

Les 11 - Frameworks

[« Vorige les](#)[» Volgende les](#)

Huiswerk

Programmeren voor Beginners - Les 11

Je hebt je huiswerk nog niet ingediend.

[Huiswerk inzenden](#)

Je hebt 1 vraagmoment. 1 moment over.

[Ga naar stel een vraag](#)[!\[\]\(d3102649f02e825ddb76dc3de0190154_img.jpg\) Markeer als deelgenomen aan deze les](#)

Inhoudsopgave

- 11.1** Inleiding
- 11.2** Wat is een framework?
- 11.3** Python-frameworks
- 11.4** Werken met Flask
- 11.5** Repository forken
- 11.6** Flask installeren
- 11.7** De map 'templates'
- 11.8** De map 'static'
- 11.9** Het hoofdbestand 'flask_app.py'
- 11.10** Routes
- 11.11** Routingfuncties
- 11.12** De functie 'render_template()'
- 11.13** Jinja
- 11.14** Project 'IJssalon'
- 11.15** Samenvatting
- 11.16** Vraagmoment
- 11.17** Oefenopgaven
- 11.18** Huiswerkopgaven

11.1 Inleiding

In de vorige les heeft u kennismoecht met **pakketten**: kant-en-klare uitbreidingen die u vrij kunt gebruiken in uw Python-project. Deze zorgen ervoor dat u een deel van uw programma niet zelf hoeft te schrijven.

Er zijn pakketten die samen een structuur vormen die dusdanig omvangrijk is, dat we spreken van **frameworks**. Dit pakket bevat de basis voor uw project, ideaal voor een beginner. Veel van de code is namelijk al voor u geschreven en vaak hoeft u voor een werkend resultaat alleen maar een aantal zaken aan te passen/in te stellen en wat van uw eigen code toe te voegen.

Omdat de taal Python in deze cursus centraal staat, richten we ons in deze les op Python-frameworks. De meeste hiervan richten zich op het ontwikkelen van websites.

We gaan werken met het framework Flask. Hiermee ontdekt u wat een framework is, hoe het werkt en hoeveel tijd u kunt besparen door ervan gebruik te maken.

Leerdoelen

Aan het einde van deze les weet u:

- wat een framework is en welke frameworks er zoal bij een bepaalde taal horen;
- wat een framework u zoal kan bieden;
- het verschil tussen Django en Flask (en Pyramid);
- wat u moet doen om het Flask-framework te kunnen gebruiken;
- het verschil tussen een *repository* klonen en forken (en hoe u dat laatste doet);
- hoe u Flask installeert;
- wat routes zijn en hoe u ze aanmaakt;
- hoe u de Flask-server opstart;
- hoe u de functie `render_template()` gebruikt;
- wat Jinja is;
- hoe u een *list* vult met een *dictionary* (en wat het nut daarvan is).



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.

11.2 Wat is een framework?

We starten deze les met een voorbeeld:

- Ga naar www.youtube.com.
- Speel een willekeurige video af.

Wanneer u een van de video's wil bekijken, wordt deze content uit een database gehaald en aan u getoond. Dat vindt plaats dankzij een programmacode.

Zo'n code is, zeker voor beginners, ingewikkeld en vaak omvangrijk. Gelukkig bestaan er frameworks die het schrijven van zo'n programmacode (maar ook andere codes) aanzienlijk gemakkelijker maken. U kunt een **framework** namelijk zien als een raamwerk dat u kunt gebruiken als uitgangspunt voor het schrijven van een programmacode. Het frame bevat een basiscode, waaraan u code toevoegt.

Een framework is altijd geschreven in een bepaalde taal. De meeste gangbare programmeertalen kennen een eigen framework, sommige zelfs meerdere. We geven hiervan enkele voorbeelden:

Programmeertaal	Framework
C#	.NET
	Visual C#
	WPF
C++	Qt
	Boost
	Visual C++
CSS	Bootstrap
	Foundation
	Tailwind
Java	Spring
	Apache Wicket
	Grails
JavaScript	React.js
	Vue.js
	jQuery
PHP	Laravel
	Symfony
	CakePHP
Python	Django
	Flask
	Pyramid
	Enthought Tool Suite
Ruby	(Ruby on) Rails
	Scorched
	Cuba
Swift	Quick
	Vapor
	SwiftMonkey

Het doel van al deze (en andere) frameworks is het bieden van een gemeenschappelijke structuur en werkwijze met herbruikbare stukjes code, zodat programmeurs niet telkens vanaf nul hoeven te starten. Een framework neemt u dus veel werk uit handen, waardoor u veel tijd (en dus geld) bespaart.

Welke stukjes code een framework u precies biedt, verschilt per framework. Een framework kan bestaan uit verschillende componenten, zoals:

- ondersteuningsprogramma's;
- compilers;
- codebibliotheeken;
- tool sets;
- API's.

Al deze onderdelen moet u zien als hulpmiddelen, die u helpen applicaties te ontwikkelen en systemen te creëren.

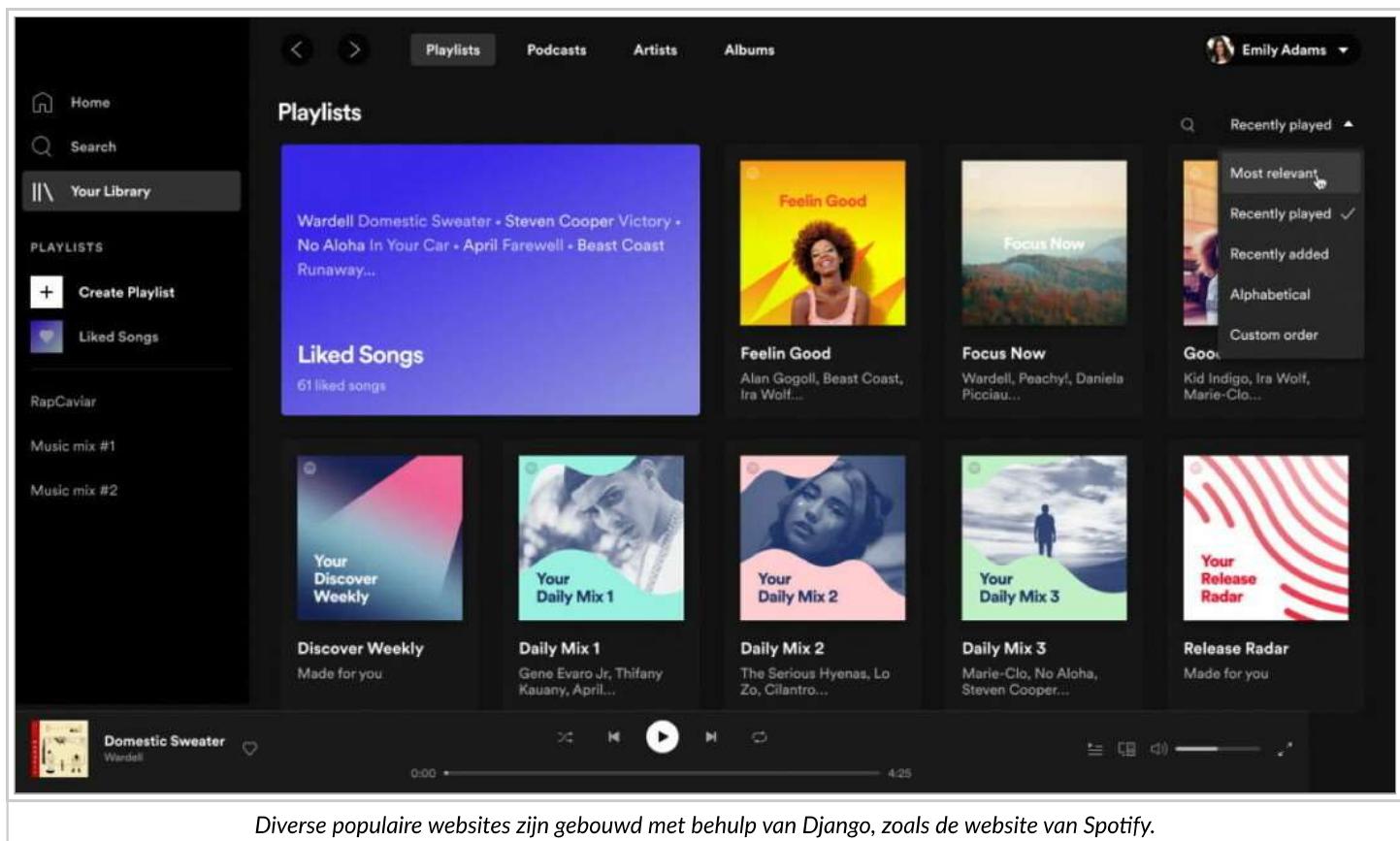
11.3 Python-frameworks

In deze cursus programmeert u met Python. Voor deze programmeertaal zijn vrij veel frameworks in omloop. In deze paragraaf besteden we extra aandacht aan enkele van deze frameworks.

11.3.1 Django

Django is een framework dat gebaseerd is op de programmeertaal Python. Het framework werd in 2003 ontwikkeld door Simon Willison en Adrian Holovaty. Die laatste was een groot fan van gitarist Django Reinhardt, die door een brand een aantal vingers niet meer kon gebruiken. Dit weerhield hem er niet van om een nieuwe speelstijl te ontwikkelen, waarbij hij meestal slechts twee vingers gebruikte. Holovaty gaf het framework de naam Django, omdat het volgens hem zo gemakkelijk in het gebruik was, dat men niet meer dan twee vingers nodig zou hebben.

In 2005 werd het framework voor het eerst vrijgegeven als een opensourceproject, waardoor het gratis te gebruiken is. Sindsdien is het gegroeid tot het meest gebruikte Python-framework. Vele populaire websites zijn gebouwd met dit framework, zoals YouTube, Spotify en Dropbox.



Diverse populaire websites zijn gebouwd met behulp van Django, zoals de website van Spotify.

Waardoor is Django zo populair geworden?

- Het framework heeft niet alleen een hoge ontwikkelsnelheid (Django is eenvoudig op te zetten), maar ook een hoge reactiesnelheid.
- Daarnaast staat Django bekend om zijn strakke ontwerpen.
- Deze ontwerpen zijn **responsive**: de applicaties passen zich aan aan de grootte van het scherm waarop ze bekeken worden.
- Django is modulair, zowel in de breedte als in de diepte. Dit maakt het framework flexibel, efficiënt en geschikt voor projecten van allerlei omvang.
- Door de vele gebruikers is er ook sprake van een grote **community**, dus er is veel hulp beschikbaar voor iemand die vragen of problemen heeft (zie hiervoor www.djangoproject.com). Daarnaast zorgt deze grote community ervoor dat het framework continu in ontwikkeling is.
- En uiteraard: het is gratis!

