

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.


The image shows a screenshot of a desktop application window titled "Invoice Generator". The window has a light gray background and standard macOS window controls (red, yellow, green buttons) in the top-left corner. The interface is organized into several sections. At the top, it says "Select or Enter Patient Name:" followed by a dropdown menu labeled "Select Patient" with a blue checkmark icon. Below this are three buttons: "Retrieve Patient Data", "Change Patient Data", and "Generate New Patient". A section labeled "Patient Information:" contains a large, empty white rectangular box. Below that is a "Treatment Date:" label with a corresponding empty text input field. This is followed by a "Select Treatment Package:" label and a dropdown menu with a blue double-arrow icon. Below the dropdown is an "Add Treatment" button. A section labeled "Treatment Dates and Packages:" contains another large, empty white rectangular box. At the bottom of the window are two buttons: "Generate Invoice" and "New Invoice".

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.

The screenshot shows the 'Invoice Generator' interface. At the top, there's a section 'Select or Enter Patient Name:' with a dropdown menu labeled 'Select Patient'. The dropdown is open, showing a list of patient names: Bertram Gertz, Christa Barkholz, Olga Seidel B.Eng., Ing. Evelin Franke, Isabel Jäntsch, Berthold Herrmann B.Eng., Kordula Schuchhardt, Geraldine Thies, Olena Köster, and Dipl.-Ing. Oskar Naser. Below the dropdown, there's a 'Treatment Date:' field, a 'Select Treatment Package:' dropdown, an 'Add Treatment' button, and a 'Treatment Dates and Packages:' table. At the bottom, there are 'Generate Invoice' and 'New Invoice' buttons.

Figure 2: The user can select patients from the database

The screenshot shows the 'Invoice Generator' interface with the patient 'Isabel Jäntsch' selected. Below the selection, there are buttons for 'Retrieve Patient Data', 'Change Patient Data', and 'Generate New Patient'. The 'Patient Information:' section displays the following details: Name: Isabel Jäntsch, Address: Waltraut-Dussen van-Ring 1, 02127 Belzig, Phone: +49 (0) 6095 573337, Email: Isabel_Jäntsch@web.de, Birthdate: 1949-12-22, Insurance Status: 4, and Diagnosis: FA72.0 Ankylosing hyperostosis. Below this, there's a 'Treatment Date:' field, a 'Select Treatment Package:' dropdown, an 'Add Treatment' button, and a 'Treatment Dates and Packages:' table. At the bottom, there are 'Generate Invoice' and 'New Invoice' buttons.

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.

The screenshot shows the 'Invoice Generator' interface with the patient 'Isabel Jäntsch' selected. Below the selection, there are buttons for 'Retrieve Patient Data', 'Change Patient Data', and 'Generate New Patient'. The 'Patient Information:' section displays the same details as in Figure 3. Below this, the 'Treatment Date:' field is now filled with '28.10.2023'. The 'Select Treatment Package:' dropdown is set to 'Package C: \$200'. Below this, there's an 'Add Treatment' button and a 'Treatment Dates and Packages:' table. The table now contains one entry: 'Date: 28.10.2023, Package: Package B, Price: \$150'. At the bottom, there are 'Generate Invoice' and 'New Invoice' buttons.

Figure 4: The user can input details about the treatments

Invoice Generator

Select or Enter Patient Name:
 ▼

Retrieve Patient Data
 Change Patient Data
 Generate New Patient

Patient Information:
 Name: Isabel Jäntschi
 Address: Waltraut-Dussen van-Ring 1
 02127 Belzig
 Phone: +49 (0) 6095 573337
 Email: Isabel_Jäntschi@web.de
 Birthdate: 1949-12-22
 Insurance Status: 4
 Diagnosis: FA72.0 Ankylosing hyperostosis

Treatment Date:

Select Treatment Package:
 ▼

Add
 Package A: \$100
 Package B: \$150
 Package C: \$200

Treatment Dates and Packages:
 Date: 28.10.2023, Package: Package B, Price: \$150
 Date: 28.10.2023, Package: Package C, Price: \$200

Generate Invoice
 New Invoice

Figure 5: The user can input dates and packages

Invoice Generator

Select or Enter Patient Name:
 ▼

Retrieve Patient Data
 Change Patient Data
 Generate New Patient

Patient Information:
 Name: Isabel Jäntschi
 Address: Waltraut-Dussen van-Ring 1
 02127 Belzig
 Phone: +49 (0) 6095 573337
 Email: Isabel_Jäntschi@web.de
 Birthdate: 1949-12-22
 Insurance Status: 4
 Diagnosis: FA72.0 Ankylosing hyperostosis

