# Python Cheat Sheet

## 1 Gegevens

- `str`
  Tekst, begint en eindigt met aanhalingstekens
  vb. `"Dit is een tekst"`

- `int`
  Gehele getallen (getallen zonder komma)
  vb. `47`

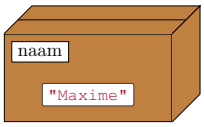
- `float`
  Getallen met komma
  vb. `3.14`

- `bool`
  Logische waarden
  `True` of `False`

- Gegevens omvormen van het de ene soort in de andere:
  `str(_)`, `int(_)`, `float(_)`, `bool(_)`

## 2 Variabelen

- Een variabele is een doosje waarin we gegevens kunnen opslaan.




- `mijnVariabele = <waarde>`
  Een nieuwe variabele maken we met het isgelijkteken `=`. De naam van de variabele komt voor het isgelijkteken en de waarde die we er in willen stoppen komt er achter.

- `mijnVariabele = <nieuweWaarde>`
  We kunnen een bestaande variabele ook een nieuwe waarde geven. Dit doen we net op dezelfde manier zoals we een nieuwe variabele maken. Maar nu bestaat de variabele al en wordt de waarde in het doosje gewoon vervangen.

- `print(mijnVariabele)`
  De waarde in een variabele kan je gebruiken door de naam van de variabele te typen op de plaats waar je deze waarde nodig hebt. Deze waarde kan je dan gebruiken zoals je dat met een gewone waarde zou doen.

- `mijnVariabele = None`
  Moet het doosje leeg zijn? Dan gebruik je de waarde `None`. In python geeft de speciale waarde `None` aan dat het doosje leeg is.

## 3 Voorwaardelijk uitvoering

- `if voorwaarde:`
  Een `if`-statement laat je toe om een blok code enkel uit te voeren als een bepaalde voorwaarde resulteert tot `True`. Het `if`-statement is altijd het begin van een `if ... elif ... else`-statement.

- `elif andereVoorwaarde:`
  Na een `if`-statement kan je meerdere `elif`-statements toevoegen als je wil. Een `elif`-statement laat je toe om een blok code enkel uit te voeren als de bijhorende voorwaarde `True`is en alle voorgaande voorwaarden `False`zijn. Een `if`-statement hoeft niet altijd gevolgd te worden door een `elif`-statement als je dat niet nodig hebt.

- `else:`
  Tot slot laat het `else`-statement je toe om een blok code enkel uit te voeren als al de voorgaande voorwaardes allemaal `False`zijn. Dit zijn de voorwaardes van het `if`-statement en alle eventuele `elif`-statements. Net zoals een `elif`-statement, moet je het `else`-statement ook niet altijd gebruiken. Maar als je het `else`-statement gebruikt hebt, dan moet het wel altijd het laatste statement zijn in een `if ... elif ... else`-statement.

## 4 Voorwaardelijk lus

- `while voorwaarde:`
  Een `while`-lus gebruik je om een blok code te blijven herhalen. De uitvoering stopt pas tot de voorwaarde evalueert tot `False`.

- `break`
  Het `break`-statement stopt een lus meteen. En dat meteen op de plaats waar het `break`-statement zich bevindt. Een nieuwe herhaling wordt niet meer uitgevoerd.

- `continue`
  De resterende uitvoering van de blok code wordt overgeslagen vanaf waar het `continue`-statement wordt uitgevoerd. In de plaats wordt meteen een nieuwe herhaling opgestart, beginnend met een volgende evaluatie van de voorwaarde.

## 5 Herbruikbare blokken

- `def functienaam(): ...`
  Met het `def`-statement definiër je een functie. De code in de functie wordt niet meteen uitgevoerd, maar pas wanneer je deze functie aanroept.

- `functienaam()`
  Door de functienaam te combineren met haakjes, roep je de functie aan. Nu wordt de code in de functie uitgevoerd.

- `def functienaam(arg0: type, arg1: type, ...): ...`
  Functies kunnen één of meerdere argumenten hebben. Argumenten zijn waardes die je aan de functie doorgeeft. Binnen de functie kan je deze argumenten gebruiken als gewone variabelen.

- `def functienaam() -> returnType: return ...`
  Het `return`-statement
  - beëindigt de uitvoering van een functie meteen, zelfs als er nog code had achter gestaan.

  - kan gevolgd worden door een waarde. Dan wordt deze waarde door de functie teruggegeven, en kan je verder gebruiken in de code.
  - mag je alleen gebruiken binnen in een functie.

- `def functienaam(arg00: type, arg1: type, ...) -> returnType: return ...`
  Natuurlijk kan een functie zowel argumenten als een `return`-waarde hebben.

- Enkele voorbeelden van functies die je waarschijnlijk al gebruikt hebt:
  - `print(...)`
    toont die de argumenten die we meegeven op het scherm.
  - `waarde = input(...)`
    toont de tekst van het argument op het scherm, en geeft de input van de gebruiker daarna terug als `str`.
  - `int(...)`, `float(...)`, `str(...)`, ...
    probeert om de waarde die we als argument meegeven om te zetten naar het respectievelijke gegevens type.