11.3.2 Flask

Naast Django is ook Flask een opensourceframework dat vaak wordt gekozen door Python-programmeurs. De ontwikkeling van het framework was in eerste instantie een 1 aprilgrap, maar werd uiteindelijk serieus opgepakt door een groep programmeurs genaamd Pocco (onder leiding van Armin Ronacher).

Het framework werd uiteindelijk in 2010 uitgebracht. Ook dit framework is gebruikt bij de bouw van vele populaire websites, zoals Netflix, Pinterest en LinkedIn.



Diverse populaire websites zijn gebouwd met behulp van Flask,
zoals de website van Netflix.

Waardoor is Flask zo populair geworden?

- Volgens programmeurs is het eenvoudig om Flask onder de knie te krijgen en kan men er al snel een eenvoudige website mee opzetten.
- Het wordt gezien als een 'lichtgewicht' framework. Hiermee doelt men op het feit dat de basis van het framework eenvoudig is, wat als voordeel heeft dat u als programmeur meer zelf mag (maar ook moet) bepalen.
- Ook Flask is gratis te gebruiken.

Later in deze les zullen we dieper ingaan op dit framework: u gaat een volledig functionerende website programmeren met Flask als basis.

11.3.3 Pyramid

Django en Flask zijn verreweg de populairste Python-frameworks voor het bouwen van een website of webapplicatie. Er zijn echter meer frameworks die interessant voor u kunnen zijn.

Een van die frameworks is Pyramid. Dit framework zit qua toepassing tussen Django (wat complexer, geschikt voor grotere websites/webapplicaties) en Flask (wat minimalistischer, geschikt voor een vrijere invulling) in. Bij de start van een Pyramid-project worden namelijk meer basisbestanden aangemaakt dan bij Flask, maar minder dan bij Django.

Ook Pyramid is een opensourceframework en dus gratis te gebruiken.



Wilt u meer informatie over de verschillen tussen Django, Flask en Pyramid? Lees dan [dit \(Engelstalige\) artikel](#).

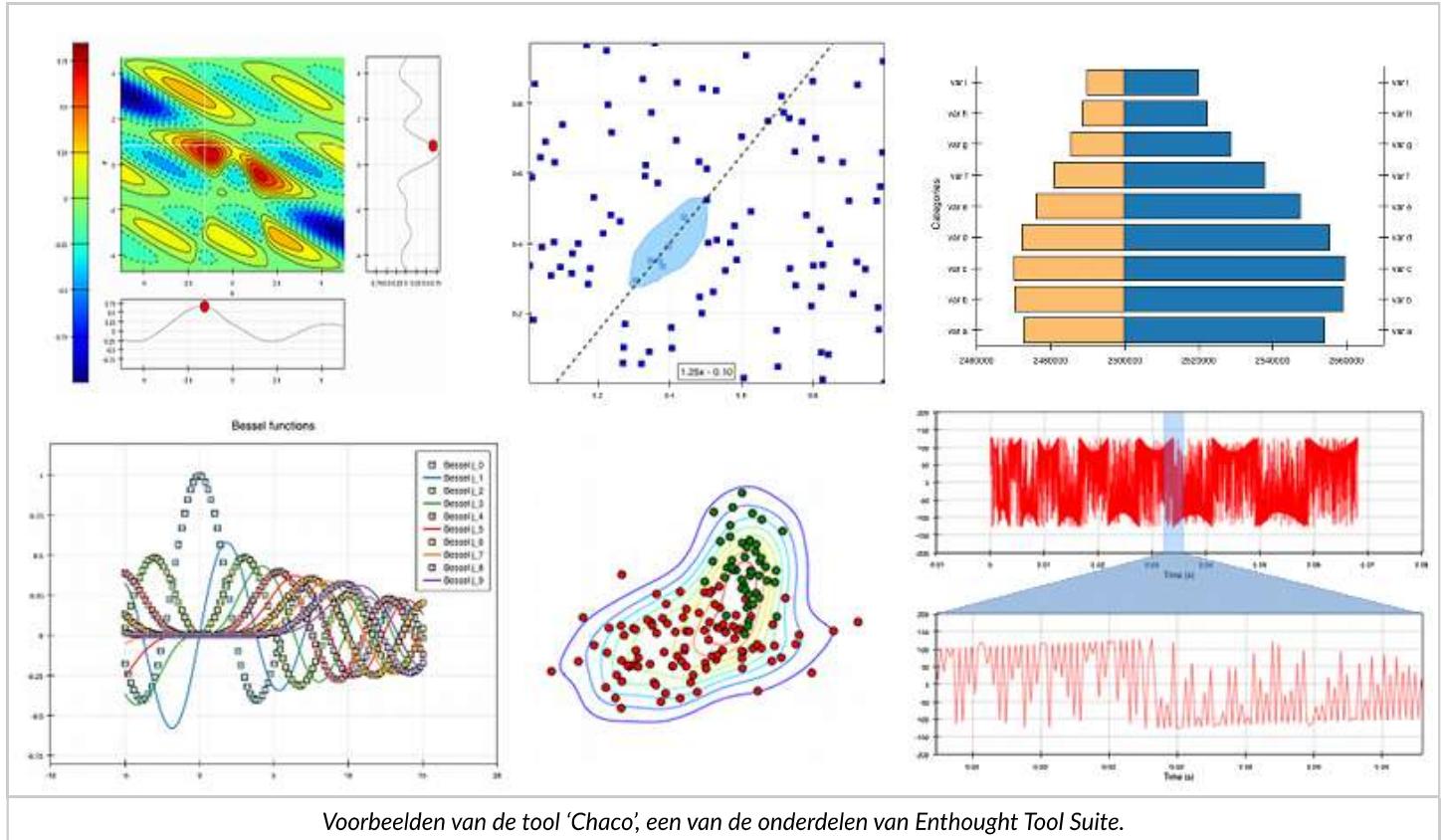
11.3.4 Enthought Tool Suite

Als laatste voorbeeld van een Python-framework noemen we het wat minder bekende Enthought Tool Suite (ETS).

Zoals we eerder vermeldden, worden de meeste Python-frameworks ingezet om webapplicaties te vervaardigen. Enthought Tool Suite is een voorbeeld van een framework dat gebruikt wordt voor een ander doel: het is gemaakt voor data-analisten.

Enthought Tool Suite is een collectie van *software tools*. Daarnaast bevat het een framework dat de basis legt voor het ontwikkelen van een computerapplicatie (die dus normaliter niet als webapplicatie gepubliceerd wordt)

Het framework legt de basis voor een applicatie die geschikt is voor het omgaan met grote hoeveelheden data: een software-eigenschap die voor data-analisten onmisbaar is!



11.4 Werken met Flask

We gaven het al eerder aan: volgens programmeurs is het eenvoudig om Flask onder de knie te krijgen en kunt u er al snel een eenvoudige website mee opzetten. We gaan dit framework daarom toevoegen aan ons 'project IJssalon'.

Het framework Flask heeft eigenlijk maar vier dingen nodig om een set van bestanden te veranderen in een heuse website:

1. De installatie van het pakket 'Flask'
2. Een map genaamd 'templates'
3. Een map genaamd 'static'
4. Een hoofdbestand, meestal genaamd 'app.py' of 'flask_app.py'

Omdat de templatebestanden (die de uiteindelijke webpagina's gaan vormen) geschreven zijn in HTML (en niet in Python), mogen we niet van u verwachten dat u deze zelf schrijft. Doordat u kunt werken met *GitHub* en *Git-repositories*, kunnen we u echter op eenvoudige wijze voorzien van deze bestanden.

Eerder in deze cursus leerde u hoe u een *repository* kunt **klonen**. In deze les gaat u echter niet klonen, maar **forken**. Er is een belangrijk verschil tussen deze twee handelingen:

- Bij **forken** maakt u een eigen kopie van de *repository*.
- Bij **klonen** blijft uw kopie altijd verbonden met de originele *repository*.

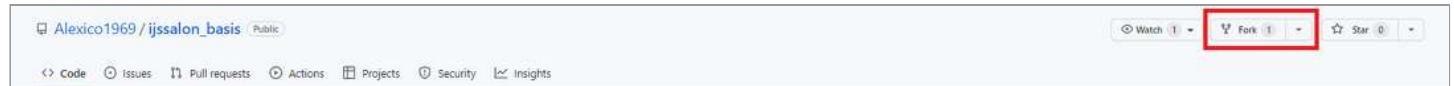
Omdat veel cursisten de *repository* zullen gebruiken, is klonen in dit geval niet wenselijk. We beginnen de volgende paragraaf daarom met het forken van een *repository*.

11.5 Repository forken

In deze paragraaf leggen we u stap voor stap uit hoe u een bestaande *repository* kunt forken.

- Ga naar www.github.com.
- Klik rechtsboven op “Sign in”.
- Log in met uw gegevens.
- Open een nieuw browservenster (of tabblad).
- Ga naar https://github.com/Alexico1969/ijssalon_basis.

Rechtsboven in beeld ziet u nu de tekst “Fork” staan:



- Klik op deze knop.

Alle instellingen die nu in beeld ziet, zijn al de correcte instellingen. U kunt daarom direct bevestigen dat u de *repository* wilt forken.

- Klik op de groene knop “Create fork”.

Er is nu een kopie van de *repository* aangemaakt. Deze kunt u als *repository* vinden in uw eigen account.

- Sluit de pagina https://github.com/Alexico1969/ijssalon_basis.
- Laat uw eigen GitHub-pagina nog openstaan.

De volgende stap is het maken van een lokale kopie van de zojuist geforkte *repository*. Dat doet u als volgt:

- Open VS Code.
- Sluit (indien nodig) alle bestanden, mappen en projecten die nog openstaan.
- Klik in de Activity Bar op de knop “Explorer”.
- Klik op “Clone Repository”.

Boven in het scherm wordt u gevraagd om de URL van de te klonen *repository* in te voeren. Die vindt u op uw GitHub-pagina.

- Ga naar de browser.
- Ververs de pagina.
- Klik links in beeld op de naam van de *repository*.

The home for all developers — including you.

Welcome to your personal dashboard, where you can find an introduction to how GitHub works, tools to help you build software, and help merging your first lines of code.

Start writing code

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

testNHA / name your new repository...

