***Doktersadministratie Systeem - Functies en Uitleg***

**1. Inlezen van doktersgegevens uit CSV**

**We lezen doktersgegevens uit een CSV-bestand in en slaan deze op in een Python-lijst van dictionaries.**

**Hierdoor kunnen we de gegevens eenvoudig beheren en manipuleren.**

**Codevoorbeeld:**

import csv

def lees\_csv\_bestand(bestand):

dokters = []

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

reader = csv.DictReader(file)

for row in reader:

dokters.append(row)

return dokters

dokters = lees\_csv\_bestand('dokters.csv')

print(dokters) # Controleer of de gegevens goed zijn ingelezen

**2. Doktersgegevens weergeven in tabel**

**Met deze functie kunnen we de gegevens van de dokters overzichtelijk weergeven in een tabel.**

**We maken gebruik van de 'tabulate'-module om dit op een nette manier te doen.**

**Codevoorbeeld:**

from tabulate import tabulate

def toon\_dokters(dokters):

print(tabulate(dokters, headers="keys", tablefmt="grid"))

toon\_dokters(dokters)

**3. Dokters toevoegen**

**Hier voegen we een nieuwe dokter toe door de gebruiker te vragen om de naam, specialisatie, ziekenhuis en ereloon.**

**De gegevens worden vervolgens opgeslagen in de dokterslijst.**

**Codevoorbeeld:**

def voeg\_dokter\_toe(dokters):

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

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

ziekenhuis = input("Voer het ziekenhuis in: ")

ereloon = input("Voer het ereloon in: ")

nieuwe\_dokter = {

"Naam": naam,

"Specialisatie": specialisatie,

"Ziekenhuis": ziekenhuis,

"Ereloon": ereloon

}

dokters.append(nieuwe\_dokter)

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

voeg\_dokter\_toe(dokters)

toon\_dokters(dokters)

**4. Dokters verwijderen**

**Met deze functie kunnen we een dokter verwijderen door te zoeken op naam.**

**De gebruiker wordt gevraagd om de naam van de dokter die verwijderd moet worden.**

**Codevoorbeeld:**

def verwijder\_dokter(dokters):

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

for dokter in dokters:

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

dokters.remove(dokter)

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

return

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

verwijder\_dokter(dokters)

toon\_dokters(dokters)

**5. Zoeken naar een dokter**

**Hier kunnen we dokters zoeken op basis van hun naam of specialisatie.**

**De gebruiker kan een zoekterm invoeren en de resultaten worden gefilterd en weergegeven.**

**Codevoorbeeld:**

def zoek\_dokter(dokters, zoekterm, op\_specialisatie=False):

resultaten = []

if op\_specialisatie:

resultaten = [dokter for dokter in dokters if zoekterm.lower() in dokter["Specialisatie"].lower()]

else:

resultaten = [dokter for dokter in dokters if zoekterm.lower() in dokter["Naam"].lower()]

if resultaten:

toon\_dokters(resultaten)

else:

print(f"Geen resultaten gevonden voor '{zoekterm}'.")

zoekterm = input("Voer de naam of specialisatie in om te zoeken: ")

zoek\_dokter(dokters, zoekterm)

**6. Dokters wijzigen**

**Hier kunnen we de gegevens van een bestaande dokter aanpassen, zoals het ziekenhuis of het ereloon.**

**De gebruiker wordt gevraagd om de wijzigingen door te voeren.**

**Codevoorbeeld:**

def wijzig\_dokter(dokters):

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

for dokter in dokters:

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

nieuw\_ziekenhuis = input(f"Nieuw ziekenhuis voor {naam}: ")

nieuw\_ereloon = input(f"Nieuw ereloon voor {naam}: ")

dokter["Ziekenhuis"] = nieuw\_ziekenhuis

dokter["Ereloon"] = nieuw\_ereloon

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

return

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

wijzig\_dokter(dokters)

toon\_dokters(dokters)

**7. Dokters sorteren op naam**

**Met deze functie sorteren we de lijst met dokters alfabetisch op hun naam. Hierdoor kunnen we eenvoudig dokters vinden.**

**Codevoorbeeld:**

def sorteer\_op\_naam(dokters):

dokters.sort(key=lambda dokter: dokter["Naam"].lower())

toon\_dokters(dokters)

sorteer\_op\_naam(dokters)

