**Sectie 1: Studentenbeheer en Puntenbeheer**

**Hoofdstuk 1: Studentenbeheer**

**1. Studenten toevoegen:**

def voeg\_student\_toe(studenten):

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

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

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

betaald = input("Heeft de student het collegegeld betaald? (ja/nee): ").lower()

nieuwe\_student = {

"Naam": naam,

"Leeftijd": leeftijd,

"Woonplaats": woonplaats,

"Betaald": betaald == "ja",

"Punten": {}

}

studenten.append(nieuwe\_student)

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

studenten = []

**2. Studenteninformatie bijwerken:**

def update\_student(studenten):

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

for student in studenten:

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

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

student["Leeftijd"] = nieuwe\_leeftijd

nieuwe\_woonplaats = input(f"Nieuwe woonplaats voor {naam}: ")

student["Woonplaats"] = nieuwe\_woonplaats

betaald = input("Heeft de student het collegegeld betaald? (ja/nee): ").lower()

student["Betaald"] = betaald == "ja"

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

return

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

**3. Studenten verwijderen:**

def verwijder\_student(studenten):

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

studenten[:] = [student for student in studenten if student["Naam"].lower() != naam.lower()]

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

**4. Betaalstatus beheren:**

def betaalstatus(studenten):

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

for student in studenten:

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

if student["Betaald"]:

print(f"Student {naam} heeft betaald.")

else:

print(f"Student {naam} heeft nog niet betaald.")

return

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

Hoofdstuk 2: Puntenbeheer

**5. Punten toevoegen per student:**

def voeg\_punten\_toe(studenten):

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

vak = input("Voor welk vak wil je punten toevoegen?: ")

punten = float(input(f"Voer het cijfer in voor {vak}: "))

for student in studenten:

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

if vak not in student["Punten"]:

student["Punten"][vak] = []

student["Punten"][vak].append(punten)

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

return

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

**6. Gemiddelde berekenen per student:**

def bereken\_gemiddeld\_student(student):

total\_punten = []

for vak in student["Punten"]:

total\_punten.extend(student["Punten"][vak])

if total\_punten:

return round(sum(total\_punten) / len(total\_punten), 2)

return 0.0

**7. Gemiddelde per vak berekenen:**

def gemiddeld\_per\_vak(studenten, vak):

punten = []

for student in studenten:

if vak in student["Punten"]:

punten.extend(student["Punten"][vak])

if punten:

return round(sum(punten) / len(punten), 2)

return 0.0

**8. Toppresteerders identificeren:**

def toppresteerders(studenten):

beste\_student = max(studenten, key=lambda student: bereken\_gemiddeld\_student(student))

print(f"Beste student is {beste\_student['Naam']} met een gemiddeld cijfer van {bereken\_gemiddeld\_student(beste\_student)}.")

**Sectie 2: Bestandsbeheer en Geavanceerde Statistieken**

**Hoofdstuk 3: Bestandsbeheer (CSV-import/export)**

**9. Studenten importeren vanuit CSV:**

import csv

def lees\_studenten\_van\_csv(bestand):

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

reader = csv.DictReader(file)

studenten = list(reader)

return studenten

**10. Punten importeren vanuit CSV:**

def lees\_punten\_van\_csv(bestand, studenten):

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

reader = csv.DictReader(file)

for row in reader:

naam = row["Naam"]

vak = row["Vak"]

punten = float(row["Punten"])

for student in studenten:

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

if vak not in student["Punten"]:

student["Punten"][vak] = []

student["Punten"][vak].append(punten)

break

**11. Studenten exporteren naar CSV:**

def schrijf\_studenten\_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"Studentengegevens geëxporteerd naar {bestand}")

**Hoofdstuk 4: Geavanceerde Statistieken**

**12. Gemiddelde cijfers per klas:**

def gemiddeld\_per\_klas(studenten):

if studenten:

totaal = sum([bereken\_gemiddeld\_student(student) for student in studenten])

gemiddeld = totaal / len(studenten)

print(f"Gemiddelde cijfer per klas: {gemiddeld:.2f}")

else:

print("Geen studenten in de lijst.")

**13. Standaardafwijking van cijfers:**

import statistics

def standaardafwijking\_cijfers(studenten):

alle\_punten = []

for student in studenten:

alle\_punten.extend([cijfer for vak in student["Punten"].values() for cijfer in vak])

if len(alle\_punten) > 1:

print(f"Standaardafwijking van de cijfers: {statistics.stdev(alle\_punten):.2f}")

else:

print("Niet genoeg gegevens voor standaardafwijking.")