Public Anyone on the internet can see this repository

Private You choose who can see and commit to this repository

Create a new repository

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

testNHA / README.md Create

1 - 🧑 Mi, I'm @testNHA
2 - 🌐 I'm interested in ...
3 - 🎓 I'm currently learning ...
4 - 🚗 I'm looking to collaborate on ...
5 - 📩 How to reach me ...
6 -

Universe2022

Let's build from here

Join the global developer event for cloud, security, community, and AI.

Register today and get a 20% off early bird discount.

GitHub Copilot

Get suggestions for lines of code and entire functions in real-time

Learn more about Copilot

Klik op de naam van de repository.

- Klik op de groene knop "Code".
- Kopieer de repository-URL.

Code

main · 1 branch · 0 tags

This branch is up to date with Alexico1969/ijssalon_basis:main.

Alexico1969 initial commit

- static initial commit
- templates initial commit
- .gitignore initial commit
- algemene_functies.py initial commit
- helper.py initial commit
- reclame.py initial commit
- tijdelijk.py initial commit

Code

Clone

HTTPS SSH GitHub CLI

You don't have any public SSH keys in your GitHub account. You can add a new public key, or try cloning this repository via HTTPS.

git@github.com:testNHA/ijssalon_basis.git

Use a password-protected SSH key.

Open with GitHub Desktop

Open with Visual Studio

About

Basisproject voor Project IJssalon - NHA

0 stars

0 watching

2 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Kopieer de repository-URL.

- Ga terug naar VS Code.
- Plak de code in het invoerveld waar nu Provide repository URL or pick a repository source staat.
- Druk op Enter.

Er wordt u nu gevraagd waar u de lokale kopie wil opslaan.

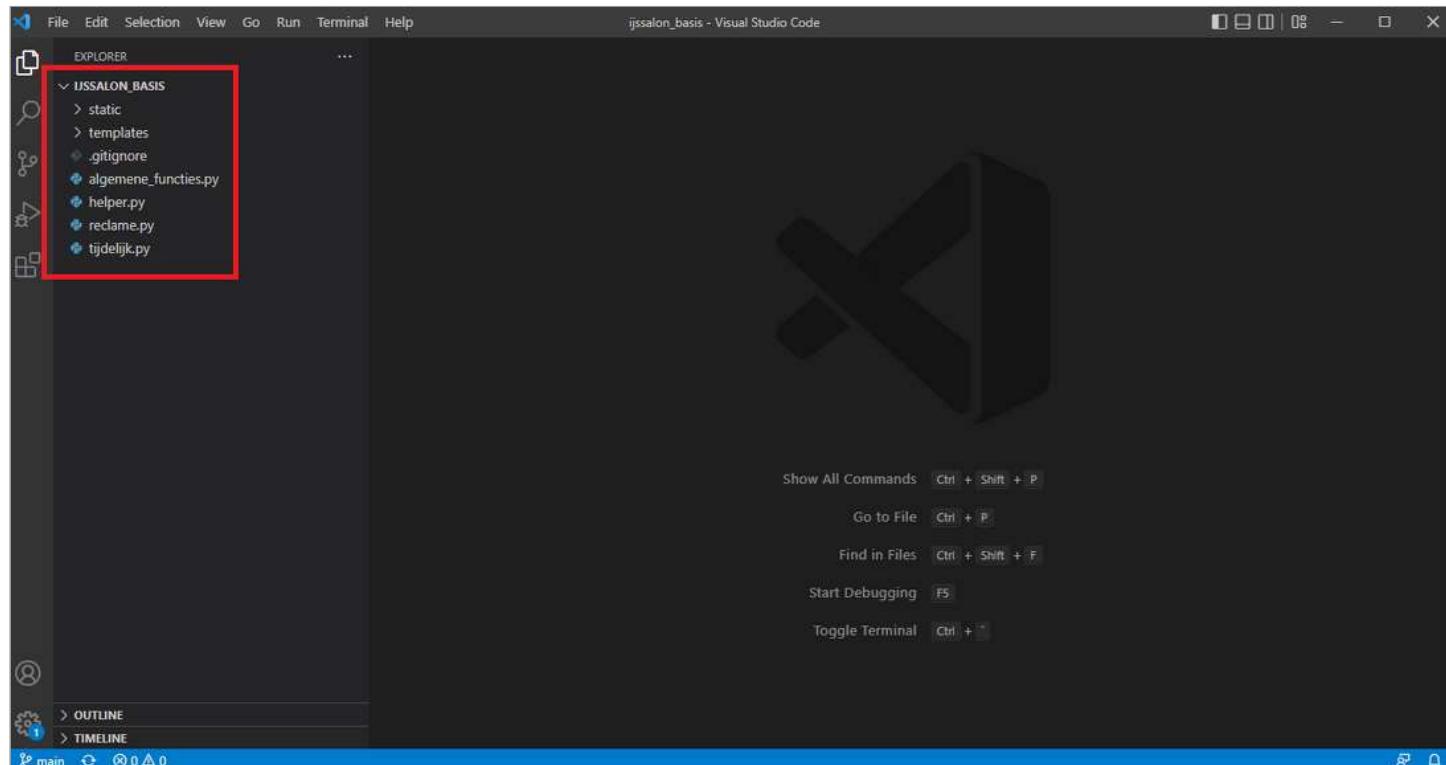
- Selecteer een locatie op uw harde schijf.
- Klik op "Select Repository Location".

U krijgt nu de vraag of u de lokale repository wil openen in VS Code:



- Klik op "Open".

Als het klopt, ziet u deze mappen en bestanden nu geopend in VS Code:



11.6 Flask installeren

Een framework als Flask wordt niet standaard geïnstalleerd op het moment dat u Python installeert. Dat zou ook vreemd en inefficiënt zijn, want een programmeur die helemaal niet van plan is met Python een website te maken, heeft die code helemaal niet nodig.

Omdat wij wél een website willen vervaardigen met behulp van Flask, gaan we het framework installeren. Dat doet u als volgt:

- Ga naar VS Code.
- Open de Terminal.
- Typ `pip install Flask` en druk op Enter.

Het duurt even voordat er output in de *Terminal* verschijnt, maar uiteindelijk zal duidelijk worden dat de installatie is gestart. Zodra de installatie is afgerond, ziet u een melding in de *Terminal*.

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

pip + ×

```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\ijssalon_basis>pip install Flask
Collecting Flask
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
    ██████████ | 101 kB 1.6 MB/s
Collecting importlib-metadata>=3.6.0; python_version < "3.10"
  Downloading importlib_metadata-5.0.0-py3-none-any.whl (21 kB)
Collecting Werkzeug>=2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
    ██████████ | 232 kB 2.2 MB/s
Collecting Jinja2>=3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
    ██████████ | 133 kB 1.1 MB/s
Collecting click>=8.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    ██████████ | 96 kB 1.2 MB/s
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting zipp>=0.5
  Downloading zipp-3.9.0-py3-none-any.whl (5.8 kB)
Collecting MarkupSafe>=2.1.1
  Downloading MarkupSafe-2.1.1-cp39-cp39-win_amd64.whl (17 kB)
Collecting colorama; platform_system == "Windows"
  Downloading colorama-0.4.5-py2.py3-none-any.whl (16 kB)
Installing collected packages: Jinja2, importlib-metadata, MarkupSafe, Werkzeug, Jinja2, colorama, click, itsdangerous, Flask
Successfully installed Flask-2.2.2 Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 colorama-0.4.5 importlib-metadata-5.0.0 itsdangerous-2.1.2 zipp-3.9.0
```

C:\Users\MSI\Documents\ijssalon_basis>

Zodra de installatie is afgelopen, ziet u ongeveer dit in beeld.

De versies kunnen iets verschillen, maar dat is geen probleem.

U kunt op de volgende manier controleren of Flask goed is geïnstalleerd:

- Ga naar een nieuwe regel in de Terminal.
- Typ `Flask --version` en druk op Enter.

U ziet dan ongeveer het volgende verschijnen als respons:

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

cmd + ×

```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\ijssalon_basis>Flask --version
Python 3.9.4
Flask 2.2.2
Werkzeug 2.2.2
```

C:\Users\MSI\Documents\ijssalon_basis>

Ook hier kunnen de versies iets verschillen, maar ook nu is dat geen probleem.

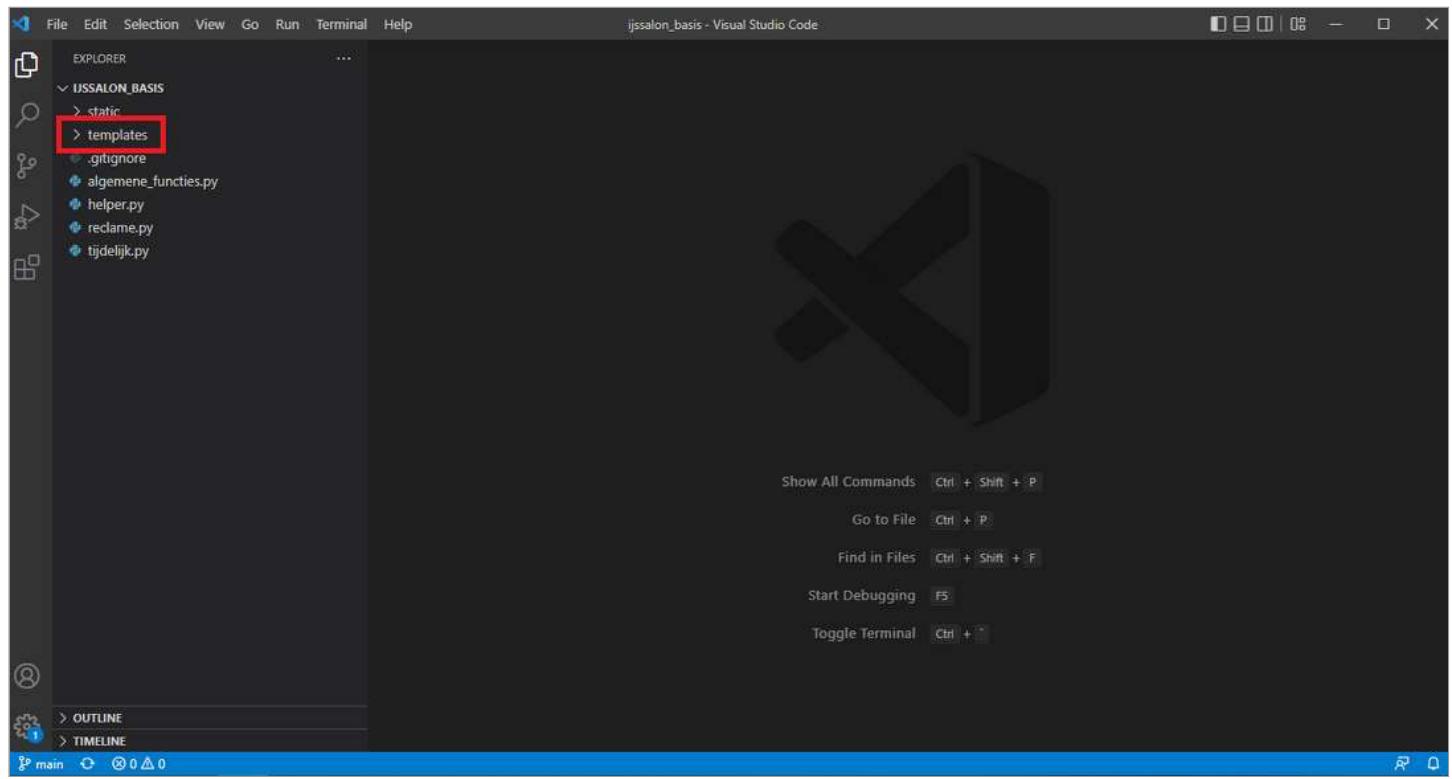


Werkzeug is een extra pakket dat automatisch met Flask wordt geïnstalleerd. Het bevat een aantal handige extra functies, die zich toespitsen op het werken met [WSGI](#).

