        end
        results(row, col) = result;
    end
end
end
```

5

Error!



In more dire circumstances we can issue our own **error messages** that halt execution of an M-file

```
function result = analyze(x, y, z)
if (nargin < 2)
    error('At least two inputs are required');
elseif (nargin == 2)
    ...
end
```

```
>> n = analyze(3)
??? Error using ==> analyze
At least two inputs are required
```

6

Nicely formatted, useful *error messages*

Format strings can be used with the `error` function:

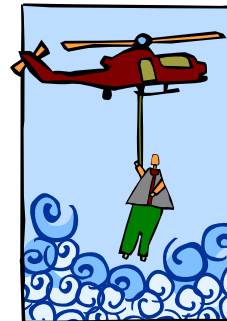
```
function results = analyzeImage (image)
[rows cols] = size(image);
results = zeros(rows, cols, 'uint8');
for row = 2:rows
    for col = 2:cols
        result = <... some computation ...>
        if (result < 0)
            error('negative result for row %u, col %u: %4.2f',
                row, col, result);
        end
        results(row, col) = result;
    end
end
```

7

Graceful recovery

We can prevent a MATLAB error from halting execution of an M-file by handling the error with a `try-catch-end` statement

```
try
    statements to execute
catch
    code to handle errors generated
    by above statements
end
```



8

When bad things happen to good programs

Some events are beyond our control:

```
try
    data = textread('data.txt', '%s');
catch
    disp('file read was not successful');
    data = rand(10,10);
end
result = analyze(data);
```



9

Error analysis using `lasterr` or `lasterror`

```
function result = matrixMultiply (A, B)
try
    result = A * B;
catch
    errmsg = lasterr;
    if (strfind(errormsg, 'Inner matrix dimensions'))
        disp('** Wrong dimensions for matrix multiplication')
    elseif (strfind(errormsg, ...
        'not defined for values of class'))
        disp('** Both arguments must be double matrices');
    end
    result = A;
end
disp('done');
```



10