# Python

- ChatGPT
- Wat is Python?
- Opdrachten
- API en JSON
- Strings
- JSON hoogste leeftijd
- JSON Leeftijd en naam
- API and JSON
- Flask installatie
- Flask form
- Form aanpassen
- Project Boekenrecensie-applicatie
- Project RSVP
- Python 3 - Game of Pong

# ChatGPT

## Gebruik ChatGPT, maar gebruik het goed!

In deze module moeten jullie ChatGPT gebruiken. Maar voordat we dat gaan doen, eerst even een soort van regel:

Deze regel luidt:

> Gebruik alle bronnen die je kan vinden om zo snel als je kunt iets te ontwikkelen, maar zorg er altijd voor dat je elke regel code begrijpt.

En waarom is dat?

Veel bedrijven zijn super afhankelijk van hun software. Zo is er een bedrijf 'Knight Capital' waar het volgende mee is gebeurd.

> *Tijdens de fatale handelsdag in 2012 werd een nieuwe handelssoftware geïmplementeerd bij Knight Capital, maar er werd een bug geïntroduceerd in het algoritme van de software. Hierdoor begon de software onjuiste orders te plaatsen op verschillende aandelenmarkten.*
>
> *De ongecontroleerde en foutieve orderstroom zorgde ervoor dat Knight Capital enorme hoeveelheden aandelen kocht of verkocht tegen verkeerde prijzen. Dit resulteerde in een verlies van ongeveer $440 miljoen binnen enkele minuten.*

Het bedrijf ging failliet.

Dit incident heeft de financiële sector veranderd. Er kwam meer regelgeving, en alle software werd nog beter getest. Als programmeur in de financiële sector kam een nieuwe regel:

"Als de software werkt, maar je begrijpt niet precies wat en hoe het werkt dan mag het niet in productie."

Als je dus stukken software kopieert van het internet of van ChatGPT en je weet niet precies hoe het werkt dan plak je al die stukjes code aan elkaar waarvan telkens mogelijk kleine onbedoelde foutjes code zitten. Als die onbedoelde kleine stukjes software zullen vroeg of laat tot een storing leiden. Het hangt natuurlijk af van in welke sector je werkt, hoe erg dan de gevolgen van een storing zijn.

Dus gebruik zoveel ChatGPT als je wil, maar zie je een commando dat je niet begrijpt, laat het je dan uitleggen. Begrijp je het dan nog niet, gebruik het dan niet, zoek een andere oplossing!

Vanaf nu zul je ook meer worden beoordeeld op het kunnen uitleggen *hoe code werkt*.

## Voordelen van het snappen van hoe code werkt:

- Je vindt sneller bugs/fouten.
- Er zitten minder fouten in jouw code.
- Er zit minder onnodige code in je project en daarom is je code veiliger. Onnodige code wordt namelijk veel gebruikt om te kunnen hacken.
- Iedereen kan aan ChatGPT code geven en vragen wat er fout aan is. Maar omdat je het echt begrijpt kun jij ook bugs oplossen die ChatGPT niet kan oplossen. Daarom ben je meer $$$$ waard dan iemand anders. En door ChatGPT worden je skills steeds belangrijker en unieker.

---

Dus gebruik alle hulpmidellen die je ter beschikking staan, maar zorg dat je kan uitleggen wat je hebt gemaakt.

# Python

In deze module(s) gaan we Python leren. Niet alles wordt meer stap-voor-stap uitgelegd. Vraag ChatGPT om uitleg. Vraag bijvoorbeeld aan chatGPT hoe je een if-then-else maakt.

Als je bijvoorbeeld aan ChatGPT de volgende vraag stelt:

> *Hoe maak je in Python een if-the-else?*

Dan krijg je een uitleg en een voorbeeld. Mocht je dat niet krijgen dan kun je ook om een voorbeeld vragen!

# Opdracht

Maak een txt bestand en leg daarin in eigen woorden uit wat het vercshil is van een if-then-else in PHP en in Python.

Noem minimaal twee veschillen.

# Wat is Python?

## Wat Python?

Guido van Rossum is een Nederlandse computerprogrammeur die bekend staat als de maker van de programmeertaal Python.

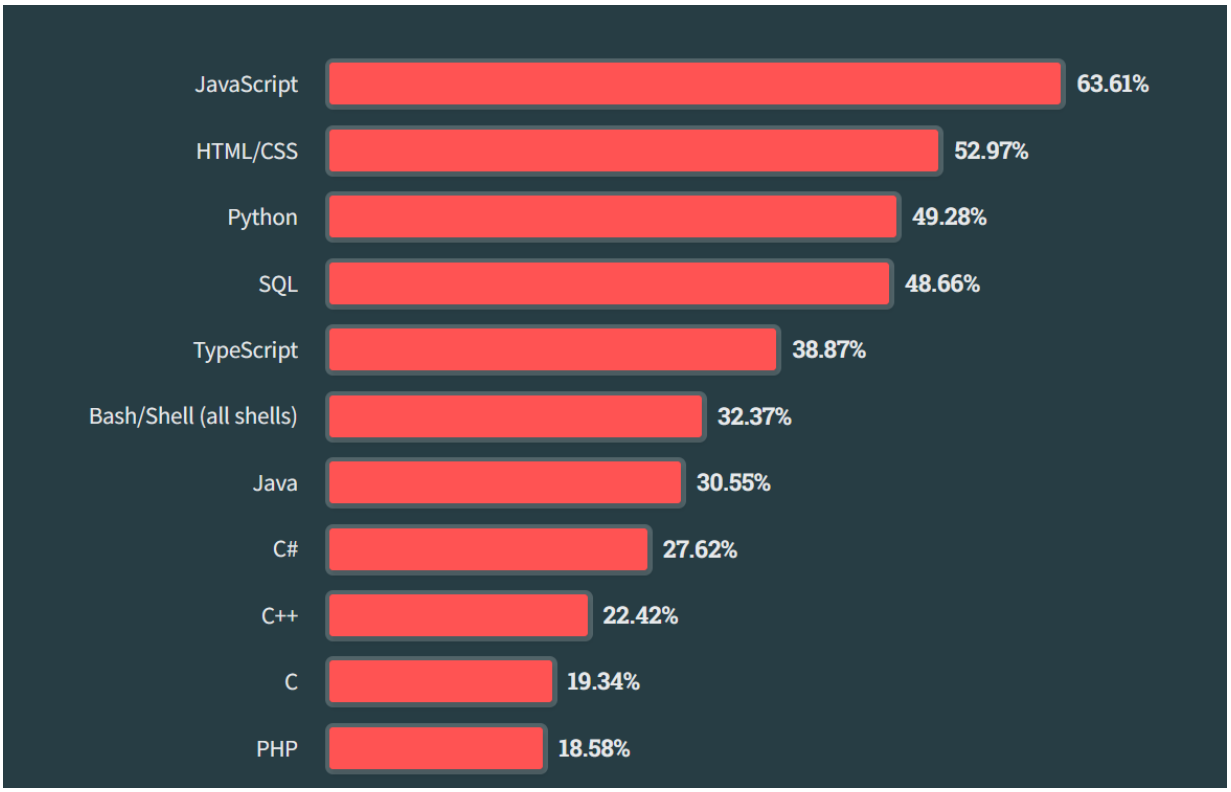
Python heeft dus Nederlandse roots.

Maar wat heeft Python zo bekendgemaakt?

- Python heeft veel library's waarmee je heel makkelijk bepaalde zaken kan doen, zo zijn er tegenwoordig veel library's waarmee je een AI-model kan maken. Maar er is bijvoorbeeld ook een Library om via de Canvas-API gegevens met Canvas uit te wisselen. De Canvas Monitor maakt daar gebruik van en is gedeeltelijk geschreven in Python.
- Python werkt platformonafhankelijk. Als je iets programmeert op Windows dan draait dat ook op een mac of op Linux.
- Er zijn Frameworks (net als Yii, Laravel en React) gebouwd rond Python. De bekendste zijn: Django en Flask. Met Flask gaan we later kennis maken.

## Populariteit

Python is momenteel (2023) erg populair en staat volgens de TIOBE index op nummer 1.



(Stackoverflow "*Most Popular Technology*" 2023)

Volgens gebruikers van Stack Overflow was Python de op twee na populairste technology (na JavaScript en HTML).

Let op: dit is wat andes als *meest gebruikte* technology, daarbij komt PHP vaak op nummer één uit.

Python wordt ook veel gebruikt in het onderwijs (HBO en Universiteit).

# Voordelen Python

Waarom zou je Python in plaats van PHP gebruiken?

- Over het algemeen wordt de code van Python gezien als beter leesbaar, waardoor je het eerder begrijpt.
- PHP is gemaakt voor web-ontwikkeling.
  Python is veel algemener en je kunt het voor alles gebruiken:
  - data-analyse, automatiseren van beheer (bijv. automatisch updaten van computers),
  - met bepaalde Python library's kan je GUI ontwikkelen,
  - netwerk-prorammeren (dat zijn programma's die bijvoorbeeld het netwerkverkeer kunnen analyseren) en
  - (proto type) game programmering.

# Installatie Python

Om Python op een Windows-machine te installeren, kun je de volgende stappen volgen:

1. Ga naar de officiële Python-website op https://www.python.org/downloads/.
2. Klik op de knop "Downloaden" onder de meest recente versie van Python (bijvoorbeeld Python 3.9.6).
3. Scroll naar beneden op de downloadpagina en selecteer de juiste installatieprogramma voor je Windows-besturingssysteem. Kies tussen de 32-bits (x86) of 64-bits (x86-64) versie, afhankelijk van je systeemconfiguratie.
4. Nadat het installatieprogramma is gedownload, dubbelklik je erop om het uit te voeren.
5. In het installatieprogramma wordt een dialoogvenster geopend. Zorg ervoor dat je het selectievakje "Add Python to PATH" aanvinkt en klik op "Install Now" om de standaardinstallatie te starten.
6. De installatie begint en Python wordt op je systeem geïnstalleerd. Het kan enige tijd duren. Zorg ervoor dat je de optie "Disable path length limit" selecteert als deze wordt weergegeven.
7. Na de installatie wordt er een dialoogvenster geopend met de titel "Setup was successful". Vink het selectievakje "Disable path length limit" aan als deze optie beschikbaar is en klik op "Close" om de installatie te voltooien.
8. Om te controleren of Python correct is geïnstalleerd, open je CMD (Win + R, typ "cmd" en druk op Enter) en typ je "python --version" gevolgd door Enter. Je zou de geïnstalleerde versie van Python moeten zien.

Gefeliciteerd! Je hebt Python succesvol geïnstalleerd op je Windows-machine. Nu kun je Python gebruiken om scripts uit te voeren, applicaties te ontwikkelen en meer.

# Inspringen

Allereerst....**superbelangrijk**, in Python is het inspringen belangrijk. Doe je dit niet op de juiste manier, dan werkt je code niet!

```
import random


def raad_het_getal():
    willekeurig_getal = random.randint(1, 100)
    aantal_pogingen = 0
```

```
    while True:
        gok = int(input("Raad het getal tussen 1 en 100: "))

        if gok == willekeurig_getal:
            print(f"Gefeliciteerd! Je hebt het juiste getal geraden in {aantal_pogingen} poging(en).")
            break
        elif gok < willekeurig_getal:
            print("Te laag! Probeer het opnieuw.")
        else:
            print("Te hoog! Probeer het opnieuw.")

raad_het_getal()
```

Maak een bestand aan (in VCS of andere editor) en zet het bovenstaande Python programmaatje er in. Voer het programma uit en bestudeer hoe het werkt.

Je ziet na elke : begint er een programma-blok. Op regel 3 begin je met het maken van een functie. Je springt dan in en alles wat op dit niveau is ingesprongen (of verder) hoort bij de functie. Regel 19 is dus de eerste regel die niet meer bij de functie hoort.

Regel 11 en 12 hoort bij de ***if***, 14 bij de ***elif*** en 16 bij de ***else***.

Probeer het spel uit.

Er is een klein foutje gemaakt; als je het spel hebt gespeeld dan wordt er gezegd dat je het getal hebt geraden in 0 pogingen.

# Opdracht 1

Pas de code aan zodat als je het getal hebt geraden het aantal pogingen wordt afgedrukt. Dus je drukt bijvoorbeeld af:

> "*Goed zo je hebt het getal binnen 6 beurten geraden.*"

Tip: om sneller te kunnen testen kun je het spel ook even veranderen in 'Raad een getal tussen 1 en 10'. Let wel; op dat je de juiste versie van het spel inlevert en dat is de versie waarbij je een getal van 1..100 moet raden.

# Inleveren

Aangepaste python code

(nakijken: import random 100 aantal_pogingen while aantal_pogingen aantal pogingen raad)

# Opdracht 2

Pas de code aan zodat je na 10 beurten het spel stopt en de gebruiker verteld dat hij het maximale aantal beurten van 10 heeft gehaald. Laat de gebruiker weten wat het getal was en dat hij heeft verloren.

Eis: verander de hiervoor de oneindige loop ***while true:***

# Inleveren

Aangepaste python code

(nakijken: import random 100 aantal_pogingen while aantal_pogingen aantal_pogingen  aantal pogingen raad)

# Opdacht 3

In Python wordt vaak een API gebruikt. API's geven vaak JSON-output, bijvoorbeeld:

```
{
  "personen": [
    {
      "naam": "Alice",
      "leeftijd": 25,
      "stad": "Amsterdam"
    },
    {
      "naam": "Bob",
      "leeftijd": 32,
      "stad": "Rotterdam"
```

```
    },
    {
      "naam": "Charlie",
      "leeftijd": 42,
      "stad": "Utrecht"
    }
  ]
}
```

Maak dit JSON bestand aan en noem het "data.json". Dus maak een nieuw bestand en zet de bovenstaande gegevens in dit nieuwe bestand.

Maak het volgende Python script in dezelfde directory/folder aan.

```
import json

# JSON-bestand lezen
with open("data.json") as json_bestand:
    data = json.load(json_bestand)

# Gegevens verwerken
personen = data["personen"]
for persoon in personen:
    naam = persoon["naam"]
    leeftijd = persoon["leeftijd"]
    stad = persoon["stad"]
    print(f"Naam: {naam}, Leeftijd: {leeftijd}, Stad: {stad}")
```

Controleer of je code werkt.

Zoek op wat het commando ***import JSON*** doet.

Opdracht regel 10, 11 en 12 zou je weg kunnen laten, maar dan moet je wel regel 13 aanpassen.

Pas regel 13 aan zodat de code werkt zonder regel 10, 11 en 12.

# Inleveren

Aangepaste python code

(import json personen for !["personen] !['leeftijd] !["stad"] naam leeftijd stad)

# Opdracht 5

Pas daarna de JSON aan zodat iedereen een telefoonnummer krijgt en pas de code aan zodat het telefoonnummer wordt afgedrukt. Het telefoonnummer wordt als laatste afgedrukt (dus na de stad).

(import json personen for naam leeftijd stad telefoonnummer)

# Inleveren

Aangepaste python code

--

# Opdrachten

We springen meteen in het diepe. Jullie hebben nu al veel programmeerervaring (en ervaring met Google en ChatGPT) dus hier gaan we.

Let erop dat ervoor zorgt dat je alles goed begrijpt, begrijp je het niet vraag het dan aan ChatGPT.

Vraag bijvoorbeeld aan ChatGPT "Hoe moet ik inspringen in Python?", of "Kun je een voorbeeld geven van inspringen in Python?"

# Inspringen

Allereerst....superbelangrijk, in Python is het inspringen belangrijk. Doe je dit niet op de juiste manier, dan werkt je code niet!

```
import random

def raad_het_getal():
    willekeurig_getal = random.randint(1, 100)
    aantal_pogingen = 0

    while True:
        gok = int(input("Raad het getal tussen 1 en 100: "))

        if gok == willekeurig_getal:
            print(f"Gefeliciteerd! Je hebt het juiste getal geraden in {aantal_pogingen} poging(en).")
            break
        elif gok < willekeurig_getal:
            print("Te laag! Probeer het opnieuw.")
        else:
            print("Te hoog! Probeer het opnieuw.")

raad_het_getal()
```

Maak een bestand aan (in VCS of andere editor) en zet het bovenstaande Python programmaatje er in. Voer het programma uit en bestudeer hoe het werkt.

Je ziet na elke : begint er een programma-blok. Op regel 3 begin je met het maken van een functie. Je springt dan in en alles wat op dit niveau is ingesprongen (of verder) hoort bij de functie. Regel 19

is dus de eerste regel die niet meer bij de functie hoort.

Regel 12 en 13 hoort bij de ***if***, 15 bij de ***else if*** en 17 bij de ***else***.

Probeer het spel uit.

Er is een klein foutje gemaakt; als je het spel hebt gespeeld dan wordt er gezegd dat je het getal hebt geraden in 0 pogingen.

# Opdracht 1

Pas de code (raad een getal) aan zodat als je het getal hebt geraden het aantal pogingen juist wordt afgedrukt.

Tip: om sneller te kunnen testen kun je het spel ook even veranderen in 'Raad een getal tussen 0 en 10'.

Je kunt ChatGPT gebruiken, maar laat het niet al je code genereren, maar vraag kleine stukjes zodat je begrijpt wat er gebeurt.

# Inleveren

Aangepaste python code

(nakijken: import random aantal_pogingen while aantal_pogingen aantal pogingen raad)

# Opdracht 2

Pas de code aan zodat je na 10 beurten het spel stopt en de gebruiker verteld dat hij het maximale aantal beurten van 10 heeft gehaald. Laat de gebruiker weten wat het getal was en dat hij heeft verloren.

Eis: verander de hiervoor de oneindige loop ***while true:***

# Inleveren

Aangepaste python code

(nakijken: import random aantal_pogingen while aantal_pogingen aantal_pogingen  aantal pogingen raad)

# Opdacht 3

In Python wordt vaak een API gebruikt. API's geven vaak JSON output, bijvoorbeeld:

```
{
  "personen": [
    {
      "naam": "Alice",
      "leeftijd": 25,
      "stad": "Amsterdam"
    },
    {
      "naam": "Bob",
      "leeftijd": 32,
      "stad": "Rotterdam"
    },
    {
      "naam": "Charlie",
      "leeftijd": 42,
      "stad": "Utrecht"
    }
  ]
}
```

Maak dit JSON bestand aan.

En maak het volgende Python script.

```
import json


# JSON-bestand lezen
with open("data.json") as json_bestand:
    data = json.load(json_bestand)
```

```
# Gegevens verwerken
personen = data["personen"]
for persoon in personen:
    naam = persoon["naam"]
    leeftijd = persoon["leeftijd"]
    stad = persoon["stad"]
    print(f"Naam: {naam}, Leeftijd: {leeftijd}, Stad: {stad}")
```

Controleer of je code werkt.

Zoek op wat het commando ***import JSON*** doet.

Opdracht regel 10, 11 en 12 zou je weg kunnen laten, maar dan moet je wel regel 13 aanpassen.

Pas regel 13 aan zodat de code werkt zonder regel 10, 11 en 12.

# Inleveren

Aangepaste python code

import json personen for !["personen] ![‘leeftijd] !["stad"] naam leeftijd stad

# Opdracht 4

Pas daarna de JSON aan zodat iedereen een telefoonnummer krijgt en pas de code aan zodat het telefoonnummer wordt afgedrukt. Het telefoonnummer wordt als laatste afgedrukt (dus na de stad).

(import json personen for naam leeftijd stad telefoonnummer)

# Inleveren

Aangepaste python code

--

# API en JSON

## Wat is een API?

Een API (Application Programming Interface) is een toegangspoort vanuit een website om informatie op te vragen.

Het is eigenlijk een soort database (zoals onze SQL-database) maar dan op een andere server. Je kunt als gebruiker die informatie opvragen, maar de informatie die op een website staat si door een programma lastig te verwerken. Daarom is er een soort standaard formaat om gegevens te delen en dat formaat heet JSON. De meeste API's gebruiken dan ook JSON als een formaat om gegevens van de ene site naar de andere te sturen.

Zo maakt de Canvas Monitor gebruik van de Canvas API en kan het daarmee gegevens ophalen uit Canvas over jouw voortgang. De reden dat de Canvas Monitor niet altijd up-to-date is, is omdat er een apart (Python) programma is dat om bepaalde tijden via de API vraagt of er updates zijn.

Via een API kan je meestal gegevens opvragen, maar ook veranderen. Voorlopig gaan wij kijken naar het lezen van gegevens via een API. Er zijn verschillende soorten API's.

Wil je nog wat meer voorbeeld van API;s dan wordt er in dit korte YouTube filmpje nog een keer uitgelegd wat een API is.

https://www.youtube.com/watch?v=AwXoCmz4yMg&pp=ygUKd2F0IGlzIEFQSQ%3D%3D

## Een API gebruiken in Python

Ga naar:

https://www.wijs.ovh/c-api/

dit is een API interface waarbij je alle gegevens van alle landen van de wereld kan opvragen.

Check de JSON die de API geeft.

We gaan met een Python programma een lijstje maken van alle landen en hun hoofdsteden.

Omdat te doen hebben we een Python Library nodig. Deze zorgt ervoor dat je een webpagina kan opvragen. Deze library heet *requests* en met dit commando importeren we de library (die is vergelijkbaar met *require* in PHP).

Na de import lezen we de informatie uit de API op.

```
import requests


response = requests.get(f"https://www.wijs.ovh/c-api/")
if response.status_code == 200:
  countries_data = response.json()
else:
  print("No Luck, I cannot get data from the API")

# print( countries_data[0]['name']['common'], countries_data[0]['capital'][0], countries_data[0]['car']['side'] )
# print( countries_data[113]['name']['common'], countries_data[113]['capital'][0],
countries_data[113]['car']['side'] )
```

Try it out!

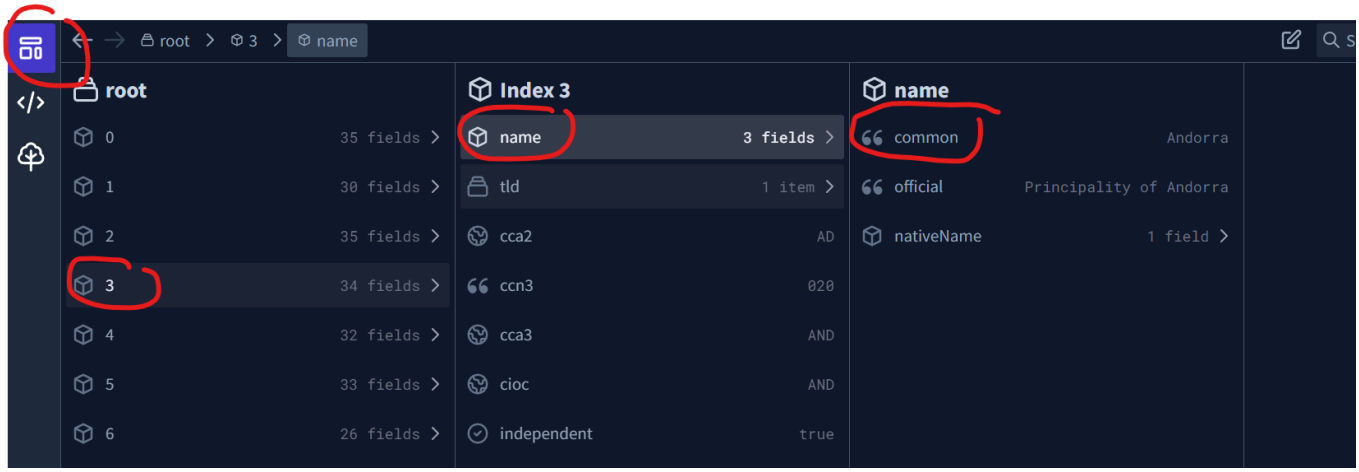
Deze code leest voert een API-call uit en krijgt JSON terug waarin informatie over alle landen van de wereld staat.

Je kunt in je browser de JSO oproepen door de URL te gebruiken die in de code staat.

Die JSON is een heleboel data en hoe weten we nu wat en hoe we daar gegevens uit kunnen halen?

# JSON ontcijferen

Je hebt hiervoor een hulpmiddel nodig. Er zijn er meerdere, maar ik raad https://jsonhero.io/ aan.



Als je het JSON-bestand inleest (dat kan door de URL in de tool mee te geven dan zie je een interface zo als in het plaatje.

In de eerste kolom staat 0,1,2,3,4, ..... dat zijn de record. In dit geval staat alle informatie over één land in een zo'n record.

Stel ons JSON bestand staat in de variabele countries_data, dan kunnen we het derde (of eigenlijk vierde want we beginnen weer met 0 te tellen) land selecteren met:

    countries_data[3]

Nu hebben we alle data van het derde land In de volgende kolom staan alle items die in een land staan. Als eerste zien we name, we kiezen nu de name door:

    countries_data[3][name]

In de derde kolom staan nu weer drie items, common, official, en nativeName. Als we de common naam willen, dan akn dat door:

    countries_data[3][name][common]

In de code staan op de laatste twee regels een paar voorbeelden. Loop deze na en controleer of deze werken.

Zie je wat countries_data[0]['car']['side'] dit afdrukt? Probeer het uit!

Stel we willen alle namen van alle landen afdrukken. Dat kan met:

```
print(countries_data[0][name][common])
print(countries_data[1][name][common])
print(countries_data[2][name][common])
....
....
```

Niet handig dus we gebruiken een loop in Python. Net als we in PHP foreach zouden gebruiken, doen we dit in Python op de volgende manier:

```
for item in countries_data
  print(print(item[name][comon]))
```

De variabele item wordt in de loop telkens countries_data[0], countries_data[1], countries_data[2], countries_data[3], etc. etc.

# Opdracht

Maak een txt bestand en beantwoord de volgende vraag:

1. Als je de hoofdstad afdrukt van een land dan gebruik je countries_data[0]['capital'][0] Waarom staat die [0] erachter en waarom denk je dat dat nodig is?

# Opdracht

Druk alle namen van alle landen af met daarachter de munteenheid (currency).

Gebruik niet de afkorting maar de volledige naam, zoals:

- Jordan - Jordian Dinar
- Northern Mariana Islands - United States Dollar
- Serbia - Serbian Dinar

# Inleveren

1. Python code (py bestand) en;
2. een schermafdruk van de CMD waarin je de code hebt gedraaid.

Idee voor later.....

```
import requests
import random

def get_random_countries(num_countries):
    response = requests.get(f"https://restcountries.com/v3.1/all")
    if response.status_code == 200:
        countries_data = response.json()
        random_countries = random.sample(countries_data, num_countries)
        return random_countries
    else:
        return None
```

```
def play_game():
    random_countries = get_random_countries(10)
    if random_countries is None:
        print("Failed to retrieve country data. Please try again later.")
        return

    total_points = 0

    for country in random_countries:
        name = country['name']['common']
        population = country['population']

        print(f"Guess the population of {name}: ")
        guess = input()

        try:
            guess = int(guess)
            difference = abs(guess - population)
            points = max(0, 100 - difference // 1000000)  # Calculate points based on difference

            print(f"The population of {name} is {population}")
            print(f"You guessed {guess}. You scored {points} points.\n")

            total_points += points
        except ValueError:
            print("Invalid input. Skipping the country.")

    print(f"Game over! Your total score is {total_points}.")

play_game()
```

# Strings

Strings zijn best een beetje bijzonder in Python.

# JSON hoogste leeftijd

Maak de volgende code af.

In het programma wordt een JSON-structuur gegeven waarin persoonsnamen staan met hun leeftijd.

Maak de code af zodat de maximale leeftijd wordt gevonden.

```
import json

# JSON data
json_data = """
{
   "mensen": [
      {"naam": "Johan Vermeulen", "leeftijd": 23},
      {"naam": "Ahmed Al-Hassan", "leeftijd": 26},
      {"naam": "María Rodríguez", "leeftijd": 30},
      {"naam": "Emma de Vries", "leeftijd": 28},
      {"naam": "Mohammed Abdulrahman", "leeftijd": 35}
   ]
}
"""

def vind_max_leeftijd(json_data):
   # Zet de JSON data om naar een Python object
   data = json.loads(json_data)

   # Initieer de maximale leeftijd variabele
   max_leeftijd = 0

   # TODO: Schrijf hier de code die door de data loopt,
   #       de leeftijden vergelijkt en de maximale leeftijd vindt

   return max_leeftijd

# Test de functie
print(vind_max_leeftijd(json_data))  # Dit zou 35 moeten printen
```

# Inleveren

Gebruik de code van hierboven en vul de code aan (bij # TODO) zodat de maximale leeftijd wordt gevonden.

Lever de aangepaste en werkende code, gebruik de naam leeftijd-<jouw-naam>.py

# JSON Leeftijd en naam

Gebruik de code van de vorige opdracht, maar zorg dat de functie twee waarden returned; de hoogste leeftijd en de naam van de persoon met de hoogste leeftijd.

De laatste twee regels van de code worden.

```
naam, leeftijd = vind_oudste_persoon(json_data)
print(f'De oudste persoon is {naam} met de leeftijd van {leeftijd} jaar.')
```

Pas de functie aan zodat de naam en leeftijd worden afgedrukt.

# Inleveren

Aangepaste code json4-<jouw-naam>.py.

# API and JSON

We hebben de volgende code

```
import requests
import json

def vind_langste_username():
    # Verzend een GET verzoek naar de JSONPlaceholder API
    response = requests.get("https://jsonplaceholder.typicode.com/users")

    # Zet de JSON response om naar een Python object
    data = json.loads(response.text)

    # Initieer de lengte variabele voor de langste gebruikersnaam en de bijbehorende gebruiker
    max_len = 0
    user_met_max_len = ''

    # TODO: Schrijf hier de code die door de data loopt,
    #       de lengtes van de gebruikersnamen vergelijkt en de langste gebruikersnaam en bijbehorende gebruiker v

    return user_met_max_len, max_len

# Test de functie
naam, lengte = vind_langste_username()
print(f'De gebruiker met de langste gebruikersnaam is {naam} met een lengte van {lengte} karakters.')
```

Deze code laadt een JSON-bestand via een API. Vervolgens moet je de langste naam selecteren. Vul daarvoor de code aan.

Let op, je hebt de library requests nodig (import requests) deze library moet je waarschijnlijk installeren.

Dat doe je met het volgende commando.

```
pip install requests
```

Pip is de installer voor Python (zoals composer voor PHP).

Gebruik alle hulpbronnen, maar zorg dat je de code kan uitleggen.

# Inleveren

Aangevulde werkende code, *api-<jouw-naam>.py*

# Flask installatie

Met Python kan je ook web applicaties maken. Daarvoor zijn twee bekende frameworks beschikbaar, Django en Flask.

Django is een beetje de grote broer van Flask. Flask is eenvoudiger en beter geschikt voor wat eenvoudigere web applicaties.

Flask heeft niet echt een MVC-structuur, maar heeft wel routing, een mooie template engine en database-integratie.

Wij gaan kennismaken met Flask en zullen kijken naar routing en de (Jinja) template engine van Flask.

Maar eerst installeren.

# Installatie

We hebben Python al geïnstalleerd (heb je dat niet meer vraag dan Chat GPT hoe je Python moet installeren of kijk in Python L1).

Met pip installeer je Flask:

```
pip install flask
```

Maak een map waarin je jouw Flask project gaat maken, en noem die map bijvoorbeeld mijFirstFlask.

Open de nieuwe folder in VCS.

De eerste stap

# Het eerste begin

1. Maak een nieuw Flask-project en maak een `templates`-map aan in de projectdirectory. Hierin plaatsen we de HTML-templates.
2. Maak een nieuw Python-bestand, bijvoorbeeld `app.py`, en plaats het in de projectdirectory.

3. Open het Python-bestand (`app.py`) met een teksteditor en voeg de volgende code toe:

```
from flask import Flask, render_template


app = Flask(__name__)


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


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

4. Maak een nieuw HTML-bestand in de templates-map, index.html, en open het met een teksteditor.
   Voeg deze HTML-code toe aan het index.html-bestand en zet achter Welkom jouw naam.

```
<!DOCTYPE html>
<html>
<head>
    <title>Mijn Flask-project</title>
</head>
<body>
    <h1>Welkom <vul hier jouw naam in> bij mijn Flask-project!</h1>
    <p>Dit is een basisstructuur voor een Flask-project.</p>
</body>
</html>
```

5. Sla zowel het Python-bestand (`app.py`) als het HTML-bestand (`index.html`) op.
6. Ga naar de opdrachtprompt of terminalvenster en navigeer naar de projectdirectory waar het Python-bestand zich bevindt.
7. Voer het volgende commando in: `python app.py` (of `python3 app.py` als je meerdere Python-versies hebt geïnstalleerd).
8. Flask start de ontwikkelingsserver en geeft een URL weer, bijvoorbeeld `http://127.0.0.1:5000/`.
9. Open de weergegeven URL in je webbrowser en je zou de inhoud van het `index.html`-bestand moeten zien, inclusief de welkomstboodschap.

Als je deze boodschap ziet, betekent dit dat Flask correct is geïnstalleerd en functioneert. Je kunt de boodschap in de `hello`-functie wijzigen naar elk gewenst bericht.

# Flask form

In deze opdracht ga je in Flask je een eenvoudige webpagina maken met een formulier. In het formulier vraag je de gebruiker om zijn naam en vervolgens gebruik je die naam voor een begroeting.

1. Maak een nieuw Python-bestand, bijvoorbeeld `app.py`, en open het met een teksteditor.
2. Voeg de volgende code toe aan het Python-bestand:

*python*

```
from flask import Flask, render_template, request

app = Flask(__name__)

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

@app.route('/greet', methods=['POST'])
def greet():
    name = request.form.get('name')
    return render_template('greet.html', name=name)

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

3. Maak een `templates`-map in dezelfde directory als het Python-bestand.
4. Maak een nieuw HTML-bestand genaamd `index.html` in de `templates`-map en open het met een teksteditor.
5. Voeg de volgende code toe aan het `index.html`-bestand:

```
<!DOCTYPE html>
<html>
<head>
    <title>Flask Opdracht</title>
</head>
<body>
    <h1>Vul je naam in:</h1>
    <form action="/greet" method="POST">
        <input type="text" name="name" required>
        <input type="submit" value="Verzenden">
    </form>
```

```
</body>
</html>
```

6. Maak een nieuw HTML-bestand genaamd `greet.html` in de `templates`-map en open het met een teksteditor.
7. Voeg de volgende code toe aan het `greet.html`-bestand:

```
<!DOCTYPE html>
<html>
<head>
   <title>Flask Opdracht</title>
</head>
<body>
   <h1>Hallo, {{ name }}!</h1>
</body>
</html>
```

8. Sla zowel het Python-bestand (`app.py`), het `index.html`-bestand als het `greet.html`-bestand op.
9. Ga naar de opdrachtprompt of terminalvenster en navigeer naar de projectdirectory waar het Python-bestand zich bevindt.
10. Voer het volgende commando in: `python app.py.`
11. Flask start de ontwikkelingsserver en geeft een URL weer, `http://127.0.0.1:5000/`.
12. Open de weergegeven URL in je webbrowser en je zou een formulier moeten zien waar je je naam kunt invullen en verzenden.
13. Nadat je op "Verzenden" hebt geklikt, zou je begroet moeten worden met de boodschap "Hallo, [naam]!" waarbij `[naam]` de ingevoerde naam is.

Met deze eenvoudige opdracht kun je een formulier maken in Flask en de ingevoerde naam gebruiken om een begroeting weer te geven. Je kunt deze opdracht verder aanpassen en uitbreiden met meer functionaliteit en stijling naar wens.

Zorg ervoor dat je begrijpt hoe dit werkt, want in een volgende opdracht moet je zelf een formulier maken.

# JINJA

De HTML-template is een zogenaamde JINJA template. In een JINJA template kan je python variabelen kan je afdrukken door de variabelen tussen {{ }} te zetten.

Als je (via ChatGPT) meer informatie wilt geef dan aan dat je in Flask met een JINJA template werkt.

# Inleveren

1. Schermafdruk van de gehele browser van de pagina waarin je de naam moet invullen, en
2. schermafdruk van de gehele browser van de pagina waarin de boodschap wordt getoond.

# Form aanpassen

Neem de code die je bij de vorige opdracht hebt gemaakt als uitgangspunt.

Pas de code die aan de route /greet is verbonden aan:

```
@app.route('/greet', methods=['POST'])
def greet():

 current_time = datetime.now().strftime('%H:%M:%S')
    current_date = datetime.now().strftime('%Y-%m-%d')

name = request.form.get('name') return render_template('greet.html', name=name)
```

Je ziet dat er twee variabelen zijn bijgekomen, *current_time* en *current_date.*

In de HTML template moet je deze waarden afdrukken. Je moet daarvoor twee dingen doen:

1. Zorg ervoor dat je de waarden current_time en current_date doorgeeft aan de HTML-template.
   Pas daarvoor de laatste regel van de def greet():  aan.
   Gebruik Internet of ChatGPT om uit te vinden hoe dat moet.
2. Pas de Jinja template aat en zorg ervoor dat de datum en tijd wordt afgedrukt.

# Inleveren

1. Aangepaste code in app.py (waar de def greet() in staat, en
2. aangepaste Jinja template, waar de datum en tijd wordt afgedrukt.

# Project Boekenrecensie-applicatie

## Doel:

Het creëren van een webapplicatie waar gebruikers boeken kunnen zoeken, beoordelen en recensies kunnen achterlaten. De app kan ook boekgegevens ophalen van een externe API zoals Google Books.

## Benodigdheden:

- Basiskennis van Python en Flask
- Begrip van databases (SQLAlchemy kan een goede keuze zijn voor Flask)
- Kennis van HTML, CSS voor de frontend (optioneel Bootstrap voor styling)
- Een teksteditor en een Flask-ontwikkelomgeving

## Stappen:

1. **Opzetten van de Flask-omgeving:**
   - Maak een nieuw Flask-project en stel de basisstructuur op.
   - Zorg voor routes en templates voor de belangrijkste pagina's (Home, Zoeken, Boekdetails, Recensies).
2. **Database Integratie:**
   - Ontwerp een database-schema voor het opslaan van gebruikers, boeken en recensies.
   - Gebruik SQLAlchemy om je database te integreren met Flask.
   - Maak modellen voor elke tabel in je database.
3. **Gebruikersauthenticatie:**
   - Implementeer gebruikersregistratie en -authenticatie.
   - Zorg ervoor dat gebruikers kunnen inloggen en hun eigen recensies kunnen achterlaten.

