Python

- Python 1 JSON
- Python 2 Flask
- Python 3 - Game of Pong
- Python 3 - Game of Pong Challenge

Python 1 JSON

Chat GPT

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 je 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.

image-1688543576041.png

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?

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

image-1688543664694.png

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.

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).

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

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.

Opdracht

Laat zien dat je python succesvol hebt geïnstalleerd.

Open een CMD window en type python --version.

Je ziet bijvoorbeeld dit:

Screenshot 2023-07-10 134645.png

Lever een screenshot van je eigen CMD-window in waarin je laat zien dat Python is geïnstalleerd.

--

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(
            break
        elif gok < willekeurig_getal:
            print("Te laag! Probeer het opnieuw.")
        else:
            print("Te hoog! Probeer het opnieuw.")

raad_het_getal()
```

Opdracht

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.

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

JSON 1

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.

Opdracht

Haal regel 10, 11 en 13 weg (dat zijn de drie regels en de eerste begint met naam=).

Pas nu het print commando aan zodat de juiste gegevens worden afgedrukt.

Inleveren

Aangepaste code json1-<jouw naam>.py

JSON 2

Neem voor deze opgave de code van de vorige opdracht JSON 1

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).

Inleveren

json2-<jouw-naam>.py

loop en if

We hebben nu wat geoefend met Python. We gaan nu even kijken naar een paar belangrijke kenmerken van Python. Ja mag gebruik maken van chat GPT. We gaan één stuk code maken waar we telkens een paar regels aan vast plakken.

Comments

Zet bovenaan in je code de datum en je naam in commentaar.

Loops.

Maak een loop waarmee de getallen 1 t/m 10 worden afgedrukt.

Maak een loop waarmee de getallen 12,14,16,18,....36 worden afgedrukt.

If-then-else

Maak een if-then-else. Test de variabele leeftijd. Is deze groter dan 18 druk dan de tekst af:

"Je bent 18, zorg ervoor dat je een zorgverzekering heb afgesloten".

Ben je jonger dan 18, druk dan de tekst af:

"Je hoeft nog geen zorgverzekering af te sluiten"

Inleveren

Je hebt nu een stuk Python code met commentaar, twee loops en een if-then-else.

Lever de code in loops-if-<jouw naam>.py

strings

Strings zijn variabelen waarmee je niet kan rekenen.

Jouw naam kan een string zijn.

naam = '*Ahmed Al-Hassan*' , weet je nog in PHP zou je *$naam* = '*Ahmed Al-Hassan*' gebruiken.

Maak een stukje code waarin je de volgende stappen laat zien:

1. **String Creatie** Maak een string aan met de waarde 'Hallo Wereld'.
2. **String Concatenatie** Maak twee strings, "Python" en "Programmering". Voeg deze twee strings samen om een nieuwe string te maken, "Python Programmering".
3. **String Lengte** Maak een string aan met de waarde 'OpenAI'. Gebruik de `len()` functie om de lengte van deze string te bepalen.
4. **Toegang tot elementen van de string** Maak een string 'Kunstmatige Intelligentie'. Print het vijfde karakter van de string.
5. **String Slice** Gebruik de string van de vorige oefening. Print de substring van het derde tot en met het zevende karakter.
6. **String Bewerkingen** Maak een string 'Python is leuk!'. Verander 'leuk' naar 'geweldig' in de string.

Gebruik de code die hieronder staat en vul de code aan op de plaats van puntjes.

```
# 1. String Creatie
my_string = ......
print(my_string)

# 2. String Concatenatie
string1 = 'Python'
```

```
string2 = 'Programmering'
new_string = ........
print(new_string)

# 3. String Lengte
ai_string = 'OpenAI'
lengte = ........
print(lengte)

# 4. Toegang tot elementen van de string
intelligence_string = 'Kunstmatige Intelligentie'
print(.........)

# 5. String Slice
print(.......])

# 6. String Bewerkingen
fun_string = 'Python is leuk!'
new_fun_string = ..........('leuk', 'geweldig')
print(new_fun_string)
```

Inleveren

1. De code die uit de 6 onderdelen bestaat, string-<jouw naam>.py
2. Een screenshot van je CMD window waarin je laat zien dat de code werkt en 6 maal de juiste waarde afdrukt.

JSON 3

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 4

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 en 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 bijbeho

    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} karakte
```

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*

Python 2 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.

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.