**14. Voortgangsrapport genereren:**

def genereer\_voortgangsrapport(studenten):

for student in studenten:

print(f"Student: {student['Naam']}")

for vak, cijfers in student["Punten"].items():

print(f" {vak}: {cijfers}")

print(f"Gemiddeld cijfer: {bereken\_gemiddeld\_student(student):.2f}

")

**Sectie 3: Geavanceerde Data Structuren en Manipulatie**

**Hoofdstuk 6: Geavanceerde Data Structuren en Manipulatie**

**1. 2D-lijsten voor cijfers en statistieken:**

# We gebruiken 2D-lijsten om de cijfers van studenten per vak op te slaan

cijfers = [

["Naam", "Wiskunde", "Nederlands", "Engels"],

["Alice", 8.5, 7.2, 6.9],

["Bob", 7.8, 8.0, 7.1],

["Charlie", 6.2, 7.5, 8.3]

]

# Functie om gemiddelde per student te berekenen (per rij):

def gemiddelde\_per\_student(cijfers):

for rij in cijfers[1:]:

naam = rij[0]

gemiddelde = sum(rij[1:]) / (len(rij) - 1)

print(f"{naam} heeft een gemiddeld cijfer van {gemiddelde:.2f}")

gemiddelde\_per\_student(cijfers)

**2. Werken met dictionaries voor flexibele datamanipulatie:**

# Dictionary voor het bijhouden van punten per student per vak

punten = {

"Alice": {"Wiskunde": [8.5], "Nederlands": [7.2], "Engels": [6.9]},

"Bob": {"Wiskunde": [7.8], "Nederlands": [8.0], "Engels": [7.1]},

}

# Functie om punten per student toe te voegen:

def voeg\_punten\_toe\_dict(punten, naam, vak, cijfer):

if naam in punten:

if vak not in punten[naam]:

punten[naam][vak] = []

punten[naam][vak].append(cijfer)

else:

punten[naam] = {vak: [cijfer]}

print(f"Punten toegevoegd voor {naam} in vak {vak}")

voeg\_punten\_toe\_dict(punten, "Alice", "Wiskunde", 9.0)

**3. Werken met sets voor unieke data:**

# Sets worden gebruikt om unieke steden van studenten bij te houden

studenten = [{"Naam": "Alice", "Woonplaats": "Amsterdam"},

{"Naam": "Bob", "Woonplaats": "Rotterdam"},

{"Naam": "Charlie", "Woonplaats": "Amsterdam"}]

unieke\_steden = {student["Woonplaats"] for student in studenten}

print(f"Unieke steden: {unieke\_steden}")

**4. Werken met tuples voor immutable data:**

# Tuples worden gebruikt voor gegevens die niet moeten worden gewijzigd, zoals vakken

vakken = ("Wiskunde", "Nederlands", "Engels")

print(f"De vakken zijn: {vakken}")

**5. Gemiddelden per rij berekenen:**

# Functie om gemiddelden per student (rij) te berekenen uit een 2D-lijst:

def gemiddelde\_per\_rij(cijfers):

for rij in cijfers[1:]:

gemiddelde = sum(rij[1:]) / (len(rij) - 1)

print(f"Gemiddeld cijfer voor {rij[0]}: {gemiddelde:.2f}")

gemiddelde\_per\_rij(cijfers)

**6. Werken met random sets:**

import random

# Willekeurige selectie van studenten voor een quiz

studentenlijst = ["Alice", "Bob", "Charlie"]

willekeurige\_student = random.choice(studentenlijst)

print(f"Willekeurige student geselecteerd voor de quiz: {willekeurige\_student}")

7. Data manipuleren met tabulate:

from tabulate import tabulate

# Data van studenten weergeven in een tabel

studenten\_data = [["Alice", 8.5, 7.2, 6.9], ["Bob", 7.8, 8.0, 7.1], ["Charlie", 6.2, 7.5, 8.3]]

headers = ["Naam", "Wiskunde", "Nederlands", "Engels"]

print(tabulate(studenten\_data, headers, tablefmt="grid"))

import csv

import statistics

from tabulate import tabulate

import random

# Sectie 1: Studentenbeheer en Puntenbeheer

# Lijst om studenten op te slaan

studenten = []

# Hoofdstuk 1: Studentenbeheer

def voeg\_student\_toe(studenten):

"""

Voegt een nieuwe student toe aan de lijst met studenten.

"""

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

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

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

betaald = input("Heeft de student het collegegeld betaald? (ja/nee): ").lower()

nieuwe\_student = {

"Naam": naam,

"Leeftijd": leeftijd,

"Woonplaats": woonplaats,

"Betaald": betaald == "ja",

"Punten": {}

}

studenten.append(nieuwe\_student)

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

def update\_student(studenten):

"""

Werkt de informatie van een bestaande student bij.

