Lambda expressions – Anonymous Functions

```
f = lambda a, b: a + b          → f: <function <lambda> at ...>
f(1, 2)                         → 3
f('hello ', 'world')            → 'hello world'
```

Really is anonymous, add two function to a list – the functions don't have a name

```
L = []                          → Will be a list of functions

L.append(lambda a, b: a + b)
L.append(lambda ch: ch not in 'aeiouAEIOU')

L[0](1, 2)                      → 3
L[1]('W')                       → True
```

Common use for Lambda Expressions - one-time use functions

Given a list of numbers, extract the evens

```
L = [3, 4, 5,3, 34, 32, 55, 12]
F = filter(lambda x: x % 2 == 0, L)

list(F)                         → [4, 34, 32, 12]
```

Section 4.4, Miller 3rd ed, Calculating Statistics on Data

Sample data:
Earthquake Data for a 7-day period

Day	Number of Earthquakes
Mon	20
Tue	32
Wed	21
Thu	26
Fri	33
Sat	22
Sun	18

Store data is a list

```
quakes = [20, 32, 21, 26, 33, 22, 18]
```

Section 4.4.1, Miller 3rd ed, Simple Dispersion (called range in stats)

Dispersion: A measure of how spread out the data is
maxValue - minValue

`dispersion = max(quakes) - min(quakes)` → dispersion: 15

Put it in a function: see 04-02-my-stats.py for all of these

```
def getDispersion1(L):
    return max(L) – min(L)
```

Write our own max / min functions

```
def getMax(L):
    maxSoFar = L[0]
    for item in L:
        if item > maxSoFar:
            maxSoFar = item
    return maxSoFar
```

```
def getMin(L):
    minSoFar = L[0]
    for item in L:
        if item < minSoFar:
            minSoFar = item
    return minSoFar
```

Re-write getDispersion

```
def getDispersion2(L):
    return getMax(L) – getMin(L)
```

Section 4.5.1, Miller 3rd ed, Mean (called average in stats)

Mean = average = sum / length

```python
def getMean1(L):
    return sum(L) / len(L)
```

Write our own sum / len functions

```python
def getSum(L):
    total = 0
    for num in L:
        total += num
    return total
```

```python
def getLen(L):
    count = 0
    for item in L:
        count += 1
    return count
```

Re-write getMean function

```python
def getMean2(L):
    return getSum(L) / getLen(L)
```

Section 4.5.2, Miller 3rd ed, Median (the middle value)

For *odd length list* - Assumes SORTED list

0	1	2	3	4	5	6
1	2	3	4	100	8000	16000

median = 4

Sort the list and take the middle value
middle Index = len(L) // 2
= 7 // 2
= 3

```
median = L[len(L) // 2]
```

For *even length list* - Assumes SORTED list

0	1	2	3	4	5
2	7	8	12	21	27

(8+12)/2 = 10
median = 10

Sort the list and take (middleL + middleR) / 2
middleL Index = len(L) // 2 - 1
= 6 // 2 - 1
= 3 – 1
= 2

middleR Index = len(L) // 2
= 6 // 2
= 3

To calculate median:
middleL = L[len(L) // 2 – 1]
middleR = L[len(L) // 2]
median = (middleL + middleR) / 2

Algorithm for Median
1. Sort the List
2. If ODD length list
   Return middle item
3. If EVEN length list
   Return (middleL + middleR) / 2

See 04-02-my-stats.py