Enkelvoudige variabelen - Booleans

Comparators

Een boolean variabele kan maar twee waarden aannemen (denk aan binaire getallen)

In [1]:
```
print(f"A boolean is either %s or %s" %(bool(1),bool(0)) )
```

```
A boolean is either True or False
```

Je kunt een boolean gebruiken als vlag

In [2]:
```
writePermission = False
```

Een boolean waarde volgt vaak uit een vergelijking

Gelijke of ongelijke waarde

In [3]:
```
20 == 20
```

Out[3]: True

In [4]:
```
"apples" == "apples"
```

Out[4]: True

In [5]:
```
20 != 30
```

Out[5]: True

In [6]:
```
"apples" != "peers"
```

Out[6]: True

Dit geldt uiteraard ook voor de booleans zelf

In [7]:
```
True == True
```

Out[7]: True

In [8]:
```
True != False
```

Out[8]: True

Wees bij het vergelijken van Strings alert op verschillen in syntax

In [9]:
```
"grey" != "gray"
```

Out[9]: True

In [10]:
```
"Apples".lower() == "apples"
```

Out[10]: True

In [11]:
```
"peers ".rstrip() == "peers"
```

Out[11]: True

Groter of kleiner

In [12]:
```
20 < 30
```

Out[12]: True

In [13]:
```
30 > 20
```

Out[13]: True

In [14]:
```
len("apples") > len("peers")
```

Out[14]: True

In [15]:
```
20 <= 20
```

Out[15]: True

In [16]:
```
len("apples") >= len( "apples ".rstrip() )
```

Out[16]: True

In [17]:
```
2 * 20 > 20
```

Out[17]: True

Gebruik haakjes als dit de leesbaarheid van de code verbetert

In [18]:
```
(2 * 20) > 20
```

Out[18]: True

Exercise 4

Gebruik conversie naar numerieke waarden om aan te tonen dat True > False

In [19]:
```
# Oplossing
(2*40) > 60
```

Out[19]: True

De Boolean waarden True en False moeten beiden worden omgezet naar een numerieke waarde zodat die waarden met elkaar kunnen worden vergeleken

Logische operatoren

De operator *and* genereert alleen True als beide booleans True zijn

In [20]:
```
True and ("apples" == "apples" )
```

Out[20]: True

omdat bovenstaande expressie opnieuw een boolean oplevert, kun je ketting verder uitbreiden

In [21]:
```
True and ("apples" == "apples" ) and ( (2 * 20) > 20 )
```

Out[21]: True

Een **conjunctie** is een *and-chain* van *booleaanse expressies*

Exercise 5

Maak een and-chain met variabelen *p*, *q* en *r* in de volgorde die het snelst resultaat geeft.

In [22]:
```
p = True
q = True
r = False
```

In [23]:
```
# Oplossing
p == q == r
```

Out[23]: False

Bij een and-chain zijn de variabelen met elkaar verbonden via de and-operator

De operator *or* genereert True zodra een van de booleans True is

In [24]:
```
True or ("apples" == "pears" )
```

Out[24]: True

In [25]:
```
True or ( "apples" != "peers" ) or ( 20 >= 30 )
```

Out[25]: True

Een **disjunctie** is een *or-chain* van *booleaanse expressies*

Exercise 6

Maak een or-chain met variabelen *p*, *q* en *r* in de volgorde die het snelst resultaat geeft.

In [26]:
```
p = False
q = True
r = (p and q)
```

In [27]:
```
p or q
```

Out[27]:
```
True
```

De variabele r ontbreekt nog in de or-chain

De operator *not* genereert ontkracht een booleaanse expressie

In [28]:
```
toBe = True
print(f"{toBe} or {not toBe}, that's the question..")
```

```
True or False, that's the question..
```

In [29]:
```
notToBe = False
print(f"{notToBe} or {not notToBe}, that's the question..")
```

```
False or True, that's the question..
```

Een **negatie** is de ontkenning van een *booleaanse expressies*

Vereenvoudig waar mogelijk

In [30]:
```
"apples" == "peers"
```

Out[30]:
```
False
```

in plaats van

In [31]:
```
not ("apples" != "peers")
```

Out[31]:
```
False
```