"""

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

for student in studenten:

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

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

student["Leeftijd"] = nieuwe\_leeftijd

nieuwe\_woonplaats = input(f"Nieuwe woonplaats voor {naam}: ")

student["Woonplaats"] = nieuwe\_woonplaats

betaald = input("Heeft de student het collegegeld betaald? (ja/nee): ").lower()

student["Betaald"] = betaald == "ja"

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

return

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

def verwijder\_student(studenten):

"""

Verwijdert een student uit de lijst.

"""

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} niet gevonden.")

def betaalstatus(studenten):

"""

Controleert of een student het collegegeld heeft betaald.

"""

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

for student in studenten:

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

if student["Betaald"]:

print(f"Student {naam} heeft betaald.")

else:

print(f"Student {naam} heeft nog niet betaald.")

return

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

# Hoofdstuk 2: Puntenbeheer

def voeg\_punten\_toe(studenten):

"""

Voegt punten toe voor een specifiek vak aan een student.

"""

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

vak = input("Voor welk vak wil je punten toevoegen?: ")

punten = float(input(f"Voer het cijfer in voor {vak}: "))

for student in studenten:

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

if vak not in student["Punten"]:

student["Punten"][vak] = []

student["Punten"][vak].append(punten)

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

return

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

def bereken\_gemiddeld\_student(student):

"""

Berekent het gemiddelde cijfer van een student over alle vakken.

"""

total\_punten = []

for vak in student["Punten"]:

total\_punten.extend(student["Punten"][vak])

if total\_punten:

return round(sum(total\_punten) / len(total\_punten), 2)

return 0.0

def gemiddeld\_per\_vak(studenten, vak):

"""

Berekent het gemiddelde cijfer voor een vak over alle studenten.

"""

punten = []

for student in studenten:

if vak in student["Punten"]:

punten.extend(student["Punten"][vak])

if punten:

return round(sum(punten) / len(punten), 2)

return 0.0

def toppresteerders(studenten):

"""

Identificeert de student met het hoogste gemiddelde cijfer.

"""

if studenten:

beste\_student = max(studenten, key=lambda student: bereken\_gemiddeld\_student(student))

hoogste\_gemiddelde = bereken\_gemiddeld\_student(beste\_student)

print(f"Beste student is {beste\_student['Naam']} met een gemiddeld cijfer van {hoogste\_gemiddelde}.")

else:

print("Er zijn geen studenten om te evalueren.")

# Sectie 2: Bestandsbeheer en Geavanceerde Statistieken

# Hoofdstuk 3: Bestandsbeheer (CSV-import/export)

def lees\_studenten\_van\_csv(bestand):

"""

Leest studenteninformatie in vanaf een CSV-bestand.

"""

try:

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

reader = csv.DictReader(file)

studenten = []

for row in reader:

student = {

"Naam": row["Naam"],

"Leeftijd": int(row["Leeftijd"]),

"Woonplaats": row["Woonplaats"],

"Betaald": row["Betaald"].lower() == "ja",

"Punten": {}

}

studenten.append(student)

print(f"{len(studenten)} studenten geïmporteerd vanuit {bestand}.")

return studenten

except FileNotFoundError:

print(f"Bestand {bestand} niet gevonden.")

return []

def lees\_punten\_van\_csv(bestand, studenten):

"""

Leest punten in vanaf een CSV-bestand en voegt ze toe aan de juiste studenten.

"""

try:

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

reader = csv.DictReader(file)

for row in reader:

naam = row["Naam"]

vak = row["Vak"]

punten = float(row["Punten"])

for student in studenten:

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

if vak not in student["Punten"]:

student["Punten"][vak] = []

student["Punten"][vak].append(punten)

break

print(f"Punten geïmporteerd vanuit {bestand}.")

except FileNotFoundError:

print(f"Bestand {bestand} niet gevonden.")

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

"""

Schrijft de studenteninformatie naar een CSV-bestand.

"""

if studenten:

fieldnames = ["Naam", "Leeftijd", "Woonplaats", "Betaald"]

try:

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

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

writer.writeheader()

for student in studenten:

writer.writerow({

"Naam": student["Naam"],

"Leeftijd": student["Leeftijd"],

"Woonplaats": student["Woonplaats"],

"Betaald": "ja" if student["Betaald"] else "nee"

})

print(f"Studentengegevens geëxporteerd naar {bestand}")

except Exception as e:

print(f"Fout bij het schrijven naar bestand: {e}")

else:

print("Geen studenten om te exporteren.")

# Hoofdstuk 4: Geavanceerde Statistieken

def gemiddeld\_per\_klas(studenten):

"""

Berekent het gemiddelde cijfer van alle studenten in de klas.