**8. Dokters sorteren op ereloon**

**Deze functie sorteert de dokters op basis van hun ereloon, van hoog naar laag. Het maakt het gemakkelijk om te zien welke dokters de hoogste tarieven hebben.**

**Codevoorbeeld:**

def sorteer\_op\_ereloon(dokters):

dokters.sort(key=lambda dokter: float(dokter["Ereloon"]), reverse=True)

toon\_dokters(dokters)

sorteer\_op\_ereloon(dokters)

**9. Filteren op ziekenhuis**

**Met deze functie filteren we de dokters op basis van hun ziekenhuis, zodat we snel alle dokters in een bepaald ziekenhuis kunnen zien.**

**Codevoorbeeld:**

def filter\_op\_ziekenhuis(dokters, ziekenhuis):

gefilterde\_dokters = [dokter for dokter in dokters if dokter["Ziekenhuis"].lower() == ziekenhuis.lower()]

if gefilterde\_dokters:

toon\_dokters(gefilterde\_dokters)

else:

print(f"Geen dokters gevonden in {ziekenhuis}.")

filter\_op\_ziekenhuis(dokters, "Ziekenhuis Sint-Jan")

**10. Dokters opslaan naar CSV**

**Deze functie slaat de bijgewerkte lijst van dokters op naar een CSV-bestand, zodat we de gegevens later weer kunnen inladen.**

**Codevoorbeeld:**

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

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

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

writer.writeheader()

writer.writerows(dokters)

print(f"Gegevens zijn succesvol opgeslagen in {bestand}.")

schrijf\_naar\_csv(dokters, 'dokters\_bijgewerkt.csv')

**11. Gemiddelde ereloon berekenen**

**Hier berekenen we het gemiddelde ereloon van alle dokters, zodat we een inzicht krijgen in de gemiddelde tarieven.**

**Codevoorbeeld:**

def bereken\_gemiddeld\_ereloon(dokters):

totaal\_ereloon = sum(float(dokter["Ereloon"]) for dokter in dokters)

gemiddeld\_ereloon = totaal\_ereloon / len(dokters)

print(f"Gemiddeld ereloon van alle dokters: {gemiddeld\_ereloon:.2f}")

bereken\_gemiddeld\_ereloon(dokters)

**12. Meerdere dokters tegelijk toevoegen**

**Met deze functie kunnen we meerdere dokters in één keer toevoegen door hun gegevens direct in te voeren of vanuit een batchbestand.**

**Codevoorbeeld:**

def voeg\_meerdere\_dokters\_toe(dokters):

aantal = int(input("Hoeveel dokters wil je toevoegen? "))

for \_ in range(aantal):

voeg\_dokter\_toe(dokters)

voeg\_meerdere\_dokters\_toe(dokters)

**13. Doktersinformatie exporteren naar JSON**

**We kunnen de doktersgegevens exporteren naar een JSON-bestand, een veelgebruikt formaat voor gegevensopslag.**

**Codevoorbeeld:**

import json

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

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

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

print(f"Gegevens zijn succesvol opgeslagen in {bestand}.")

exporteer\_naar\_json(dokters, 'doktersgegevens.json')

**14. GUI voor doktersinvoer**

**We maken een eenvoudige GUI waarmee we doktersgegevens kunnen invoeren via een visuele interface.**

**Codevoorbeeld:**

import tkinter as tk

from tkinter import simpledialog, messagebox

def voeg\_dokter\_toe\_gui(dokters):

root = tk.Tk()

root.withdraw() # Verberg het hoofdvenster

naam = simpledialog.askstring("Invoer", "Voer de naam van de dokter in:")

specialisatie = simpledialog.askstring("Invoer", "Voer de specialisatie in:")

ziekenhuis = simpledialog.askstring("Invoer", "Voer het ziekenhuis in:")

ereloon = simpledialog.askstring("Invoer", "Voer het ereloon in:")

if naam and specialisatie and ziekenhuis and ereloon:

nieuwe\_dokter = {"Naam": naam, "Specialisatie": specialisatie, "Ziekenhuis": ziekenhuis, "Ereloon": ereloon}

dokters.append(nieuwe\_dokter)

messagebox.showinfo("Succes", f"{naam} is toegevoegd!")

else:

messagebox.showwarning("Waarschuwing", "Alle velden moeten worden ingevuld.")

voeg\_dokter\_toe\_gui(dokters)