Extending Python (Libraries)

D. Leeuw

6 mei 2025
v.0.6.0

Over dit Document

0.1 Leerdoelen

Na bestudering van dit document heeft de lezer begrip van

- libraries
- `pip`
- het koppelen van systeembeheer-taken aan Python

0.2 Voorkennis

Van de lezer wordt verwacht dat deze kennis heeft van:

- variabelen in Python
- for en while loops
- hoe functies werken in Python

Inhoudsopgave

Hoofdstuk 1

De herschreven code

Een student heeft de onderstaande code geschreven om IP-adressen uit te vragen van een gebruiker.

```
# Functie
def vraag_ip(nummer):
    while True:
        ip = input(f"Wat is IP-adres nummer {nummer} in uw netwerk? ")
        if ip not in lst_netwerk:
            return(ip)
        print(f"IP-adres {ip} komt al in de lijst voor")


# Hoofdprogramma
lst_netwerk = []

print("We vragen om 3 IP-adressen in uw netwerk:")

for i in range(1,4):
    antw = vraag_ip(i)
    lst_netwerk.append(antw)

print("U heeft deze IP-adressen in uw netwerk:")
print(lst_netwerk)
```

We gaan met deze code verder werken. Knip en plak deze in je code editor.

Hoofdstuk 2

Het maken van een library

Splits je programma in twee bestanden. Eén met de functie en één met het hoofdprogramma. Het bestand met de functie komt er dan zo uit te zien:

```
def vraag_ip(nummer):
    while True:
        ip = input(f"Wat is IP-adres nummer {nummer}: ")
        if ip not in lst_netwerk:
            return(ip)
        print(f"IP-adres {ip} komt al in de lijst voor")
```

Sla dit stuk code op als mylib.py

Voor je hoofdprogramma blijft er dan over:

```
lst_netwerk = []

print("We vragen om 3 IP-adressen uit uw netwerk")

for i in range(1,4):
    antw = vraag_ip(i)
    lst_netwerk.append(antw)

print(lst_netwerk)
```

Sla dit stuk code op als `myprog.py`.

Nu willen we graag onze functie weer kunnen gebruiken in het hoofdprogramma, daarvoor hebben we `import`. Met `import` kunnen we externe code laden in ons programma. We voegen een `import` regel toe als eerste regel van ons hoofdprogramma:

```
import mylib
```

Let op: Bij de `import` komt er na de naam van het bestand geen `.py`! Doe je dit wel dan krijg je een error-melding:

```
Traceback (most recent call last):
  File "/home/Boeken/Programmeren/Python/code/myprog.py", line 1, in
    <module>
    import mylib.py
ModuleNotFoundError: No module named 'mylib.py'; 'mylib' is not a package
```

Een library of module zoals dat in Python heet kan uit meerdere functies bestaan. Verschillende libraries zouden een functie met dezelfde naam kunnen hebben. Als we beide libraries zouden importeren in ons programma wat moet er dan gebeuren met de twee functies die dezelfde naam hebben? Om dit uit elkaar te houden wordt bij de function-call de naam van de functie vooraf gegaan door de naam van de library. Onze functie `vraag_ip` heet dus vanaf nu `mylib.vraag_ip`. Dit heet een "namespace". Een functie behoort dus tot en bepaalde namespace. In ons geval is de namespace `mylib`.

In ons programma `myprog` zullen we de oude functie call `vraag_ip` moeten aanpassen naar `mylib.vraag_ip`. Doe dit in je eigen code.

Na de aanpassing van de naam van de functie call in `myprog.py` zal je na het starten van `myprog.py` merken dat Python een error geeft nadat je de eerste IP-adres hebt ingevuld. Python geeft als melding:

```
NameError: name 'lst_netwerk' is not defined
```

Python kent opeens de lijst lst_netwerk niet meer. De variabele die we eerst nog vanuit ons hoofdprogramma konden gebruiken in een functie kunnen we nu niet meer gebruiken! Kennelijk is een functie als onderdeel van het programma iets anders dan als het de functie uit een bestand vandaan komt. En dat klopt ook. Namespaces zijn geïsoleerd, ook wat variabelen betreft. We moeten de functie dus gaan vertellen wat onze lijst is.