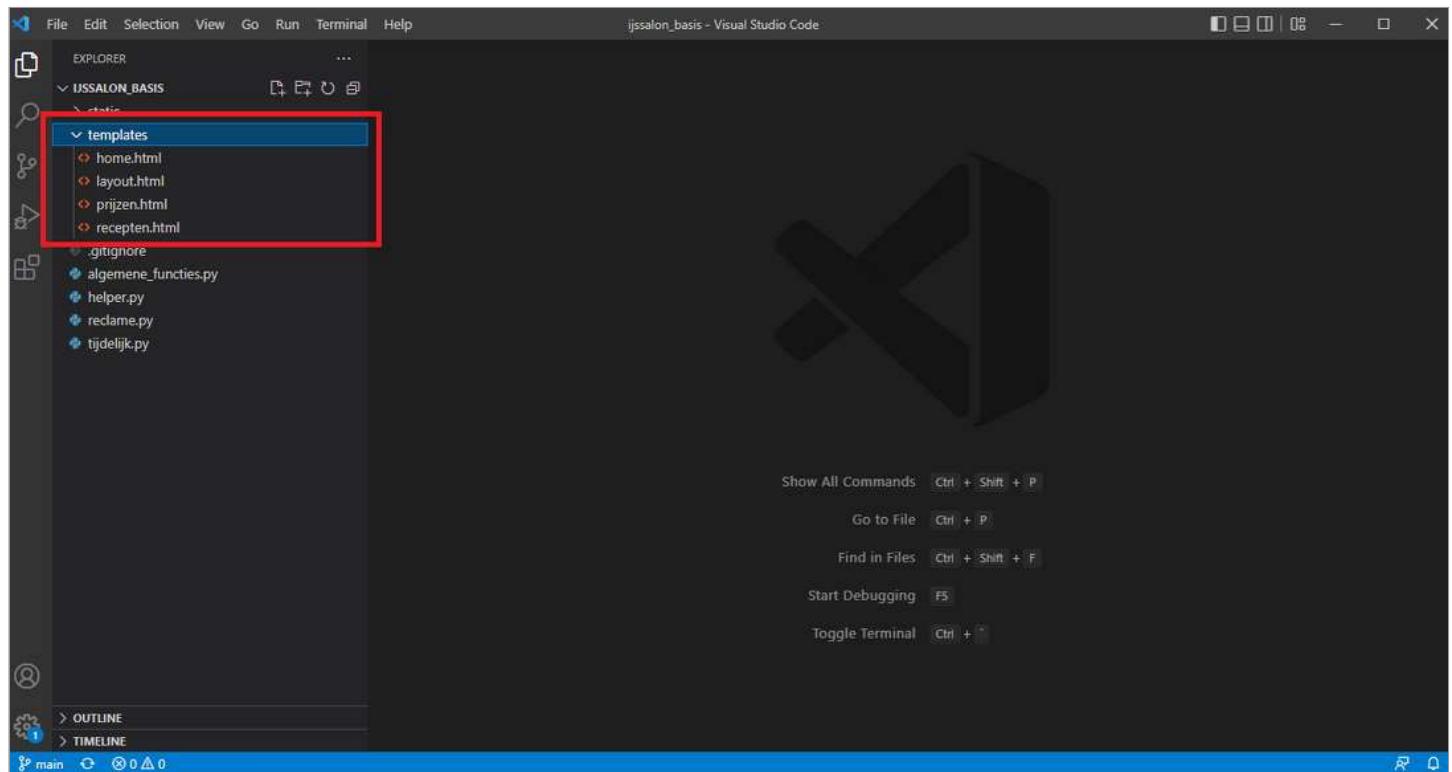
11.7 De map 'templates'

Wanneer u een website met het Flask-framework aanmaakt, heeft u een aantal standaardmappen nodig. Een daarvan is de map 'templates'.

Deze map is al voor u aangemaakt. Als u de repository correct heeft gekloond, ziet u deze map links in beeld in VS Code:



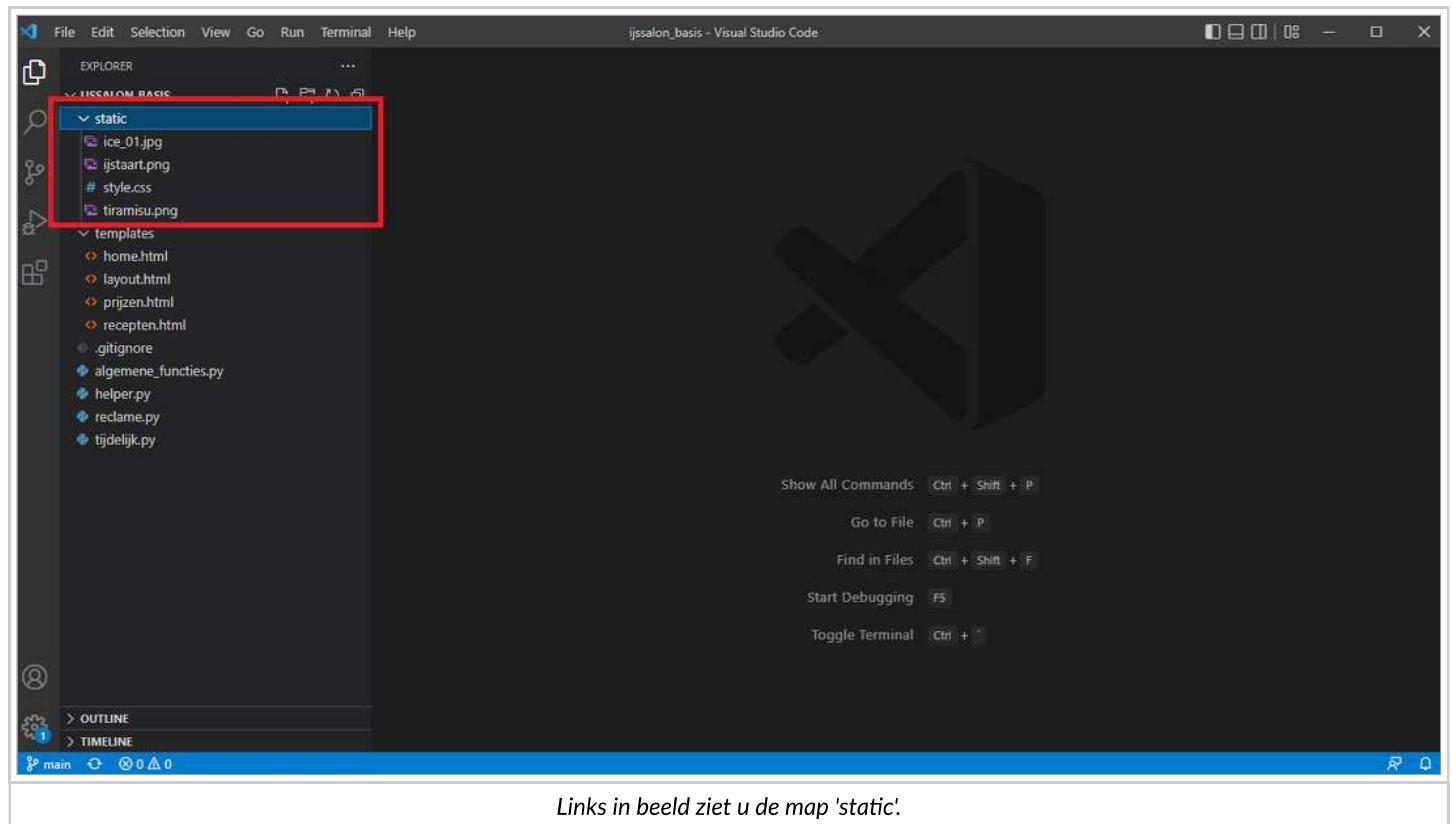
Als u de map openklapt, kunt u de inhoud ervan bekijken:



U ziet vier HTML-bestanden: onze *templates*. Het zijn de webpagina's die te zien zullen zijn op onze website. De inhoud ligt grotendeels vast, maar omdat we Flask gebruiken, kunnen we op de webpagina's bepaalde gegevens tonen die we met Python berekenen.

11.8 De map 'static'

Behalve de map 'templates' is er nog een map aanwezig: 'static'. Ook die ziet u links in beeld in VS Code en bevat al enkele bestanden.



