Project: Invoice Generator and SQL Database for an Osteopathy practice in Berlin, Germany

Project Overview

For this project, my primary objective was to create a secure and efficient Invoice Generator, tailored to the unique needs of my client. Given the sensitive nature of patient data, I ensured that all data showcased here is dummy data only, in strict compliance with privacy and security regulations. I used Python, tkinter, SQLite3 and HTML code.

Figure 1: The UI of the invoice generator

User-Friendly Interface

The heart of the Invoice Generator is its user-friendly interface. It empowers the user to effortlessly navigate and manage patient information. At the core of this interface is a dropdown menu where the user can select patients' names. Upon selection, the application seamlessly loads and displays the respective patient's data. This dynamic approach streamlines the process and ensures that the most up-to-date patient information is always readily accessible.

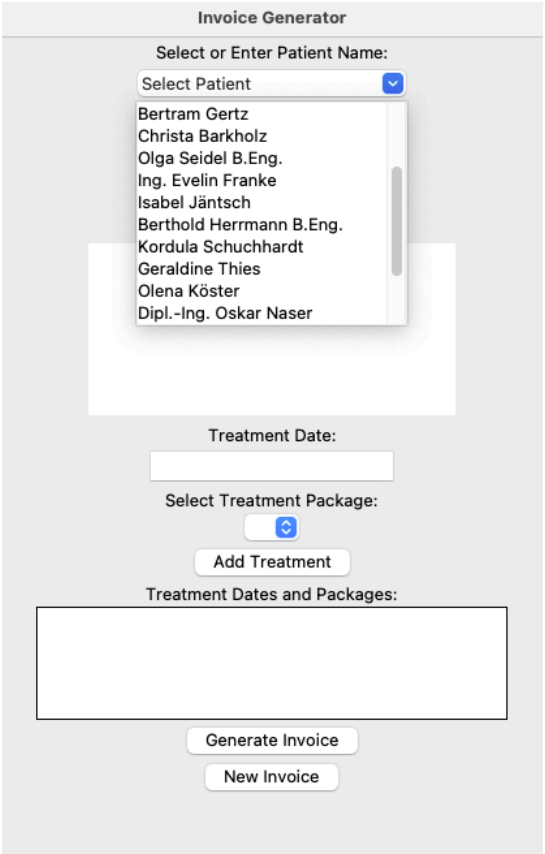

Figure 2: The user can select patients from the database

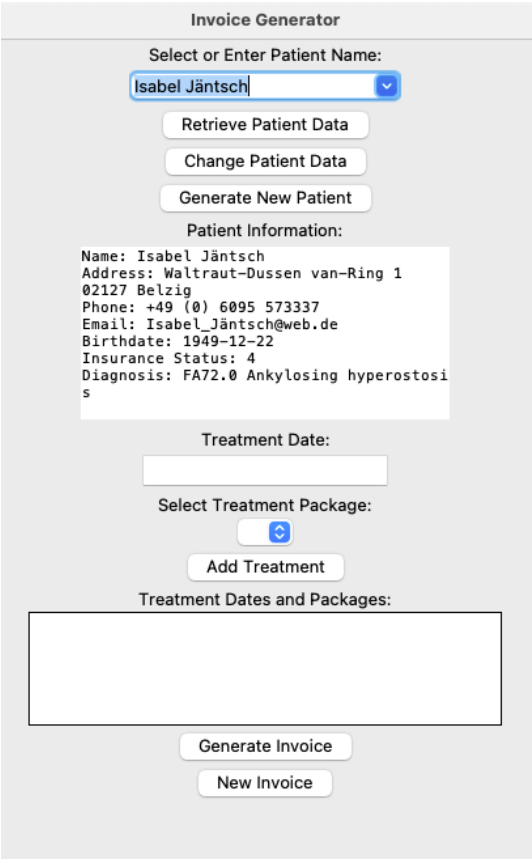

Figure 3: The invoice generator automatically retrieves the patient data from the database

Effortless Data Management

To further enhance user convenience, I incorporated a 'Change Patient Data' button. This feature enables the user to effortlessly update patient information in the database, guaranteeing data accuracy and integrity.

