

Les 7 - Programmastructuren

[« Vorige les](#)

[» Volgende les](#)

Huiswerk

Programmeren voor Beginners - Les 7

[Ga naar huiswerk](#)

Je hebt 1 vraagmoment. 1 moment over.

[Ga naar stel een vraag](#)

[Markeer als deelgenomen aan deze les](#)

Inhoudsopgave

- 7.1** Inleiding
- 7.2** Wat is een string? (herhaling)
- 7.3** String = list van karakters
- 7.4** Strings bewerken
- 7.5** Formatted strings
- 7.6** String-methods
- 7.7** Andere functies in combinatie met strings
- 7.8** Introductie van de casus
- 7.9** Map aanmaken en repo klonen
- 7.10** Remote repo aanmaken op GitHub
- 7.11** Bestand 'helper.py' toevoegen
- 7.12** Samenvatting
- 7.13** Vraagmoment
- 7.14** Oefenopgaven
- 7.15** Huiswerkopgaven

7.1 Inleiding

U heeft inmiddels al een behoorlijke basis als het gaat om programmeren in Python. Maar wat kunt u er nu eigenlijk mee in de praktijk?

De programma's die u tot nu toe heeft geschreven, waren erg klein en veelal op zichzelf staand. In deze les brengen we daar voor het eerst verandering in. We starten namelijk met het schrijven van een programma dat de rest van deze cursus centraal zal staan.

Voordat u met dit programma aan de slag gaat, gaan we wat dieper in op het gebruik van *strings*. In deze les leert u hoe u *strings* kunt bewerken en gaan we in op *formatted strings*.

Leerdoelen

Aan het einde van deze les weet u:

- dat een *string* in feite een *list* van karakters is;
- op welke manieren u een *string* kunt bewerken;
- wat een *formatted string* is en hoe u die samenstelt;
- wat een *string-method* is;
- welke andere functies u zoal kunt combineren met *strings*;
- met welke praktijkcasus u aan de slag gaat;
- hoe u een *repo* kunt klonen.



Enkele afbeeldingen in deze les zijn moeilijk leesbaar in de printversie van deze les. Wilt u een afbeelding uitvergroten? Bekijk dan de digitale versie van de les op Plaza. Klik op de afbeeldingen om ze te vergroten.

7.2 Wat is een string? (herhaling)

In een eerdere les gaven we aan dat de waarde van een variabele een getal of een ander **datatype** kan zijn. Een van die datatypes is *string*.

We zouden simpelweg kunnen zeggen dat dit een tekst is, maar dat dekt niet de hele lading. Een *string* kan namelijk niet alleen letters bevatten, maar ook cijfers, leestekens en symbolen. Daarnaast kan een *string* leeg zijn.

Enkele voorbeelden:

```
string1 = ""  
string2 = "A"  
string3 = "123ABC"  
string4 = "Python is een taal die begin jaren 90 ontworpen werd."  
string5 = " @&*( )!_)_@*&#(!_!"
```

De eerste regel code is een lege *string*, de andere vier regels laten zien hoe gevarieerd de inhoud kan zijn wanneer de *string* gevuld is.

7.3 String = list van karakters

In de vorige les maakte u kennis met het begrip *list*. Dit is een van de methodes waarmee u een reeks van waardes kunt toekennen aan een enkele variabele.

Een *list* ziet er bijvoorbeeld zo uit:

```
my_cart = ["appel", "banaan", "citroen"]
```

Een *string* zou u ook kunnen zien als een *list*: een **lijst van karakters**. Dat wordt duidelijk op het moment dat u een *string* combineert met een *for-loop*:

- Open VS Code.
- Open een map die u eerder aanmaakte, bijvoorbeeld NHA-prg).

TIP!

Weet u het nog? Een map openen doet u door in de *Menu Bar* op "File" te klikken, vervolgens te klikken op "Open Folder" en tot slot de map te selecteren.

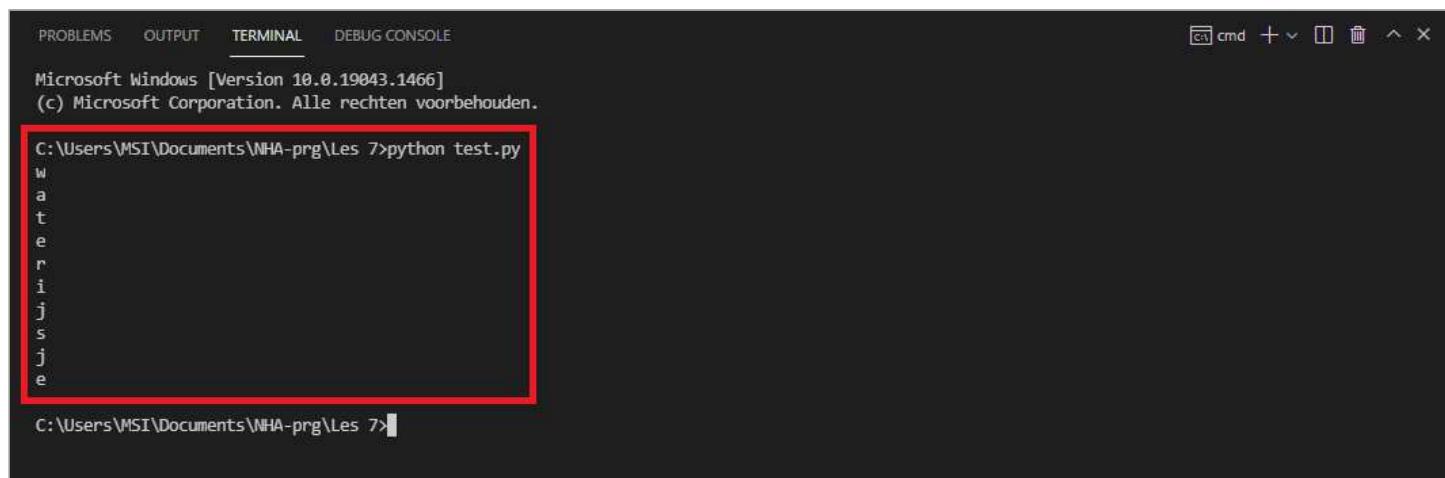
- Klik op de knop "New Folder".
- Geef de map een naam (bijvoorbeeld Les 7).
- Klik in de *Menu Bar* op "File".
- Klik op "Open Folder".
- Selecteer de map die u zojuist als laatste aanmaakte (Les 7).
- Klik in de *Side Bar* op de knop "New File".
- Geef het bestand een naam. Eindig met de extensie .py om aan te geven dat het om een Python-bestand gaat (bijvoorbeeld test.py).
- Typ de volgende code in het bestand:

```
mijn_string = "waterijsje"

for l in mijn_string:
    print(l)
```