Inleveren

Een schermafdruk van je gehele browser waarin je laat zien dat de template de welkomstboodschap wordt getoond.

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:

app.py

```
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)
```

Uitleg

Met **@app** wordt een route bepaald. Zo wordt er bijvoorbeeld bepaald dat als er wat naar */greet* wordt gepost, de functie *def greet():* wordt uitgevoerd.
render_template opent een *html tempate* en in het voorbeeld bij *greet()* wordt de variabele name meegegeven en die krijgt de waarde van name. (de eerste *name* is de naam die je in de template kan gebruiken en de tweede *name* is de variabele).
Heb je meer vragen over de code, zoek het dan op (internet, ChatGPT) of vraag het aan de docent.

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>
```

Uitleg

De templates in Flask, zijn JINJA tempaltes. Dit zijn HTML templates waarin variabelen kunnen worden afgedrukt. In dit voorbeeld {{ name }}.
Een JINJA template kan ook code bevatten dat staat tussen {% en %}, daarover later meer.

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.

Aanpassen app en form

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.

Welke dag?

Inleiding

Weet jij op welke dag van de week jij bent geboren?

We gaan een formulier waarin we de gebruiker om een datum vragen. Python bepaald dan op welke dag van de week deze datum valt en toont dat aan de gebruiker.

Jij voert dus (bijvoorbeeld) jouw geboortedag in en jouw programma berekend dan dat jij op een maandag bent geboren.

App.py

De app.py ziet er als volgt uit:

```
from flask import Flask, render_template, request
import datetime

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
```

```
def index():
    if request.method == 'POST':
        date_string = request.form['date']
        date_object = datetime.datetime.strptime(date_string, "%Y-%m-%d")
        day_as_number = date_object.weekday()
        days_of_week_array = ['Maandag', 'Dinsdag', 'Woensdag', 'Vrijdag', 'Zaterdag', 'Zondag']
        day_of_week = days_of_week_array[day_as_number]
        return render_template('template.html', day_of_week=day_of_week)
    return render_template('template.html')

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

De functie index() kijkt eerst of er een post is gedaan of dat er een pagina wordt opgevraagd. Als er een post is gedaan dan moet het resultaat worden berekend en worden getoond en anders moet de gebruiker eerst om input worden gevraagd (index.html).

Er zijn dus twee manieren om de functie te returnen: via de result.html template of via de index.html template.

Opdracht

Maak een werkende (Flask) app en plaats alle bestanden op de juiste plaats.

De templates die je moet gebruiken staan hieronder.Neem de code van de *app.py* over en vul de juiste template in. Vervang de 'template.html' voor de juiste template naam!

De templates zelf staan hieronder.

Je moet dus de volgende stappen uitvoeren:

1. Zet alle bestanden in de juiste folders en laat de Flask app werken.
2. Zorg dat de juiste templates worden aangeroepen; vervang hiervoor de 'template.html' in de app.py
3. Probeer te begrijpen hoe de app werkt en voorzie de app.py van commentaar.
   Plaats in ieder geval commentaar bij regels 9, 10 en 11 (de eerste drie regels na de if).
   Leg in eigen woorden uit wat deze regels doen en plaats dat in het commentaar.
4. Test of alles werkt; er zit één foutje in de code. Corrigeer deze fout en lever de gecorrigeerde code van app.py in.

Templates

We hebben twee templates, die moeten beiden in de template directory staan.

De index.html template ziet er als volgt uit:

```
<!DOCTYPE html>
<html>
<head>
    <title>Dag van de week</title>
</head>
<body>
    <h1>Vind de dag van de week</h1>
    <form method="POST" action="/">
        <label for="date">Voer een datum in:</label>
        <input type="date" id="date" name="date" required>
        <br><br>
        <input type="submit" value="Vind dag van de week">
    </form>
</body>
</html>
```

En de result.html ziet er als volgt uit:

```
<!DOCTYPE html>
<html>
<head>
    <title>Dag van de week</title>
</head>
<body>
    <h1>Dag van de week</h1>
    <p>De opgegeven datum valt op een: {{ day_of_week }}</p>
    <a href="/">Terug naar startpagina</a>
</body>
</html>
```

Inleveren

1. Zorg dat de app werkt en dat er commentaar wordt toegevoegd en lever alleen de app.py in.

Weersvoorspelling 1

We gaan een eenvoudige webapplicatie maken die een weersvoorspelling laat zien.