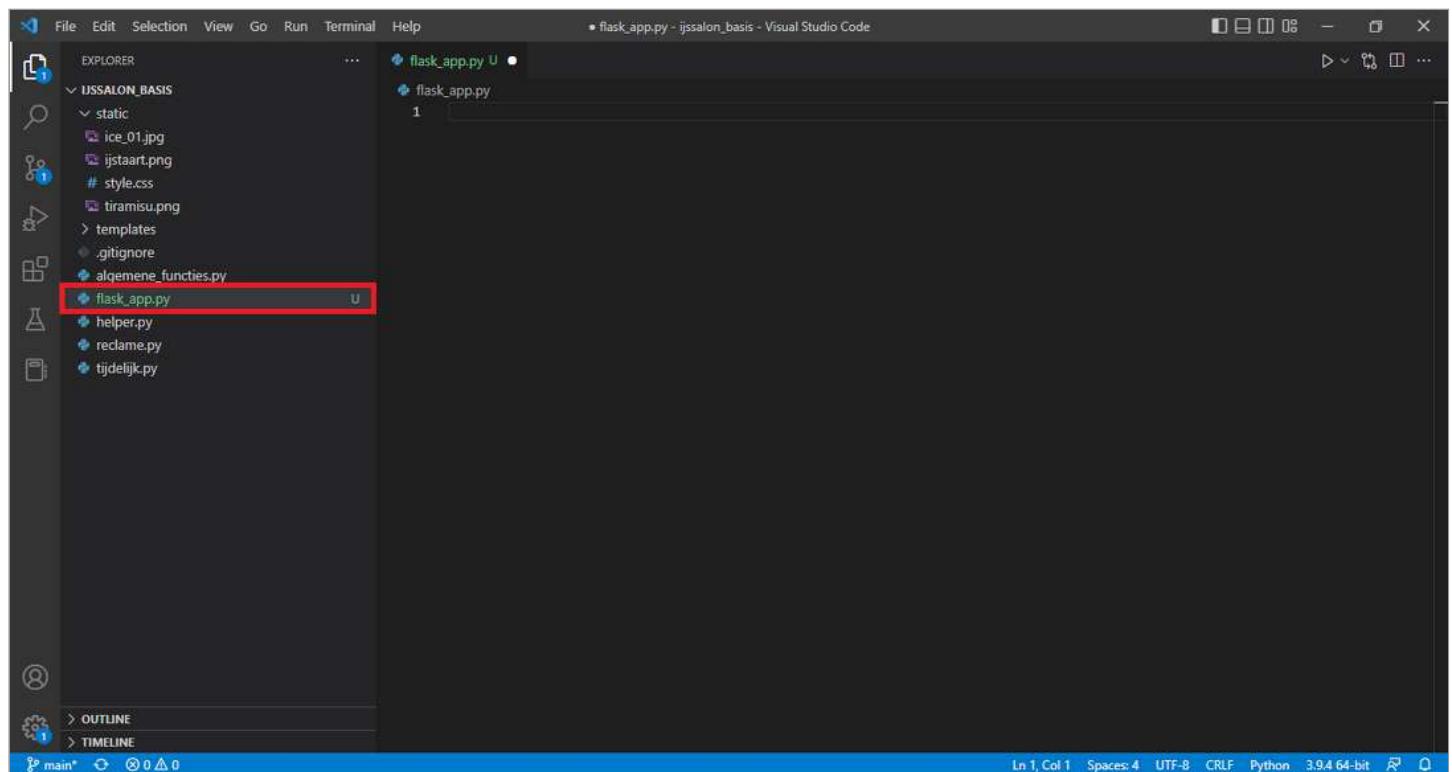
In Flask worden alle hulpbestanden in deze map geplaatst, zoals afbeeldingen, informatie over de opmaak van de templates (CSS-bestanden), geluidsbestanden en video's.

11.9 Het hoofdbestand 'flask_app.py'

Een ander vast onderdeel van ieder Flask-project is het hoofdbestand. Meestal heeft dit bestand de naam 'app.py' of 'flask_app.py'. Dit bestand is nog *niet* aanwezig en moet u nog aanmaken. Dat doet u als volgt:

- Klik in de Side Bar op de knop "New File".
- Geef het bestand de naam flask_app.py.

Het bestand verschijnt nu in de projectstructuur:





Controleer of dit bestand zich in de hoofdmap bevindt en niet in een van de submappen ('static' of 'templates')!

11.10 Routes

We kerden in deze paragraaf even terug naar het voorbeeld waarmee we deze les begonnen. Toen vroegen we u namelijk om naar www.youtube.com te surfen.

Doordat u naar www.youtube.com ging, kwam u op de homepage terecht. We hadden u echter ook kunnen vragen naar een specifieke pagina te surfen, bijvoorbeeld naar de pagina van NHA (www.youtube.com/user/NHA042010) of naar de pagina met 'te ontdekken video's' (www.youtube.com/feed/explore). Dit zijn pagina's binnen het domein 'youtube.com'.

De map 'templates' bevat, zoals gezegd, vier HTML-bestanden. Dit zijn de webpagina's die te zien zullen zijn op onze website. Elk bestand bevat de gegevens voor een aparte webpagina.

Stel, we zouden onze toekomstige website publiceren op www.mijnsite.nl. U zou dan verwachten dat u de pagina 'prijzen.html' kunt bezoeken door naar www.mijnsite.nl/prijzen.html te surfen. Zo werkt het echter niet. Daar is een goede reden voor.

Het Flask-framework zorgt ervoor dat u helemaal zelf kunt bepalen welke pagina's te zien zijn. Ook mag u zelf bepalen welk webadres moet worden ingevoerd om bij een bepaalde pagina uit te komen. Anders gezegd:

Elke pagina krijgt een eigen route.

De homepage noemen we ook wel de default-route. Daarnaast zijn er routes die naar extra pagina's leiden. Die routes moeten nog worden aangemaakt.

11.10.1 Default-route aanmaken

Als eerste maakt u de default-route aan. Zolang deze route niet aanwezig is, is de website niet toegankelijk voor bezoekers.

Voordat u de route kunt aanmaken in het hoofdbestand 'flask_app.py', moet u nog enkele zaken initialiseren. Allereerst importeert u een deel van het Flask-pakket (alleen de onderdelen die u gebruikt):

- Ga naar VS Code.
- Open het hoofdbestand 'flask_app.py'.
- Typ `from flask import Flask` en druk op Enter.

Vervolgens maakt u een variabele aan genaamd 'app':

- Typ `app=Flask(__name__)` en druk op Enter.

Omdat het hoofdbestand 'flask_app.py' zelfstandig uitgevoerd wordt en geen module is die weer door een ander Python-bestand wordt geïmporteerd, moet u de variabele 'app' gebruiken om de website te starten:

- Voeg de volgende twee regels toe aan het hoofdbestand:

```
if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

We geven een korte toelichting op deze code:

- De eerste regel checkt of dit inderdaad een programma is dat zelfstandig wordt uitgevoerd.
- Op de tweede regel gebruikt u de *method* `.run` om de website te starten:
 - Het deel `port=5000` is toegevoegd omdat de communicatie tussen onze website en de 'buitenwereld' plaatsvindt met behulp van een *port* (in ons geval de *port* genaamd 5000).
 - We starten onze website in de *debug*-modus: dat zal u helpen tijdens de ontwikkeling van de website. We geven dit aan door het toevoegen van het deel `debug=True`. Wanneer de website helemaal klaar is, dient `True` te worden veranderd in `False`.



Deze twee regels moeten altijd de laatste van het hoofdbestand zijn. Alles wat u later nog toevoegt aan deze code, moet u dus plaatsen tussen de eerste regel (`app=Flask(__name__)`) en deze laatste twee regels code.

Nu dit geregeld is, kunt u de default-route aanmaken:

- Voeg de volgende code toe aan het hoofdbestand:

```
@app.route('/')
def home():
    return "Welkom op mijn eerste Flask-website"
```

Ook deze code lichten we kort toe:

- De eerste regel is de syntax voor het aanmaken van een route: een @ gevolgd door de variabele die onze Flask-website represeneert (app) en het standaardonderdeel `.route()`.

De invulling '/' refereert aan de homepage van de website. Als er geen andere toevoeging wordt ingetypt (zoals www.youtube.com), zal deze route gebruikt worden.

- Elke route heeft een functie, te vinden op de tweede regel van dit stukje code. Deze functie bepaalt wat er gebeurt wanneer de route wordt geactiveerd.
- Op de derde regel van dit stukje staat een `return`-statement. Datgene wat we onze functie door middel van dit statement 'teruggeven', wordt op de webpagina aan de gebruiker getoond.

11.10.2 Flask-server opstarten

Door het toevoegen van de default-route kunnen we nu zeggen dat we een Flask-website hebben ontwikkeld. Om deze website te kunnen bekijken, moet u een Flask-server opstarten.

- Open een nieuwe *Terminal*.

U moet de *Terminal* laten weten welk bestand uw hoofdbestand is. Dat doet u op de volgende manier:

- Typ `set FLASK_APP=flask_app.py` en druk op Enter.

Daarna kunt u de Flask-server opstarten:

- Typ `flask run` en druk op Enter.

Als alles goed gegaan is, ziet u de volgende berichten in de *Terminal*:

```
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
flask + × □ ☰ ^ ×

Microsoft Windows [Version 10.0.19043.2130]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\Flask_tryout>set FLASK_APP=flask_app.py
C:\Users\MSI\Documents\Flask_tryout>flask run
 * Serving Flask app 'flask_app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

- Plaats uw cursor op het webadres `http://127.0.0.1:5000` in de *Terminal*. **Klik nog niet**, maar wacht tot het bericht `Follow link` (`ctrl + click`) verschijnt.
- Klik op "Follow link".

Er wordt een webbrowser geopend, waarin uw Flask-website te zien is:



- Ga terug naar VS Code.
- Klik ergens in de Terminal.
- Sluit de server door op Ctrl + C (Windows) of Command + C (Mac) te drukken.

11.10.3 Extra routes aanmaken

In deze subparagraph laten we u zien hoe u extra routes kunt aanmaken. Dat is relatief eenvoudig als u eenmaal een default-route heeft aangemaakt, want die code kunt u kopiëren.

- Ga naar VS Code.
- Open het hoofdbestand.
- Kopieer de code die hoort bij de default-route:

```
@app.route('/')
def home():
    return "Welkom op mijn eerste Flask-website"
```

- Plak de code onder de code die hoort bij de default-route.
- Pas deze code als volgt aan:

```
@app.route('/prijzen')
def prijzen():
    return "Binnenkort verschijnen hier onze actuele prijzen"
```

We voegen ook een extra route toe voor de recepten:

- Kopieer nogmaals de code die hoort bij de default-route.
- Plak de code onder de code die hoort bij de eerste extra route.
- Pas deze code als volgt aan:

```
@app.route('/recepten')
def recepten():
    return "Binnenkort verschijnen hier enkele recepten"
```



Merk op dat we niet alleen '/' veranderen in respectievelijk '/prijzen' en '/recepten': ook de namen van de functies dienen veranderd te worden in prijzen() en recepten(). In dit geval zijn de namen van de functies passend, maar in feite kunnen we iedere willekeurige naam kiezen.

11.11 Routingfuncties

Eerder in deze les heeft u de syntax `@app.route('/prijzen')` gebruikt om een route aan te maken. Direct daaronder maakte u een nieuwe functie aan: `(prijzen())`.

