

Exceptions as flow of control

- In traditional programming languages, one deals with errors by having functions return special values
- Any other code invoking a function has to check whether 'error value' was returned
- In Python, can just raise an exception when unable to produce a result consistent with function's specification
 - `raise exceptionName (arguments)`

Example

```
def getRatios(v1, v2):  
    """Assumes: v1 and v2 are lists of equal length of numbers  
    ReturnsL a list containing the meaningful values of  
        v1[i]/v2[i]"""  
    ratios = []  
    for index in range(len(v1)):  
        try:  
            ratios.append(v1[index]/float(v2[index]))  
        except ZeroDivisionError:  
            ratios.append(float('NaN')) #NaN = Not a Number  
        except:  
            raise ValueError('getRatios called with bad arg')  
    return ratios
```

Using the example

```
try:
    print getRatios([1.0, 2.0, 7.0, 6.0],
                    [1.0, 2.0, 0.0, 3.0])
    print getRatios([], [])
    print getRatios([1.0, 2.0], [3.0])
except ValueError, msg:
    print msg
```

```
[1.0, 1.0, nan, 2.0]
```

```
[]
```

```
getRatios called with bad argument
```

Compare to traditional code

```
def getRatios(v1, v2):  
    """Assumes: v1 and v2 are lists of equal length of numbers  
    ReturnsL a list containing the meaningful values of  
        v1[i]/v2[i]"""  
    ratios = []  
    if len(v1) != len(v2):  
        raise ValueError('getRatios called with bad arg')  
    for index in range(len(v1)):  
        v1Elt = v1[index]  
        v2Elt = v2[index]  
        if (type(v1Elt) not in (int, float))\  
            or (type(v2Elt) not in (int, float)):  
            raise ValueError('getRatios called with bad arg')  
        if v2Elt == 0.0:  
            ratios.append(float('NaN')) #NaN = Not a Number  
        else:  
            ratios.append(v1Elt/v2Elt)  
    return ratios
```

Compare to traditional code

- Harder to read, and thus to maintain or modify
- Less efficient
- Easier to think about processing on data structure abstractly, with exceptions to deal with unusual or unexpected cases