API

We halen de informatie van https://www.weatherapi.com registreer je daar voor een gratis account. Je krijgt dan een API key, zeg maar een sleutel die toegang geeft tot de weersvoorspelling.

Als je de key hebt dan gaan we verder.

Flask app

Maak een nieuwe Flask app. Maak een nieuwe folder en noem die bijvoorbeeld *weather* .

In *weather* maak je een bestand app.py en je maakt twee folders *templates* en *static* .

Screenshot 2023-07-11 210134.png

In het bestand app.py zetten we de volgende code.

```
from flask import Flask, render_template, request
import requests

app = Flask(__name__)

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

@app.route('/forecast', methods=['POST'])
def forecast():
    country = request.form.get('country')
    city = request.form.get('city')

    api_key = 'your_weather_api_key'  # Vervang dit met je eigen WeatherAPI-sleutel

    url =
f'http://api.weatherapi.com/v1/forecast.json?key={api_key}&q={city},{country}&days=3'

    response = requests.get(url)
    data = response.json()

    forecast_days = data['forecast']['forecastday']

    return render_template('forecast.html', country=country, city=city,
```

```
forecast_days=forecast_days)


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

Plaats de persoonlijke API key die je hebt gekregen in de code.

De applicatie werkt als volgt:

1. Je gaat naar localhost:5000 en via de route functie in de app wordt de pagina geopend die in de template index.html staat.
2. Deze *index.html* is een form waarin de gebruiker wordt gevraagd om input; de naam van het land en de naam van de stad.
3. Het formulier wordt gepost naar de route /forecast en daar wordt het ingevoerd land en stad gebruikt om via de API een weersvoorspelling op te vragen.
4. De voorspelling wordt getoond met de template *forecast.html.*

Zorg dat je dit goed begrijpt want je moet zo een paar onderdelen zelf aanvullen.

Allereerst moet je de twee templates maken in de templates folder:

1. index.html
2. forecast.html

In index.html zet je de volgende code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Weersvoorspelling</title>
</head>
<body>
    <h1>Weersvoorspelling</h1>
    <form action="!!zet hier de juiste action in!!" method="POST">
        <label for="country">Selecteer een land:</label>
        !! Zet hier de html-code in om een land in te kunnen voeren.!!
        !! Zorg ervoor dat je juiste naam gebruikt, want die heb je nodig in de code die je aanr
        <label for="city">Selecteer een stad:</label>
        <input type="text" id="city" name="city" required><br><br>
        <input type="submit" value="Zoek weersvoorspelling">
    </form>
</body>
</html>
```

Let op in de index.html is het formulier niet helemaal af:

1. je moet nog invullen waarnaar toe het formulier moet worden gepost en;
2. je moet nog een <input> maken.

In de code staat met !! aangegeven waar je iets moet aanpassen. Dus alles tussen !! en !! moet je vervangen door eigen code!

In forecast.html zet je de volgende code.

```
<!DOCTYPE html>
<html>
<head>
    <title>Weersvoorspelling</title>
</head>
<body>
    <h1>Weersvoorspelling voor {{ city }}, {{ country }}</h1>
    <table>
        <tr>
            <th>Datum</th>
            <th>Min Temperatuur</th>
            <th>Max Temperatuur</th>
            <th>Weersomstandigheden</th>
        </tr>
        {% for day in forecast_days %}
            <tr>
                <td>{{ day['date'] }}</td>
                <td>{{ day['day']['mintemp_c'] }}°C</td>
                <td>{{ day['day']['maxtemp_c'] }}°C</td>
                <td>{{ day['day']['condition']['text'] }}</td>
            </tr>
        {% endfor %}
    </table>
</body>
</html>
```

De forecast template is helemaal compleet.

Pas de template index.html aan en maak de applicatie werkend.

Inleveren

1. De aangepast index.html.
2. Een screenshot waarin je de weers voorspelling laat zien van Amsterdam. Vul hiervoor bij land *Netherlands* en bij stad *Amsterdam.*

Zorg dat alles goed werkt want in de volgende opdracht gaan we onze app uitbreiden.

Voorbeeld

Screenshot 2023-07-11 211742.png

Weersvoorspelling 2

We nemen de code van de vorige opdracht als uitgangspunt.