Treatment Date:

Select Treatment Package:
 Package A: \$100 ▼

Add Treatment

Treatment Dates and Packages:
 Date: 28.10.2023, Package: Package B, Price: \$150
 Date: 28.10.2023, Package: Package C, Price: \$200
 Date: 30.10.2023, Package: Package A, Price: \$100

Generate Invoice
 New Invoice

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.

```
osteopathy_4.py x create_html.py
1 import tkinter as tk
2 from tkinter import ttk
3 import sqlite3
4 from tkinter import messagebox
5 from tkinter import simpledialog
6
7 app = tk.Tk()
8 app.title("Invoice Generator")
9
10 # Function to populate the Combobox with patient names from the database
11 def populate_patient_names():
12     conn = sqlite3.connect("patienten.db")
13     cursor = conn.cursor()
14     cursor.execute("SELECT name FROM patients")
15     patients = cursor.fetchall()
16     conn.close()
17     return [patient[0] for patient in patients]
18
19 # Function to change patient data
20 def change_patient_data():
21     selected_name = selected_patient.get()
22
23     # Check if a patient is selected
24     if selected_name != "Select Patient" and selected_name != "Enter New Name":
25         conn = sqlite3.connect("patienten.db")
26         cursor = conn.cursor()
27
28         # Retrieve patient data based on the selected name
29         cursor.execute(_sql: "SELECT * FROM patients WHERE name=?", _parameters: (selected_name,))
30         patient_data = cursor.fetchone()
31
32         conn.close()
33
34         if patient_data:
35             # Create a new window for changing patient data
36             change_patient_window = tk.Toplevel(app)
37             change_patient_window.title("Change Patient Data")
38
39             # Labels and Entry widgets for patient details
40             name_label = tk.Label(change_patient_window, text="Name:")
41             name_label.pack()
42             name_entry = tk.Entry(change_patient_window)
43             name_entry.insert(index=0, patient_data[1]) # Set the current name
44             name_entry.pack()
45
46             address_label = tk.Label(change_patient_window, text="Address:")
47             address_label.pack()
48             address_entry = tk.Entry(change_patient_window)
49             address_entry.insert(index=0, patient_data[2]) # Set the current address
50             address_entry.pack()
51
52             phone_label = tk.Label(change_patient_window, text="Phone:")
53             phone_label.pack()
54             phone_entry = tk.Entry(change_patient_window)
55             phone_entry.insert(index=0, patient_data[3]) # Set the current address
56             phone_entry.pack()
57
58             email_label = tk.Label(change_patient_window, text="Email:")
59             email_label.pack()
60             email_entry = tk.Entry(change_patient_window)
61             email_entry.insert(index=0, patient_data[4]) # Set the current address
62             email_entry.pack()
63
64             birthday_label = tk.Label(change_patient_window, text="Birthday:")
65             birthday_label.pack()
66             birthday_entry = tk.Entry(change_patient_window)
67             birthday_entry.insert(index=0, patient_data[5]) # Set the current address
68             birthday_entry.pack()
69
70             insurance_label = tk.Label(change_patient_window, text="Insurance:")
71             insurance_label.pack()
72             insurance_entry = tk.Entry(change_patient_window)
73             insurance_entry.insert(index=0, patient_data[6]) # Set the current address
74             insurance_entry.pack()
75
76             diagnosis_label = tk.Label(change_patient_window, text="Diagnosis:")
77             diagnosis_label.pack()
78             diagnosis_entry = tk.Entry(change_patient_window)
79             diagnosis_entry.insert(index=0, patient_data[7]) # Set the current address
80             diagnosis_entry.pack()
81
82 populate_patient_names()
```