Invoice Generation

One of the central features of this project is the generation of invoices. This process is designed to be as smooth as possible for the user. To create an invoice, the user can select a date for the treatment and choose a treatment package. Multiple treatment dates can be added, and the system dynamically calculates the invoice's length.

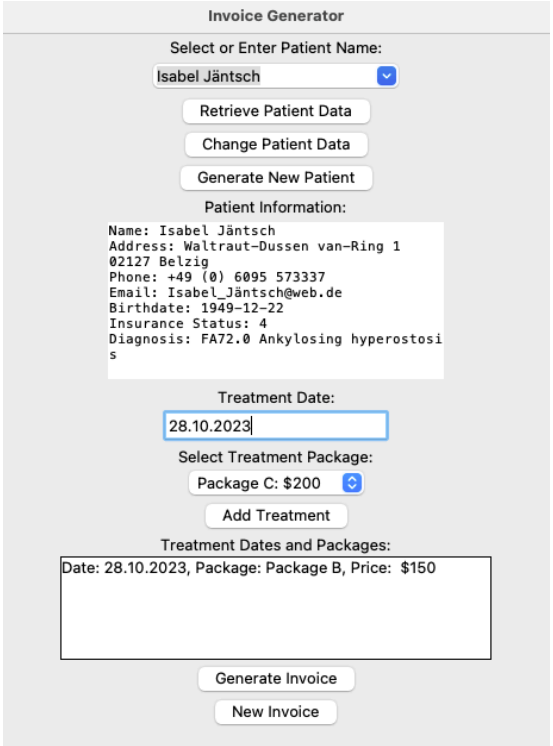

Figure 4: The user can input details about the treatments

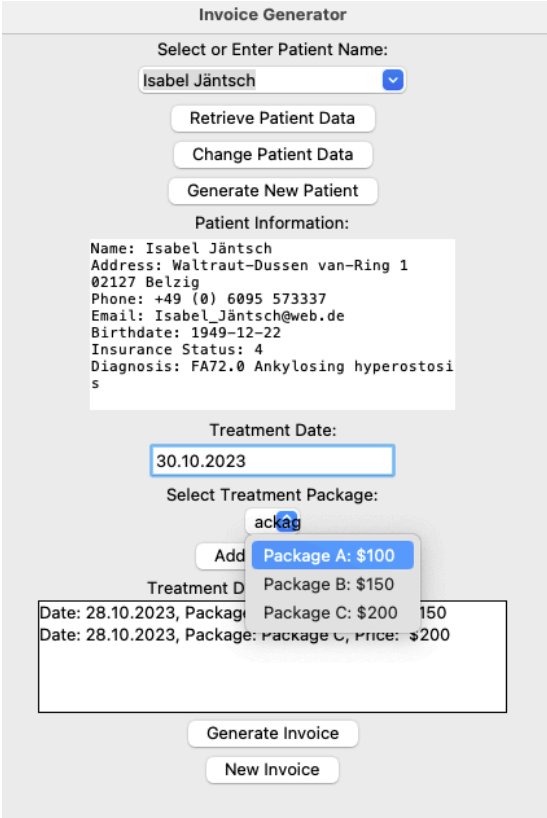

Figure 5: The user can input dates and packages

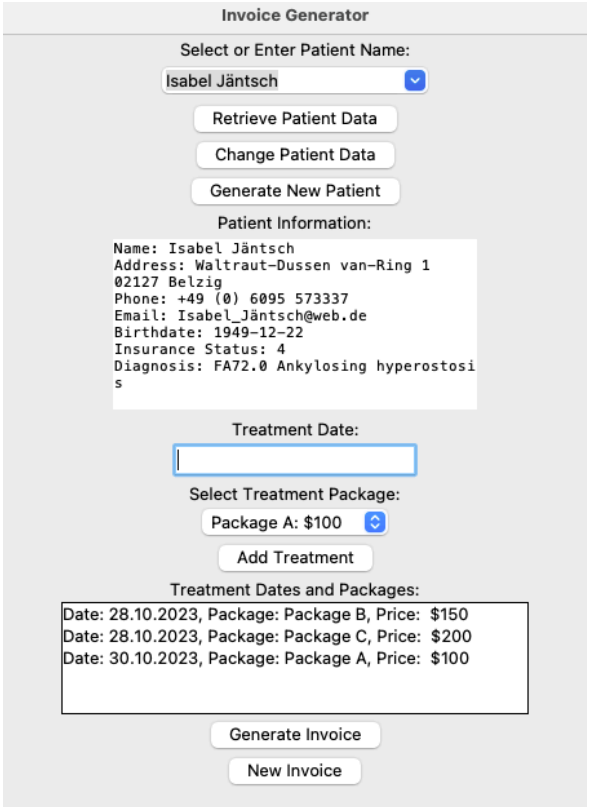

Figure 6: The details are added up to create an invoice

Dynamic HTML to PDF Conversion

Upon user request, the program dynamically generates an HTML file, designed to adapt to the number of treatment dates, ensuring a neat and professional layout. From this HTML file, a PDF is automatically generated, serving as the final invoice. This streamlined approach simplifies the invoicing process, making it efficient and user-friendly.

Seamless Database Integration

To further enhance patient history and data management, this Invoice Generator seamlessly integrates with an SQL database. Newly generated treatment entries are automatically added to the patient's history, ensuring a comprehensive and up-to-date record.

This Invoice Generator project represents a successful fusion of user convenience, data security, and professional output.

Code Snippets

In the following section, I present selected code snippets that illustrate the technical aspects of this project. These snippets showcase the implementation details and the behind-the-scenes functionality of the Invoice Generator.

