Python & Micro Python

Een introductie, deel 2

python™

6 januari 2024

Python Introductie

- Waarom Python
  - Eenvoudig te leren
  - Gestructureerd
    - Een instruktie per regel
    - Inspringen voor structuur
  - Interprenter
    - Elke regel wordt vertaald en uitgevoerd

Python Introductie

- De volgende items zijn behandeld
  - De editor Thonny, een basic python editor
  - De python shell (terminal)
    - Invoer van instukties en expressies
    - Het resultaat wordt direct getoond
  - Python programma
    - Meerdere instrukties en expressies achter elkaar
    - Een regel per instruktie of expressie
    - Men dient zelf te zorgen voor het tonen van resultaat
  - Python invoer en uitvoer
    - Het print statement
      - print ("Hallo wereld !") → **Hallo Wereld !**
      - print (print(1, "+", 2, "=", 1+2) → **1 + 2 = 3**
    - Het input statement (impliciet behandeld)
      - tekst = input ("Geef een tekst: ")

Python Introductie

- De volgende items zijn behandeld
  - Python basis bewerkingen
    - Reken operatoren
      - Optellen +
      - Aftrekken -
      - Vermenigvuldigen *
      - Delen /
      - Machtsverheffen **
      - Integer delen //
      - Modulo %
    - Standaard variabelen
      - String tekst = “Hallo”
      - Integer i = 25
      - Float x = 42.57

Python Introductie

- De volgende items zijn behandeld
  - Python expressies
    - Simpele expressies — 15 + 2
    - Uitgebreide expressies — 5 * 2 – 3 + 4 / 2
      Meneer Van Dalen wacht Op Antwoord
    - String expressies — “Hallo” + “ “ + “Wereld”
      2 * “Hoi”
    - Boolean expressies
      - Waarde: — True, False
      - Vergelijkingen
        - Kleiner dan — <
        - Kleiner dan of gelijk aan — <=
        - Gelijk aan — ==
        - Groter dan of gelijk aan — =>
        - Groter dan — >
        - Niet gelijk aan — !=

Python Introductie

- De volgende items zijn behandeld
  - Python expressies
    - Boolean expressies (vervolg)
      - Logische operatoren
        - not — not (a)
        - and — (a) and (b)
        - or — (a) or (b)
      - De “in” operator
        - a in (2,8,9,22)
        - s in (‘rood’,’wit’,’blauw’)

Python Introductie

- Programmeren
  - Python expressies
    - print() funktie
    - input() funktie
    - float() funktie
  - Python conditioneel statement
    - if: – elif: – else:
  - Python lussen
    - for:
    - while:
    - while True: break

Python Introductie (deel 2)

- Programmeren
  - Het standaard print() commando heeft een aantal beperkingen
    - Een output regel per print() commando
    - Python bepaalt zelf hoe de variabelen weergegeven worden
    - Meerdere variabelen zijn mogelijk, maar worden gescheiden door een komma, terwijl in de output er altijd een spatie toegevoegd wordt tussen twee elementen.
    - Voorbeeld:

```
>>> print(5.0,"+",7.5,"=",5.0+7.5)
    5.0 + 7.5 = 12.5
>>>
```

  - Geformateerd printen
    - Bij geformateerd printen bepaal je zelf hoe de output eruit moet zien
    - Je kunt print() commando's zodanig definiëren, zodat een volgend print() statement verder gaat op 'n manier die jij wil.
    - Je kunt bij het print() commando aangeven wat het scheidings teken is.

Python Introductie (deel 2)

- Programmeren
  - Het scheidings teken in het print() statement bepalen
    - Als je niets verteld, is het scheidings teken een spatie (“ “)
    - Met behulp van een speciale parameter **sep** kun je het scheidings teken definiëren.
    - Voorbeeld:

```
>>> print(5.0,"+",7.5,"=",5.0+7.5)
   5.0 + 7.5 = 12.5
>>>
```

    - We voegen nu de parameter ***,sep=“”*** toe (oftewel een lege string):

```
>>> print(5.0,"+",7.5,"=",5.0+7.5,sep="")
   5.0+7.5=12.5
>>>
```

    - Je ziet dat nu de spatie tussen de verschillende parameters is verdwenen

Python Introductie (deel 2)

- Programmeren
  - Het scheidings teken in het print() statement bepalen
    - Een ander voorbeeld, een scheidings teken “x”:

```
>>> print("X","X","X", sep="x")
  XxXxX
>>>
```

    - Merk op, dat je maar twee keer een kleine “x” ziet. Dat komt, omdat het een scheidings teken betreft. Na de laatste grote “X”, volgt er niets meer (dus ook geen scheidingsteken).