Continued

```
osteopathy_4.py x create_html.py
40 name_label = tk.Label(change_patient_window, text="Name:")
41 name_label.pack()
42 name_entry = tk.Entry(change_patient_window)
43 name_entry.insert(index=0, patient_data[1]) # Set the current name
44 name_entry.pack()
45
46 address_label = tk.Label(change_patient_window, text="Address:")
47 address_label.pack()
48 address_entry = tk.Entry(change_patient_window)
49 address_entry.insert(index=0, patient_data[2]) # Set the current address
50 address_entry.pack()
51
52 phone_label = tk.Label(change_patient_window, text="Phone:")
53 phone_label.pack()
54 phone_entry = tk.Entry(change_patient_window)
55 phone_entry.insert(index=0, patient_data[3]) # Set the current address
56 phone_entry.pack()
57
58 email_label = tk.Label(change_patient_window, text="Email:")
59 email_label.pack()
60 email_entry = tk.Entry(change_patient_window)
61 email_entry.insert(index=0, patient_data[4]) # Set the current address
62 email_entry.pack()
63
64 birthday_label = tk.Label(change_patient_window, text="Birthday:")
65 birthday_label.pack()
66 birthday_entry = tk.Entry(change_patient_window)
67 birthday_entry.insert(index=0, patient_data[5]) # Set the current address
68 birthday_entry.pack()
69
70 insurance_label = tk.Label(change_patient_window, text="Insurance:")
71 insurance_label.pack()
72 insurance_entry = tk.Entry(change_patient_window)
73 insurance_entry.insert(index=0, patient_data[6]) # Set the current address
74 insurance_entry.pack()
75
76 diagnosis_label = tk.Label(change_patient_window, text="Diagnosis:")
77 diagnosis_label.pack()
78 diagnosis_entry = tk.Entry(change_patient_window)
79 diagnosis_entry.insert(index=0, patient_data[7]) # Set the current address
80 diagnosis_entry.pack()
81
82 populate_patient_names()
```

```
ostheopathy_4.py x create_html.py
82 # Add more labels and Entry widgets for other patient details (phone, email, etc.)
83
84 # Function to save the changes
85 def save_changes():
86     new_name = name_entry.get()
87     new_address = address_entry.get()
88     new_email = email_entry.get()
89     new_phone = phone_entry.get()
90     new_birthday = birthday_entry.get()
91     new_insurance = insurance_entry.get()
92     new_diagnosis = diagnosis_entry.get()
93     # Get other updated details similarly...
94
95     conn = sqlite3.connect("patienten.db")
96     cursor = conn.cursor()
97
98     # Update the patient's data in the database
99     cursor.execute( _sql: """UPDATE patients SET name=?, address=?, phone=?, email=?, birthdate=?,
100 insurance_status=?, diagnosis=? WHERE name=?""", _parameters: (new_name, new_address, new_phone,
101 new_email, new_birthday, new_insurance,
102 new_diagnosis, selected_name))
103
104     conn.commit()
105     conn.close()
106
107 # Close the change patient window
108 change_patient_window.destroy()
109
110 # Show a success message
111 messagebox.showinfo( title: "Success", message: "Patient data updated")
112
113 # Button to save the changes
114 save_button = tk.Button(change_patient_window, text="Save Changes", command=save_changes)
115 save_button.pack()
116
117 else:
118     # Display a message if the selected patient is not found
119     messagebox.showerror( title: "Error", message: "Patient not found in the database.")
120
121 else:
122     # Display a message if no patient is selected or a new name is entered
123     messagebox.showerror( title: "Error", message: "Please select a patient from the dropdown or enter a new name.")
124
125 populate_patient_names()
```

```
ostheopathy_4.py x create_html.py
123 # Function to reset all fields to their initial state
124 def reset_app():
125     selected_patient.set("Select Patient")
126     patient_combobox.set("Select Patient")
127     #name_entry.delete(0, tk.END)
128     patient_info_text.delete( index: 1.0, tk.END) # Clear patient information text
129     treatment_date_entry.delete( first: 0, tk.END)
130     selected_package.set("")
131
132
133 # Clear treatment display by re-creating the listbox
134 treatment_display_listbox.destroy()
135 treatment_display_listbox = tk.Listbox(app, height=5, width=40)
136 treatment_display_listbox.pack()
137
138
139 # Function to create a new invoice and reset the app to its original state
140 def new_invoice():
141     reset_app()
142
143
144 def add_new_patient():
145     # Create a separate window for adding a new patient
146     new_patient_window = tk.Toplevel(app)
147     new_patient_window.title("Add New Patient")
148
149     # Labels and Entry widgets for patient details
150     name_label = tk.Label(new_patient_window, text="Name:")
151     name_label.pack()
152     name_entry = tk.Entry(new_patient_window)
153     name_entry.pack()
154
155     street_label = tk.Label(new_patient_window, text="Street/nr:")
156     street_label.pack()
157     street_entry = tk.Entry(new_patient_window)
158     street_entry.pack()
159
160     postcode_label = tk.Label(new_patient_window, text="Postcode/City:")
161     postcode_label.pack()
162     postcode_entry = tk.Entry(new_patient_window)
163     postcode_entry.pack()
164
165     ...
166
167 populate_patient_names()
```

```

ostheopathy_4.py × create_html.py
175 birthday_label = tk.Label(new_patient_window, text="Birthday(YYYY-MM-DD):")
176 birthday_label.pack()
177 birthday_entry = tk.Entry(new_patient_window)
178 birthday_entry.pack()
179
180 insurance_label = tk.Label(new_patient_window, text="Insurance Status:")
181 insurance_label.pack()
182 insurance_entry = tk.Entry(new_patient_window)
183 insurance_entry.pack()
184
185 diagnosis_label = tk.Label(new_patient_window, text="Diagnosis:")
186 diagnosis_label.pack()
187 diagnosis_entry = tk.Entry(new_patient_window)
188 diagnosis_entry.pack()
189
190 # Function to save the new patient to the database
191 def save_patient():
192     new_name = name_entry.get()
193     new_address = street_entry.get() + "\n" + postcode_entry.get()
194     new_email = email_entry.get()
195     new_phone = phone_entry.get()
196     new_birthday = birthday_entry.get()
197     new_insurance = insurance_entry.get()
198     new_diagnosis = diagnosis_entry.get()
199     print(new_diagnosis)
200
201     # Check if any of the fields are empty
202     if not all([new_name, new_address, new_email, new_phone, new_birthday, new_diagnosis, new_insurance]):
203         messagebox.showerror(title="Error", message="All fields must be filled")
204         return
205     print("hallo")
206
207     conn = sqlite3.connect("patienten.db")
208     cursor = conn.cursor()
209
210     # Insert the new patient into the database
211     cursor.execute(_sql: """INSERT INTO patients (name, address, phone,
212         email, birthdate, insurance_status, diagnosis) VALUES (?, ?, ?, ?, ?, ?, ?)""",
213         _parameters: (new_name, new_address, new_phone, new_email, new_birthday, new_insurance, new_diagnosis))
214     conn.commit()
215     conn.close()
216     """ CREATE TABLE IF NOT EXISTS patients (
populate_patient_names()