ostheopathy_4.py × create_html.py

```python
import tkinter as tk
from tkinter import ttk
import sqlite3
from tkinter import messagebox
from tkinter import simpledialog

app = tk.Tk()
app.title("Invoice Generator")

# Function to populate the Combobox with patient names from the database
def populate_patient_names():
    conn = sqlite3.connect("patienten.db")
    cursor = conn.cursor()
    cursor.execute("SELECT name FROM patients")
    patients = cursor.fetchall()
    conn.close()
    return [patient[0] for patient in patients]

# Function to change patient data
def change_patient_data():
    selected_name = selected_patient.get()

    # Check if a patient is selected
    if selected_name != "Select Patient" and selected_name != "Enter New Name":
        conn = sqlite3.connect("patienten.db")
        cursor = conn.cursor()

        # Retrieve patient data based on the selected name
        cursor.execute( __sql: "SELECT * FROM patients WHERE name=?",  __parameters: (selected_name,))
        patient_data = cursor.fetchone()

        conn.close()

        if patient_data:
            # Create a new window for changing patient data
            change_patient_window = tk.Toplevel(app)
            change_patient_window.title("Change Patient Data")

            # Labels and Entry widgets for patient details
            name_label = tk.Label(change_patient_window, text="Name:")
            name_label.pack()
            name_entry = tk.Entry(change_patient_window)
```

populate_patient_names()