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

Het resultaat ziet er dan als volgt uit:



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE cmd + ×

Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
w
a
t
e
r
i
j
s
j
e

C:\Users\MSI\Documents\NHA-prg\Les 7>
```

De letters van de *string* waterijsje zijn een voor een uitgeprint. De verschillende karakters (hier: letters) zijn in feite de elementen van de *list* `mijn_string`.

Probeer nu dit eens:

- Verwijder de tweede en derde regel van uw code.
- Typ de volgende code op de tweede regel:

```
print(mijn_string[4])
```

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

De letter r wordt op het scherm getoond bij uitvoering van deze code:

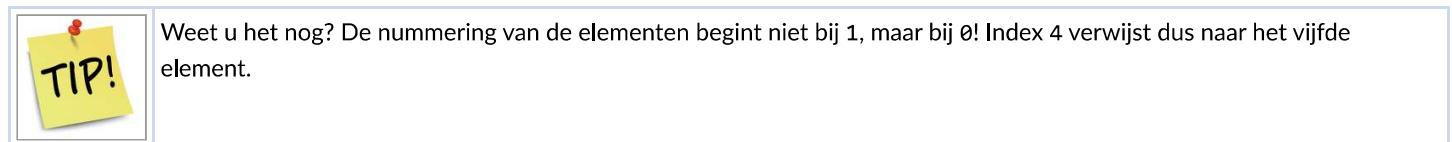
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. Alle rechten voorbehouden.

```
C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
r
```

C:\Users\MSI\Documents\NHA-prg\Les 7>

Dit karakter is dus het vijfde onderdeel van de *list*.



7.4 Strings bewerken

In ieder computerprogramma is er sprake van het verwerken van informatie, op welke manier dan ook. Informatie komt in beeld en geluid, maar vaak ook in tekst.

Tekst wordt vaak in variabelen bewaard gedurende het uitvoeren van het programma. Het zal u dan ook niet verbazen dat Python veel manieren kent om op een of andere manier een *string* te bewerken. We behandelen in de volgende subparagrafen de meest voorkomende manieren.

7.4.1 Samenvoegen

In Python kunt u *strings* eenvoudig samenvoegen door ze bij elkaar op te tellen alsof het getallen zijn. U gebruikt daarvoor het plusje (+).

- Vervang de code in het bestand door de volgende code:

```
str1 = "zon"
str2 = "licht"

totaal_str = str1 + str2

print(totaal_str)
```

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

Het resultaat van *zon* + *licht* is *zonlicht*:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. Alle rechten voorbehouden.

```
C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
zonlicht
```

C:\Users\MSI\Documents\NHA-prg\Les 7>

Het is ook mogelijk om *strings* op te tellen met een spatie ertussen. Bijvoorbeeld:

- Vervang de code in het bestand door de volgende code:

```
voornaam = "Jan"
achternaam = "de Vries"

naam = voornaam + " " + achternaam
```

```
print(naam)
```

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

Het resultaat bevat twee spaties:



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
Jan de Vries
C:\Users\MSI\Documents\NHA-prg\Les 7>
```

Een van die spaties was al aanwezig in de *string* die hoorde bij de variabele achternaam, de andere heeft u handmatig toegevoegd tussen de twee variabelen.



U kunt *strings* dus samenvoegen door ze op te tellen, maar dat is ook de enige rekenkundige bewerking die u ermee kunt doen. Aftrekken, vermenigvuldigen en delen zijn niet mogelijk, uitsluitend optellen.

Maar: het is wel mogelijk om een *string* te vermenigvuldigen met een getal! Zo zal `print(10 * "")` resulteren in `*****`.

7.4.2 Splitsen en sub-strings

Eerder in deze les gaven we aan dat u een *string* kunt zien als een *list* van karakters. Daardoor is het mogelijk om slechts delen van een *string* te nemen en bijvoorbeeld te printen.

