**Uitgebreid Puntenboekje Systeem - Functies en Uitleg**

**1. Studentgegevens inlezen uit CSV**

**We lezen studenten en hun basisgegevens in vanuit een CSV-bestand, wat de basis vormt voor het puntenboekje.**

**Codevoorbeeld:**

import csv

def lees\_studenten\_csv(bestand):

studenten = []

with open(bestand, newline='', encoding='utf-8') as file:

reader = csv.DictReader(file)

for row in reader:

studenten.append(row)

return studenten

studenten = lees\_studenten\_csv('studenten.csv')

print(studenten)

**2. Studenten toevoegen**

**We voegen een nieuwe student toe aan het systeem. De gebruiker wordt gevraagd om de naam, leeftijd, en klas in te voeren.**

**Codevoorbeeld:**

def voeg\_student\_toe(studenten):

naam = input("Voer de naam van de student in: ")

leeftijd = input("Voer de leeftijd in: ")

klas = input("Voer de klas in: ")

nieuwe\_student = {

"Naam": naam,

"Leeftijd": leeftijd,

"Klas": klas,

"Punten": []

}

studenten.append(nieuwe\_student)

print(f"Student {naam} is toegevoegd.")

voeg\_student\_toe(studenten)

**3. Studenten verwijderen**

**Met deze functie kunnen we een student verwijderen door de naam in te voeren.**

**Codevoorbeeld:**

def verwijder\_student(studenten):

naam = input("Voer de naam van de student in die je wilt verwijderen: ")

for student in studenten:

if student["Naam"].lower() == naam.lower():

studenten.remove(student)

print(f"Student {naam} is verwijderd.")

return

print(f"Student {naam} is niet gevonden.")

verwijder\_student(studenten)

**4. Zoeken naar een student**

**Deze functie zoekt naar een student op basis van naam.**

**Codevoorbeeld:**

def zoek\_student(studenten, naam):

for student in studenten:

if student["Naam"].lower() == naam.lower():

print(f"Gevonden: {student}")

return student

print(f"Student {naam} niet gevonden.")

return None

naam = input("Voer de naam van de student in om te zoeken: ")

zoek\_student(studenten, naam)

**5. Punten invoeren voor een student**

**Met deze functie kunnen we punten voor een bepaalde student invoeren.**

**Codevoorbeeld:**

def voeg\_punten\_toe(studenten):

naam = input("Voer de naam van de student in: ")

student = zoek\_student(studenten, naam)

if student:

punten = float(input(f"Voer de punten in voor {naam}: "))

student["Punten"].append(punten)

print(f"Punten toegevoegd voor {naam}.")

voeg\_punten\_toe(studenten)

**6. Studentgegevens wijzigen**

**We kunnen de gegevens van een bestaande student, zoals leeftijd of klas, wijzigen.**

**Codevoorbeeld:**

def wijzig\_student(studenten):

naam = input("Voer de naam van de student in die je wilt wijzigen: ")

student = zoek\_student(studenten, naam)

if student:

nieuwe\_leeftijd = input(f"Nieuwe leeftijd voor {naam}: ")

nieuwe\_klas = input(f"Nieuwe klas voor {naam}: ")

student["Leeftijd"] = nieuwe\_leeftijd

student["Klas"] = nieuwe\_klas

print(f"Gegevens van {naam} zijn bijgewerkt.")

wijzig\_student(studenten)

**7. Punten van een student weergeven**

**We kunnen alle ingevoerde punten van een student weergeven.**

**Codevoorbeeld:**

def toon\_punten\_student(studenten):

naam = input("Voer de naam van de student in om de punten te zien: ")

student = zoek\_student(studenten, naam)

if student:

print(f"Punten van {naam}: {student['Punten']}")

toon\_punten\_student(studenten)

**8. Studenten sorteren op naam**

**Deze functie sorteert de lijst van studenten alfabetisch op naam.**

**Codevoorbeeld:**

def sorteer\_op\_naam(studenten):

studenten.sort(key=lambda student: student["Naam"].lower())

for student in studenten:

print(student)

sorteer\_op\_naam(studenten)