API output bestuderen

Zet in de app.py na de regel waar de url variabele wordt gemaakt (waarschijnlijk regel 18), de vogelende regel code:

```
print(f"{url}")
```

Draai het programma nog een keer en zoek in het cmd window de url die je net hebt afgerdukt op.

Screenshot 2023-07-11 212742.png

De API key is hier onleesbaar gemaakt.

Kopieer de URL in de browser en bestudeer de JSON die je terug krijgt.

Als je goed kijkt dan zie je ook dat de tijd dat de zon opkomt en ondergaat ook in JSON staat (*sunrise* en *sunset*).

Pas de code aan zodat je de volgende output krijgt:

Screenshot 2023-07-11 213749.png

1. Bedenk eerst welk bestand (template) je moet aanpassen.
2. Zoek in de JSON hoe de structuur in elkaar zit en hoe je de tijden van *sunrise* en *sunset* kan ophalen.
3. Bereid de tabel uit met twee kolommen en vergeet de kolom headers <th> niet.
4. Maak met CSS of Bootsptrap (of een ander CSS framework) jouw tabel netjes zoals in het voorbeeld.

Succes!

Inleveren

1. forecast.html
2. een schermafdruk van de tabel waarin de twee extra kommen *Zon op* en *Zon onder* worden afgedrukt.

Weersvoorspelling 3

We gaan een laatste stukje toevoegen aan onze code.

De opdracht is om de voorspelling er als volgt uit te laten zien.

Screenshot 2023-07-11 215151.png

Wat er is bijgekomen zijn de drie icoontje boven aan de tabel.

1. In de JSON die de API terug geeft staat een link naar het juiste icoontje (zoek naar 'icon').
2. Gebruik die link om een <img> af te drukken in de template.
3. Zet dan de datum (die je al hebt want die wordt in de eerste kolom van de tabel afgedrukt), voor het icoontje.

Nu wordt het lastig.

Probeer de datum om te zetten in een dag van de week (eerst in het Engels) en als het je lukt daarna in het Nederlands. Gebruik internet en ChatGPT.

Inleveren

1. app.py
2. een schermafdruk van je geheel browser met daarin de de dag van de week en de icoontjes boven de tabel zoals in het voorbeeld is aangegeven.

Succes!

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.

image-1716544749432.png

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)

# Toetsenbordbinding
wn.listen()
wn.onkeypress(paddle_up, "w")
wn.onkeypress(paddle_down, "s")
```

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

image-1716630823508.png

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

image-1716632670176.png

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.

Python 3 - Game of Pong Challenge

(moet nog uitgewerkt worden; samenwerkingsopdracht?)

Samenwerken

Deze Challenge heeft 5 opdrachten die allemaal moeten worden gemaakt.

- Iedereen maakt ten minste één opdracht en je helpt elkaar.
- Iedereen moet de code snappen en kunnen uitleggen.
- Jullie presenteren samen de complete code en laten zien hoe het werkt.

De opdrachten zijn deze keer in het Engels, maar je kan deze natuurlijk laten vertalen door AI.

1 - Advanced Ball Movement

Now let's make the movement of the ball more interesting. You should make the ball go faster when it hits the paddle, and randomly change the direction of the ball within a small range.

Instructions

1. Ensure that the speed of the ball increases by 5% each time it hits a paddle.
2. Make the ball have a small random variation in angle when it hits a paddle to make the game more unpredictable.
3. Test your code to check if the ball responds correctly.

Deliverables

1. The code (.py)
2. A short explanation, in a.txt file, about how you implemented the implementation of the variations in speed and angle.
   In this .txt file, you specify the members of the team and the author(s) of the modified

code.

2 - Score Multiplier and Bonus Points

Introduction

In this assignment, we'll introduce a score multiplier feature. If a player hits the ball multiple times without missing, their score will increase faster. This concept is similar to a combo multiplier in many games, encouraging players to keep the ball in play to maximize their score.

Task

1. **Multiplier Logic:**
   - Create a multiplier variable starting at 1.
   - Every time the ball hits the paddle without the player missing, increase the multiplier by 0.5 (up to a maximum of 3x).
   - When the player misses, reset the multiplier to 1.

2. **Update Score Function:**
   - Modify the `update_score()` function to display the current multiplier alongside the score.

Example Code Snippet:

```
# Initialize the multiplier
multiplier = 1.0

# Update the score function to include the multiplier
def update_score():
    pen.clear()
    pen.write(f"Score: {score} Levens: {lives} Multiplier: {multiplier}x", align="center",
font=("Courier", 24, "normal"))

