Birkbeck
UNIVERSITY OF LONDON

Introduction to Programming

Python Lab 6:
Relational Operators and
Boolean Variables

Ping Brennan (p.brennan@bbk.ac.uk)

5 November 2019 or
21 February 2020

Getting Started

- Create a new folder in your disk space with the name **PythonLab6**

- Launch the Python Integrated Development Environment (IDLE) - begin with the **Start** icon in the lower left corner of the screen.

- If you are in a DCSIS laboratory, search using the keyword **Python** and click on **IDLE (Python 3.6 64-bit)**

  A window with the title **Python 3.6.2** should appear. This window is the *Shell*.

Getting Started (2)

- If you are in the ITS laboratory MAL 109, then right mouse click on the **Start** icon in the lower left corner of the screen.

  A list of menu options should appear and click on ***Search***. Type ***Python*** in the search text box at the bottom of the pop-up window. A list of Apps should appear and select

  Python 3.4 IDLE(PythonGUI)

  A window with the title **Python 3.4.3 Shell** should appear. This window is the *Shell*.

- In the *Shell* click on **File**. A drop down menu will appear.

  Click on **New File.** A window with the `title` **Untitled** should appear. This window is the *Editor*.

Getting Started (3)

- In the *Editor*, click on **File**, and then in the drop down menu click on **Save As...** .

  A window showing a list of folders should appear.

  – To search any folder on the list, double click on the folder.

  – Find the folder PythonLab6 and double click on it.

  – In the box **File name** at the bottom of the window

    1. Type `TruthTable.py`
    2. Then click on the button **Save** in the lower right corner of the window.

  The title of the *Editor* should change to show the location of the file `TruthTable.py`.

Objectives of the exercises set

- Understand the use of relational operators

  < <= > >= == !=

  to compare two values, such as two numbers.

Table 1 Relational Operators	
Python	**Description**
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
==	Equal
!=	Not equal

Table 2 Relational Operator Examples	
Expression	**Value**
2 < 2	False
2 <= 3	True
5 > 6	False
5 >= (3-1)	True
5 == 6	False
5 != 6	True

Objectives of the exercises set (2)

- Understand the use of variables of type `bool` that have the value `True` or the value `False`.

  For example,

```
passed = True    # value of variable passed is True

frozen = False   # value of variable frozen is False
```

- `True` and `False` are reserved words in Python.

Objectives of the exercises set (3)

- Understand the use of Boolean operators (`and` , `or`) to combine expressions (i.e. conditions), and the `not` Boolean operator.

- The `and` operator gives the result `True` only when **both** conditions (A, B) are `True`.

A	**B**	**A** `and` **B**
True	True	True
True	False	False
False	True	False
False	False	False

- The `or` operator gives the result `True` if at least **one** of the conditions (A, B) is `True`.

A	**B**	**A** `or` **B**
True	True	True
True	False	True
False	True	True
False	False	False

- The `not` operator is used to invert a condition.

A	`not` **A**
True	False
False	True

Program `TruthTable.py`: Truth Table

- **Question 2: Problem statement**

Write a program that takes as input from the keyboard integer values for three variables `a` , `b` , `c`. Obtain from `a`, `b`, `c` the values of three corresponding variables `p`, `q`, `r` of type `bool`. To obtain these values use statements of the form

```
p = (a != 0)
```

The **right hand side** of the above statement has the value `True` or `False`, depending on the value of `a`. Print out the value of the Boolean valued expression

```
(p and q) or (not r)
```

For example, if `a = 3`, `b = -5`, and `c = 10`, then

Expression (condition)	`p = (a!=0)`	`q = (b!=0)`	`r = (c!=0)`	`(p and q)`	`r`	`(not r)`	`(p and q) or (not r)`
Value	True	True	True	True	True	False	**True**

Program `TruthTable.py`: Truth Table (2)

- **Problem statement** (continued)

  The truth table for the above expression displays the value of the expression for each choice of values for `p`, `q` and `r`. It follows that truth table has **eight** rows. Add code to your program to write out **all** eight rows of the truth table.

  It is **not** necessary to obtain any further input from the key board. Print out a header such as

```
  p     q     r     (p and q) or (not r)
```

  and then focus on printing out each row correctly, for example,

```
True  True  True      True
```

Value of p

Value of q

Value of r

E.g. print out 7 spaces between these two outputs.

Value of the expression: (p and q) or (not r)

The above output shows the first row of the truth table (see page 12).

Note: the | symbol is simply used to show the number of spaces needed to be displayed. Replace each | symbol with a space in the output.

Program `TruthTable.py`: Truth Table (3)

- **Problem solving** - Convert the following pseudo code into a sequence of Python statements in your program.

