An example of exceptions

- Here is an example of how we can use exceptions to handle unexpected situations in code
- Assume we are given a class list for a subject: each entry is a list of two parts – a list of first and last name for a student, and a list of grades on assignments
- We want to create a new subject list, with name, grades, and a weighted average

A simple start

```
def getSubjectStats(subject, weights):
    return [[elt[0], elt[1], avg(elt[1], weights)]
            for elt in subject]

def dotProduct(a,b):
    result = 0.0
    for i in range(len(a)):
        result += a[i]*b[i]
    return result

def avg(grades, weights):
    return dotProduct(grades, weights)/len(grades)
```

An error if no grades for a student

- If we run this on a list of students, one or more of which don't actually have any grades, we get an error:

```
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    getSubjectStats(test, weights)
  File
"/Users/ericgrimson/Documents/6.00x/subjectCode
.py", line 3, in getSubjectStats
    for elt in subject]
  File
"/Users/ericgrimson/Documents/6.00x/subjectCode
.py", line 12, in avg
    return dotProduct(grades,
weights)/len(grades)
ZeroDivisionError: float division by zero
```

Let's flag the error

```
def avg(grades, weights):
    try:
        return dotProduct(grades, weights)/len(grades)
    except ZeroDivisionError:
        print 'no grades data'
```

Running on some test data yields

```
>>> getSubjectStats(test, weights)
no grades data
[[['fred', 'flintstone'], [10.0, 5.0, 85.0], 15.5],
[['barney', 'rubble'], [10.0, 8.0, 74.0],
13.866666666666667], [['wilma', 'flintstone'], [8.0,
10.0, 96.0], 17.466666666666665], [['dino'], [], None]]
```

Note that last entry now has a 'None' object for the average grade

Or we could change policy

- Suppose we decide that a student with no grades is getting a zero in the class:

```
def avg(grades, weights):
    try:
        return dotProduct(grades, weights)/len(grades)
    except ZeroDivisionError:
        print 'no grades data'
        return 0.0
```

```
>>> getSubjectStats(test, weights)
no grades data
[[['fred', 'flintstone'], [10.0, 5.0, 85.0], 15.5],
[['barney', 'rubble'], [10.0, 8.0, 74.0],
13.866666666666667], [['wilma', 'flintstone'], [8.0,
10.0, 96.0], 17.466666666666665], [['dino'], [], 0.0]]
```

We can handle multiple exceptions

- Suppose some grades are “letter” grades. We can convert them using

```
def convertLetterGrade(grade):
    if type(grade) == int:
        return grade
    elif grade == 'A':
        return 90.0
    elif grade == 'B':
        return 80.0
    elif grade == 'C':
        return 70.0
    elif grade == 'D':
        return 60.0
    else:
        return 50.0
```

We can handle multiple exceptions

```
def avg(grades, weights):
    try:
        return dotProduct(grades, weights)/len(grades)
    except ZeroDivisionError:
        print 'no grades data'
        return 0.0
    except TypeError:
        newgr = [convertLetterGrade(elt) for elt in grades]
        return dotProduct(newgr, weights)/len(newgr)
```

We can handle multiple exceptions

```
>>> getSubjectStats(test1, weights1)
no grades data
[['fred', 'flintstone'], [10.0, 5.0, 85.0,
'D'], 10.0], [['barney', 'rubble'], [10.0,
8.0, 74.0, 'B'], 11.25], [['wilma',
'flintstone'], [8.0, 10.0, 96.0, 'A'],
11.875], [['dino'], [], 0.0]]
```