- Vervang de code in het bestand door de volgende code:

```
mijn_string = "ham, eieren, koffie en thee"

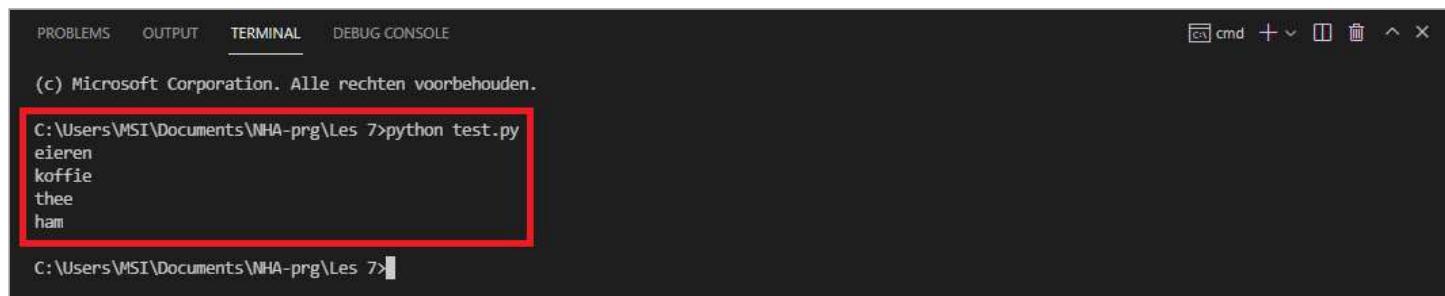
print(mijn_string[5:11])
print(mijn_string[13:19])
print(mijn_string[23:])
print(mijn_string[:3])
```

Als u deze *string* opdeelt in karakters, kunt u de uitkomsten van de vier print-acties voorspellen:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
h	a	m	,		e	i	e	r	e	n	,		k	o	f	f	i	e		e	n		t	h	e	e

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

Het resultaat ziet er dan als volgt uit:



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
eieren
koffie
thee
ham
C:\Users\MSI\Documents\NHA-prg\Les 7>
```

Waren uw voorspellingen juist?



Weet u het nog? [5:11] betekent index 5 tot index 11, dus niet index 5 tot en met index 11. Met [23:] geeft u aan dat u index 24 en verder wil printen. [:3] betekent daarentegen dat u alle elementen voor index 3 wil printen, oftewel index 0 tot en met 2.

7.5 Formatted strings

In een eerdere les maakte u al kennis met *formatted strings*. Hiermee kunt u variabelen integreren in een `print`-statement. Bijvoorbeeld:

Input	Output
<pre>a = "Maandag" b = "Dinsdag" c = "Woensdag" print(f"Eerst komt {a}, dan komt {b} en dan komt {c}.")</pre>	Eerst komt Maandag, dan komt Dinsdag en dan komt Woensdag.

Het gebruik van een *formatted string* is een veelgebruikte manier om variabelen te combineren met een tekst. Hoe dat werkt en waarom het beter is om dat te doen, laten we zien aan de hand van een voorbeeld. We kijken daarbij al 'stiekem' even naar onze ijssalon:

U zou een variabele op de volgende manier kunnen combineren met tekst:

```
prijs = "2 euro"

print("Een ijsje met 2 bolletjes kost " + prijs + ".")
```

Hoewel er hier is gebruikgemaakt van een *formatted string*, is deze code juist. U krijgt immers de gewenste output:

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE cmd + ×

Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
Een ijsje met 2 bolletjes kost 2 euro.

C:\Users\MSI\Documents\NHA-prg\Les 7>
```

Het is echter aanbevelenswaardig om de `print`-opdracht op de volgende manier te formuleren:

```
prijs = "2 euro"

print(f"Een ijsje met 2 bolletjes kost {prijs}.")
```

De uitvoer zal exact hetzelfde zijn, maar deze werkwijze heeft drie belangrijke voordelen:

1. De code is beter leesbaar.
2. Er is een kleinere kans op fouten. (Als u het op een andere manier doet, moet u namelijk variabelen en *string* bij elkaar optellen, met een wirwar van enkele en dubbele aanhalingsstekens. Het is dan een stuk minder overzichtelijk wat voor *string* u aan het knutselen bent en daardoor foutgevoelig.)
3. De computer voert de bewerking sneller uit.

Kortom: als u variabelen en tekst wil combineren, kunt het beste werken met een *formatted string*. De code bouwt u dan zo op:

```
print(f"string")
```

Het **rode** deel van de code toont waar u uw *string* plaatst. Het is erg belangrijk om een **f** voor de *string* te plaatsen. Die **f** zorgt er namelijk voor dat de Python-interpreter begrijpt dat alle elementen die tussen de accolades `{}` staan, variabelen zijn. De *interpreter* vult tijdens het printen de waarden van deze variabelen in.

7.6 String-methods

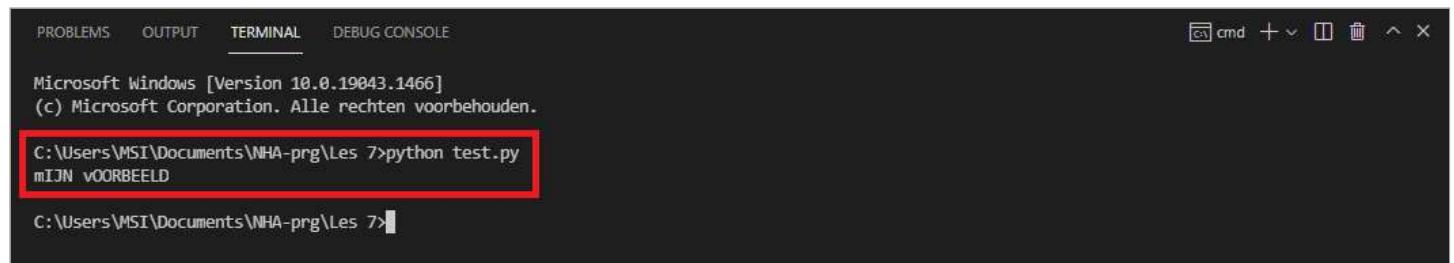
Omdat *strings* vaak worden gebruikt, bevat de *The Python Standard Library* vrij veel functies die 'iets' met een *string* doen.