"""

if studenten:

total = sum([bereken\_gemiddeld\_student(student) for student in studenten])

gemiddeld = total / len(studenten)

print(f"Gemiddeld cijfer per klas: {gemiddeld:.2f}")

else:

print("Geen studenten in de lijst.")

def standaardafwijking\_cijfers(studenten):

"""

Berekent de standaardafwijking van alle cijfers in de klas.

"""

alle\_punten = []

for student in studenten:

for punten in student["Punten"].values():

alle\_punten.extend(punten)

if len(alle\_punten) > 1:

std\_dev = statistics.stdev(alle\_punten)

print(f"Standaardafwijking van de cijfers: {std\_dev:.2f}")

else:

print("Niet genoeg gegevens voor standaardafwijking.")

def genereer\_voortgangsrapport(studenten):

"""

Genereert een voortgangsrapport voor alle studenten.

"""

for student in studenten:

print(f"Student: {student['Naam']}")

for vak, cijfers in student["Punten"].items():

print(f" {vak}: {cijfers}")

gemiddeld = bereken\_gemiddeld\_student(student)

print(f"Gemiddeld cijfer: {gemiddeld:.2f}\n")

# Sectie 3: Geavanceerde Data Structuren en Manipulatie

# Hoofdstuk 6: Geavanceerde Data Structuren en Manipulatie

def gemiddelde\_per\_student(cijfers):

"""

Berekent het gemiddelde cijfer per student uit een 2D-lijst.

"""

for rij in cijfers[1:]:

naam = rij[0]

gemiddelde = sum(rij[1:]) / (len(rij) - 1)

print(f"{naam} heeft een gemiddeld cijfer van {gemiddelde:.2f}")

def voeg\_punten\_toe\_dict(punten, naam, vak, cijfer):

"""

Voegt punten toe aan een student in een punten-dictionary.

"""

if naam in punten:

if vak not in punten[naam]:

punten[naam][vak] = []

punten[naam][vak].append(cijfer)

else:

punten[naam] = {vak: [cijfer]}

print(f"Punten toegevoegd voor {naam} in vak {vak}")

def main():

"""

Hoofdprogramma dat het menu toont en gebruikersinteracties beheert.

"""

global studenten

while True:

print("\nStudentenbeheer en Puntenbeheer")

print("1. Voeg student toe")

print("2. Werk studentinformatie bij")

print("3. Verwijder student")

print("4. Controleer betaalstatus")

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

print("6. Toon gemiddelde per student")

print("7. Toon gemiddelde per vak")

print("8. Toon toppresteerders")

print("9. Importeer studenten vanuit CSV")

print("10. Importeer punten vanuit CSV")

print("11. Exporteer studenten naar CSV")

print("12. Toon gemiddelde cijfer per klas")

print("13. Toon standaardafwijking van cijfers")

print("14. Genereer voortgangsrapport")

print("15. Afsluiten")

keuze = input("Maak een keuze (1-15): ")

if keuze == '1':

voeg\_student\_toe(studenten)

elif keuze == '2':

update\_student(studenten)

elif keuze == '3':

verwijder\_student(studenten)

elif keuze == '4':

betaalstatus(studenten)

elif keuze == '5':

voeg\_punten\_toe(studenten)

elif keuze == '6':

for student in studenten:

gemiddeld = bereken\_gemiddeld\_student(student)

print(f"{student['Naam']} heeft een gemiddeld cijfer van {gemiddeld:.2f}")

elif keuze == '7':

vak = input("Voor welk vak wil je het gemiddelde zien?: ")

gemiddeld = gemiddeld\_per\_vak(studenten, vak)

if gemiddeld > 0:

print(f"Gemiddelde voor {vak}: {gemiddeld}")

else:

print(f"Geen cijfers gevonden voor {vak}.")

elif keuze == '8':

toppresteerders(studenten)

elif keuze == '9':

bestand = input("Voer de bestandsnaam in van het CSV-bestand met studenten: ")

studenten = lees\_studenten\_van\_csv(bestand)

elif keuze == '10':

bestand = input("Voer de bestandsnaam in van het CSV-bestand met punten: ")

lees\_punten\_van\_csv(bestand, studenten)

elif keuze == '11':

bestand = input("Voer de bestandsnaam in om studenten naar te exporteren: ")

schrijf\_studenten\_naar\_csv(studenten, bestand)

elif keuze == '12':

gemiddeld\_per\_klas(studenten)

elif keuze == '13':

standaardafwijking\_cijfers(studenten)

elif keuze == '14':

genereer\_voortgangsrapport(studenten)

elif keuze == '15':

print("Programma wordt afgesloten.")

break

else:

print("Ongeldige keuze, probeer opnieuw.")

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

main()