Inleiding tot Python

Brecht Baeten

15 september 2015

1 Wat is Python?

Python is een open source programmeer taal. Voor ingenieurstoepassingen of wetenschappelijke projecten is Python interessant aangezien er een aantal Python Modules bestaan die hier specifiek op gericht zijn. Deze modules maken het beheren van data of uitvoeren van complexe wiskundige algoritmes zeer eenvoudig, vandaar de populariteit. Python werkt volledig object georiënteerd, dit laat toe efficiente en leesbare code te schrijven. Het schijven van leesbare code is één van de hoekstenen van Python aangezien code veel vaker gelezen wordt dan geschreven. Python code hoeft ook niet gecompileerd te worden (althans niet door de gebruiker). Python is een geïnterpreteerde programmeertaal wat inhoud dat code rechtstreeks uitgevoerd kan worden zonder eerst te compileren. Een andere groot voordeel van Python boven bijvoorbeeld Matlab is dat het gratis is! Iedereen, wetenschappers, studenten, particulieren, grote of kleine bedrijven kan Python gratis downloaden en gebruiken. In veel Linux distributies word Python standaard meegeleverd wat zelfs de installatie overbodig maakt.

2 Installatie in Windows

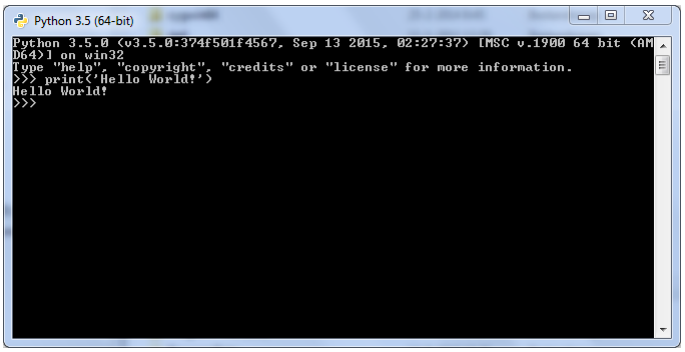
De nieuwste versie van Python kan worden gedownload via `https://www.python.org/downloads/`. Hier vind je de broncode en installer executables voor verschillende platformen. Downloaden, uitvoeren en klaar. Tijdens de installatie is het interessant om Python aan je Windows search path toe te voegen, zo kan je python steeds van in een command line interface openen. Het is ook interessant om de installatie locatie te veranderen naar bijvoorbeeld "C://python34".

Je kan nu al Python starten door op python.exe te klikken. Er opent een console waarin je python commando's kan invoeren (zie Figuur 1). Typ hier bijvoorbeeld:

```
print('Hello World!')
```

Druk op enter en je hebt je eerste python commando uitgevoerd.

Python code kan opgeslagen worden in eenvoudige tekst bestanden met de extentie ".py". Om deze te schrijven is een goede text editor met syntax highlighting aan te raden. In Windows is *notepad++* een goed, open source alternatief met syntax highlighting voor Python en een groot aantal andere talen ingebouwd. *notepad++* kan worden gedownload vanop `https://notepad-plus-plus.org/download/`.



Figuur 1: Python console met eerste commando

Omdat Python vanuit een commandline interface zal worden aangeroepen is het interessant om een deftige console te installeren. *ConEmu* is een open source console emulator met gelijkenissen aan *bash* in Linux. Deze is te downloaden vanaf `http://sourceforge.net/projects/conemu/files/latest/download`. Simpelweg downloaden, unzippen in een folder naar keuze en klaar.

3 Scripts

Omdat het telkens opnieuw invoeren van commando's achter elkaar niet zo praktisch is, is het interessant om een reeks commando's op te slaan in een script en dit uit te voeren. Open een teksteditor en typ **`print`**`('Hello World!')` en sla het bestand op als hello_world.py. Om het bestand uit te voeren open je een console, navigeer naar de locatie waar je het bestand hebt opgeslagen, typ `python hello\_world.py` en druk op enter.

