Returning Tuples for Unpacking

Recall we can loop through a list of tuples and unpack the values within them

In [1]:
```
stock_prices = [('APPL', 200), ('GOOG',400),('MSFT',100)]
```

In [2]:
```
for item in stock_prices:
    print(item)
```

```
('APPL', 200)
('GOOG', 400)
('MSFT', 100)
```

We can individually grab and item in the tuple like so:

In [4]:
```
for ticker, price in stock_prices:
    print(ticker)
```

```
APPL
GOOG
MSFT
```

In [6]:
```
for ticker, price in stock_prices:
    print(price+(0.1*price))
```

```
220.0
440.0
110.0
```

Simlarly, functions often return tuples, to easily return multiple results for later use

In [10]:
```
work_hours = [('Abby',100), ('Billy', 4000), ('Cassie', 800)]
```

The employee of the month function will return both the name and number of hours worked for the top performed *judged by number of hours worked)

In [11]:
```
def employee_check(work_hours):
    # Set some max value to initially beat, like zero hours (set a default value)
    current_max = 0
    # Set some empty value before the loop
    employee_of_month = ''

    for employee, hours in work_hours:
        if hours > current_max:
            current_max = hours
            employee_of_month = employee
        else:
            pass
    # Notice the indentation here
    return (employee_of_month,current_max)
```

In [12]:
```
employee_check(work_hours)
```

Out[12]:
```
('Billy', 4000)
```

```
result = employee_check(work_hours)
```

In [16]:
```
name,hours = employee_check(work_hours)
```

In [17]:
```
result
```

Out[17]:
```
('Billy', 4000)
```

In [18]:
```
name
```

Out[18]:
```
'Billy'
```

In [19]:
```
hours
```

Out[19]:
```
4000
```

Pictured above, we performed tuple unpacking with a function call.

In [ ]:
```

```