Om de functie te vertellen wat onze lijst is hebben we een extra paramater bij de functie definitie nodig. We noemen de tweede parameter `net_lst`, de lijst krijgt dus binnen de functie een andere naam dan in het hoofdprogramma. We gebruiken de naam van de lijst bij de `if` in de functie en zullen dus ook daar de naam moeten aanpassen. Met deze wijzigingen komt onze library functie er zo uit te zien:

```
def vraag_ip(nummer,net_lst):
    while True:
        ip = input(f"Wat is IP-adres nummer {nummer}: ")
        if ip not in net_lst:
            return(ip)
        print(f"IP-adres {ip} komt al in de lijst voor")
```

In ons hoofdprogramma moet de functie call nu 2 argumenten hebben:

```
    antw = mylib.vraag_ip(i,lst_netwerk)
```

Waarom hebben we deze splitsing aangebracht? Daar zijn een aantal redenen voor:

Leesbaarheid Door functies in hun eigen bestand op te slaan wordt het hoofdprogramma overzichtelijker

Beheer We kunnen verschillende functies groeperen in hun eigen bestand

Hergebruik We kunnen de functies nu in verschillende programma's gebruiken, wat programmeer werk scheelt

Hoofdstuk 3

Libraries van anderen

Er zijn vele duizenden programmeurs op deze wereld die werken met Python. Vele programma's worden dagelijks in Python geschreven. Al deze programmeurs schrijven natuurlijk ook functies om hun leven makkelijker te maken en met het Internet is het eenvoudig om deze functies met elkaar te delen. Wat iemand anders al gedaan heeft hoeven wij niet meer te doen. Collecties van functies worden verzameld in een library die gedeeld kan worden.

Graag zouden we een simpele manier hebben om de code van anderen op ons systeem te installeren. Dat doen we door de libraries te verpakken als packages, zogenaamde modules die we binnen Python kunnen installeren met het commando `pip`. In dit hoofdstuk gaan we nader op `pip` in.

3.1 pip

We hebben gezien dat we in Python zelf programma's kunnen schrijven en dat we functies kunnen gebruiken om eenmaal geschreven code te hergebruiken. Het idee zou nu ontstaan kunnen zijn dat het ook mogelijk zou moeten zijn om functies van iemand anders te gebruiken. Dat kan natuurlijk door te knippen en plakken, maar kan dat ook makkelijker? Ja, dat kan door code van anderen te downloaden op ons systeem en deze vervolgens te gebruiken. Het beheer (downloaden, installeren en eventueel weer verwijderen) doen we met `pip`.

`pip` is de package manager van Python.

Gebruik de list optie om een lijst te krijgen van geïnstalleerde packages:

```
$ pip list
```

Om meer te weten te komen van een package gebruiken we de show optie:

```
$ pip show pip
```

Met de install en uninstall opties kunnen we packages installeren en de-installeren. In de opdrachten gaan we hiermee werken.

Hoofdstuk 4

Vragen en opdrachten

4.1 Vragen

1. Hoe heet de software die we gebruiken om Python-code van anderen te installeren?

2. Wat is het commando dat we moeten geven om shutil te installeren op ons systeem?

3. Als ik uit de module shutil de functie disk_usage() wil gebruiken, hoe ziet de functie-call er dan uit?

4.2 Opdrachten

Opdrachten

1. Schijfruimte controleren

   Doel Werken met functies en modules.

   Opdracht Schrijf een functie "controleer_schijfruimte()" die controleert hoeveel vrije ruimte er beschikbaar is op de C-schijf en een waarschuwing geeft als dit minder dan 5 GB is.

   Tip Gebruik de module "shutil" en de functie "disk_usage()"

   Voorbeeldoutput output naar de gebruiker:

   ```
   De vrije ruimte op C:\ is 10.4 GB - voldoende
   ```

   of:

   ```
   Waarschuwing: minder dan 5 GB beschikbaar!
   ```

2. Automatisch gebruikers aanmaken

   Doel Functies met parameters schrijven.

   Opdracht Maak een functie "maak_gebruiker_aan(gebruikersnaam)" die een nieuwe gebruiker zou kunnen aanmaken (simulatie, geen echte systeemverandering).

   Uitbreiding Laat de functie een bevestiging printen, zoals:

   ```
   Gebruiker 'student123' succesvol aangemaakt.
   ```

   Bonus Voeg een standaard wachtwoord toe als extra parameter.

3. Logbestanden opschonen

   Doel Werken met bestandsbeheer via functies.

   Opdracht Schrijf een functie "verwijder_oude_logs(pad)" die alle bestanden met '.log' in een opgegeven map verwijderd die ouder zijn dan 30 dagen.

   Tip Gebruik "os", "os.path" en "datetime"

   Voorbeeldinput De functie call zou er zo uit kunnen zien:

   ```
   verwijder_oude_logs("C:/logs")
   ```

4. Systeemrapport genereren

   Doel Functies combineren en rapporteren.

   Opdracht Maak een functie "genereer_systeemrapport()" die:

   - de hoeveelheid RAM laat zien,
   - de CPU-info toont,
   - het IP-adres weergeeft.

   Tip Gebruik modules zoals "psutil" en "socket"

   Voorbeeldoutput De output zou er zo uit kunnen zien:

   ```
   Systeemrapport:
   RAM: 8 GB
   CPU: Intel Core i5
   IP-adres: 192.168.1.101
   ```