**9. Studenten sorteren op gemiddelde cijfer**

**Deze functie sorteert de studenten op basis van hun gemiddelde cijfer.**

**Codevoorbeeld:**

def sorteer\_op\_gemiddelde(studenten):

studenten.sort(key=lambda student: sum(student["Punten"])/len(student["Punten"]) if student["Punten"] else 0, reverse=True)

for student in studenten:

print(student)

sorteer\_op\_gemiddelde(studenten)

**10. Gemiddelde cijfer van de klas berekenen**

**Met deze functie berekenen we het gemiddelde cijfer van de hele klas.**

**Codevoorbeeld:**

def bereken\_klas\_gemiddelde(studenten):

totale\_punten = sum(sum(student["Punten"]) for student in studenten if student["Punten"])

aantal\_punten = sum(len(student["Punten"]) for student in studenten if student["Punten"])

klas\_gemiddelde = totale\_punten / aantal\_punten if aantal\_punten else 0

print(f"Gemiddelde cijfer van de klas: {klas\_gemiddelde:.2f}")

bereken\_klas\_gemiddelde(studenten)

**11. Standaardafwijking berekenen voor cijferverdeling**

**Deze functie berekent de standaardafwijking van de cijfers binnen de klas.**

**Codevoorbeeld:**

import math

def bereken\_standaardafwijking(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"] if student["Punten"]]

gemiddelde = sum(alle\_punten) / len(alle\_punten) if alle\_punten else 0

variatie = sum((punt - gemiddelde) \*\* 2 for punt in alle\_punten) / len(alle\_punten) if alle\_punten else 0

standaardafwijking = math.sqrt(variatie)

print(f"Standaardafwijking: {standaardafwijking:.2f}")

bereken\_standaardafwijking(studenten)

**12. Hoogste en laagste cijfer vinden**

**We zoeken het hoogste en laagste cijfer in de klas.**

**Codevoorbeeld:**

def hoogste\_laagste\_cijfer(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"] if student["Punten"]]

hoogste = max(alle\_punten) if alle\_punten else None

laagste = min(alle\_punten) if alle\_punten else None

print(f"Hoogste cijfer: {hoogste}, Laagste cijfer: {laagste}")

hoogste\_laagste\_cijfer(studenten)

**13. Aantal studenten met voldoende halen**

**Deze functie telt het aantal studenten met een voldoende.**

**Codevoorbeeld:**

def aantal\_voldoendes(studenten):

voldoendes = sum(1 for student in studenten if any(punt >= 5.5 for punt in student["Punten"]))

print(f"Aantal studenten met een voldoende: {voldoendes}")

aantal\_voldoendes(studenten)

**14. Cijfers opslaan naar CSV**

**De cijfers en studenten worden opgeslagen in een CSV-bestand.**

**Codevoorbeeld:**

def schrijf\_naar\_csv(studenten, bestand):

with open(bestand, mode='w', newline='', encoding='utf-8') as file:

writer = csv.DictWriter(file, fieldnames=studenten[0].keys())

writer.writeheader()

writer.writerows(studenten)

print(f"Studenten en cijfers succesvol opgeslagen in {bestand}.")

schrijf\_naar\_csv(studenten, 'studenten\_cijfers.csv')

**15. Punten exporteren naar JSON**

**We kunnen de punten van studenten exporteren naar een JSON-bestand.**

**Codevoorbeeld:**

import json

def exporteer\_naar\_json(studenten, bestand):

with open(bestand, 'w') as file:

json.dump(studenten, file, indent=4)

print(f"Studentgegevens en cijfers succesvol opgeslagen in {bestand}.")

exporteer\_naar\_json(studenten, 'studenten\_punten.json')

**16. Punten importeren uit JSON**

**We kunnen punten en gegevens van studenten importeren uit een JSON-bestand.**

**Codevoorbeeld:**

def lees\_studenten\_json(bestand):

with open(bestand, 'r') as file:

studenten = json.load(file)

print(f"Studentgegevens succesvol geladen vanuit {bestand}.")

return studenten

studenten = lees\_studenten\_json('studenten\_punten.json')