Momenteel is de functie `prijzen()` erg karg:

```
def prijzen():
    return "Binnenkort verschijnen hier onze actuele prijzen"
```

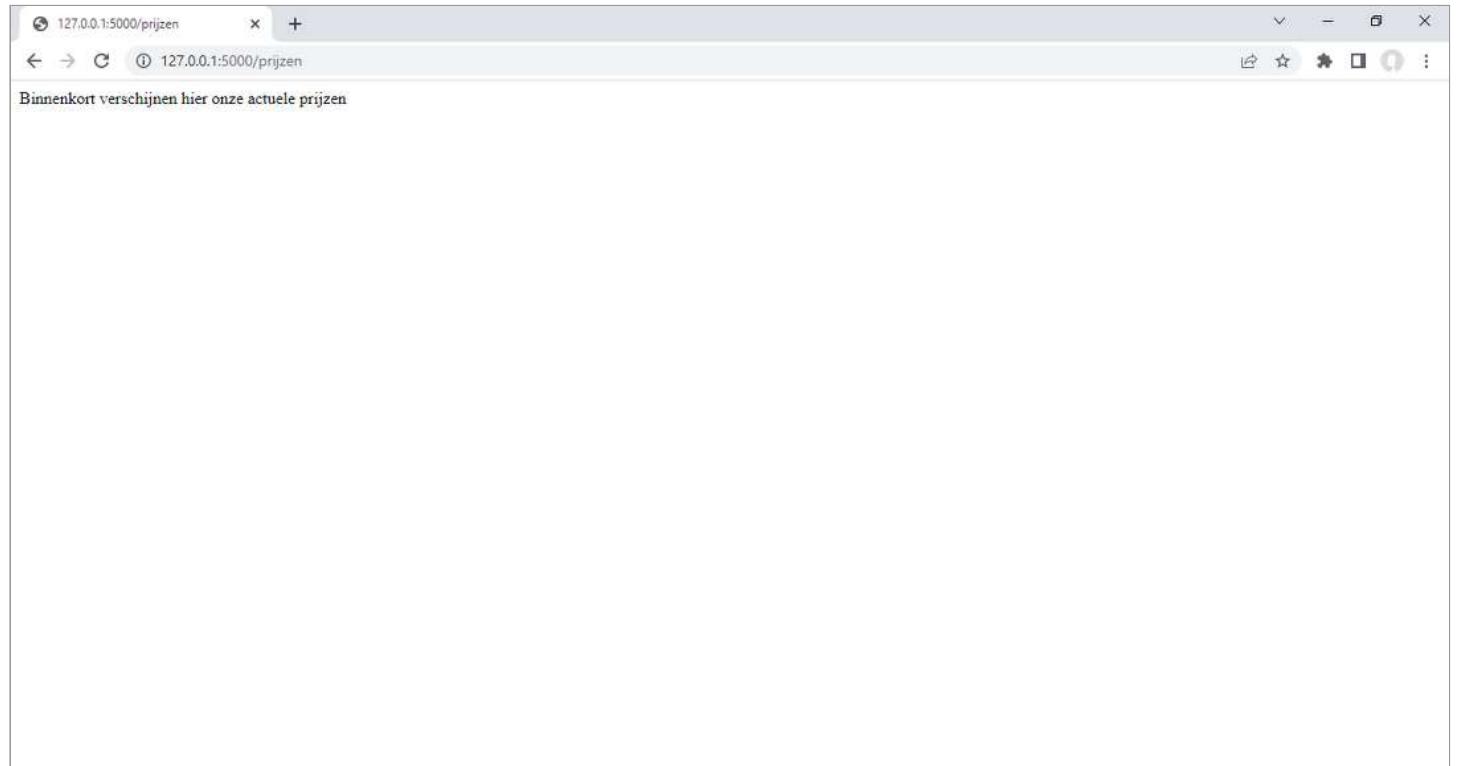
Het enige wat we doen, is een enkele *string* teruggeven met het `return`-commando. Dat gaan we in de volgende paragraaf veranderen. Eerst gaan we kijken of de routes eigenlijk wel werken.

- Ga naar de *Terminal*.
- Start de server op door `flask run` te typen en op Enter te drukken.
- Plaats uw cursor op het webadres `http://127.0.0.1:5000` in de *Terminal*. **Klik nog niet**, maar wacht tot het bericht *Follow link (ctrl + click) verschijnt*.
- Klik op “Follow link”.

Er wordt een webbrowser geopend, waarin uw Flask-website te zien is.

- Verander het IP-adres in de adresbalk in `127.0.0.1:500/prijzen`.
- Druk op Enter.

Als het goed is, ziet u nu de tekst verschijnen die u heeft gecodeerd op de derde regel van de code die hoort bij deze route:



Op een vergelijkbare manier kunt u de tekst die hoort bij de route ‘recepten’, in beeld brengen:



- Ga terug naar *VS Code*.
- Klik ergens in de *Terminal*.
- Sluit de server door op *Ctrl + C* (*Windows*) of *Command + C* (*Mac*) te drukken.

11.12 De functie 'render_template()'

Het is goed dat de routes werken, maar onze website oogt nog een beetje saai. In deze paragraaf laten we u zien hoe u in plaats van deze platte tekst een professioneel ogende pagina kunt laten zien. Daarvoor gebruiken we de Flask-functie `render_template()`.

Om de functie te kunnen gebruiken, moet u deze functie eerst importeren:

- Ga naar *VS Code*.
- Open het hoofdbestand.
- Breid de code op de eerste regel als volgt uit:

```
from flask import Flask, render_template
```

Ook is het nodig om de `home()`-functie aan te passen:

- Wijzig de code van de default-route als volgt:

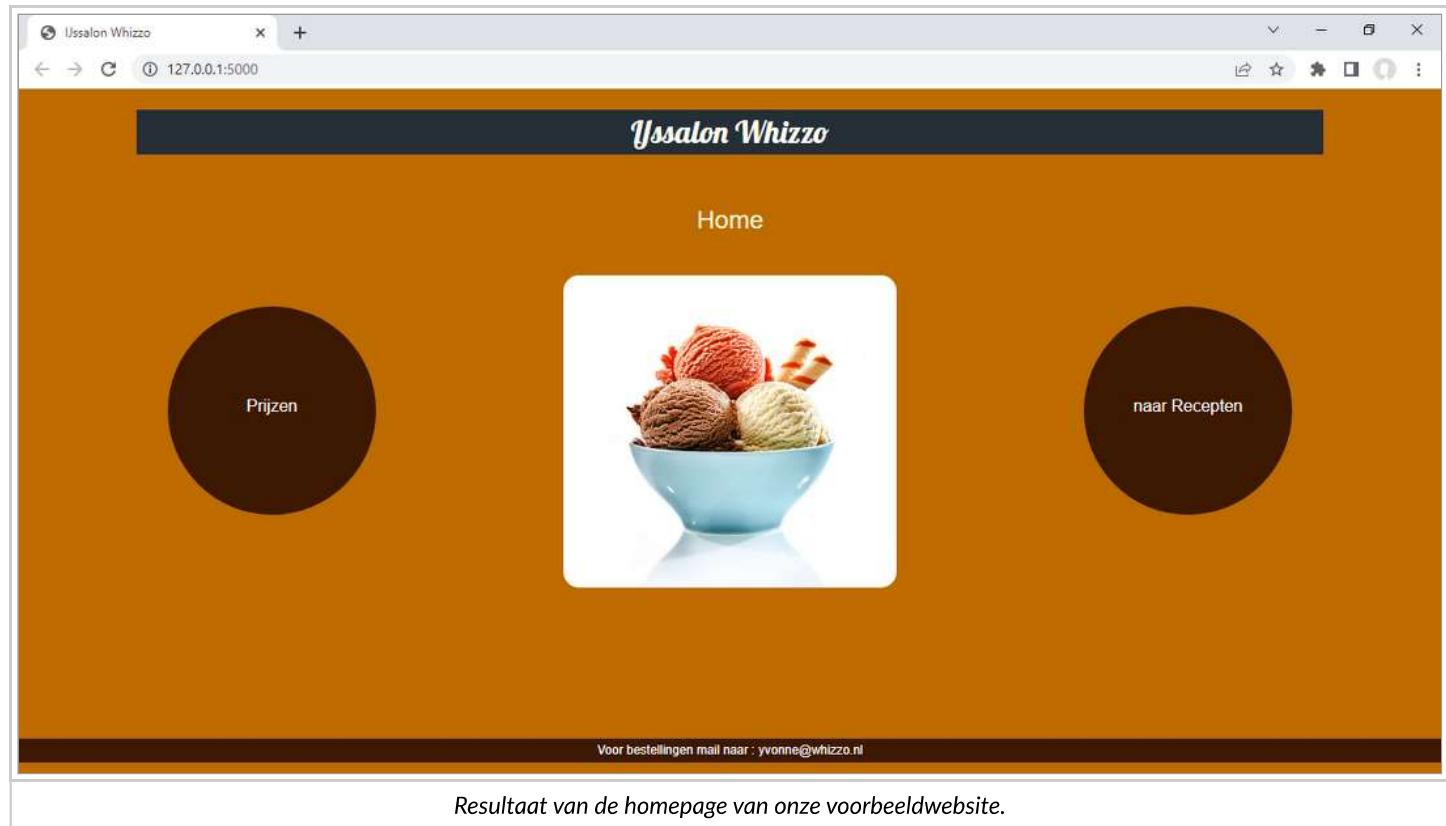
```
@app.route('/')
def home():
    return render_template("home.html")
```

In plaats van de enkele `string` (*Welkom op mijn eerste Flask-website*) geven we nu een functie terug (de functie `render_template()` met als argument "home.html").

Flask zal nu automatisch gaan zoeken naar een map 'templates' (die al voor u aangemaakt was). Eenmaal gevonden zal Flask in die map gaan zoeken naar het bestand 'home.html'.

- Sla het hoofdbestand op.
- Ga naar de *Terminal*.
- Start de server op door `flask run` te typen en op Enter te drukken.
- Plaats uw cursor op het webadres `http://127.0.0.1:5000` in de *Terminal*. **Klik nog niet**, maar wacht tot het bericht *Follow link (ctrl + click) verschijnt*.
- Klik op "Follow link".

Er wordt een webbrowser geopend, waarin uw Flask-website te zien is. Het resultaat zal u verbazen!



Resultaat van de homepage van onze voorbeeldwebsite.

- Ga terug naar VS Code.
- Klik ergens in de Terminal.
- Sluit de server door op Ctrl + C (Windows) of Command + C (Mac) te drukken.