```

```

ostheopathy_4.py × create_html.py
215     conn.close()
216     """ CREATE TABLE IF NOT EXISTS patients (
217     patient_id INTEGER PRIMARY KEY,
218     name TEXT,
219     address TEXT,
220     phone TEXT,
221     email TEXT,
222     birthdate TEXT,
223     insurance_status TEXT,
224     diagnosis TEXT"""
225     # Close the new patient window
226     new_patient_window.destroy()
227
228     # Update the patient names in the Combobox
229     patient_names = populate_patient_names()
230     patient_combobox["values"] = ["Select Patient"] + patient_names + ["Enter New Name"]
231
232     # Show a success message
233     messagebox.showinfo(title="Success", message="New patient added to the database")
234
235     # Button to save the new patient
236     save_button = tk.Button(new_patient_window, text="Save", command=save_patient)
237     save_button.pack()
238
239     # Function to add treatment information to the window below
240     def add_treatment():
241         treatment_date = treatment_date_entry.get()
242         selected_package_value = selected_package.get()
243
244         # Check if treatment date and package are provided
245         if treatment_date and selected_package_value:
246             # Extract the package name and price (assuming the package name and price are separated by ":")
247             package_parts = selected_package_value.split(":")
248             package_name = package_parts[0]
249             package_price = package_parts[1]
250
251             # Display treatment information in the "Treatment Dates and Packages" Listbox
252             treatment_info = f"Date: {treatment_date}, Package: {package_name}, Price: {package_price}"
253             treatment_display_listbox.insert(tk.END, *elements: treatment_info)
254
255             # Clear the treatment date and selected package fields
256             treatment_date_entry.delete(0, tk.END)
add_new_patient() > save_patient()

```

```

255         # Clear the treatment date and selected package fields
256         treatment_date_entry.delete(first=0, tk.END)
257         selected_package.set("")
258
259
260     # Variable to store the selected patient
261     selected_patient = tk.StringVar()
262
263     # Create a label and Combobox for selecting or entering a patient's name
264     patient_name_label = tk.Label(app, text="Select or Enter Patient Name:")
265     patient_name_label.pack()
266
267     # Populate the Combobox with patient names
268     patient_names = populate_patient_names()
269     patient_combobox = ttk.Combobox(app, textvariable=selected_patient, values=patient_names)
270     patient_combobox.pack()
271
272
273     ## Bind the Combobox to the selection event
274     patient_combobox.bind("<ComboboxSelected>", handle_patient_selection)
275     selected_patient.set("Select Patient")
276     patient_combobox.set("Select Patient")
277
278     # Create an option for entering a new patient's name
279     patient_combobox["values"] = ["Select Patient"] + patient_names + ["Enter New Name"]
280
281
282     # Retrieve Patient Data button
283     def retrieve_patient_data():
284         selected_name = selected_patient.get()
285
286         # Check if a patient is selected
287         if selected_name != "Select Patient" and selected_name != "Enter New Name":
288             conn = sqlite3.connect("patienten.db")
289             cursor = conn.cursor()
290
291             # Retrieve patient data based on the selected name
292             cursor.execute(_sql: "SELECT * FROM patients WHERE name=?", _parameters: (selected_name,))
293             patient_data = cursor.fetchone()
294
295             conn.close()
296
297         add_new_patient() > save_patient()