Als een functie door middel van een punt gekoppeld wordt aan een ander Python-element, noemen we de functie een **method**. Bijvoorbeeld:

```
print("Mijn Voorbeeld".swapcase())
```

In dit voorbeeld is `.swapcase()` de *method*. Hij wordt toegepast op de *string* "Mijn Voorbeeld". Een *method* die u toepast op een *string*, noemen we een **string-method**.

Wat deze *string-method* doet, wordt duidelijk als u die uitvoert in VS Code:



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
MIJN VOORBEELD

C:\Users\MSI\Documents\NHA-prg\Les 7>
```

In de volgende subparagrafen bespreken we enkele veelgebruikte *string-methods*.



Wilt u ook kennismaken met andere *string-methods*? Bekijk dan de *The Python Standard Library* op <https://docs.python.org/3/library> of bekijk de website www.w3schools.com/python/python_ref_string.asp.

7.6.1 .lower()

Met de *string-method* `.lower()` kunt u alle hoofdletters in een *string* omzetten in kleine letters (in het Engels ook wel *lowercase letters* genoemd). Bijvoorbeeld:

Input	Output
<code>print("Mijn Voorbeeld").lower()</code>	mijn voorbeeld

7.6.2 .upper()

Met de *string-method* `.upper()` kunt u alle kleine letters in een *string* omzetten in hoofdletters (in het Engels ook wel *uppercase letters* genoemd). Bijvoorbeeld:

Input	Output
<code>print("Mijn Voorbeeld").upper()</code>	MIJN VOORBEELD

7.6.3 .split()

Met de *string-method* `.split()` kunt u een *string* opsplitsen en er een *list* van maken. Bijvoorbeeld:

Input	Output
<code>print("Mijn Voorbeeld").split(" ")</code>	['Mijn', 'Voorbeeld']

Dat wat tussen achter `.split` tussen haakjes staat, wordt gebruikt als breekpunt. Dit noemen we ook wel de **substring** en is in dit geval dus de spatie tussen Mijn en Voorbeeld.

7.6.4 .index()

Deze *string-method* geeft de index van de *substring* weer, mits die gevonden wordt. Bijvoorbeeld:

Input	Output
<code>print("Mijn Voorbeeld".index("Voorbeeld"))</code>	5

Het resultaat is 5, want dat is de index van de V van Voorbeeld.

U vraagt zich misschien af of de output niet 4: of 4:13 had moeten zijn. Zo werkt deze *string-method* echter niet. Hij begint vooraan in de *string* en kijkt per karakter of daar het woord Voorbeeld te vinden is. Bij het zesde karakter (met index 5) vindt hij het woord Voorbeeld, dus is de output gelijk aan 5.

7.6.5 .replace()

Tot slot benoemen we de *string-method* `.replace()`. Hiermee kunt u de ene *substring* vervangen door de andere. Bijvoorbeeld:

Input	Output
<code>print("Mijn Voorbeeld".replace("Voorbeeld", "Vervanging"))</code>	Mijn Vervanging

7.7 Andere functies in combinatie met strings

Er zijn nog een paar functies die vaak gebruikt worden in combinatie met strings. Dit zijn geen methods, maar wel de moeite waard om te benoemen.

7.7.1 len(string)

Als eerste benoemen we de functie `len(string)`. Hiermee kunt u de lengte van een *string* weergeven. Bijvoorbeeld:

Input	Output
<code>print(len("Mijn Voorbeeld"))</code>	14

Het resultaat is 14, want de *string* Mijn Voorbeeld bestaat uit 14 karakters.

7.7.2 int(string)

Daarnaast noemen we de functie `int(string)`. Hiermee kunt u een *string* naar een *integer* converteren. Bijvoorbeeld:

Input	Output
<code>print(int("1200"))</code>	1200

Het lijkt alsof er niets is veranderd. Als u de volgende stappen volgt, zult u echter snappen dat dit wel het geval is:

- Ga naar VS Code.
- Vervang de code in het bestand door de volgende code:

```
mijn_string = "2000"
mijn_int = int("2000")

print(mijn_string + mijn_string)
print(mijn_int + mijn_int)
```

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

Het resultaat ziet er dan als volgt uit:

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 7>python test.py
20002000
4000

C:\Users\MSI\Documents\NHA-prg\Les 7>
```

U ziet dat *strings* optellen betekent dat ze gewoon achter elkaar geplakt worden. *Integers* optellen is een wiskundige bewerking.

7.8 Introductie van de casus

We hebben er in de voorgaande lessen al een aantal keer op gehint, maar nu is het eindelijk zover: we gaan alle kennis die u tot nu toe heeft opgedaan, toepassen op een praktijkcasus. We bouwen een simpele webshop voor een ijssalon.

We geven u de casus waarmee u aan de slag zult gaan:

De familie Van Steen is erg creatief. Vooral in de keuken wordt er flink geëxperimenteerd met allerlei gerechten. Dochter Ellie heeft zich gespecialiseerd in het maken van desserts. Ze maakt zelfs haar eigen schepijs.

Uit de buurt komen er geregeld verzoekjes voor een door Ellie gemaakte ijstaart of een emmertje schepijs. Haar hobby is nu zo uit de hand gelopen, dat ze heeft besloten een winkeltje te openen. Ze is op zoek naar een pand. Maar voor het zover is, wil ze alvast een webshop starten, waar ze haar producten kan etaleren en klanten orders kunnen plaatsen.