Merk op dat u, wanneer u de extra routes bezoekt, nog altijd de 'saaie pagina's' te zien krijgt. Dit kunt u veranderen door de return-statements in de functies aan te passen en de strings te wijzigen in `render_template("prijzen.html")` en `render_template("recepten.html")`. Nadat u de wijzigingen heeft opgeslagen en de server opnieuw heeft gestart, zullen ook die pagina's er anders uitzien.

11.13 Jinja

U heeft Flask leren kennen als een framework dat routes kan genereren en zo op een veilige manier webpagina's kan aanbieden. We spreken van 'veilig', omdat bezoekers van onze website niet zomaar bestanden kunnen openen: als er geen route voor is, kan het bestand niet worden bekeken.

Flask kan echter nog veel meer! Dat kan onder meer dankzij de templatetaal Jinja, die u in de HTML-pagina's kunt integreren. Dat doet u als volgt:

- Ga naar VS Code.
- Open de pagina 'prijzen.html'.
- Bekijk regel 9 tot en met 15.

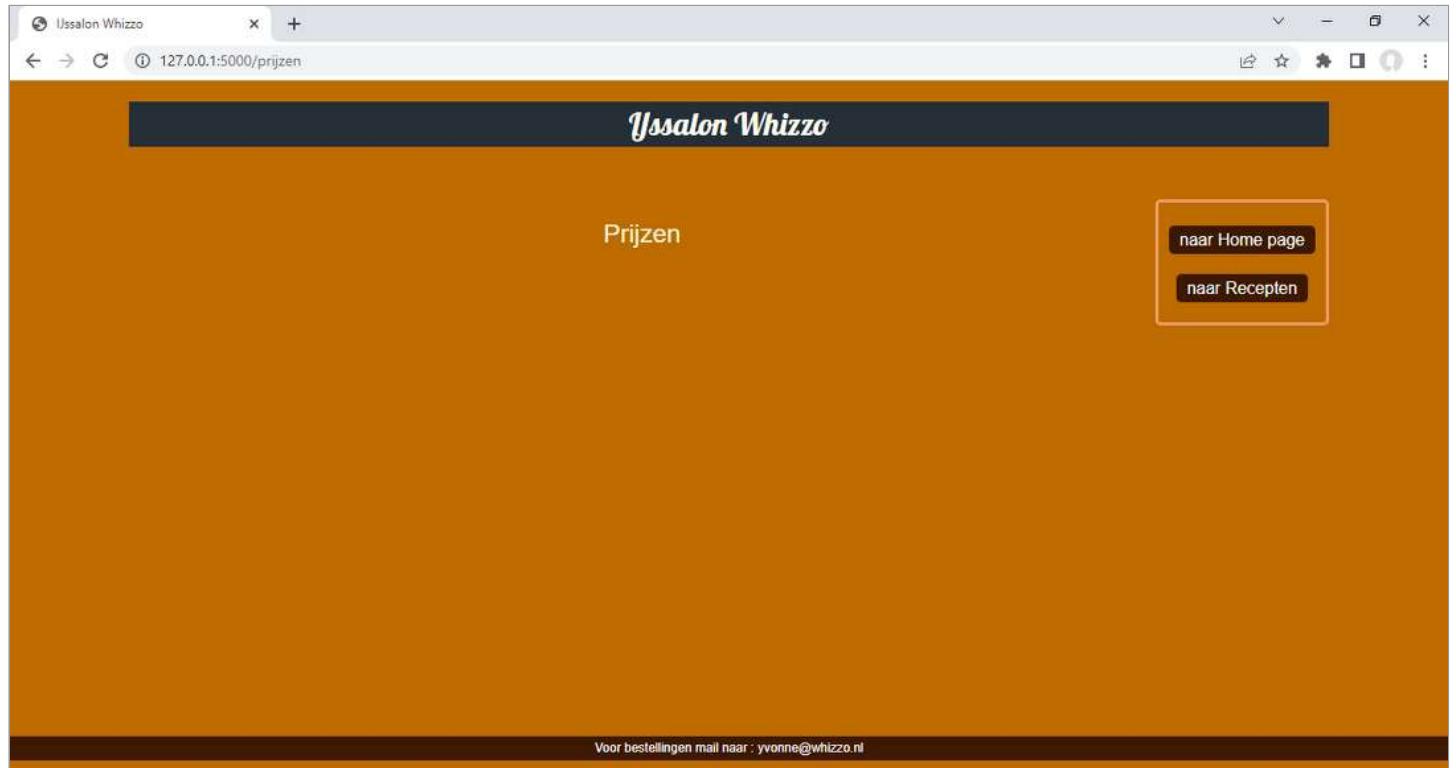
Deze regels zien er zo uit:

```
{% for item in items %}  
  <p class="items-list">  
    <span>{{item.product}}</span>  
    <span>.....</span>  
    <span>{{item.prijs}}</span>  
  </p>  
{% endfor %}
```

Omdat HTML-taal geen onderdeel is van deze cursus, bent u waarschijnlijk niet bekend met HTML-pagina's. U zult daardoor vermoedelijk ook niet zien dat dit geen normale HTML-code is, maar dat er Jinja-elementen aanwezig zijn (de code tussen de accolades).

Door het toevoegen van deze Jinja-code kunnen we gegevens uit het bestand 'flask_app.py' tonen aan de bezoeker (die de *front-end* van de HTML-pagina's ziet).

Omdat we deze informatie momenteel niet doorgeven, zijn er nog geen prijzen op de *Prijzen*-pagina te zien:



Deze informatie doorgeven doen we binnen de functie `render_template()`. We kunnen namelijk niet alleen de template zelf doorgeven (wat we eerder al deden), maar ook informatie in de vorm van argumenten. Dat klinkt wellicht ingewikkeld, maar het valt in de praktijk reuze mee.

Het ingewikkeldste is het voorbereiden van de data die we vanuit de *back-end* willen doorgeven aan de *front-end*.

We beginnen eenvoudig, door het creëren van een *list*.

- Ga naar VS Code.
- Open het hoofdbestand.
- Voeg een extra regel toe aan deze eerste extra route (*prijzen*):

```
@app.route('/prijzen')
def prijzen():
    items = []
    return "Binnenkort verschijnen hier onze actuele prijzen"
```

- Pas de vierde regel als volgt aan (indien u dat nog niet gedaan had):

```
@app.route('/prijzen')
def prijzen():
    items = []
    return render_template("prijzen.html")
```

U heeft nu een lege *list* gecreëerd met de naam 'items'. Deze gaat u vullen met... *dictionaries*! Dat stelt u in staat om zeer gestructureerd informatie door te spelen naar de template 'prijzen.html'.

U heeft eerder gewerkt met *lists* en *dictionaries*, maar het vullen van een *list* met *dictionaries* is nieuw voor u. We geven u daarom de code die u gaat gebruiken.

- Vul de *list* op de volgende manier:

```
def prijzen():
    items = [
        {
            "product": "vanille-ijs 1 liter",
            "prijs": "2 euro"
        },
        {
            "product": "chocolade-ijs 1 liter",
            "prijs": "2 euro"
        }
    ]
    return render_template("prijzen.html")
```

We hebben nu een variabele gecreëerd die een lijst van *dictionaries* bevat. Elke *dictionary* bevat de gegevens van een enkel product.

U gaat de lijst doorsturen naar de template. Dat doet u door in het return-statement een extra argument toe te voegen:

- Breid het return-statement als volgt uit:

```
return render_template("prijzen.html", items=items)
```

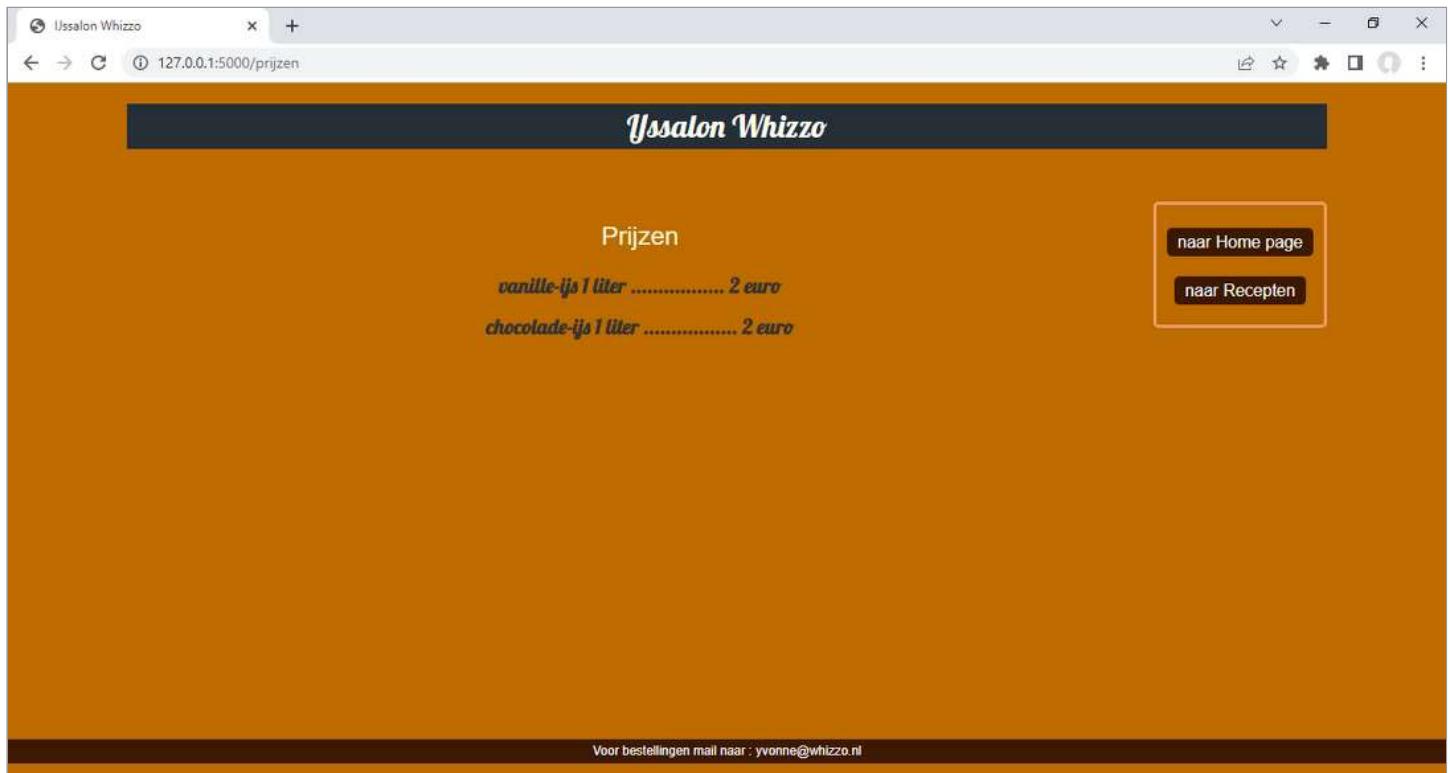
U ziet twee keer *items* staan in deze toevoeging:

- De eerste *items* is de naam van de variabele die u in de *Jinja-code* in de template 'prijzen.html' zag.
- De tweede *items* is de naam die u heeft gekozen voor de *list* in de functie 'prijzen()'. (Dat had ook elke andere willekeurige naam kunnen zijn, maar het is een goed gebruik om deze namen hetzelfde te houden.)
- Sla het hoofdbestand op.
- Ga naar de *Terminal*.
- Start de server op door `flask run` te typen en op Enter te drukken.
- Plaats uw cursor op het webadres `http://127.0.0.1:5000` in de *Terminal*. **Klik nog niet**, maar wacht tot het bericht *Follow link* (`ctrl + click`) verschijnt.
- **Klik op "Follow link".**

Er wordt een webbrowser geopend, waarin uw Flask-website te zien is.