Python heeft een aantal interessanten ingebouwde variabele types. Open een nieuw bestand, typ onderstaande commando's en sla het op als variables.py.

```
#variables
# integer
A = 1
print(A)

# float
B = 3.572
print(B)

# string
C = 'string'
print(C)

# list
D = [1,2,3]
print(D)
E = [1,'test',4,D]
print(E)
print(E[0])
print(E[1])

# dictionary
F = {3:1, 5:'test', 'foo':'bar'}
```

```
print(F)
print(F['foo'])
```

Voor het script ditmaal uit met het commando `python -i variables.py`. De `-i` zorgt ervoor dat python na het uitvoeren van alle commando's in het script naar de interactieve console gaat. Alle reeds gedefinieerde variabelen zijn nu ook beschipkbaar in de console. Typ bijvoorbeeld:

```
E[3]
```

of:

```
A+B
```

Zoals te merken in het voorgaande voorbeeld is Python zeer flexibel met data types. Zo kunnen een *Integer* en een *Float* met elkaar opgeteld worden en kan een *List* elementen met verschillende datatypes bevatten. Een zeer interessante data type is het *Dictionary* of *dict*. Hierin kunnen key / value paren worden opgeslagen en terug opgeroepen. Zowat elke Python variabele kan een *Dictionary* key zijn wat dit type zeer flexibel maakt.

Het interessante aan elke programmeertaal is het maken van loops en voorwaarden: flow control. In python is dit zeer eenvoudig met behulp van **`for`** lussen of **`if else`** structuren:

```
# flow control
for i in range(10):
    if i%2 == 0:
        print('{} is even'.format(i))
    else:
        print('{} is oneven'.format(i))
```

In Python moet elk flow control element eindigen met een ":". De code binnen het element moet een tab inspringen. Dit zorgt voor een zekere leesbaarheid in de code. De `range(10)` functie in de for lus maakt een *List* met waarden van 0 tot en met 9. De `.format(i)` functie formatteert zijn inhoud op de plaats van de accolade's in de voorafgaande string. De format functie is eigenlijk een methode van het *String* datatype en geeft een hele reeks formatteer opties. Even zoeken op het net leert je heel veel over zulke ingebouwde functies.

Je kan over zowat elk datatype itereren. Een lus over een dictionary kan bijvoorbeeld op verschillende manieren. Ook itereren over verschillende variablen tegelijk is eenvoudig met de `zip()` functie. Indien je itereert over een *List* en de index van de variabelen wil gebruiken kan je de functie `enumerate()` gebruiken:

```
# flow control advanced
A = {'foo': 'bar',
     'spam': 'eggs',
     'bacon': 'spam'}

# dictionaries
for key in A:
    print('{}: {}'.format(key,A[key]))

for key,val in A.iteritems():
    print('{}: {}'.format(key,val))

# zip
B = [1,2,3]
C = [7,8,9]
```

```
for b,c in zip(B,C):
    print('b = {:.0f}, c = {:.3f}'.format(b,c))


# enumerate
D = [1,11,111,1111]
for i,d in enumerate(D):
    print('D[{}] = {}'.format(i,d))
```

Een set commando's die vaak op dezelfde manier gebruikt worden kan je groeperen in een functie. Een functie definieer je met het **def** keywoord, gevolgd door de functie naam, de argument namen tussen haakjes en een dubbelpunt. Argumenten kunnen verplicht of optioneel zijn. voor optionele argumenten moet een default waarde opgegeven worden in de functie definitie. Alle verplichte argumenten moeten ook voor de optionle komen in de functie definitie. Wanneer een functie veel optionele argumenten heeft is het gemakkellijk om deze via naam-waarde paren op te geven, de volgorde is dan niet meer van belang:

```
# functions
def spam(A,B,C=0,D=0):
    val = 100*A+10*B+C+0.1*D
    return val

D = spam(4,1)
print(D)

E = spam(4,1,5.3)
print(E)

F = spam(4,1,D=3,C=5)
print(F)

print(val)
```

Variabelen die binnen een functie gedefinieerd zijn bestaan ook enkel in de functie namespace. Bovenstaande code geeft dus een error omdat `val` enkel in de functie namespace gedefinieerd is, niet in de globale namespace. Functies kunnen wel gebruik maken van variabelen in de globale namespace maar niet omgekeerd. Wanneer de bovenstaande code runt krijg je dan ook de volgende foutmelding:

```
Traceback (most recent call last):
  File "C:\examples\functions.py", line 10, in <module>
    print(val)
NameError: name 'val' is not defined
```

4 Numpy en Matplotlib

Er bestaan enorm veel modules waarin specifieke functie gedefinieerd zijn voor gebruik in Python. Twee voor Ingenieurs interessante modules zijn *Numpy*, een lineair algebra module en *Matplotlib* een module voor het maken van wetenschappelijke figuren van hoge kwaliteit. Voor beide kunnen onofficiele windows installers worden gedownload via `http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy` en `http://www.lfd.uci.edu/~gohlke/pythonlibs/#matplotlib`. Gewoon de binary kiezen die overeenkomt met jou systeem architectuur, downloaden en installeren.

5 Object georiënteerd

6 Modules