Python Introductie (deel 2)

- Programmeren
  - Wat moet er met het volgende print() statement gebeuren
    - Als je niets verteld, gaat een volgend print() statement verder op de volgende regel.
    - Je kunt dit gedrag veranderen door middel van de speciale parameter **end**.
    - Bijvoorbeeld, een programma'tje:

```
print("X", end="")
print("Y", end="")
print("Z")
>>> %Run -c $EDITOR_CONTENT
    XYZ
>>>
```

    - De end="“ parameter geeft aan dat de default "newline" vervangen moet worden door een lege string. Vandaar de output "XYZ" in de shell.

Python Introductie (deel 2)

- Programmeren
  - En natuurlijk kun je beide speciale parameters combineren!
    - Bijvoorbeeld, een ander programma'tje:

```
print("A","b","c", sep="", end=" ")
print(1,2,3, sep="")
>>> %Run -c $EDITOR_CONTENT
   Abc 123
>>>
```

    - Je ziet dat hier de scheidings spaties weg zijn en dat de newline na het eerste statement veranderd is in een spatie.

Python Introductie (deel 2)

- Programmeren
  - Het **.format()** statement
  - Het.format statement kun je gebruiken om de opgegeven waardes te plaatsen in een string.
  - Als voorbeeld:
  - “De text bevat {} en {}” .format(“een”,”twee”). De twee combinaties ‘{} ‘ worden vervangen door de twee parameters in het .format() statement.

```
>>> "de tekst bevat {} en {}".format("een","twee")
'de tekst bevat een en twee'
>>>
```

Python Introductie (deel 2)

- Programmeren
  - Het **.format()** statement
  - Dit kunnen we dus ook in een print() commando toepassen
  - Bijvoorbeeld, het programma:

```
a = 10
b = 25
som = a + b
print("De som van {} plus {} is: {}".format(a,b,som))
```

  - Geeft als het uitgevoerd wordt:

```
>>> %Run -c $EDITOR_CONTENT
    De som van 10 plus 25 is: 35
>>>
```

Python Introductie (deel 2)

- Programmeren
  - Het **.format()** statement
  - Met behulp van .format() kunnen we bij getallen ook aangeven, hoeveel cijfers we achter de komma willen zien.
  - Bijvoorbeeld:

```
>>> a = 10.2
>>> print ("A is:{:.3f}".format(a))
   A is:10.200
>>>
```

  - Let op, als je meerdere {} sets gebruikt in de string, zal de .format{} deze van links naar rechts op volgorde vervangen. Wil je dit anders, zul je tussen de {} de index van de reeks in .format() moeten geven.
  - Bijvoorbeeld: print ("reeks is: {2} {1} {0} ".format(1,2,3))

Python Introductie (deel 2)

- Programmeren
  - Het **.format()** statement
  - String parameters:
    - {:7} = lengte is (minimaal) zeven karakters, vooraf aangevuld met spaties
    - {:d} = Integer getal
    - {:f} = Floating point getal
    - {:5d} = Integer getal, (minimaal) 5 posities lang
    - {:7f} = Floating point getal, (minimaal) 7 posities lang
    - {:6.2f} = Floating point getal, (minimaal) 6 posities lang, waarvan 2 cijfers achter de komma.
    - *Er zijn nog andere mogelijkheden, welke ik nu niet behandel (bv. Rechts/Links uitlijnen)*

Python Introductie (deel 2)

- Programmeren
  - Funkties
  - Een funktie is een stukje code, die maar één keer gedefinieerd wordt, maar vele malen aangeroepen kan worden.
  - Een funktie hoeft geen waarde terug te geven, maar als er iets terug gegeven wordt, hoeft dat niet beperkt te blijven tot een waarde.
  - Het voordeel van een funktie is, dat je de funktie op zichzelf kunt optimaliseren. En als de funktie eenmaal goed werkt, kun je de funktie ook zonder nadenken als een "zwarte doos" (blackbox) gebruiken.