- Navigeer in de browser naar de pagina met prijzen.

U kunt nu zien dat de prijzen weergegeven worden op de pagina:



- Ga terug naar VS Code.
- Klik ergens in de Terminal.
- Sluit de server door op Ctrl + C (Windows) of Command + C (Mac) te drukken.

11.14 Project 'IJsSalon'

Ons project is bijna klaar. Het enige wat we nog toevoegen, zijn de recepten op de daarvoor bedoelde pagina. Ook hier beginnen we met het aanmaken van een lege *list*:

- Ga naar VS Code.
- Open het hoofdbestand.
- Voeg een extra regel toe aan deze eerste extra route (recepten):

```
@app.route('/recepten')
def recepten():
    items = []
    return "Binnenkort verschijnen hier enkele recepten"
```

- Pas de vierde regel als volgt aan (indien u dat nog niet gedaan had):

```
@app.route('/recepten')
def recepten():
    items = []
    return render_template("recepten.html")
```

Ook hier gaat u de lege *list* vullen met *dictionaries*. U past dit keer echter een trucje toe dat u in staat stelt om bij elk recept een passende afbeelding weer te geven. De afbeelding is in feite het recept zelf, opgeslagen als afbeelding:

- Vul de *list* op de volgende manier:

```
def recepten():
    items = [
        {
            "recept": "Tiramisu di nona",
            "img": "tiramisu.png"
        },
    ]
```

```

    }
    "recept": "IJstaart met chocolade",
    "img": "ijstaart.png"
}
]

return render_template("recepten.html")

```

- Breed het return-statement als volgt uit:

```
return render_template("recepten.html", items=items)
```

- Sla het hoofdbestand op.
- Ga naar de *Terminal*.
- Start de server op door flask run te typen en op Enter te drukken.
- Plaats uw cursor op het webadres <http://127.0.0.1:5000> in de *Terminal*. **Klik nog niet**, maar wacht tot het bericht **Follow link (ctrl + click) verschijnt**.
- Klik op "Follow link".

Er wordt een webbrowser geopend, waarin uw Flask-website te zien is.

- Navigeer in de browser naar de pagina met recepten.

U ziet nu de recepten als beeld. U heeft de recepten echter niet in tekstvorm maar als afbeelding geplaatst.

The screenshot shows a web browser window with the title 'Yssalon Whizzo'. The URL in the address bar is '127.0.0.1:5000/recepten'. The page content is as follows:

Tiramisu di nona

Klop de eidooiers met de heft van de suiker tot een zeer romig mengsel, dit duurt zeker 5 minuten. Voeg vervolgens de mascarpone toe aan het eidooiermengsel en klop dit tot een homogene massa. Doe de eiwitten met een snuf zout in een vettvrije kom en klop ze stijf. Voeg lepel voor lepel de resterende suiker toe. Spatel het stijfgeklopte eiwit door het mascarponemengsel en meng dit tot een egale crème.

Meng de espresso met de drank naar keuze. Doop hier de lange vingers één voor één kort in en bedek daarmee de bodem van een (oven)schaal van ongeveer 25x18 centimeter. Verdeel hier de heft van het tiramisumengsel overeen en strijk dit glad. Herhaal beide stappen en zet de tiramisu minimaal 4 uur in de koelkast om op te stijven. Bestrooi voor het serveren met een dikke laag cacaopoeder.

IJstaart met chocolade

Bekleed de binnenkant van een cakevorm van 25 centimeter met plasticfolie en laat dit aan de zijkanten overhangen. Doe de gecondenseerde melk, de slagroom en het vanillemerg in een kom en klop dit met een (hand)mixer met garde(s) tot een romig geheel, tot het de dikte van lobbige geklopte slagroom heeft. Meng daar de gehakte chocolade door.

Doe de jam in een fijne zeef en druk deze er met de bolle kant van een lepel doorheen, zodat de pitjes achterblijven. Verdeel een dun laagje van het roommengsel over de bodem van de vorm en strijk dit glad. Sprengel daar met een lepel wat van de gezeepte jam over. Schep daar weer voorzichtig een dun laagje van het roommengsel op en strijk weer glad. Ga zo door tot de vorm vol zit. Klap het overhangende plasticfolie naar binnen. Zet de ijstaart minimaal 1 nacht in de vriezer.

Haal voor het serveren de ijstaart met behulp van het plasticfolie uit de vorm en plaats de taart op een schaal. Verwijder het folie voorzichtig en maak de ijstaart af met het verse rode fruit.

Voor bestellingen mail naar : yvonne@whizzo.nl

U heeft de recepten niet in tekstvorm maar als afbeelding geplaatst.

- Ga terug naar VS Code.
- Klik ergens in de *Terminal*.
- Sluit de server door op Ctrl + C (Windows) of Command + C (Mac) te drukken.

11.15 Samenvatting

In deze les hebben we ons verdiept in Flask, een van de meest gebruikte frameworks voor het vervaardigen van een Python-website.

Na de installatie zag u dat een Flask-project twee mappen kent: 'templates' en 'static'. De eerste map bevat de HTML-bestanden die de *front-end* van de website vormen, de tweede map bevat hulpbestanden (zoals afbeeldingen).

Ook heeft u gezien hoe u routes kunt aanmaken. Dankzij routes heeft u volledige controle over wat gebruikers kunnen zien en wat niet. Elke route heeft een functie die bepaalt wat er precies gebeurt wanneer een gebruiker een bepaald gedeelte van onze website bezoekt. Met de functie `render_template()` legde u vervolgens een link naar de HTML-bestanden in de map 'templates'. Daarmee heeft u een volledig functionele website vervaardigd, die vervolgens gevoed werd door data vanuit de *back-end*.

In het hoofdbestand heeft u Python gebruikt om data op te stellen, die u vervolgens als argument meegaf tijdens het aanroepen van de functie `render_template()`.

Daarna lieten we u nog zien hoe u Jinja-code kunt invoegen in HTML-bestanden, waardoor data vanuit de *back-end* konden worden weergegeven op de onze webpagina's.

Tot slot heeft u alles samengebracht in het project 'IJssalon'.

11.16 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 teksvak kunt typen.



Het teksvak 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.

11.17 Oefenopgaven

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

Wat is een framework?

→ Bekijk antwoord

Waarom is het voor beginners ideaal om met frameworks te werken?

→ Bekijk antwoord

Welk van de volgende frameworks is geen Python framework?

- Django
- Enthought Tool Suite
- Flask

Voor welke programmeertaal is het framework .NET van toepassing?

→ Bekijk antwoord

Django staat bekend om zijn grote *community*. Waarom is een grote *community* een groot voordeel?

→ Bekijk antwoord

Een voordeel van Django is de modulaire opbouw, zowel in de breedte als in de diepte. Waarom is dit een voordeel?

→ Bekijk antwoord

Flask wordt gezien als een 'lichtgewicht' framework. Wat wordt hiermee bedoeld?

→ Bekijk antwoord

Wat is het verschil tussen forken en klonen?

→ Bekijk antwoord

Met welk commando installeert u het Flask-framework op uw computer?

→ Bekijk antwoord

Welke twee mappen heeft u nodig als u een website met het Flask-framework aanmaakt?

→ Bekijk antwoord

Elke pagina krijgt zijn eigen route. Was is de naam van de route die hoort bij de homepage van de website?

→ Bekijk antwoord

Als u een route aanmaakt, is een return-statement een onderdeel van de code die u schrijft. Wat is het nut van dit statement?

→ Bekijk antwoord

Leg in eigen woorden uit wat de Flask-functie `render_template()` doet.

 [Bekijk antwoord](#)

Wat is Jinja?

 [Bekijk antwoord](#)

11.18 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. Ga naar *GitHub* en log in op uw account. Maak een nieuwe *repository* aan met de naam 'Flask_tryout'.
2. Open *VS Code*. Sluit (indien nodig) alle bestanden, mappen en projecten die nog openstaan. Klik daarna in de *Activity Bar* op de knop "Explorer" en clone de zojuist aangemaakte *repository*. Open de *repository* in *VS Code*.
3. Voeg een nieuw bestand aan de *repository* toe en geef dit de naam 'flask_app.py'. Plaats in dit bestand een code die voor het volgende zorgt:
 - Er wordt een deel van het Flask-pakket geïmporteerd.
 - Er wordt een variabele aangemaakt met de naam app.
 - Er is een default-route aangemaakt, waarbij het return-statement de tekst "<p>Hello world!</p>" teruggeeft.

4. Typ de volgende code in de *Terminal*:

```
git checkout -b main (gevolgd door een Enter).
```

Geef aan wat u als boodschap van de *Terminal* terugkrijgt en leg in eigen woorden uit wat dit betekent.

5. Maak een nieuwe lokale versie van het project aan. Gebruik daarvoor twee commando's. Plaats bij het tweede commando de passende tekst basic Flask code tussen aanhalingstekens.
6. Verander de tekst tussen <p> en </p> in een tekst naar keuze.
7. Update de *repository* op *GitHub*.
8. Laat de *Terminal* weten dat het zojuist aangemaakte bestand uw hoofdbestand is.
9. Start de Flask-server en open uw Flask-website in de browser. Maak een printscreen van de pagina in de browser.

Stuur de URL van de aangemaakte *repository* op *GitHub*, de printscreen en een antwoord op de vraag bij stap 4 via Plaza naar uw docent. (Door het opsturen van de URL kan de docent niet alleen het eindresultaat zien, maar ook alle tussentijdse wijzigingen.)