**17. Histogram van cijfers tonen**

**We kunnen een histogram tonen van de cijferverdeling in de klas.**

**Codevoorbeeld:**

import matplotlib.pyplot as plt

def toon\_histogram(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"] if student["Punten"]]

plt.hist(alle\_punten, bins=10, edgecolor='black')

plt.title("Cijferverdeling")

plt.xlabel("Cijfers")

plt.ylabel("Frequentie")

plt.show()

toon\_histogram(studenten)

**18. Gemiddelde cijfer per vak berekenen**

**Deze functie berekent het gemiddelde cijfer voor elk vak.**

**Codevoorbeeld:**

def bereken\_gemiddelde\_per\_vak(studenten, vakken):

vak\_gemiddelden = {vak: 0 for vak in vakken}

for vak in vakken:

vak\_punten = [student["Punten"][vak] for student in studenten if vak in student["Punten"]]

vak\_gemiddelden[vak] = sum(vak\_punten) / len(vak\_punten) if vak\_punten else 0

print(vak\_gemiddelden)

vakken = ["Wiskunde", "Nederlands", "Engels"]

bereken\_gemiddelde\_per\_vak(studenten, vakken)

**19. Aantal herkansingen per student**

**Deze functie telt het aantal herkansingen per student.**

**Codevoorbeeld:**

def tel\_herkansingen(studenten):

for student in studenten:

herkansingen = sum(1 for punt in student["Punten"] if punt < 5.5)

print(f"{student['Naam']} heeft {herkansingen} herkansingen.")

tel\_herkansingen(studenten)

**20. Wiskundige berekeningen op basis van cijfergegevens**

**Deze functie voert extra wiskundige berekeningen uit, zoals mediaan en modus.**

**Codevoorbeeld:**

from statistics import median, mode

def extra\_wiskundige\_berekeningen(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"] if student["Punten"]]

mediaan = median(alle\_punten) if alle\_punten else None

modus = mode(alle\_punten) if alle\_punten else None

print(f"Mediaan: {mediaan}, Modus: {modus}")

extra\_wiskundige\_berekeningen(studenten)

import csv

import json

import math

import matplotlib.pyplot as plt

from statistics import median, mode

# Functie om studenten in te lezen uit een CSV-bestand

def lees\_studenten\_csv(bestand):

studenten = []

try:

with open(bestand, newline='', encoding='utf-8') as file:

reader = csv.DictReader(file)

for row in reader:

# Zet 'Punten' om naar een lijst van floats

row['Punten'] = [float(punt) for punt in row.get('Punten', '').split(',') if punt]

studenten.append(row)

print(f"{len(studenten)} studenten geladen uit {bestand}.")

except FileNotFoundError:

print(f"Bestand {bestand} niet gevonden. Beginnen met een lege studentenlijst.")

return studenten

# Functie om een student toe te voegen

def voeg\_student\_toe(studenten):

naam = input("Voer de naam van de student in: ")

leeftijd = input("Voer de leeftijd in: ")

klas = input("Voer de klas in: ")

nieuwe\_student = {

"Naam": naam,

"Leeftijd": leeftijd,

"Klas": klas,

"Punten": []

}

studenten.append(nieuwe\_student)

print(f"Student {naam} is toegevoegd.")

# Functie om een student te verwijderen

def verwijder\_student(studenten):

naam = input("Voer de naam van de student in die je wilt verwijderen: ")

for student in studenten:

if student["Naam"].lower() == naam.lower():

studenten.remove(student)

print(f"Student {naam} is verwijderd.")

return

print(f"Student {naam} is niet gevonden.")

# Functie om een student te zoeken

def zoek\_student(studenten, naam):

for student in studenten:

if student["Naam"].lower() == naam.lower():

return student

print(f"Student {naam} niet gevonden.")

return None

# Functie om punten toe te voegen voor een student

def voeg\_punten\_toe(studenten):

naam = input("Voer de naam van de student in: ")

student = zoek\_student(studenten, naam)

if student:

punten = float(input(f"Voer de punten in voor {naam}: "))

student["Punten"].append(punten)

print(f"Punten toegevoegd voor {naam}.")