Python Introductie (deel 2)

- Programmeren
  - Basic Python funkties
  - Python heeft een aantal basis funkties, waarvan we er al een tweetal besproken hebben:
    - print() - output functie
    - input() - input functie, geeft een string terug
  - Daarnaast kent python zogenaamt "type casting", waarbij een meegegeven parameter als juist type wordt terug gegeven:
    - str() - geeft altijd een tekst terug
    - float() - geeft een floating point getal terug als de parameter als een getal geïdentificeert kan worden (anders krijg je een runtime error)
    - int() - geeft een integer getal terug, als de parameter als een getal geïdentificeert kan worden. In het geval van een floating point, zal alleen het gehele deel van het getal teruggegeven worden.

Python Introductie (deel 2)

- Programmeren
  - Andere eenvoudige basic Python funkties:
    - abs() - Het absolute (positieve) getal (-3.5 → 3.5)
    - max() - Het grootste van de twee meegegeven parameters
    - min() - Het kleinste van de twee meegegeven parameters
    - pow() - De eerste parameter wordt verheven tot de tweede parameter (machtsverheffen)
    - round() - Geeft alleen het gehele deel terug van een float parameter. Als er een tweede parameter meegegeven wordt, geeft de tweede parameter het deel achter de komma.
    - len() - Geeft de lengte van de meegegeven string terug.
  - Waarschijnlijk zijn er nog wel meerdere basis functies, maar die heb ik momenteel niet paraat.

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven.
  - Zoals al eerder aangegeven, kun je zelf funkties schrijven, die bepaalde taken in jouw programma afwerken. Zo'n funktie kun je dan optimaliseren, zonder dat je je druk moet maken over de rest van het programma.
  - Als de funktie dan af is, kun je de rest van je programma maken, zonder na te denken over de werking van de funktie. De funktie kun je nu als zwarte doos gebruiken.
  - Syntax:
    - Def <funktie naam>(<parameter lijst>):
      <akties>

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven.
  - Laten we eens aannemen, we willen een funktie maken die "Tot ziens!" print.
  - Daarvoor dienen we een funktie te definieren, die we even totZiens zullen noemen. De funktie heeft verder geen parameters nodig, dus wordt de definitie:
    - **def totZiens():**
  - In ons programma kunnen we de funktie simpelweg aanroepen met:
    - **totZiens()**

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven.
  - Het programma wordt nu:

```
def totZiens():
    print("Tot ziens!")

print("Hallo !")
totZiens()
```

  - De output is:

```
>>> %Run -c $EDITOR_CONTENT
Hallo !
Tot ziens!
>>>
```

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven, parameters meegeven
  - Aan een funktie kun je parameters gebruiken, die je in het hoofd programma meegeeft. Bijvoorbeeld:

```
def hallo( naam ):
    print("Hallo {}!".format(naam))
```

  - In je programma kun je nu de funktie aanroepen met verschillende namen:

```
hallo( "Jan" )
hallo( "Mies" )
hallo( "Karel" )
```

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven, parameters meegeven
  - Als je nu het totale programma draait, krijg je als output:

```
>>> %Run -c $EDITOR_CONTENT
    Hallo Jan!
    Hallo Mies!
    Hallo Karel!
>>>
```

  - Je ziet dus, dat de meegegeven parameter binnen de funktie verwerkt wordt, afhankelijk van de waarde van de parameter.

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven, waarde(s) terug geven
  - De vorige funkties gaven geen waarde terug, maar het is mogelijk om een funktie iets te laten doen en dan een resultaat terug te geven. Bijvoorbeeld, laten we eens een maximaal(a,b) functie maken. Het resultaat zal zijn dat de funktie de grootste waarde van de twee terug geeft.

```
def maximaal(a,b):
    if a > b:
        maxwaarde = a
    else:
        maxwaarde = b
    return maxwaarde
```

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven, waarde(s) terug geven
  - We kunnen nu de funktie vanuit het hoofd programma aanroepen en krijgen de grootste waarde terug.

```
print("De hoogste waarde van {} en {} is: {}".format(2,5,maximaal(2,5)))
```

  - De output wordt dan:

```
>>> %Run -c $EDITOR_CONTENT
   De hoogste waarde van 2 en 5 is: 5
>>>
```

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven, waarde(s) terug geven
  - Het aantal waardes wat we teruggeven, is niet gebonden aan een maximum. Bijvoorbeeld, we maken een funktie die twee getallen meekrijgt, de inhoud van die twee getallen omwisselt en weer teruggeeft:

```
def wisselOm(a,b):
    tijdelijk = a
    a = b
    b = tijdelijk
    return a,b
```

  - *Merk op: Deze funktie kan eenvoudiger, aan jou de opgave hoe.*

Python Introductie (deel 2)