Caption

ostheopathy_4.py × create_html.py

```python
            name_label = tk.Label(change_patient_window, text="Name:")
            name_label.pack()
            name_entry = tk.Entry(change_patient_window)
            name_entry.insert( index: 0, patient_data[1])  # Set the current name
            name_entry.pack()

            address_label = tk.Label(change_patient_window, text="Address:")
            address_label.pack()
            address_entry = tk.Entry(change_patient_window)
            address_entry.insert( index: 0, patient_data[2])  # Set the current address
            address_entry.pack()

            phone_label = tk.Label(change_patient_window, text="Phone:")
            phone_label.pack()
            phone_entry = tk.Entry(change_patient_window)
            phone_entry.insert( index: 0, patient_data[3])  # Set the current address
            phone_entry.pack()

            email_label = tk.Label(change_patient_window, text="Email:")
            email_label.pack()
            email_entry = tk.Entry(change_patient_window)
            email_entry.insert( index: 0, patient_data[4])  # Set the current address
            email_entry.pack()

            birthday_label = tk.Label(change_patient_window, text="Birthday:")
            birthday_label.pack()
            birthday_entry = tk.Entry(change_patient_window)
            birthday_entry.insert( index: 0, patient_data[5])  # Set the current address
            birthday_entry.pack()

            insurance_label = tk.Label(change_patient_window, text="Insurance:")
            insurance_label.pack()
            insurance_entry = tk.Entry(change_patient_window)
            insurance_entry.insert( index: 0, patient_data[6])  # Set the current address
            insurance_entry.pack()

            diagnosis_label = tk.Label(change_patient_window, text="Diagnosis:")
            diagnosis_label.pack()
            diagnosis_entry = tk.Entry(change_patient_window)
            diagnosis_entry.insert( index: 0, patient_data[7])  # Set the current address
            diagnosis_entry.pack()
```

populate_patient_names()

ostheopathy_4.py × create_html.py

```python
            # Add more labels and Entry widgets for other patient details (phone, email, etc.)

            # Function to save the changes
            def save_changes():
                new_name = name_entry.get()
                new_address = address_entry.get()
                new_email = email_entry.get()
                new_phone = phone_entry.get()
                new_birthday = birthday_entry.get()
                new_insurance = insurance_entry.get()
                new_diagnosis = diagnosis_entry.get()
                # Get other updated details similarly...

                conn = sqlite3.connect("patienten.db")
                cursor = conn.cursor()

                # Update the patient's data in the database
                cursor.execute( __sql: """UPDATE patients SET name=?, address=?, phone=?, email=?, birthdate=?,
                insurance_status =?, diagnosis=? WHERE name=?""",  __parameters: (new_name, new_address, new_phone,
                                                                    new_email, new_birthday, new_insurance,
                                                                    new_diagnosis, selected_name))
                conn.commit()
                conn.close()

                # Close the change patient window
                change_patient_window.destroy()

                # Show a success message
                messagebox.showinfo( title: "Success",  message: "Patient data updated")

            # Button to save the changes
            save_button = tk.Button(change_patient_window, text="Save Changes", command=save_changes)
            save_button.pack()
        else:
            # Display a message if the selected patient is not found
            messagebox.showerror( title: "Error",  message: "Patient not found in the database.")
    else:
        # Display a message if no patient is selected or a new name is entered
        messagebox.showerror( title: "Error",  message: "Please select a patient from the dropdown or enter a new name.")
```

populate_patient_names()

ostheopathy_4.py × create_html.py

```python
# Function to reset all fields to their initial state
def reset_app():
    selected_patient.set("Select Patient")
    patient_combobox.set("Select Patient")
    #name_entry.delete(0, tk.END)
    patient_info_text.delete( index1: 1.0, tk.END)  # Clear patient information text
    treatment_date_entry.delete( first: 0, tk.END)
    selected_package.set("")


    # Clear treatment display by re-creating the Listbox
    treatment_display_listbox.destroy()
    treatment_display_listbox = tk.Listbox(app, height=5, width=40)
    treatment_display_listbox.pack()


# Function to create a new invoice and reset the app to its original state
def new_invoice():
    reset_app()


def add_new_patient():
    # Create a separate window for adding a new patient
    new_patient_window = tk.Toplevel(app)
    new_patient_window.title("Add New Patient")

    # Labels and Entry widgets for patient details
    name_label = tk.Label(new_patient_window, text="Name:")
    name_label.pack()
    name_entry = tk.Entry(new_patient_window)
    name_entry.pack()

    street_label = tk.Label(new_patient_window, text="Street/nr:")
    street_label.pack()
    street_entry = tk.Entry(new_patient_window)
    street_entry.pack()

    postcode_label = tk.Label(new_patient_window, text="Postcode/City:")
    postcode_label.pack()
    postcode_entry = tk.Entry(new_patient_window)
    postcode_entry.pack()
```