4. **Integratie van een Boeken-API:**
    - Kies een externe boeken-API zoals Google Books.
    - Implementeer functionaliteit waarmee gebruikers boeken kunnen zoeken via de API.
    - Laat basisinformatie over de boeken zien en bied de mogelijkheid om deze toe te voegen aan je eigen database.
5. **Recensie Functionaliteit:**
    - Sta gebruikers toe recensies en beoordelingen voor boeken achter te laten.
    - Toon deze recensies op de boekdetailpagina's.
6. **Frontend Ontwikkeling:**
    - Ontwerp de frontend met HTML/CSS.
    - Gebruik Bootstrap of een vergelijkbaar framework om de ontwikkeling te versnellen en de app responsief te maken.
7. **Extra Features (optioneel):**
    - Voeg functionaliteiten toe zoals het aanbevelen van boeken op basis van beoordelingen of genres.
    - Implementeer een systeem voor gebruikers om boeken aan hun favorieten toe te voegen.
8. **Testen en Debuggen:**
    - Test de applicatie grondig.
    - Zorg voor foutafhandeling en zorg dat de app soepel werkt.

# Opdracht

1. Maak een planning en verdeel de taken en bepaal het benodigd aantal uren. Laat de planning goedkeuren.
2. Nadat een grof ontwerp. Oftewel een ontwerp op hoofdlijnen.
3. Maak de code met Python en Flask. Zorg dat alles werkt en dat de GUI er netjes uit ziet. Gebruik een CSS framework zoals Bootstrap.

# Project RSVP

## Inleiding

Een RSVP-applicatie is een soort software of webapplicatie die wordt gebruikt voor het beheren van uitnodigingen en reacties (RSVP's) voor evenementen. RSVP staat voor "Répondez s'il vous plaît", een Franse uitdrukking die "Antwoord alstublieft" betekent. In de context van evenementenplanning, wordt het gebruikt om aan te geven dat de genodigden worden gevraagd om te bevestigen of ze al dan niet aanwezig zullen zijn.

## Kernfuncties van een RSVP-applicatie:

1. **Uitnodigingen Versturen:**
   - De organisator van een evenement kan digitale uitnodigingen versturen naar de gastenlijst. Dit kan via e-mail of via een link naar de RSVP-applicatie.
2. **Reacties Beheren:**
   - Genodigden kunnen hun aanwezigheid bevestigen of afwijzen via de applicatie. Ze kunnen vaak ook specifieke voorkeuren of opmerkingen toevoegen, zoals dieetbeperkingen of plus-één informatie.
3. **Evenementendetails:**
   - De applicatie toont details over het evenement, zoals datum, tijd, locatie en eventuele andere relevante informatie.
4. **Gastenlijstbeheer:**
   - De organisator kan de gastenlijst en de reacties in real-time volgen. Dit helpt bij het plannen van de logistiek van het evenement, zoals catering, zitplaatsen en accommodatie.
5. **Communicatie:**
   - De applicatie kan worden gebruikt voor verdere communicatie met de gasten, zoals updates over het evenement, herinneringen of post-evenement bedankjes.
6. **Integraties:**
   - Geavanceerde RSVP-applicaties kunnen integreren met kalenderapps, e-maildiensten, sociale media en zelfs ticketing-systemen.

# Gebruiksscenario's:

- **Bruiloften en Feesten:** Voor persoonlijke evenementen zoals bruiloften, verjaardagsfeestjes, jubilea, waar de organisator een duidelijk beeld wil hebben van wie er zal komen.
- **Zakelijke Evenementen:** Conferenties, workshops, bedrijfsfeesten, waarbij het belangrijk is om de opkomst te beheren en te plannen volgens het aantal deelnemers.
- **Openbare Evenementen:** Culturele evenementen, concerten, gemeenschapsevenementen, waarbij de organisatoren de opkomst moeten schatten en beheren.

# Opdracht

1. Maak een planning en verdeel de taken en bepaal het benodigd aantal uren. Laat de planning goedkeuren.
2. Nadat een grof ontwerp. Oftewel een ontwerp op hoofdlijnen.
3. Maak de code met Python en Flask. Zorg dat alles werkt en dat de GUI er netjes uit ziet. Gebruik een CSS framework zoals Bootstrap.

# Python 3 - Game of Pong

# *Opdracht 1 - het scherm*

## Inleiding

In deze module gaan we een oud spel 'Pong' maken. Pong is een oud Arcade-spel en je keert een hiermee een paar basis concepten van game development.

Deze module is niet makkelijk. Lees goed wat er staat en probeer de code te begrijpen. Begrijp je de code niet, dan loop je vast verderop in de cursus.

Houd je aan de opdracht. Een alternatieve versie van Pong (via AI) wordt niet geaccepteerd. Tevens wordt code waar niet om is gevraagd ook afgekeurd.

Nogmaal lees goed wat er staat en vraag telkens aan het eind van de opdracht of je alles begrijpt. Indien niet lees de opdracht dan nog een keer door. Vraag een mede student en als je er dan nog niet uitkomt vraag de docent.

## turtle

We gaan gebruik maken van de turtle module van Python.

```
import turtle
```

De `turtle` module in Python is een eenvoudige manier om grafische tekeningen te maken door middel van een virtuele "schildpad" die over het scherm beweegt en lijnen tekent.

De turtle-module gebruiken we om grafische objecten te maken en te verplaatsen. Het helpt ons om het speelveld, de paddle, en de bal te tekenen.

Nadat de turtle module is geïmporteerd, maken we een window (venster) voor het spel.

```
# Venster instellen
wn = turtle.Screen()
wn.title("Pong voor <Jouw Naam>")
wn.bgcolor("black")
wn.setup(width=800, height=600)
wn.tracer(0)
```

```
input("Press any key to continue...") # tijdelijke toevoeging t.b.v. testen
```

Hier maken we een venster met de titel "Pong voor<Jouw Naam>". Vervang <Jouw Naam> door jouw naam.

De achtergrondkleur is zwart en de afmetingen van het venster zijn 800 bij 600 pixels. Met `wn.tracer(0)` zorgen we ervoor dat het scherm alleen ververst als wij dat willen, wat handig is voor het maken van animaties.

# Opdracht

Test de code.

Leg in je eigen woorden uit waarom regel 8 in de code staat. Wat heeft regel 8 te maken met testen?

Wat gebeurt er als deze regel er niet in staat.

# Inleveren

Korte uitleg in eigen woorden waarom regel 8 in de code staat.

# *Opdracht 2 - paddle*

In deze opdracht maken we een paddle voor de speler. De paddle is een rechthoek die de speler kan verplaatsen om de bal terug te kaatsen.

We gaan ook leren wat coördinaten zijn.

De code voor de paddle:

```
# Paddle
paddle = turtle.Turtle()
paddle.shape("square")
paddle.color("white")
paddle.shapesize(stretch_wid=6, stretch_len=1)
```

```
paddle.penup()
paddle.goto(0, 0)


wn.update()
```

## Uitleg:

- We maken een turtle-object genaamd `paddle`.
- `paddle.speed(0)` zorgt ervoor dat de paddle direct getekend wordt zonder animatie.
- `paddle.shape("square")` geeft de paddle de vorm van een vierkant.
- `paddle.color("white")` maakt de paddle wit.
- `paddle.shapesize(stretch_wid=6, stretch_len=1)` maakt de paddle rechthoekig door de breedte te stretchen.
- `paddle.penup()` zorgt ervoor dat de paddle niet tekent als hij beweegt.
- `paddle.goto(0, 0)` plaatst de paddle op het midden van het scherm verschijnt.
- `wn.update()` zorgt ervoor dat het scherm wordt geüpdatet.

## Coördinaten

Het scherm is verdeeld in coördinaten. Een coördinaat is (0,0) en dat is het midden van het scherm.

Het eerste cijfer wordt de x-waarde genoemd en is de horizontale positie (links-rechts).

Het tweede cijfer wordt de y-waarde genoemd en is de verticale positie (boven-beneden).

> Het eerste cijfer in een coördinaat is de x-waarde en de tweede coördinaat is de y-waarde. De x-waarde geeft de horizontale postie weer en de y-waarde geeft de vertikale positie weer.

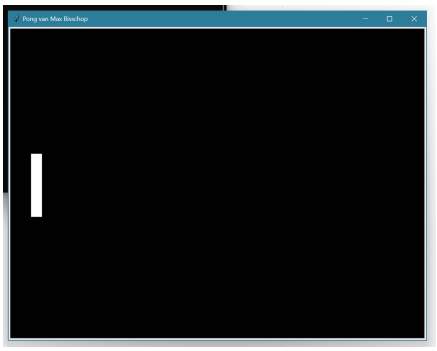
Voorbeelden

| | |
|---|---|
| (10,0) | x-waarde is 10 en de y-waarde is 0; 10 punten rechts van het midden. |
| (0,10) | 10 punten boven het midden. |
| (-10,0) | 10 punten links van het midden. |
| (0,-10) | 10 punten onder het midden |

| | |
|---|---|
| (10,-10) | 15 punten rechts en dan 10 naar beneden vanuit het midden. |

# Opdracht

Verander de code zodat de paddle aan de linkerkant van het scherm staat. Hiervoor moet je dus de coördinaten aanpassen.



Test je programma uit en zorg ervoor dat je een witte paddle ziet aan de **linkerkant** van het venster.

# Inleveren

Screenshot van het scherm waarin je de paddle ziet.

# *Opdracht 3 - paddle bewegen*

We gaan ervoor zorgen dat je de paddle kan bewegen.

De paddle kan twee richtingen op: up (naar boven) en down (naar beneden).

We maken eerst twee functies die de paddle up of down kunnen bewegen.

```
def paddle_up():
    y = paddle.ycor()
    if y < 250:
        y += 20
    paddle.sety(y)
```

```
def paddle_down():
    y = paddle.ycor()
    if y > -240:
        y -= 20
    paddle.sety(y)
```

### Uitleg:

- `paddle_up()` verhoogt de y-coördinaat van de paddle met 20 als deze nog niet de bovenkant van het scherm heeft bereikt.
- `paddle_down()` verlaagt de y-coördinaat van de paddle met 20 als deze nog niet de onderkant van het scherm heeft bereikt.

Als we deze code plaatsen dan gebeurt er nog niets. Dat komt omdat we de coördinaten wel veranderen, maar we doen geen scherm-update.

Gedurende het spel moeten we telkens een scherm update uitvoeren. Dit doen we in een loop die we voor nu even 'oneindig maken' (stopt niet vanzelf).

```
while True:
    wn.update()
```

Deze loop wordt ook wel de *game loop* genoemd.

> De game-loop is het programma-onderdeel dat steeds controleert of er iets gebeurt en tekent telkens een nieuw scherm tekent.

# Opdracht

Test je spel uit en als het goed is, kan je de paddle bewegen.

Tip: Omdat de code nu een *game-loop* heeft, zal je deze moeten onderbreken met een CTRL-C in de command prompt waar je het spel hebt gestart.

# Inleveren

De gehele werkende code (.py) tot nu toe.

# *Opdracht 4 - de bal*

Om te beginnen maken we een bal, op eenzelfde manier als we de paddle hebben gemaakt.

Deze code staat boven de *game-loop*.

```
# Bal
ball = turtle.Turtle()
ball.shape("square")
ball.color("white")
ball.penup()
ball.goto(0, 0)
ball.dx = 0.1
ball.dy = -0.1
```

## Uitleg:

- `ball = turtle.Turtle()`: Maakt een nieuw turtle-object genaamd `ball`.
- `ball.speed(1)`: Stelt de animatiesnelheid van de bal in.
- `ball.shape("square")`: Geeft de bal de vorm van een vierkant.
- `ball.color("white")`: Maakt de bal wit.
- `ball.penup()`: Voorkomt dat de bal lijnen tekent tijdens beweging.
- `ball.goto(0, 0)`: Plaatst de bal in het midden van het scherm.
- `ball.dx = 0.2`: Stelt de horizontale snelheid van de bal in.
- `ball.dy = -0.2`: Stelt de verticale snelheid van de bal in.

## Test

Test je code uit. Als het goed is zie je een stilstaande bal in het midden van het scherm.

## Beweging

Waarom beweegt de bal niet??

Dat komt omdat de coördinatoren van de bal niet worden aangepast.

In de game-loop, voor het updaten van het scherm, gaan we dit doen met deze code.

```
# Beweeg de bal
ball.setx(ball.xcor() + ball.dx)
ball.sety(ball.ycor() + ball.dy)
```

### Uitleg:

- `ball.setx(ball.xcor() + ball.dx)`: Verplaatst de bal horizontaal door de huidige x-coördinaat te verhogen met `ball.dx`.
- `ball.sety(ball.ycor() + ball.dy)`: Verplaatst de bal verticaal door de huidige y-coördinaat te verhogen met `ball.dy`.

## Test

Test je code uit. Je ziet nu een bewegende bal, die vrij snel het beeld uit verdwijnt.

# Opdracht

Leg in eigen woorden uit waarom de bal uit het scherm verdwijnt.

# Inleveren

Korte uitleg waarom de bal van het scherm verdwijnt in eigen woorden.

# *Opdracht 5 - bal stuitert terug*

Je weet nu dus waarom de bal van het scherm verdwijnt?

De vraag is nu: wat doen we eraan?

Het punt is dat als de bal de randen van ons scherm nadert de bal terug moet 'stuiteren'.

Dat betekent dat we moeten detecteren wanneer de bal de randen bereikt en dat we dan de richting van de bal moeten veranderen.

Laten we proberen te detecteren als de bal de rand raakt en als dat zo is dan veranderen we de richting.

Daarvoor plaatsen we deze code in de game-loop.

```
# detecteer randen van het scherm
if (ball.xcor() > 200 or ball.xcor() < -200 ):
   ball.dx *= -1
if (ball.ycor() > 200 or ball.ycor() < -200 ):
   ball.dy *= -1
```

De ball.dx en ball.dy geven de snelheid in horizontalen- en verticale richting aan.

Er staat dus: als de x-coördinaat van de bal groter dan of kleiner dan een bepaalde waarde is, vermenigvuldig dan de horizontale snelheid met -1. Stel de snelheid is 1 dat wordt deze dan -1 en stel de snelheid is -1 dan wordt die 1. Dus als de snelheid naar rechts is, dan gaat die naar links en andersom.

Hetzelfde gebeurt met de y-coördinaat.

## Test

Test je code uit. Je ziet nu een bewegende bal, die terug stuitert, maar niet precies bij de randen van het scherm.

## Taak 1

Pas de code aan zodat de bal werkelijk aan de rand van het scherm terug 'stuitert'. Pas hiervoor de coördinaten in de code aan.

## Taak 2

Zorg ervoor dat de bal aan alle kanten van het scherm terug stuitert, maar **niet** als die aan de linkerkant komt.

Dus de bal stuitert terug als die aan de onder-, boven of rechterkant komt, maar niet als die aan de linkerkant komt.

# Inleveren

De aangepaste volledige code tot nu toe.

# *Opdracht 6 - raak of mis?*

De bel verdwijnt nu als die aan de linkerkant van het scherm komt. Dat is goed, behalve als daar de paddle staat. We moeten dus code maken in de game-loop die detecteert of de ball en de paddle elkaar raken.

```
# Detecteer botsing met paddle
if (ball.dx < 0 and ball.xcor() < -350): # ball beweegt naar links en zit bij de linker zijkant.
    if (paddle.ycor() - 60 < ball.ycor() < paddle.ycor() + 60): # bal 'raakt' de bal
        ball.dx *= -1 # beweeg de bal de andere kant uit (horizontaal)
        ball.dy *= -1 # beweeg de bal de andere kant uit (verticaal)
    else:
        ball.dx = 0
        ball.dy = 0
```

## Uitleg:

- `ball.dx < 0`: Controleert of de bal naar links beweegt.
- `ball.xcor() < -360`: Controleert of de bal bij/over de linker zijlijn is..
- `paddle.ycor() - 60 < ball.ycor() < paddle.ycor() + 60`: Controleert of de bal binnen het bereik van de paddle is. De paddle was size 6. Dit betekent 6 eenheden en in de turtle library is 1 eeheid 20 pixels. De paddle is dus 6*20 = 120 pixels hoog. Vanuit het midden gezien is dat dus 60 pixels omhoog en 60 pixels omlaag.
- `ball.dx *= -1`: Verandert de richting van de bal horizontaal.
- `ball.dx *= 0` en `ball.dy *= 0`: Zet de bal stil. De horizontale- en verticale snelheid wordt 0.

