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 5: Summary & Exercises
Making Choices

The Boolean Type, Boolean Operators, Relational Operators, Choosing which Statements to Execute

---

"The best thing about a boolean is even if you are wrong, you are only off by a bit."

– Anonymous

Lecture 5: Summary

In this lecture you learned the following:

- Python uses Boolean values, `True` and `False`, to represent what is true and what isn't. Programs can combine these values using three operators: `not`, `and`, and `or`.
- Boolean operators can also be applied to numeric values. `0`, `0.0`, the empty string, and `None` are treated as `False`; all other numeric values and strings are treated as `True`. It is best to avoid applying Boolean operators to non-Boolean values.
- Relational operators such as "equals" and "less than" relate values and produce a Boolean result.

Relational Operators

Operator	Operation
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`==`	equal to
`!=`	not equal to
`in`	contained in
`is`	object identity

- When different operators are combined in an expression, the order of precedence from highest to lowest is arithmetic, relational, and then Boolean.
- `if` statements control the flow of execution. As with function definitions, the bodies of `if` statements are indented, as are the bodies of `elif` and `else` clauses.

Lecture 5: Exercises

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

Exercise 1:

What value does each expression produce? Verify your answers by typing the expressions into Python.

a. `True and not False`

b. `True and not false` (Notice the capitalization!)

c. `True or True and False`

d. `not True or not False`

e. `True and not 0`

f. `52 < 52.3`

g. `1 + 52 < 52.3`

h. `4 != 4.0`

Exercise 2:

Variables `a` and `b` refer to Boolean values.

a. Write an expression that produces `True` iff both variables are `True`.

b. Write an expression that produces `True` iff `a` is `False`.

c. Write an expression that produces `True` iff at least one of the variables is `True`.

Exercise 3:

Variables `full` and `empty` refer to Boolean values. Write an expression that produces `True` iff at most one of the variables is `True`.

Exercise 4: Exclusive Or

We have learned Python's `or` operator is *inclusive*. That is, `True or True` evaluates to `True`. An "exclusive or" operator would evaluate this expression to `False`, more akin to everyday language.

You want an automatic wildlife camera to switch on if the light level is less than 0.01 lux or if the temperature is above freezing, but not if both conditions are true. (You should assume that function `turn_camera_on` has already been defined.)

Your first attempt to write this is as follows:

```
if (light < 0.01) or (temperature > 0.0):
    if not ((light < 0.01) and (temperature > 0.0)):
        turn_camera_on()
```

A friend says that this is an exclusive `or` and that you could write it more simply as follows:

```
if (light < 0.01) != (temperature > 0.0):
    turn_camera_on()
```

Is your friend right? If so, explain why. If not, give values for `light` and `temperature` that will produce different results for the two fragments of code.

Exercise 5:

We have learned about the built-in function `abs` in Lecture 3. Variable `x` refers to a number. Write an expression that evaluates to `True` if `x` and its absolute value are equal and evaluates to `False` otherwise. Assign the resulting value to a variable named `result`.

Exercise 6:

Write a function named `different` that has two parameters, `a` and `b`. The function should return `True` if `a` and `b` refer to different values and should return `False` otherwise.

Exercise 7:

Variables `population` and `land_area` refer to `float` objects.

a. Write an if statement that will print the population if it is less than 10,000,000.

b. Write an `if` statement that will print the population if it is between 10,000,000 and 35,000,000.

c. Write an `if` statement that will print "Densely populated" if the land density (number of people per unit of area) is greater than 100.

d. Write an `if` statement that will print "Densely populated" if the land density (number of people per unit of area) is greater than 100, and "Sparsely populated" otherwise.

Exercise 8:

Function `convert_to_celsius` from Lecture 3 converts from Fahrenheit to Celsius. Wikipedia, however, discusses eight temperature scales: Kelvin, Celsius, Fahrenheit, Rankine, Delisle, Newton, Réaumur, and Rømer.
Visit http://en.wikipedia.org/wiki/Comparison_of_temperature_scales to read about them.

a. Write a `convert_temperatures(t, source, target)` function to convert temperature `t` from source units to target units, where source and target are each one of "Kelvin", "Celsius", "Fahrenheit", "Rankine", "Delisle", "Newton", "Réaumur", and "Rømer" units.

   Hint: On the Wikipedia page there are eight tables, each with two columns and seven rows. That translates to an awful lot of **`if`** statements – at least 8 * 7 – because each of the eight units can be converted to the seven other units. Possibly even worse, if you decided to add another temperature scale, you would need to add at least sixteen more **`if`** statements: eight to convert from your new scale to each of the current ones and eight to convert from the current ones to your new scale.

   A better way is to choose one canonical scale, such as Celsius.

   Your conversion function could work in two steps: convert from the source scale to Celsius and then from Celsius to the target scale.

b. Now if you added a new temperature scale, how many **`if`** statements would you need to add?

Exercise 9:

Assume we want to print a strong warning message if a pH value is below 3.0 and otherwise simply report on the acidity. We try this if statement:

```
>>> ph = 2
>>> if ph < 7.0:
...     print(ph, 'is acidic.')
... elif ph < 3.0:
...     print(ph, 'is VERY acidic! Be careful.')
...
2 is acidic.
```

This prints the wrong message when a pH of 2 is entered. What is the problem, and how can you fix it?

Exercise 10:

The following code displays a message(s) about the acidity of a solution:

```
ph = float(input('enter the pH level: '))
if ph < 7.0:
    print("It's acidic!")
elif ph < 4.0:
    print("It's a strong acid!")
```

a. What message(s) are displayed when the user enters 6.4?

b. What message(s) are displayed when the user enters 3.6?

c. Make a small change to one line of the code so that both messages are displayed when a value less than 4 is entered.