*Ellie heeft bij NHA een **cursus Webdesign** gevolgd, waarmee het haar is gelukt om de ‘voorkant’ van de website (ook wel ‘front end’ of ‘client side’ genoemd) zelf te maken. Voor de achterkant (ook wel ‘back-end’ of ‘server-side’ genoemd) heeft ze een specialist nodig.*



Nu is het uw beurt! U gaat Ellie helpen, want Python is uitermate geschikt om de programmatuur van die achterkant te verzorgen. U heeft voor uzelf een stappenplan uitgewerkt:

Stap	Beschrijving	Les
1	Alle initiële bestanden en mappen aanmaken	7
2	Versiebeheer lokaal organiseren (Git)	7
3	Versiebeheer remote organiseren (GitHub)	7
4	De bestanden 'helper.py' en 'tijdelijk.py' coderen	7
5	Functies aanmaken	8
6	Functies importeren	8
7	Het bestand 'helper.py' uitbreiden met een functie	9
8	De bestanden 'hoofd_programma.py', 'boekhouding.py' en 'presentatie.py' coderen	10
9	Flask-framework installeren en bijbehorende bestanden coderen	11

Zoals u ziet, gaat u in deze les aan de slag met de eerste stappen.

Goed om te weten: Ellie heeft een deel van de website al gemaakt en weet – net als u – hoe Git en GitHub werken.



De creatieve Ellie van der Steen is uw opdrachtgever.

7.9 Map aanmaken en repo klonen

In een eerdere les heeft u leren werken met *Git* en *GitHub*. In die les noemden we al enkele voorbereidingen voor deze les. In deze paragraaf borduren we daarop voort.

Als eerste gaat u een map aanmaken:

- Open *VS Code*.
- Open een map die u eerder aanmaakte (bijvoorbeeld *NHA-prg*).
- Klik op de knop “New Folder”.
- Geef de map een naam (bijvoorbeeld *IJssalon*).
- Klik in de *Menu Bar* op “File”.
- Klik op “Open Folder”.
- Selecteer de map die u zojuist aanmaakte.

Daarna kloont u een bestaande *repository* (of kortweg ‘repo’). In een eerdere les leerde u dat dit een map is die alle bestanden van een project bevat. In dit geval is de *repository* gemaakt door uw opdrachtgever, Ellie.

- Open een nieuwe *Terminal*.
- Typ de volgende code:

```
git clone https://github.com/Alexico1969/IJssalon-start
```

- Druk op Enter.

U ziet dan de volgende code in de *Terminal* verschijnen:

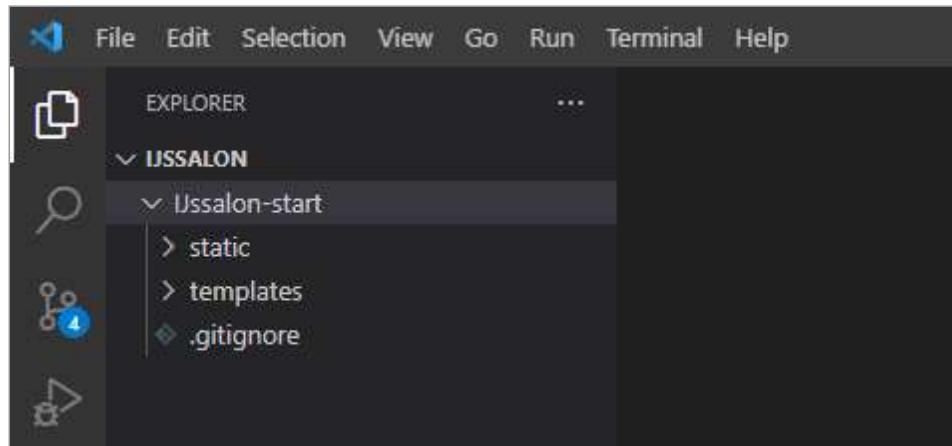
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Microsoft Windows [Version 10.0.19043.1526]
(c) Microsoft Corporation. Alle rechten voorbehouden.

```
C:\Users\MSI\Documents\NHA-prg\IJssalon>git clone https://github.com/Alexico1969/IJssalon-start
Cloning into 'IJssalon-start'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (15/15), done.
Receiving objects: 100% (20/20), 113.86 KiB | 2.92 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

C:\Users\MSI\Documents\NHA-prg\IJssalon>

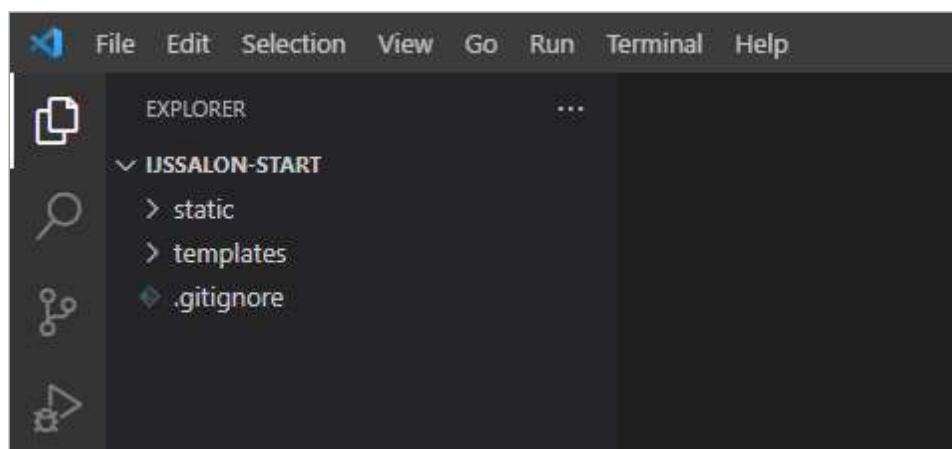
Als het proces voltooid is, ziet u dat er een nieuwe map is aangemaakt:



In de map zijn twee submappen aanwezig (en het losse bestand '.gitignore'), maar die zijn nu nog niet van belang. U gaat aan de slag met de hoofdmap, genaamd 'IJssalon-start'. Die opent u in VS Code.

- Klik in de *Menu Bar* op "File".
- Klik op "Open Folder".
- Navigeer naar de map die u zojuist heeft aangemaakt. Zorg ervoor dat u ver genoeg doorklikt: als u de submappen in beeld ziet, bent u op de juiste plaats.
- Bevestig uw keuze.

Als het goed is, ziet u nu in de *Side Bar* dat de map 'IJssalon-start' de actieve map is:



7.10 Remote repo aanmaken op GitHub

De volgende stap is het aanmaken van een *remote repository* op *GitHub*. In een eerdere les leerde u dat dit de connectie is tussen de map die u in *Git* geopend heeft en de *repository*.

- Ga naar www.github.com.
- Klik rechtsboven op "Sign in".
- Log in met uw gegevens.
- Klik na het inloggen op "New" om een nieuwe repository aan te maken.

- Geef de repository een logische naam, bijvoorbeeld `ijssalon_ellie`.
- Scrol naar beneden en klik op “Create repository”.

Nadat u op deze knop heeft gedrukt, komt u terecht op de *repository page* van uw nieuwe pagina. In de adresbalk van uw browser staat nu de link naar uw *remote repo*, bijvoorbeeld:



- Kopieer deze link.
- Ga naar VS Code.
- Ga naar een nieuwe regel in de Terminal.
- Typ `git remote set-url origin, maar druk nog niet op Enter!`
- Plaats een spatie na `origin`.
- Plak de eerder gekopieerde link na deze spatie.
- Druk nu wél op Enter.
- Typ `git push origin main` en druk op Enter.

Voordat het proces gestart wordt, zou het kunnen zijn dat u nog een keer moet inloggen. Kies dan voor het inloggen via de browser. Als u GitHub nog open heeft staan (en nog ingelogd bent), dan zal het proces alsnog starten.

- Ververs de GitHub-pagina zodra het proces voltooid is.

U ziet nu de submappen (en het losse bestand ‘.gitignore’) staan:

File	Commit Message	Date	Commits
<code>static</code>	initial commit	3 days ago	1
<code>templates</code>	initial commit	3 days ago	1
<code>.gitignore</code>	initial commit	3 days ago	1

U heeft nu een lokale *repo* (op uw eigen computer), die gesynchroniseerd is met de *remote repo* op GitHub.

7.11 Bestand ‘helper.py’ toevoegen

We splitsen de functionaliteit van de achterkant van webshop in twee delen: het hoofddeel (dat we ‘app.py’ zullen noemen) en een hulpdeel (dat we ‘helper.py’ noemen). Dat doen we vooral ter voorbereiding op een latere les.

- Ga naar VS Code.
- Controleer of de map ‘IJssalon-start’ nog als *folder* geopend is.
- Klik in de *Side Bar* op de knop “New File”.
- Geef het bestand de naam `helper.py`.
- Plaats de volgende code in het bestand:

```
tekst="header"
lengte = len(tekst) + 4
print()
print(lengte * "*")
print(f"* {tekst} *")
print(lengte * "*")
print()
```

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

Het resultaat ziet er dan als volgt uit:



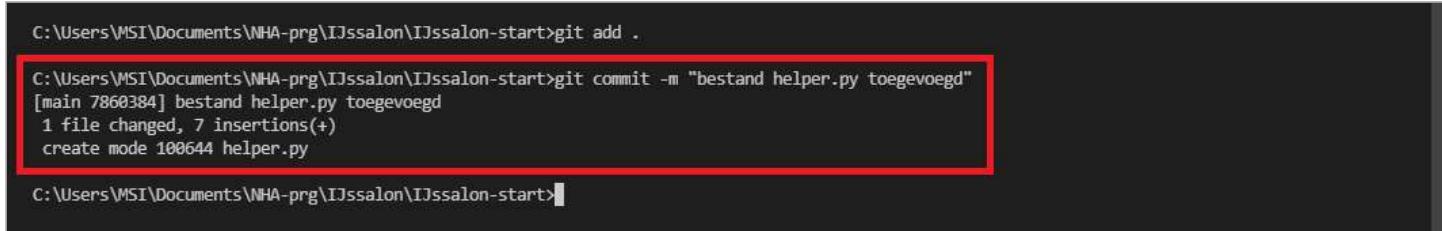
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1526]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>python helper.py
*****
* header *
*****
```

In de volgende stap geeft u aan dat u deze wijzigingen definitief wil opslaan:

- Ga naar een nieuwe regel in de *Terminal*.
- Typ `git add .` en druk op Enter. (Vergeet de `.` niet!)
- Typ `git commit -m "bestand helper.py toegevoegd"` en druk op Enter.

De volgende boodschap verschijnt:

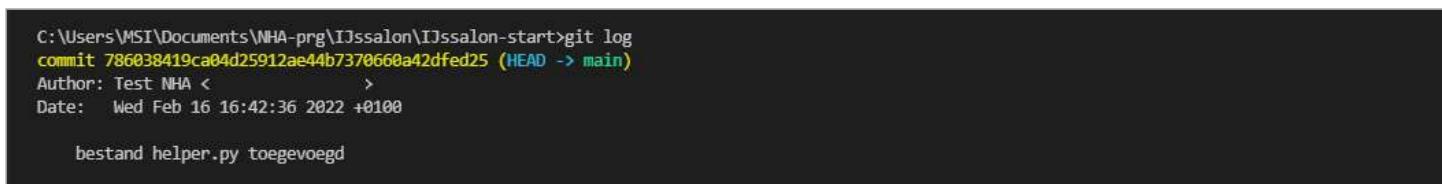


```
C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>git add .
C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>git commit -m "bestand helper.py toegevoegd"
[main 7860384] bestand helper.py toegevoegd
 1 file changed, 7 insertions(+)
 create mode 100644 helper.py
```

Er is nu een nieuwe versie aangemaakt van het project. Of dat goed is gegaan, kunt u op de volgende manier controleren:

- Ga naar een nieuwe regel in de *Terminal*.
- Typ `git log` en druk op Enter.