## Test

Test je code uit. Op zich moet het spel nu werken. De bal verdwijnt als je hem mist maar als je de bal raakt met de paddle dan stuitert de bal terug.

# Opdracht

Maak een variabele score.

```
# Score variabele
score = 0
```

Telkens als je de bal raakt dan verhoog je de score.

Om te testen of het werkt, druk je telkens nadat de score is, verhoogt de variabele score af.

```
score = score + 1
print(score)
```

Speel het spel en houd je cmd-windows in de gaten. Als het goed is wordt de score daar afgedrukt.

# Inleveren

Eens schermafdruk van het gehele cmd-window waarin de score wordt afgedrukt.

# *Opdracht 7 - Score op scherm*

Weg hebben een variabele score. Deze gaan we op het scherm plaatsen.

Om de score bij te houden en op het scherm te tonen maken we een nieuw turtle object.

```
# Score variabele
score = 0

# Pen om de score weer te geven
pen = turtle.Turtle()
pen.speed(0)
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 260)
pen.write("Score: 0", align="center", font=("Courier", 24, "normal"))

def update_score():
    pen.clear()
    pen.write("Score: {}".format(score), align="center", font=("Courier", 24, "normal"))
```

We hebben nu een 'lege score', om die te vullen maken we een functie.

```
def update_score():
    pen.clear()
```

```
    pen.write("Score: {}".format(score), align="center", font=("Courier", 24, "normal"))
```

We hebben de functie om de score op het scherm af te drukken. Wat we nu moeten doen is de score telkens als die is veranderd opnieuw op het scherm zetten.

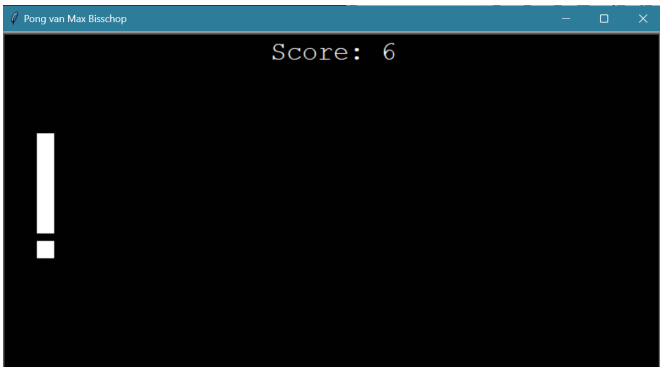
```
update_score()  # Werk de score weergave bij
```

Als de code werkt dan kan de print() die we in de vorige opdracht hebben geplaatst uit de code worden gehaald.

# Inleveren

Eens schermafdruk van het spel waarbij de score wordt getoond. Zorg ervoor dat er een score hoger dan 0 wordt getoond.

# Voorbeeld




# *Opdracht 8 - Levens*

Telkens als het spel stop omdat je de bal niet hebt geraakt me de paddle, stop het spel.Het spel stopt door de bal stil te zetten. De x- en y-snelheid wordt op 0 gezet.

Dit gaan we aanpassen. Je begint met drie levens en telkens als je de bal mist, stopt het spel voor 3 seconden. Als je nog genoeg levens hebt, zal het spel na 2 seconden weer verder gaan. De bal verandert van richting (terug naar recht) en de snelheid wordt weer op de beginwaarde 0.1 gezet.

Als je levens op zijn dan stopt het spel en wordt er in het midden van het scherm 'Game Over' getoond.

# Stappenplan

## Stap 1 - Laat "Game Over" zien.

Maak een functie die de tekst game over laat zien.

```
def game_over():
    pen.goto(0, 0)
    pen.write("Game Over", align="center", font=("Courier", 36, "normal"))
```

Roep deze functie aan op de plaats in de game-loop waar het spel eindigt.

## Stap 2a - Maak een variabele die het aantla levens bijhoud

Net zoals je de score bijhoud, houd je ook het aantal levens bij. Maak een variabele en zet de initiële waarde op 3. Zorg dat op de juisten plaats in de game-loop het aantal levens wordt veminderd met één.

## Stap 2b - Game stopt alleen als aantal levens 0 is.

Nu veranderd het moment dat we game-over laten zien. Alleen als het aantal levens 0 is, laten we game over zien.

Op de plaats in de code waar je de bal hebt gemist, kunnen er twee dingen gebeuren:

1. Je controleert of het aantal levens 0 is, dan zet je de snelheid van de bal op 0 en roep je de functie game_over() aan.
2. Is het aantal levens niet 0, dan vermidner je het aantal levens met 1 en laat je de bal terug stuiteren alsof je heb had geraakt met de paddle.

## Stap 3 - druk jet aantal levens af

Je kunt hiervoor de bestaande functie aanpassen en naast de score ook het aantal levens aanpassen.

```
def update_score():
    pen.clear()
    pen.write("Score: {} Levens: {}".format(score, lifes), align="center", font=("Courier", 24, "normal"))
```

Zorg er wel voor dat als je het aantal levens aanpast, je deze functie aanroept zodat de score wordt ververst.

Zorg er ook voor dat er bij het begin van het spel "Score: 0 Levens: 3" wordt getoond.

### Stap 4 - pauzeer de game als je de bal mist

Met het volgende commando kan je de game 3 seconden pauzeren (vergeet niet de module time te importeren).

```
time.sleep(3)
```

Zodra je de bal mist, dan pauzeer je de game voor 3 seconden.

# Inleveren

Eens schermafdruk van het spel waarbij de score en het aantal levens wordt getoond.

# Voorbeeld


# *Opdracht 9 - be creative!*

Maak een variatie op het spel. Kies zelf één of meer van de volgende opties.

1. De bals stuitert nu altijd op een bepaalde manier. Dat komt omdat de ball.dx en ball.dy een vaste waarde hebben 0,1 of -0.1. Je kunt dit laten variëren. Dat kan je at random doen, maar je kan het ook laten afhangen waar op de paddle je de bal raakt. Raak je hem aan de boven of onderkant dan kan je wat doen met de ball.dy, maak hem dan bijvoorbeeld 01.15 of -0.15.
2. Je zou de snelheid langzaam kunnen opvoeren. Bijvoorbeeld na elke 5 punten gaat de bal iets sneller. Vergeert niet de snelheid weer aan te passen na het verliezen van een leven.

3. Je kan de paddle groter en kleiner maken. Bijvoorbeeld na elke 5 punten iets kleiner. Vergeet hem dan niet na het verliezen van een leven weer op de originele grootte te zetten.
4. Je zou de paddle in het midden een gat kunnen geven. Komt de bal precies in het gat dn krijg je er een leven bij.
5. Pas de kleur van de bal aan al naar gelang je meer punten scoort.

Heb jij nog een ander idee, bespreek dit dan met de docent.

## Inleveren

Beschrijf op de eerste regels in de code in commentaar wat je hebt gekozen en hoe je dat gedaan hebt. Lever daarna je gehele code in.

# *Opdracht 10 - Quiz*

Vraag in de klas (op school) of je de bijbehorende quiz mag maken. De quiz bevat 7 vragen over deze module.

Je krijgt 7 vragen en je moet er minimaal 5 goed hebben (dat is 70% of hoger).

Je hebt maximaal 3 pogingen.

## Inleveren

Schermafdruk met een uitslag van 70% og hoger.