Inputs

1. Read in the first integer value and store it in the variable `a`.

```
a = int(input("Enter the first integer"))
```

2. Read in the second integer value and store it in the variable `b`.

   Next, read in the third integer value and store it in the variable `c`.

Process the inputs to give the output (result).

3. Write a statement of the form below to obtain from `a` the value of Boolean variable `p`. The Boolean expression `(a != 0)` is `True` only when the value of `a` is not equal to `0`.

```
p = (a != 0)
```

4. Write a statement similar to step 3 to obtain from `b` the value of Boolean variable `q`, i.e.

```
q = (b != 0)
```

   Next, write a similar statement to obtain from `c` the value of Boolean variable `r`.

Output

5. Print out the value of the Boolean valued expression

```
(p and q) or (not r)
```

Program `TruthTable.py`: Truth Table (4)

- **Problem solving** (continued)

6. Create a variable named `spaces` and store seven spaces as follows:

```
spaces = "       "     # chosen seven spaces
```

Output

7. Print out a header such as

```
  p      q      r      (p and q) or (not r)
```

8. Print out the first row of the truth table for the Boolean valued expression `(p and q) or (not r)` as follows:

Inputs

```
p = True                  # assign value True
q = True                  # assign value True
r = True                  # assign value True
```

Process the inputs and then output.

```
print(p, q, r, spaces, (p and q) or (not r))
```

9. Add code similar to step 8 to simply print out the remaining seven rows of the truth table by using a combination of values, `False` and `True`, for the variables `p`, `q` and `r`. **Hint**: see the truth table on page 12.

Program `TruthTable.py`: Truth Table (5)

- Provide a comment at the beginning of the program to explain the purpose of the program together with your name and the date. Save your program to the file `TruthTable.py` and then run it.

Hint: Below shows the truth table for the Boolean valued expression `(p and q) or (not r)` that uses a combination of values, `False` and `True`, for the variables `p`, `q` and `r`.

p	q	r	(p and q)	(not r)	(p and q) or (not r)
True	True	True	True	False	True
True	True	False	True	True	True
True	False	True	False	False	False
True	False	False	False	True	True
False	True	True	False	False	False
False	True	False	False	True	True
False	False	True	False	False	False
False	False	False	False	True	True

Program `Comparison.py`: Comparison

- Create a new Editor for a new file called `Comparison.py`
- **Question 3: Problem statement**

  Write a program that takes as input from the keyboard integer values for three variables `x`, `y`, `z`.

  Construct a single Boolean expression that has the value `True` if exactly two of the variables `x`, `y`, `z` have the same value and that has the value `False` otherwise.

  Test your program using a range of inputs.

  Hint: Experiment with Boolean expressions which solve parts of the problem, and then join these Boolean expressions together using `and` or `or` as appropriate.

  For example the Boolean valued expression

```
x == y
```

  has the value `True` if `x` has the same value as `y`, otherwise it has the value `False`.

Program `Comparison.py`: Comparison (2)

- **Problem solving** - Convert the following pseudo code into a sequence of Python statements in your program.

Inputs

1. Read in the first integer and store it in the variable `x`

```
x = int(input("Enter the first integer: "))
```

2. Read in the second integer and store it in the variable `y`.
3. Read in the third integer and store it in the variable `z`.

Process the inputs and then output the result.

4. Use the Boolean expressions below to solve parts of the problem (three different cases to consider):

```
((x == y) and (x != z))    # case 1
((x == z) and (y != z))    # case 2
((y == z) and (y != x))    # case 3
```

   Use the Boolean operator **`or`** to combine **all** three Boolean expressions above into **a single Boolean expression**, and print out the value of this expression.

Program `Comparison.py`: Comparison (3)

- Provide a comment at the beginning of the program to explain the purpose of the program together with your name and the date.
- Save your program to the file `Comparison.py` and then run it.
- For example, if

```
x = 1
y = 1
z = 3
```

Then the Boolean expression

```
(x == y) and (x != z)
```

gives a `True` value.

Supplementary Questions for Private Study

- The laboratory worksheet contains supplementary questions in section 4 for private study.

- You are encouraged to complete the supplementary questions at home, or in the laboratory if you have time after completing questions 2 to 3.

Appendix A Examples on using the Boolean operators: `and` , `or` , `not`

- Example 1 shows the use of the Boolean operator `and`

```
A = (5 < 10)          # yields True
B = (10 > 3)          # yields True
result = (A and B)    # yields True
```

- Example 2 shows the use of the Boolean operator `or`

```
A = (5 < 10)          # yields True
B = (10 < 3)          # yields False
result = (A or B)     # yields True
```

- Example 3 shows the use of the Boolean operator `not`

```
A = (5 < 10)          # yields True
result = not A        # yields False
```