while-lussen (loop)

De while-instructie in Python is een van de meest algemene manieren om iteratie uit te voeren. Een while-instructie zal herhaaldelijk een enkele instructie of een groep instructies uitvoeren als de voorwaarde waar is. De reden dat het een 'lus (loop)' wordt genoemd, is omdat de code-instructies **herhaaldelijk** worden doorlopen totdat niet langer aan de voorwaarde wordt voldaan.

Het algemene formaat van een while-lus is:

```
while test:
    code statements
else:
    final code statements
```

Laten we eens kijken naar een paar eenvoudige while-lussen in actie.

```
x = 0

while x < 10:
    print('x is currently: ',x)
    print(' x is still less than 10, adding 1 to x')
    x+=1
```

```
x is currently:  0
 x is still less than 10, adding 1 to x
x is currently:  1
 x is still less than 10, adding 1 to x
x is currently:  2
 x is still less than 10, adding 1 to x
x is currently:  3
 x is still less than 10, adding 1 to x
x is currently:  4
 x is still less than 10, adding 1 to x
x is currently:  5
 x is still less than 10, adding 1 to x
x is currently:  6
 x is still less than 10, adding 1 to x
x is currently:  7
 x is still less than 10, adding 1 to x
x is currently:  8
 x is still less than 10, adding 1 to x
x is currently:  9
 x is still less than 10, adding 1 to x
```

Merk op hoe vaak de print-statements voorkwamen (occurred) en hoe de while-

lus doorging totdat aan de True-voorwaarde werd voldaan, wat eenmaal x==10 voorkwam. Het is belangrijk op te merken dat zodra dit gebeurde, de code stopte. Laten we eens kijken hoe we een else-statement kunnen toevoegen:

```
x = 0

while x < 10:
    print('x is currently: ',x)
    print(' x is still less than 10, adding 1 to x')
    x+=1

else:
    print('All Done!')
```

```
x is currently:  0
 x is still less than 10, adding 1 to x
x is currently:  1
 x is still less than 10, adding 1 to x
x is currently:  2
 x is still less than 10, adding 1 to x
x is currently:  3
 x is still less than 10, adding 1 to x
x is currently:  4
 x is still less than 10, adding 1 to x
x is currently:  5
 x is still less than 10, adding 1 to x
x is currently:  6
 x is still less than 10, adding 1 to x
x is currently:  7
 x is still less than 10, adding 1 to x
x is currently:  8
 x is still less than 10, adding 1 to x
x is currently:  9
 x is still less than 10, adding 1 to x
All Done!
```

break, continue, pass

We kunnen de instructies break, continue en pass gebruiken in onze loops om extra functionaliteit toe te voegen voor verschillende gevallen. De drie statements worden gedefinieerd door:

```
break: Breaks out of the current closest enclosing loop.
continue: Goes to the top of the closest enclosing loop.
pass: Does nothing at all.
```

Denkend aan break en continue statements, ziet het algemene formaat van de

while-lus er als volgt uit:

```
while test:
    code statement
    if test:
        break
    if test:
        continue
else:
```

break en continue statements kunnen overal in de body van de lus verschijnen, maar we zullen ze meestal verder genest plaatsen in combinatie met een if statement om een actie op basis van een voorwaarde.

Laten we doorgaan en enkele voorbeelden bekijken!

```
x = 0

while x < 10:
    print('x is currently: ',x)
    print(' x is still less than 10, adding 1 to x')
    x+=1
    if x==3:
        print('x==3')
    else:
        print('continuing...')
        continue
```

```
x is currently:  0
 x is still less than 10, adding 1 to x
continuing...
x is currently:  1
 x is still less than 10, adding 1 to x
continuing...
x is currently:  2
 x is still less than 10, adding 1 to x
x==3
x is currently:  3
 x is still less than 10, adding 1 to x
continuing...
x is currently:  4
 x is still less than 10, adding 1 to x
continuing...
x is currently:  5
 x is still less than 10, adding 1 to x
continuing...
x is currently:  6
 x is still less than 10, adding 1 to x
```

```
continuing...
x is currently:  7
 x is still less than 10, adding 1 to x
continuing...
x is currently:  8
 x is still less than 10, adding 1 to x
continuing...
x is currently:  9
 x is still less than 10, adding 1 to x
continuing...
```

Merk op hoe we een afgedrukte instructie hebben als x==3, en een wordt afgedrukt terwijl we verder gaan door de buitenste while-lus. Laten we een keer pauzeren x == 3 en kijken of het resultaat klopt:

```
x = 0

while x < 10:
    print('x is currently: ',x)
    print(' x is still less than 10, adding 1 to x')
    x+=1
    if x==3:
        print('Breaking because x==3')
        break
    else:
        print('continuing...')
        continue
```

```
x is currently:  0
 x is still less than 10, adding 1 to x
continuing...
x is currently:  1
 x is still less than 10, adding 1 to x
continuing...
x is currently:  2
 x is still less than 10, adding 1 to x
Breaking because x==3
```

Merk op dat het andere else statement niet werd bereikt en dat doorgaan nooit werd afgedrukt!

Na deze korte maar eenvoudige voorbeelden zou u zich op uw gemak moeten voelen bij het gebruik van while-statements in uw code.

** Een woord van waarschuwing (caution) echter! Het is mogelijk om een oneindig (infiniet) lopende lus te maken met while statements. Bijvoorbeeld:**

```
# VOER DEZE CODE NIET UIT!!!! DO NOT RUN THIS CODE!!!!
while True:
```

```
print("I'm stuck in an infinite loop!")
```

Een korte opmerking: als je *deed* de bovenstaande cel hebt uitgevoerd, klik dan op het Kernel-menu hierboven om de kernel opnieuw te starten!