# Functie om studentgegevens te wijzigen

def wijzig\_student(studenten):

naam = input("Voer de naam van de student in die je wilt wijzigen: ")

student = zoek\_student(studenten, naam)

if student:

nieuwe\_leeftijd = input(f"Nieuwe leeftijd voor {naam}: ")

nieuwe\_klas = input(f"Nieuwe klas voor {naam}: ")

student["Leeftijd"] = nieuwe\_leeftijd

student["Klas"] = nieuwe\_klas

print(f"Gegevens van {naam} zijn bijgewerkt.")

# Functie om punten van een student weer te geven

def toon\_punten\_student(studenten):

naam = input("Voer de naam van de student in om de punten te zien: ")

student = zoek\_student(studenten, naam)

if student:

print(f"Punten van {naam}: {student['Punten']}")

# Functie om studenten te sorteren op naam

def sorteer\_op\_naam(studenten):

studenten.sort(key=lambda student: student["Naam"].lower())

for student in studenten:

print(student)

# Functie om studenten te sorteren op gemiddelde cijfer

def sorteer\_op\_gemiddelde(studenten):

studenten.sort(

key=lambda student: sum(student["Punten"]) / len(student["Punten"]) if student["Punten"] else 0,

reverse=True

)

for student in studenten:

print(student)

# Functie om het gemiddelde cijfer van de klas te berekenen

def bereken\_klas\_gemiddelde(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"]]

if alle\_punten:

klas\_gemiddelde = sum(alle\_punten) / len(alle\_punten)

print(f"Gemiddelde cijfer van de klas: {klas\_gemiddelde:.2f}")

else:

print("Geen punten beschikbaar om het gemiddelde te berekenen.")

# Functie om de standaardafwijking te berekenen

def bereken\_standaardafwijking(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"]]

if alle\_punten:

gemiddelde = sum(alle\_punten) / len(alle\_punten)

variatie = sum((punt - gemiddelde) \*\* 2 for punt in alle\_punten) / len(alle\_punten)

standaardafwijking = math.sqrt(variatie)

print(f"Standaardafwijking: {standaardafwijking:.2f}")

else:

print("Geen punten beschikbaar om de standaardafwijking te berekenen.")

# Functie om het hoogste en laagste cijfer te vinden

def hoogste\_laagste\_cijfer(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"]]

if alle\_punten:

hoogste = max(alle\_punten)

laagste = min(alle\_punten)

print(f"Hoogste cijfer: {hoogste}, Laagste cijfer: {laagste}")

else:

print("Geen punten beschikbaar om hoogste en laagste cijfers te bepalen.")

# Functie om het aantal studenten met een voldoende te tellen

def aantal\_voldoendes(studenten):

voldoendes = sum(1 for student in studenten if any(punt >= 5.5 for punt in student["Punten"]))

print(f"Aantal studenten met een voldoende: {voldoendes}")

# Functie om studenten en punten op te slaan naar CSV

def schrijf\_naar\_csv(studenten, bestand):

if studenten:

with open(bestand, mode='w', newline='', encoding='utf-8') as file:

fieldnames = ["Naam", "Leeftijd", "Klas", "Punten"]

writer = csv.DictWriter(file, fieldnames=fieldnames)

writer.writeheader()

for student in studenten:

# Converteer de puntenlijst naar een door komma's gescheiden string

student\_copy = student.copy()

student\_copy['Punten'] = ','.join(map(str, student['Punten']))

writer.writerow(student\_copy)

print(f"Studenten en cijfers succesvol opgeslagen in {bestand}.")

else:

print("Geen studenten om op te slaan.")

# Functie om studenten en punten te exporteren naar JSON

def exporteer\_naar\_json(studenten, bestand):

if studenten:

with open(bestand, 'w', encoding='utf-8') as file:

json.dump(studenten, file, indent=4)

print(f"Studentgegevens en cijfers succesvol opgeslagen in {bestand}.")

else:

print("Geen studenten om op te slaan.")