De laatste versie verschijnt bovenaan:



```
C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>git log
commit 786038419ca04d25912ae44b7370660a42dfed25 (HEAD -> main)
Author: Test NHA <           >
Date:   Wed Feb 16 16:42:36 2022 +0100

  bestand helper.py toegevoegd
```



Het kan zijn dat u op de Q-toets van uw toetsenbord moet drukken om verder te kunnen gaan.

Ten slotte gaat u uw *remote repo* als volgt op GitHub updaten:

- Ga naar een nieuwe regel in de *Terminal*.
- Typ `git push origin main` en druk op Enter.

U ziet ongeveer de volgende boodschap in de *Terminal*:

```
C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 425 bytes | 70.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/testNHA/ijssalon_ellie
  5e6376c..7860384 main -> main

C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>
```



Misschien is het u opgevallen: in les 4 typten we alleen `git push`, hier typen we `git push origin main`. Het commando `git push` is het basiscommando om uw lokale *repo* te uploaden naar *GitHub*. Zolang de *remote repo* op *GitHub* de enige versie van uw project is, kunt u dit commando probleemloos gebruiken. Dat wordt anders wanneer u meerdere versies / branches / ... van hetzelfde project heeft. Als u dan `git push` gebruikt, zou u uw lokale *repo* naar een andere *remote repo* kunnen uploaden. In dat geval is `git push origin main` beter, want het is specifieker. Dit commando geeft precies aan naar welke *remote repo* de lokale *repo* gekopieerd moet worden.

7.12 Samenvatting

In deze les heeft u zich voornamelijk gericht op het bewerken en gebruiken van *strings*. U heeft *strings* samengevoegd, gesplitst en *substrings* gevormd. Ook heeft u gezien hoe handig *formatted strings* kunnen zijn en dat het gebruik ervan de voorkeur verdient boven het 'optellen' van *strings* en variabelen.

Ook heeft u een begin gemaakt met het praktijkproject: u helpt Ellie met het bouwen van haar online ijssalon. Dankzij dit project kunt u alles wat u in deze lessen geleerd heeft en nog gaat leren, toepassen in de praktijk.

U bent begonnen met het klonen van een bestaande *repository*, waarna u een remote *repository* heeft aangemaakt op *GitHub*. Daarnaast heeft u het bestand 'helper.py' toegevoegd aan zowel de lokale als de online *repo*.

7.13 Vraagmoment

Heeft u vragen over deze les? Dan kunt u die stellen via een vraagmoment.

U herkent een vraagmoment aan het spreekballonnetje dat op Plaza in het overzicht "Lessen" bij een les staat. Door op deze spreekballoon te klikken, komt u op een nieuwe pagina, waar u uw vraag in het tekstvak kunt typen.



Het tekstvak mag maximaal vijfhonderd tekens bevatten. Zorg er dus voor, dat u uw vraag/vragen bondig formuleert.

Nadat u op "Verstuur vraag" heeft geklikt, wordt uw vraag automatisch naar uw docent verstuurd. Zodra uw docent de vraag heeft beantwoord, verschijnt er op uw dashboard bij de opleiding een spreekballoon.

U kunt per les één keer vragen stellen. Verzamel dus uw vragen over de les, voordat u deze instuurt. Als u gebruik heeft gemaakt van het vraagmoment, verdwijnt het spreekballonnetje bij de les.

7.14 Oefenopgaven

De volgende opgaven werkt u als oefenopgaven voor uzelf uit. De uitwerkingen van deze opgaven kunt u meteen zelf controleren.

Bekijk de volgende code:

```
for c in "ijshoorntje":
    print(c)
```

Wat is het resultaat als u deze code uitvoert?

Bekijk antwoord

Bekijk de volgende code:

```
print("20" + "20")
```

Wat is het resultaat als u deze code uitvoert?

→ Bekijk antwoord

Bekijk de volgende code:

```
print("Mijn favoriete smaak is aardbei"[5:10])
```

Wat is het resultaat als u deze code uitvoert?

→ Bekijk antwoord

Bekijk de volgende code:

```
print("Mijn favoriete smaak is aardbei"[21:])
```

Wat is het resultaat als u deze code uitvoert?

→ Bekijk antwoord

Stel, u heeft een variabele `smaak` met de waarde `citroen`.

U wil de volgende zin printen:

Ik houd van <smaak>ijs

Daar waar nu `<smaak>` staat, moet uiteindelijk de waarde van de variabele `smaak` komen te staan. Welke code moet u uitvoeren om dit voor elkaar te krijgen?

→ Bekijk antwoord

Bekijk de volgende code:

```
print("IJshoorntje".swapcase())
```

Wat is het resultaat als u deze code uitvoert?

→ Bekijk antwoord

Stel, u wil een *list* creëren genaamd `woorden`. Deze *list* bevat alle woorden die de zin `Mijn favoriete smaak is aardbei` bevat als losse *string*-elementen. Welke code moet u uitvoeren om dit voor elkaar te krijgen?