# Update multiplier in the game loop
```

```
if hit_paddle:
    multiplier = min(multiplier + 0.5, 3)
    score += int(1 * multiplier)
else:
    multiplier = 1
```

Deliverable

1. A .txt document
2. A short description of a few lines in a.txt file, about how your code works.
   In this .txt file you specify the members of the Team and the author(s) of the modified code.

3 - Handling Game State and Pausing

Introduction

Managing game state is crucial in game development. In this assignment, you'll implement a pause functionality, allowing players to pause and resume the game.

Task

1. **Game State Variable:**
   - Introduce a `game_state` variable that can be either "playing" or "paused".
2. **Pause and Resume Functions:**
   - Create functions to pause and resume the game. When paused, the game loop should not update the ball or paddle positions.
   - Bind a keyboard key (e.g., "p") to toggle between "playing" and "paused".

Example Code Snippet:

```
game_state = "playing"

def toggle_pause():
    global game_state
    if game_state == "playing":
        game_state = "paused"
    else:
        game_state = "playing"

wn.listen()
wn.onkeypress(toggle_pause, "p")

while True:
    if game_state == "playing":
        # Game logic updates
        wn.update()
    else:
        # Game is paused; do not update game logic
        pass
```

Deliverable

1. A `.py` file with the pause functionality implemented. Include a comment explaining how the pause feature is beneficial in games.
2. In this .py file you specify the members of the team and the author(s) in comments. Place this at the top of your code.

4 - Power-Ups and Advanced Features

Introduction

Adding power-ups can make games more engaging. In this assignment, you'll introduce a new feature: power-ups that appear randomly and affect the game when collected.

Task

1. ## Power-Up Implementation:

   - Create a new turtle object representing a power-up. Randomly place it on the screen every 20 seconds.
   - If the ball hits the power-up, apply a random effect such as increasing the paddle size, slowing down the ball, or adding an extra life.

2. ## Effect Duration:

   - Ensure that power-up effects last only for a certain period (e.g., 10 seconds), after which the game returns to normal.

Example Code Snippet

```
import random

def spawn_power_up():
    power_up = turtle.Turtle()
    power_up.shape("circle")
    power_up.color("blue")
    power_up.penup()
    x = random.randint(-350, 350)
    y = random.randint(-250, 250)
    power_up.goto(x, y)
    return power_up

def apply_power_up(effect):
    global paddle, ball
    if effect == "increase_paddle":
        paddle.shapesize(stretch_wid=8)
        # Reset after 10 seconds
        wn.ontimer(lambda: paddle.shapesize(stretch_wid=6), 10000)
    elif effect == "slow_ball":
        ball.dx *= 0.5
        ball.dy *= 0.5
        wn.ontimer(lambda: reset_ball_speed(), 10000)

# Example effect application in the game loop
```

```
if ball.distance(power_up) < 20:
    apply_power_up(random.choice(["increase_paddle", "slow_ball"]))
```

Deliverable

1. A `.py` file with the power-up feature implemented. Include comments explaining each power-up and its impact on the game.
2. In this .py file you specify the members of the team and the author(s) in comments. Place this at the top of your code.

5 - Collision Detection and Physics

Introduction

Improving collision detection can make a game feel more realistic. In this assignment, we'll refine the collision detection to account for the ball's velocity and angle, providing a more physics-based gameplay experience.

Task

1. ### Advanced Collision Detection:
   - Adjust the ball’s direction based on where it hits the paddle (top, middle, or bottom). If it hits the top, it should bounce off at a sharper angle; if it hits the middle, it should bounce back more vertically.

2. ### Implement Angled Bounces:
   - Modify the game so that when the ball hits the paddle near its edges, it alters its `dx` and `dy` to create an angled bounce.

Example Code Snippet

```
def check_collision():
    global ball, paddle
    if ball.distance(paddle) < 50 and ball.xcor() < -340:
```

```
        # Calculate where the ball hit the paddle
        hit_pos = ball.ycor() - paddle.ycor()
        ball.dy = hit_pos * 0.05
        ball.dx *= -1
```

Deliverable

1. A `.py` file with advanced collision detection and physics implemented. Include a brief explanation of how physics enhances gameplay realism.
2. In this .py file you specify the members of the team and the author(s) in comments. Place this at the top of your code.

--