- Programmeren
  - Zelf funkties schrijven, waarde(s) terug geven
  - Ook deze funktie kunnen we in een programma gebruiken:

```
x = 10
y = 20
print("Voor wisselOm: x={} en y={}".format(x,y))
x,y = wisselOm(x,y)
print("Na wisselOm:   x={} en y={}".format(x,y))
```

  - De output van deze funktie wordt nu:

```
>>> %Run -c $EDITOR_CONTENT
   Voor wisselOm: x=10 en y=20
   Na wisselOm:    x=20 en y=10
>>>
```

Python Introductie (deel 2)

- Programmeren
  - Modules
  - Een module is eigenlijk een verzamelig van een of meerdere funkties in een separate python file.
  - Deze python file (module), kun je in elk programma importeren en van daaruit aanroepen.
  - Hierdoor kun je een verzamelijng van funkties maken, die je in meerdere programma's toepast, zodat je niet telkens opnieuw het wiel moet uitvinden.

Python Introductie (deel 2)

- Programmeren
  - Modules importeren
  - Importeren van funkties uit modules kan op twee manieren:
    1. **import <module>**
       In het programma roep je dan de betreffende funktie aan met:
       modulenaam. funktienaam(<parameterlijst>)
    2. **from <module> import <funktie>**
       In het programma roep je nu de betreffende funktie aan met:
       funktienaam(<parameterlijst>)

Python Introductie (deel 2)

- Programmeren
  - Modules maken, maak je eigen module:

```
def hallo( naam ):
    print("Hallo {}!".format(naam))

def maximaal(a,b):
    if a > b:
        return a                # Met een return verlaat je de funkie
    return b                    # dus a is hier niet groter dan b

def wisselOm(a,b):
    return b,a                  # Ha,ha, ha, zo simpel kan het dus ook
```

  - Bewaar deze code als bijvoorbeeld ***mijnModule.py*** in de folder waar jouw programma's staan.

Python Introductie (deel 2)

- Programmeren
  - Nu kun je simpelweg de funkties uit je module gebruiken,
  - Als voorbeeld het programma'tje wat we eerder gebruikt hebben met de funktie "maximaal":

```
from mijnModule import maximaal

print("De hoogste waarde van {} en {} is: {}".format(2,5,maximaal(2,5)))
```

  - Opnieuw wordt de output:

```
>>> %Run -c $EDITOR_CONTENT
    De hoogste waarde van 2 en 5 is: 5
>>>
```

Python Introductie (deel 2)

- Programmeren
  - Of op de andere manier de funkties uit je module gebruiken,
  - Als voorbeeld het programma'tje met de funktie "wisselOm":

```
import mijnModule

x = 10
y = 20
print("Voor wisselOm: x={} en y={}".format(x,y))
x,y = mijnModule.wisselOm(x,y)
print("Na wisselOm:   x={} en y={}".format(x,y))
```

  - Opnieuw wordt de output:

```
>>> %Run -c $EDITOR_CONTENT
   Voor wisselOm: x=10 en y=20
   Na wisselOm:    x=20 en y=10
>>>
```

Python Introductie (deel 2)

- Programmeren
  - Modules algemeen

  - Binnen python zijn er voor heel veel problemen modules geschreven, die je op dezelfde manier aanroept. Kijk maar eens naar "het "blink" programma'tje uit de vorige presentatie!

```
1  from machine import Pin      # Importeer de Pin funktie uit de machine module
2  import time                  # Importeer de time module
3  led = Pin(25, Pin.OUT)       # Defineer GPIO poort 25 als output voor de
4                               # interne led op de Pico
5  while True:
6      led.toggle()             # Schakel de led status om: Aan <-> Uit
7      time.sleep(1)            # Wacht één seconde
```

Python Introductie

- Tot hier toe mijn introductie in Python.
- Er is zeker nog veel meer over te vertellen,
  maar dat gaat buiten deze introductie.
- Wil je meer over de materie weten, kan ik je zeker aanraden om de programmeursleerling te bestuderen, of eens aan te sluiten bij de training van HCC!Programmeren.

- Vanaf mijn kant wil ik jullie bedanken voor de aandacht
  en succes wensen met python programma'tjes maken.
- Mocht je nog tegen rariteiten aanlopen, aarzel niet om eens op 'n inloop middag binnen te lopen, of je probleem in de werkgroep te gooien ☺

Python Introductie

Zijn er nog vragen ?

Python Introductie

That's all, Folks