→ Bekijk antwoord

Bekijk de volgende code:

```
print("Mijn favoriete smaak is aardbei".index("smaak"))
```

Wat is het resultaat als u deze code uitvoert?

→ Bekijk antwoord

Stel, u wil de volgende zin printen:

Mijn favoriete smaak is aardbei

Daar waar nu aardbei staat, moet echter citroen komen te staan. Welke code moet u uitvoeren om dit voor elkaar te krijgen?

→ Bekijk antwoord

Bekijk de volgende code:

```
woord = "chocolade"
aantal = _____(woord)
print(f"Er zitten {aantal} letters in het woord {woord}.")
```

Wat hoort op de open plaats (_____) in deze code?

→ Bekijk antwoord

Bekijk de volgende code:

```
print("Mijn favoriete smaak is aardbei".index("Mijn"))
```

Wat is het resultaat als u deze code uitvoert?

→ Bekijk antwoord

Bekijk de volgende code:

```
zin1 = "Dit is een aardbei"
zin2 = zin1.swapcase()
zin3 = zin1 + " " + zin2
print(zin3)
```

Wat is het resultaat als u deze code uitvoert?

→ Bekijk antwoord

7.15 Huiswerkopgaven

Maak deze huiswerkopgaven en stuur ze via Plaza naar uw docent. Deze opgaven worden nagekeken en voorzien van een cijfer. Mocht u een onvoldoende cijfer behalen, dan krijgt u een herkansing.

De volgende stappen vormen samen de huiswerkopgaven:

1. 1. Voeg een nieuw bestand toe aan de map 'IJssalon-start'. Geef deze map de naam `tijdelijk.py`.
2. Plaats in het bestand '`tijdelijk.py`' een *dictionary* genaamd `prijzen`. Zorg ervoor dat deze *dictionary* de volgende *key-value pairs* bevat:

Key	Value
aardbei	3
vanille	4
chocolade	5

3. Plaats in het bestand 'tijdelijk.py' een variabele met de naam aanbieding. Deze variabele heeft de waarde van het dictionary-element vanille, vermenigvuldigd met 0.8.

4. Plaats in het bestand 'tijdelijk.py' een variabele met de naam reclame_tekst. Deze variabele heeft de volgende zin als waarde:

Vandaag in de aanbieding: vanille-ijs, 1 liter - slechts € <aanbieding>

Daar waar nu <aanbieding> staat, plaatst u de variabele met de naam aanbieding. Dat doet u door het maken van een *formatted string*. Voeg ook een print-statement toe dat de waarde van de variabele reclame_tekst uitvoert naar het scherm.

5. Als u de variabele reclame_tekst uitprint, ziet u dat die er een beetje raar uitziet:

```

PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1526]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\WHA-prg\IJssalon\IJssalon-start>python tijdelijk.py
Vandaag in de aanbieding: vanille-ijs, 1 liter - slechts € 2.4000000000000004

C:\Users\MSI\Documents\WHA-prg\IJssalon\IJssalon-start>

```

Er zijn meerdere oplossingen mogelijk. U verwijdert de print-opdracht uit de code en creëert een extra variabele met de naam reclame_tekst2. Die variabele geeft u de waarde `reclame_tekst[:_____]`. Op de open plek na de dubbele punt plaatst u de index die hoort bij de eerste 0 na de komma.

Print daarna de waarde van deze nieuwe variabele `reclame_tekst2`.

6. Een oom van Ellie heeft een sportvliegtuigje, waarachter hij een spandoek kan hangen. Dan moeten de letters van de reclametekst wel allemaal in hoofdletters.

Verwijder de print-opdracht uit de code en voeg een extra variabele toe met de naam `reclame_tekst3`. Die variabele heeft als waarde dezelfde tekst als de variabele `reclame_tekst2`, maar dan in hoofdletters.

Print daarna de waarde van deze nieuwe variabele `reclame_tekst3`.

7. Voor de drukker van het spandoek is het handiger als de reclametekst voor achter het vliegtuigje een *list* van woorden is.

Verwijder de print-opdracht uit de code en voeg een extra variabele toe met de naam `reclame_tekst4`. Die variabele heeft als waarde een *list* van alle woorden van de *string* `reclame_tekst3`.

Print daarna de waarde van deze nieuwe variabele `reclame_tekst4`.

8. Ellie wil zien hoe de woorden eruitzien als ze onder elkaar gedrukt worden. Dat zou goed kunnen werken voor een flyer die ze in de buurt wil verspreiden.

Verwijder de print-opdracht uit de code en creëer een *for-loop* die door de elementen van de *list* `reclame_tekst4` itereert. Gebruik de naam `e1` als variabele voor de *for-loop*.

Print daarna alle elementen op het scherm.

9. Het resultaat van de print-opdracht staat in hoofdletters. Voor de flyer wil Ellie echter liever geen hoofdletters. Pas daarom de *string-method* `.lower()` toe op de variabele `e1`.
10. Ellie is nog niet helemaal tevreden, het kan nog beter. Ze wil dat alleen de elementen die vijf of meer karakters hebben in hoofdletters worden geprint (en bij vier of minder karakters in kleine letters).

Voeg een *if-statement* toe aan de *for-loop* die de lengte van de variabele `e1` controleert en het juiste formaat print.
11. Maak een nieuwe lokale versie van het project aan. Gebruik daarvoor twee commando's, waarbij u bij het tweede commando deze tekst tussen aanhalingstekens plaatst:
`"bestand tijdelijk.py gecodeerd"`
12. Update uw *remote repo* op *GitHub*.
13. Stuur de URL van uw *remote repo* op *GitHub* via Plaza naar uw docent. Die kan dan niet alleen het eindresultaat zien, maar ook alle tussentijdse wijzigingen.