Introduction to Computer Programming

1.3 Operators

Comparison Operators

Comparison operators (==, !=, <, > ....) compare values 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 [39]:

```
A = 1.0

B = 2

C = type(A) == type(B)

print(C)
```

```
False
```

Logical Operators

Comparison operators compare two values.

Logical operators combine *multiple* expressions/variables with `True` / `False` (boolean) values and outout a *single* `True` / `False` (boolean) value.

Logical operators:
`and`
`or`
`not`

`X and Y`

Output:
`True` if statement `X` **and** statement `Y` *both* true.
Otherwise `False` .

`X or Y`

Output:
`True` if statement `X` **or** statement `Y` true.
Otherwise `False` .

Examples:

$10 < 9$ `False`

$20 <= 20$ `True`

In [3]:

```
print(10 < 9 and 20 <= 20)
```

```
False
```

In [4]:

```
print(10 < 9 or 20 <= 20)
```

```
True
```

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

In [6]:

```
a = 12

print(a < 0)

print(not a < 0)
```

```
False
True
```

Operator Precedence

1. Parentheses
2. Arithmetic operators (top to bottom)

`**`	Exponent__
`*` , `/` , `//` , `\%`	Multiplication, division, floor division, modulo (evaluated left to right)
`+` , `-`	Addition, subtraction (evaluated left to right)

3. Comparison operators: `<` , `<=` , `>` , `>=` , `!=` , `==` (evaluated left to right)
4. Logical `not`
5. Logical `and`
6. 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 [41]:

```
# Variables
time = 12.00          # current time
lunch_starts = 13.00  # time lunch starts
lunch_ends =   14.00  # time lunch ends


# Logical and comparsion operators to find lunchtime True / False value
lunchtime = time >= lunch_starts and time < lunch_ends


print("Is it lunchtime?", lunchtime)
```

```
Is it lunchtime? False
```

If we change the value of `time` , the program output changes.

Stacking Comparison Operators

Extract from example program:

```
lunchtime = time >= lunch_starts and time < lunch_ends
```

We can rewrite *stacking* the comparison operators:

```
time >= lunch_starts and time < lunch_ends
```

is the same as

```
lunch_starts <= time < lunch_ends
```

Summary

- **Arithmetic operators** (+, -, /, * ....)
  Used with numeric values to perform mathematical operations (behave differently with strings).
- **Comparison operators** (==, !=, <, > ....)
  Compare two *variables*.
  Outcome of a comparison is a *Boolean* (True or False) value.
- **Logical operators** (`and` , `or`)
  Compare Boolean `True` or `False` values (e.g. outcomes of two *comparison operations*) to form logic statements.
  Outcome of a logical operation is a *Boolean* (True or False) value.
  Logical `not` operator returns the inverse Boolean value of a comparison.

In-class Demos

Example 1:
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 [43]:

```
A, B, C = 1, -1, 7

answer = A<0 or B<0 or C<0

print('negatives?', answer)
```

```
negatives? True
```

Example 2:
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 [45]:

```
t = 13.30
Ls = 13.00
Le = 14.00
Ws = 8.00
We = 17.00

lunchtime = Ls <= t < Le

work_time = Ws <= t < We and not lunchtime

print(lunchtime)
print(work_time)
```

```
True
False
```

In [ ]: