Physics Without Frontiers

The Abdus Salam International Centre for Theoretical Physics

Practical Programming
in Python

Inspired by 'Practical Programming' by Paul Gries, Jennifer Campbell, Jason Montojo

Lecture 9: Summary & Exercises
Repeating Code: Loops

Loops, Iteration, Nested Loops, Controlling Loops

"In theory, practice is simple"
– Trygve Reenskaug

Lecture 9: Summary

In this lecture you learned the following:

- Repeating a block is a fundamental way to control a program's behavior. A for loop can be used to iterate over the items of a list, over the characters of a string, and over a sequence of integers generated by built-in function range.
- The most general kind of repetition is the while loop, which continues executing as long as some specified Boolean condition is true. However, the condition is tested only at the beginning of each iteration. If that condition is never false, the loop will be executed forever.
- The break and continue statements can be used to change the way loops execute.
- Control structures like loops and conditionals can be nested inside one another to any desired depth.

Lecture 9: Exercises

When writing code, only use Python concepts that have been introduced in the lectures already.

Exercise 1:

The variable `celegans_phenotypes` refers to the list

```
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
```

Write a loop that prints out all the values in `celegans_phenotypes`, one per line.

Exercise 2:

The variable `half_lives` refers to the list

```
[87.74, 24110.0, 6537.0, 14.4, 376000.0]
```

Write a loop that prints out all the values in `half_lives`, all on a single line.

Exercise 3:

The variable `whales` refers to the list

```
[5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
```

Write a loop that adds `1` to all the values in `whales` and stores the result in a new list called `more_whales`. The original `whales` list should not be modified.

Exercise 4:

In this exercise, you'll create a nested list and then write code that performs operations on that list.

a. Create a nested list where each element of the outer list contains the atomic number and atomic weight for an alkaline earth metal. The values are beryllium (4 and 9.012), magnesium (12 and 24.305), calcium (20 and 40.078), strontium (38 and 87.62), barium (56 and 137.327), and radium (88 and 226). Assign the list to variable `alkaline_earth_metals`

b. Write a `for` loop to print all the values in `alkaline_earth_metals`, with the atomic number and atomic weight for each alkaline earth metal on a different line.

c. Write a `for` loop to create a new list called `number_and_weight` that contains the elements of `alkaline_earth_metals` in the same order but not nested.

Exercise 5:

The following function doesn't have a docstring or comments. Write enough of both to make it easy for another programmer to understand what the function does and how. Think of a good name for the function. Then compare your solution with those of at least two other people. How similar are they? Why do they differ?

```
def mystery_function(values):
    result = []
    for sublist in values:
        result.append([sublist[0]])
        for i in sublist[1:]:
            result[-1].insert(0, i)
    return result
```

Exercise 6:

In Lecture 9 you saw a loop that prompted users until they typed `quit`. This code won't work if users type `Quit` or `QUIT` – or anything else that is not exactly `quit`. Modify the code such that the loop terminates independent of the capitalization.

Exercise 7:

Consider the following statement, which creates a list of populations of countries in eastern Asia (China, DPR Korea, Hong Kong, Mongolia, Republic of Korea, and Taiwan) in millions:

```
country_populations = [1295, 23, 7, 3, 47, 21]
```

Write a `for` loop that adds up all the values and stores them in variable `total`. (Hint: Give `total` an initial value of zero, and, inside the loop body, add the population of the current country to total.)

Exercise 8:

You are given two lists, `rat_1` and `rat_2`, that contain the daily weights of two rats over a period of ten days. Assume the rats never have exactly the same weight. Write statements to do the following:

a. If the weight of rat 1 is greater than that of rat 2 on day 1, print "Rat 1 weighed more than rat 2 on day 1."; otherwise, print "Rat 1 weighed less than rat 2 on day 1.".

b. f rat 1 weighed more than rat 2 on day 1 and if rat 1 weighs more than rat 2 on the last day, print "Rat 1 remained heavier than Rat 2."; otherwise, print "Rat 2 became heavier than Rat 1."

c. If your solution to the previous exercise used nested if statements, then do it without nesting, or vice versa.

Exercise 9:

Print the numbers in the range 33 to 49 (inclusive).

Exercise 10:

Print the numbers from 1 to 10 (inclusive) in descending order, all on one line.

Exercise 11:

Using a loop, sum the numbers in the range 2 to 22 (inclusive), and then calculate the average.

Exercise 12:

Consider this code:

```
def remove_neg(num_list):
    """
    Remove all negative numbers from num_list.

    >>> numbers = [-5, 1, -3, 0, 2]
    >>> remove_neg(numbers)
    >>> numbers
    [1, 0, 2]
    """
    for num in num_list:
        if num < 0:
        num_list.remove(num)
```

When

```
remove_neg([1, 2, 3, -3, 6, -1, -3, 1])
```

is executed, it produces

```
[1, 2, 3, 6, -3, 1]
```

The for loop traverses the elements of the list, and when a negative value (like -3 at position 3) is reached, it is removed, shifting the subsequent values one position earlier in the list (so 6 moves into position 3). The loop then continues on to process the next item, skipping over the value that moved into the removed item's position. If there are two negative numbers in a row (like -1 and -3), then the second one won't be removed.

Rewrite the code to avoid this problem.

Exercise 13:

Using nested `for` loops, print a right triangle of the character `T` on the screen where the triangle is one character wide at its narrowest point and seven characters wide at its widest point:

```
T
TT
TTT
TTTT
TTTTT
TTTTTT
TTTTTTT
```

Exercise 14:

Using nested `for` loops, print the triangle described in the previous exercise with its hypotenuse on the left side:

```
      T
     TT
    TTT
   TTTT
  TTTTT
 TTTTTT
TTTTTTT
```

Exercise 15:

Redo the previous two exercises using `while` loops instead of `for` loops.

Exercise 16:

Variables `rat_1_weight` and `rat_2_weight` contain the weights of two rats at the beginning of an experiment. Variables `rat_1_rate` and `rat_2_rate` are the rate that the rats' weights are expected to increase each week (for example, 4 percent per week).

a. Using a `while` loop, calculate how many weeks it would take for the weight of the first rat to become 25 percent heavier than it was originally.

b. Assume that the two rats have the same initial weight, but rat 1 is expected to gain weight at a faster rate than rat 2. Using a while loop, calculate how many weeks it would take for rat 1 to be 10 percent heavier than rat 2.