populate_patient_names()

ostheopathy_4.py × create_html.py

```python
    birthday_label = tk.Label(new_patient_window, text="Birthday(YYYY-MM-DD):")
    birthday_label.pack()
    birthday_entry = tk.Entry(new_patient_window)
    birthday_entry.pack()

    insurance_label = tk.Label(new_patient_window, text="Insurance Status:")
    insurance_label.pack()
    insurance_entry = tk.Entry(new_patient_window)
    insurance_entry.pack()

    diagnosis_label = tk.Label(new_patient_window, text="Diagnosis:")
    diagnosis_label.pack()
    diagnosis_entry = tk.Entry(new_patient_window)
    diagnosis_entry.pack()

    # Function to save the new patient to the database
    def save_patient():
        new_name = name_entry.get()
        new_address = street_entry.get() + "\n" + postcode_entry.get()
        new_email = email_entry.get()
        new_phone = phone_entry.get()
        new_birthday = birthday_entry.get()
        new_insurance = insurance_entry.get()
        new_diagnosis = diagnosis_entry.get()
        print(new_diagnosis)

        # Check if any of the fields are empty
        if not all([new_name, new_address, new_email, new_phone, new_birthday, new_diagnosis, new_insurance]):
            messagebox.showerror( title: "Error",  message: "All fields must be filled")
            return
        print("hallo")

        conn = sqlite3.connect("patienten.db")
        cursor = conn.cursor()

        # Insert the new patient into the database
        cursor.execute( __sql: """INSERT INTO patients (name, address, phone,
                        email, birthdate,insurance_status, diagnosis) VALUES (?, ?, ?, ?, ?, ?, ?)""",
                         __parameters: (new_name, new_address, new_phone, new_email, new_birthday, new_insurance, new_diagnosis))
        conn.commit()
        conn.close()
        """ CREATE TABLE IF NOT EXISTS patients (
```

populate_patient_names()

ostheopathy_4.py × create_html.py

```python
        conn.close()
        """ CREATE TABLE IF NOT EXISTS patients (
        patient_id INTEGER PRIMARY KEY,
        name TEXT,
        address TEXT,
        phone TEXT,
        email TEXT,
        birthdate TEXT,
        insurance_status TEXT,
        diagnosis TEXT"""
        # Close the new patient window
        new_patient_window.destroy()

        # Update the patient names in the Combobox
        patient_names = populate_patient_names()
        patient_combobox["values"] = ["Select Patient"] + patient_names + ["Enter New Name"]

        # Show a success message
        messagebox.showinfo( title: "Success",  message: "New patient added to the database")

    # Button to save the new patient
    save_button = tk.Button(new_patient_window, text="Save", command=save_patient)
    save_button.pack()

# Function to add treatment information to the window below
def add_treatment():
    treatment_date = treatment_date_entry.get()
    selected_package_value = selected_package.get()

    # Check if treatment date and package are provided
    if treatment_date and selected_package_value:
        # Extract the package name and price (assuming the package name and price are separated by ":")
        package_parts = selected_package_value.split(":")
        package_name = package_parts[0]
        package_price = package_parts[1]

        # Display treatment information in the "Treatment Dates and Packages" Listbox
        treatment_info = f"Date: {treatment_date}, Package: {package_name}, Price: {package_price}"
        treatment_display_listbox.insert(tk.END,  *elements: treatment_info)

        # Clear the treatment date and selected package fields
        treatment_date_entry.delete( first: 0,  tk.END)
```

add_new_patient() › save_patient()

ostheopathy_4.py × create_html.py

```python
        # Clear the treatment date and selected package fields
        treatment_date_entry.delete(first: 0, tk.END)
        selected_package.set("")


# Variable to store the selected patient
selected_patient = tk.StringVar()

# Create a label and Combobox for selecting or entering a patient's name
patient_name_label = tk.Label(app, text="Select or Enter Patient Name:")
patient_name_label.pack()

# Populate the Combobox with patient names
patient_names = populate_patient_names()
patient_combobox = ttk.Combobox(app, textvariable=selected_patient, values=patient_names)
patient_combobox.pack()


## Bind the Combobox to the selection event
#patient_combobox.bind("<<ComboboxSelected>>", handle_patient_selection)
selected_patient.set("Select Patient")
patient_combobox.set("Select Patient")

# Create an option for entering a new patient's name
patient_combobox["values"] = ["Select Patient"] + patient_names + ["Enter New Name"]


# Retrieve Patient Data button
def retrieve_patient_data():
    selected_name = selected_patient.get()

    # Check if a patient is selected
    if selected_name != "Select Patient" and selected_name != "Enter New Name":
        conn = sqlite3.connect("patienten.db")
        cursor = conn.cursor()

        # Retrieve patient data based on the selected name
        cursor.execute(__sql: "SELECT * FROM patients WHERE name=?", __parameters: (selected_name,))
        patient_data = cursor.fetchone()

        conn.close()
```

add_new_patient() › save_patient()

ostheopathy_4.py × create_html.py

```python
        if patient_data:
            # Clear previous data in the "Patient Information" window
            patient_info_text.delete(index1: 1.0, tk.END)

            # Display patient data in the "Patient Information" window
            patient_info_text.insert(tk.END, chars: f"Name: {patient_data[1]}\n")
            patient_info_text.insert(tk.END, chars: f"Address: {patient_data[2]}\n")
            patient_info_text.insert(tk.END, chars: f"Phone: {patient_data[3]}\n")
            patient_info_text.insert(tk.END, chars: f"Email: {patient_data[4]}\n")
            patient_info_text.insert(tk.END, chars: f"Birthdate: {patient_data[5]}\n")
            patient_info_text.insert(tk.END, chars: f"Insurance Status: {patient_data[6]}\n")
            patient_info_text.insert(tk.END, chars: f"Diagnosis: {patient_data[7]}\n")
        else:
            # Display a message if the selected patient is not found
            patient_info_text.delete(index1: 1.0, tk.END)
            patient_info_text.insert(tk.END, chars: "Patient not found in the database.")
    else:
        # Display a message if no patient is selected or a new name is entered
        patient_info_text.delete(index1: 1.0, tk.END)
        patient_info_text.insert(tk.END, chars: "Please select a patient from the dropdown or enter a new name.")


retrieve_button = tk.Button(app, text="Retrieve Patient Data", command=retrieve_patient_data)
retrieve_button.pack()

# Change Patient Data button (you can implement its functionality)
change_patient_data_button = tk.Button(app, text="Change Patient Data", command=change_patient_data)
change_patient_data_button.pack()

# Generate New Patient button (you can implement its functionality)
generate_patient_button = tk.Button(app, text="Generate New Patient", command=add_new_patient)
generate_patient_button.pack()
# Display Patient Information
patient_info_label = tk.Label(app, text="Patient Information:")
patient_info_label.pack()
patient_info_text = tk.Text(app, height=10, width=40)
patient_info_text.pack()

# Entry for Treatment Date and Treatment Package Selection
treatment_date_label = tk.Label(app, text="Treatment Date:")
treatment_date_entry = tk.Entry(app)
```

add_new_patient() › save_patient()

ostheopathy_4.py × create_html.py

```python
treatment_date_label = tk.Label(app, text="Treatment Date:")
treatment_date_entry = tk.Entry(app)
treatment_date_label.pack()
treatment_date_entry.pack()

treatment_packages = [
    "Package A: $100",
    "Package B: $150",
    "Package C: $200"
]

selected_package = tk.StringVar()
package_menu = tk.OptionMenu(app, selected_package, *values: *treatment_packages)
package_menu_label = tk.Label(app, text="Select Treatment Package:")
package_menu_label.pack()
package_menu.pack()

# Add Treatment Button
add_treatment_button = tk.Button(app, text="Add Treatment", command=add_treatment)
add_treatment_button.pack()

# Display Treatment Dates and Packages
treatment_display_label = tk.Label(app, text="Treatment Dates and Packages:")
treatment_display_label.pack()
treatment_display_listbox = tk.Listbox(app, height=5, width=40)
treatment_display_listbox.pack()

# Generate Invoice Button
generate_invoice_button = tk.Button(app, text="Generate Invoice", command=create_html)
generate_invoice_button.pack()

# New Invoice Button
new_invoice_button = tk.Button(app, text="New Invoice", command=new_invoice)
new_invoice_button.pack()

app.mainloop()
```

add_new_patient() › save_patient()

ostheopathy_4.py create_html.py ×

```python

def make_html(leistungen_und_preise):
    html_code = '''<div class="page-container"></div>
    <div class="logo-container"><img src="https://osteopathie-chiropraktik.com/wp-content/uploads/thegem-logos/logo_3e15e9cbd2b863b62ad43ef77878d5af_2x.png" alt="" width="223" heig
    <table class="invoice-info-container" style="height: 126px; width: 558px;">
    <tbody>
    <tr style="height: 21px;">
    <td class="client-name" rowspan="2" style="height: 34px; width: 340.602px;">{{name_entry}}</td>
    <td style="height: 21px; width: 216.398px;">Andr&eacute; Franken</td>
    </tr>
    <tr style="height: 13px;">
    <td style="height: 13px; width: 216.398px;">Thuyring 19</td>
    </tr>
    <tr style="height: 29px;">
    <td style="width: 340.602px; height: 29px;">{{adress_entry}}</td>
    <td style="width: 216.398px; height: 29px;"></td>
    </tr>
    <tr style="height: 21px;">
    <td style="height: 21px; width: 340.602px;">Rechnungsdatum: <strong>{{datum_entry}}</strong></td>
    <td style="height: 21px; width: 216.398px;">12101 Berlin</td>
    </tr>
    <tr style="height: 21px;">
    <td style="height: 21px; width: 340.602px;">Rechnungsnummer: <strong>{{rechnungs_entry}}</strong></td>
    <td style="height: 21px; width: 216.398px;"><a href="mailto:info@osteopathie-chiropraktik.com">info@osteopathie-chiropraktik.com</a></td>
    </tr>
    <tr style="height: 21px;">
    <td style="height: 21px; width: 340.602px;">Diagnosen: {{diagnose_entry}}</td>
    <td style="height: 21px; width: 216.398px;"></td>
    </tr>
    </tbody>
    </table>'''

    # Öffne eine HTML-Tabelle für die Leistungen und Preise
    html_code += '<table class="line-items-container" style="width: 526px; height: 84px;" height="81">\n'
    html_code += '<thead>\n'
    html_code += '<tr style="height: 21px;">\n'
    html_code += '<th class="heading-quantity" style="width: 10.0078px; height: 21px;"> </th>\n'
    html_code += '<th class="heading-description" style="width: 131.914px; height: 21px;">Datum</th>\n'
    html_code += '<th style="width: 172.484px; height: 21px;"><strong>Leistung</strong></th>\n'
    html_code += '<th class="heading-price" style="width: 10.0078px; height: 21px;"> </th>\n'
    html_code += '<th class="heading-subtotal" style="width: 200.586px; height: 21px;">Preis</th>\n'
    html_code += '</tr>\n'
```

make_html() › for entry in leistungen_und_pre...

ostheopathy_4.py create_html.py

```python
    # Schleife durch die Liste der Leistungen und Preise
    for entry in leistungen_und_preise:
        html_code += '<tr style="height: 21px;">\n'
        html_code += '<td style="width: 10.0078px; height: 21px;"> </td>\n'
        html_code += '<td style="width: 131.914px; height: 21px;">{{bh_tag_entry}}</td>\n'
        html_code += '<td style="width: 172.484px; height: 21px;">' + entry["leistung"] + '</td>\n'
        html_code += '<td class="right" style="width: 10.0078px; height: 21px;"> </td>\n'
        html_code += '<td class="bold" style="width: 200.586px; height: 21px;">' + entry["preis"] + '</td>\n'
        html_code += '</tr>\n'

    # Schließe die HTML-Tabelle
    html_code += '</tbody>\n'
    html_code += '</table>\n'

    html_code += '''</table>
    <table class="line-items-container has-bottom-border" style="width: 469px; height: 82px;">
    <thead>
    <tr style="height: 21px;">
    <th style="width: 135.578px; height: 21px;">Bankdetails</th>
    <th style="width: 195.836px; height: 21px;">Steuernummer</th>
    <th style="width: 136.586px; height: 21px;">Total Due</th>
    </tr>
    </thead>
    <tbody>
    <tr style="height: 61px;">
    <td class="payment-info" style="width: 135.578px; height: 61px;">
    <div>IBAN: <strong>123567744</strong></div>
    <div>BIC: <strong>120000547</strong></div>
    </td>
    <td class="large" style="width: 195.836px; height: 61px;">21/291/XXXXX</td>
    <td class="large total" style="width: 136.586px; height: 61px;">{{total}}</td>
    </tr>
    </tbody>
    </table>
    <div class="footer">
    <div class="footer-info">Bitte zahlen Sie den Rechnungsbetrag innerhalb von 30 Tagen.</div>
    <div class="footer-info">Vielen Dank!</div>
    </div>'''

    # Dateiname für die HTML-Datei
    html_file_name = 'rechnung.html'
```

make_html() › for entry in leistungen_und_pre...

ostheopathy_4.py create_html.py

```python
    html_code += '</tbody>\n'
    html_code += '</table>\n'

    html_code += '''</table>
    <table class="line-items-container has-bottom-border" style="width: 469px; height: 82px;">
    <thead>
    <tr style="height: 21px;">
    <th style="width: 135.578px; height: 21px;">Bankdetails</th>
    <th style="width: 195.836px; height: 21px;">Steuernummer</th>
    <th style="width: 136.586px; height: 21px;">Total Due</th>
    </tr>
    </thead>
    <tbody>
    <tr style="height: 61px;">
    <td class="payment-info" style="width: 135.578px; height: 61px;">
    <div>IBAN: <strong>123567744</strong></div>
    <div>BIC: <strong>120000547</strong></div>
    </td>
    <td class="large" style="width: 195.836px; height: 61px;">21/291/XXXXX</td>
    <td class="large total" style="width: 136.586px; height: 61px;">{{total}}</td>
    </tr>
    </tbody>
    </table>
    <div class="footer">
    <div class="footer-info">Bitte zahlen Sie den Rechnungsbetrag innerhalb von 30 Tagen.</div>
    <div class="footer-info">Vielen Dank!</div>
    </div>'''

    # Dateiname für die HTML-Datei
    html_file_name = 'rechnung.html'

    # Öffne die HTML-Datei im Schreibmodus und schreibe den HTML-Code hinein
    with open(html_file_name, 'w') as html_file:
        html_file.write(html_code)

    print(f'Die HTML-Datei "{html_file_name}" wurde erfolgreich erstellt.')
```

make_html() › for entry in leistungen_und_pre...