Introduction to Computer Programming

1.3 Operators

University of BRISTOL

Comparison Operators

Comparison operators (==, !=, <, > ....) compare values (operands) and return a *Boolean* value: `True or False`

Commonly used comparison operators:

$==$	Equality
$!=$	Inequality
$>$	Greater than
$<$	Less than
$>=$	Greater than or equal to
$<=$	Less than or equal to

Examples

In [18]:
```
print(10 < 9)
```
```
False
```

In [2]:
```
print(15 < 20)
```
```
True
```

In [1]:
```
print(20 <= 20)
```
```
True
```

In [4]:
```
a = 1.0
b = 1
```

What is the value of `c` ?

In [5]:
```
c = a == b
print(c)
```
```
True
```

What is the value of `d` ?

In [6]:
```
d = type(a) == type(b)
print(d)
```
```
False
```

How could we change the value of `d` by changing `a` or `b` ?

Identity Operators

If two variables are *equal* this does not imply that they are *identical*.

`is` : `True` if the operands are identical
`is not` : `True` if the operands are not identical

In [19]:
```
a = 1.0
b = 1

print(a is b)
print(a is not b)
```
```
False
True
```

Logical Operators

Comparison operators compare two operands.

Logical operators:

- Compare Boolean `True` or `False` *operands* (e.g. outcomes of two *comparison operations*) to form logic statements.
- Outout a *single* `True` / `False` (boolean) value.

```
and
or
not
```

```
x and y
```

Output:
`True` if statement `x` **and** statement `y` *both* `True`.
Otherwise `False`.

Process:
Return `x` if its Boolean value is `False`; otherwise, return `y`.

```
x or y
```

Output:
`True` if statement `x` **or** statement `y` `True`.
Otherwise `False`.

Process:
Return `x` if its Boolean value is `True`; otherwise, return `y`.

Examples:

$10 < 9$ `False`

$20 <= 20$ `True`

In [24]:
```
print(10 < 9 and 20 <= 20)

```

```
False
```

In [25]:
```
print(10 < 9 or 20 <= 20)

```
```
True
```

A word of warning! : This does *not* mean "Is either `a` or `b` less than `c` ?"

In [43]:
```
a, b, c = 0, 1, 7

print(a or b < c)
```
```
True
```

`x or y` : Return `x` if its Boolean value is `True` ; otherwise, return `y` .

"Return `a` if its Boolean value is `True` ; otherwise, return `b < c` ."

In Python, numerical value 0 has the Boolean value `False` , and *all other numbers* have the Boolean value `True` .

"Is either `a` or `b` less than `c` ?" can be expressed using:

In [14]:
```
a, b, c = 0, 8, 7

print(a < c or b < c)
```
```
True
```

Example

What will be output?:

`print(b and a < c)`

(Hint: `x and y` : Return `x` if its Boolean value is `False` ; otherwise, return `y` .)

In [56]:
```
a, b, c = 1, -1, 7

```

In Python, the `not` operator negates the Boolean value of a statement, e.g.:

In [53]:
```python
a = 12

print(a < 0)

print(not a < 0)

```

```
False
True
```

Operator Precedence

1. Parentheses
2. Arithmetic operators (top to bottom)
   `**` Exponent
   `/` , `*` , `//` , `\%` Division, multiplication, floor division, modulo (evaluated left to right)
   `+` , `-` Addition, subtraction (evaluated left to right)
3. Comparison operators: `<` , `<=` , `>` , `>=` , `!=` , `==` (evaluated left to right)
4. Assignment operators `=` , `/=` , `*=` , `//=` , `\%=` , `+=` , `-=` ....
5. Identity operators `is` , `is not`
6. Logical `not`
7. Logical `and`
8. Logical `or`

Example

Write a program, using comparison and logical operators, that answers a question based on the current time of day:

> **Is it lunchtime?**
> `True` if current time is between lunch start and end times.
> `False` if not.

In [47]:
```
# Variables
t = 14.30   # Current time
ls = 13.00  # Time lunch starts
le = 14.00  # Time lunch ends

```

```
False
```

Are there any other ways we could write the expression for lunchtime using the operators we have studied so far?

Best Practises - Comments

As shown in this program, you should use comments to document your code.

When you or someone else reads a comment, they should be able to easily understand the code the comment applies to and how it functions within the rest of the program.

Points for adding comments to code:

- Use complete sentences, starting with a capital letter.
- Limit the total line length to 79 characters (vertical line in Spyder editor window).
- Don't use comments to state the obvious
  e.g. `x = y ** 2 # Assign x the value y squared`

Stacking Comparison Operators

Extract from example program:

```
lunch = t >= ls and t < le
```

We can rewrite *stacking* the comparison operators:

```
ls <= t < le
```

Summary

- Every variable has a type (`int` , `float` , `string` ....) which is automatically assigned when the variable is created.
- **Arithmetic operators** (+, -, /, * ....)
  Used with numeric values to perform mathematical operations (behave differently with strings).
- **Comparison operators** (==, !=, <, > ....)
  Compare two *operands*.
  Output is a *Boolean* (True or False) value.
  Comparison operators can be stacked e.g. `x < y <= z`
- **Identity operators** (`is` , `is not` ....)
  Checks if two *operands* are identical.
  Outout is a *Boolean* (True or False) value.
- **Logical operators** (`and` , `or`)
  Compare Boolean `True` or `False` *operands* (e.g. outcomes of two *comparison operations*) to form logic statements.
  Outout is a *Boolean* (True or False) value.
  Logical `not` operator returns the inverse Boolean value of an operand.
- **Assignment operators** (+=, -=, /= ....)
  Reassign the value of a variable.

Demos

Example

Write a program that:

1. creates 3 variables, `a` , `b` and `c` , with numerical values
2. outputs a statement that tells the user if the values include *any* negative numbers.

In [49]:

```
False
```

Example

Write a program that answers *two* questions based on the current time of day:

> **Is it lunchtime?**
> `True` if time between lunch start and end times.
> `False` if not.

> **Is it time for work?**
> `True` if time between work start and end times **and not** lunchtime.
> `False` if not.

In [51]:

```
False True
```

Are there any other ways we could have written the expressions for `lunch` or `work`?

In [ ]: