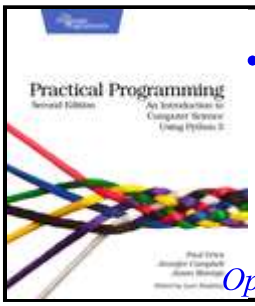
Nested Loops



- Chapter 9.5 Nesting Loops in Loops

Optional reading

Bodies of Loops

The bodies of loops can contain any statements, including other loops. When this occurs, this is known as a *nested loop*.

Here is a nested loop involving 2 `for` loops:

```
for i in range(10, 13):
    for j in range(1, 5):
        print(i, j)
```

Here is the output:

```
10 1
10 2
10 3
10 4
11 1
11 2
11 3
11 4
12 1
12 2
12 3
12 4
```

Notice that when `i` is 10, the inner loop executes in its entirety, and only after `j` has ranged from 1 through 4 is `i` assigned the value `11`.

Example of Nested Loops

```
def calculate_averages(grades):
    ''' (list of list of number) -> list of float

    Return a new list in which each item is the average of the grades in the
    inner list at the corresponding position of grades.

    >>> calculate_averages([[70, 75, 80], [70, 80, 90, 100], [80, 100]])
    [75.0, 85.0, 90.0]
    '''

    averages = []

    # Calculate the average of each sublist and append it to averages.
    for grades_list in grades:

        # Calculate the average of grades_list.
        total = 0
        for mark in grades_list:
            total = total + mark

        averages.append(total / len(grades_list))

    return averages
```

In `calculate_averages`, the *outer* `for` loop iterates through each sublist in `grades`. We then calculate the average of that sublist using a *nested*, or *inner*, loop, and add the average to the accumulator (the new list, `averages`).

Jennifer Campbell • Paul Gries
University of Toronto