# Functie om studenten en punten te importeren uit JSON

def lees\_studenten\_json(bestand):

try:

with open(bestand, 'r', encoding='utf-8') as file:

studenten = json.load(file)

# Zorg ervoor dat de puntenlijsten floats bevatten

for student in studenten:

student['Punten'] = [float(punt) for punt in student.get('Punten', [])]

print(f"Studentgegevens succesvol geladen vanuit {bestand}.")

return studenten

except FileNotFoundError:

print(f"Bestand {bestand} niet gevonden. Beginnen met een lege studentenlijst.")

return []

# Functie om een histogram van cijfers te tonen

def toon\_histogram(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"]]

if alle\_punten:

plt.hist(alle\_punten, bins=10, edgecolor='black')

plt.title("Cijferverdeling")

plt.xlabel("Cijfers")

plt.ylabel("Frequentie")

plt.show()

else:

print("Geen punten beschikbaar om een histogram te tonen.")

# Functie om het aantal herkansingen per student te tellen

def tel\_herkansingen(studenten):

for student in studenten:

herkansingen = sum(1 for punt in student["Punten"] if punt < 5.5)

print(f"{student['Naam']} heeft {herkansingen} herkansingen.")

# Functie voor extra wiskundige berekeningen

def extra\_wiskundige\_berekeningen(studenten):

alle\_punten = [punt for student in studenten for punt in student["Punten"]]

if alle\_punten:

mediaan = median(alle\_punten)

try:

modus = mode(alle\_punten)

except:

modus = "Geen unieke modus"

print(f"Mediaan: {mediaan}, Modus: {modus}")

else:

print("Geen punten beschikbaar voor extra berekeningen.")

# Hoofdprogramma

def main():

studenten = []

# Probeer studenten te laden vanuit 'studenten.csv' of 'studenten\_punten.json'

keuze = input("Wil je studenten laden uit CSV of JSON? (csv/json): ").lower()

if keuze == 'csv':

studenten = lees\_studenten\_csv('studenten.csv')

elif keuze == 'json':

studenten = lees\_studenten\_json('studenten\_punten.json')

else:

print("Geen geldige keuze. Beginnen met een lege studentenlijst.")

while True:

print("\nPuntenboekje Menu:")

print("1. Voeg student toe")

print("2. Verwijder student")

print("3. Wijzig studentgegevens")

print("4. Toon punten van student")

print("5. Voeg punten toe aan student")

print("6. Sorteer studenten op naam")

print("7. Sorteer studenten op gemiddelde cijfer")

print("8. Bereken gemiddelde cijfer van de klas")

print("9. Bereken standaardafwijking van cijfers")

print("10. Vind hoogste en laagste cijfer")

print("11. Tel aantal studenten met voldoende")

print("12. Schrijf gegevens naar CSV")

print("13. Exporteer gegevens naar JSON")

print("14. Toon histogram van cijfers")

print("15. Tel herkansingen per student")

print("16. Extra wiskundige berekeningen")

print("17. Afsluiten")

keuze = input("Maak een keuze (1-17): ")

if keuze == '1':

voeg\_student\_toe(studenten)

elif keuze == '2':

verwijder\_student(studenten)

elif keuze == '3':

wijzig\_student(studenten)

elif keuze == '4':

toon\_punten\_student(studenten)

elif keuze == '5':

voeg\_punten\_toe(studenten)

elif keuze == '6':

sorteer\_op\_naam(studenten)

elif keuze == '7':

sorteer\_op\_gemiddelde(studenten)

elif keuze == '8':

bereken\_klas\_gemiddelde(studenten)

elif keuze == '9':

bereken\_standaardafwijking(studenten)

elif keuze == '10':

hoogste\_laagste\_cijfer(studenten)

elif keuze == '11':

aantal\_voldoendes(studenten)

elif keuze == '12':

bestand = input("Voer de bestandsnaam in om op te slaan (bijv. studenten\_cijfers.csv): ")

schrijf\_naar\_csv(studenten, bestand)

elif keuze == '13':

bestand = input("Voer de bestandsnaam in om op te slaan (bijv. studenten\_punten.json): ")

exporteer\_naar\_json(studenten, bestand)

elif keuze == '14':

toon\_histogram(studenten)

elif keuze == '15':

tel\_herkansingen(studenten)

elif keuze == '16':

extra\_wiskundige\_berekeningen(studenten)

elif keuze == '17':

print("Programma wordt afgesloten.")

break

else:

print("Ongeldige keuze, probeer opnieuw.")

if \_\_name\_\_ == "\_\_main\_\_":

main()