```

```

297     if patient_data:
298         # Clear previous data in the "Patient Information" window
299         patient_info_text.delete(index1: 1.0, tk.END)
300
301         # Display patient data in the "Patient Information" window
302         patient_info_text.insert(tk.END, chars: f"Name: {patient_data[1]}\n")
303         patient_info_text.insert(tk.END, chars: f"Address: {patient_data[2]}\n")
304         patient_info_text.insert(tk.END, chars: f"Phone: {patient_data[3]}\n")
305         patient_info_text.insert(tk.END, chars: f"Email: {patient_data[4]}\n")
306         patient_info_text.insert(tk.END, chars: f"Birthdate: {patient_data[5]}\n")
307         patient_info_text.insert(tk.END, chars: f"Insurance Status: {patient_data[6]}\n")
308         patient_info_text.insert(tk.END, chars: f"Diagnosis: {patient_data[7]}\n")
309     else:
310         # Display a message if the selected patient is not found
311         patient_info_text.delete(index1: 1.0, tk.END)
312         patient_info_text.insert(tk.END, chars: "Patient not found in the database.")
313     else:
314         # Display a message if no patient is selected or a new name is entered
315         patient_info_text.delete(index1: 1.0, tk.END)
316         patient_info_text.insert(tk.END, chars: "Please select a patient from the dropdown or enter a new name.")
317
318
319     retrieve_button = tk.Button(app, text="Retrieve Patient Data", command=retrieve_patient_data)
320     retrieve_button.pack()
321
322     # Change Patient Data button (you can implement its functionality)
323     change_patient_data_button = tk.Button(app, text="Change Patient Data", command=change_patient_data)
324     change_patient_data_button.pack()
325
326     # Generate New Patient button (you can implement its functionality)
327     generate_patient_button = tk.Button(app, text="Generate New Patient", command=add_new_patient)
328     generate_patient_button.pack()
329
330     # Display Patient Information
331     patient_info_label = tk.Label(app, text="Patient Information:")
332     patient_info_label.pack()
333     patient_info_text = tk.Text(app, height=10, width=40)
334     patient_info_text.pack()
335
336     # Entry for Treatment Date and Treatment Package Selection
337     treatment_date_label = tk.Label(app, text="Treatment Date:")
338     treatment_date_entry = tk.Entry(app)
339
340     add_new_patient() > save_patient()

```

```

ostheopathy_4.py x create_html.py
336 treatment_date_label = tk.Label(app, text="Treatment Date:")
337 treatment_date_entry = tk.Entry(app)
338 treatment_date_label.pack()
339 treatment_date_entry.pack()
340
341 treatment_packages = [
342     "Package A: $100",
343     "Package B: $150",
344     "Package C: $200"
345 ]
346
347 selected_package = tk.StringVar()
348 package_menu = tk.OptionMenu(app, selected_package, *values: *treatment_packages)
349 package_menu_label = tk.Label(app, text="Select Treatment Package:")
350 package_menu_label.pack()
351 package_menu.pack()
352
353 # Add Treatment Button
354 add_treatment_button = tk.Button(app, text="Add Treatment", command=add_treatment)
355 add_treatment_button.pack()
356
357 # Display Treatment Dates and Packages
358 treatment_display_label = tk.Label(app, text="Treatment Dates and Packages:")
359 treatment_display_label.pack()
360 treatment_display_listbox = tk.Listbox(app, height=5, width=40)
361 treatment_display_listbox.pack()
362
363 # Generate Invoice Button
364 generate_invoice_button = tk.Button(app, text="Generate Invoice", command=create_html)
365 generate_invoice_button.pack()
366
367 # New Invoice Button
368 new_invoice_button = tk.Button(app, text="New Invoice", command=new_invoice)
369 new_invoice_button.pack()
370
371 app.mainloop()
372

```

add_new_patient() → save_patient()

```

ostheopathy_4.py x create_html.py
1
2 def make_html(leistungen_und_preise):
3     html_code = '''<div class="page-container"></div>
4     <div class="logo-container"></div>
5     <table class="invoice-info-container" style="height: 126px; width: 558px;">
6     <tbody>
7     <tr style="height: 21px;">
8     <td class="client-name" colspan="2" style="height: 34px; width: 340.602px;">{{name_entry}}</td>
9     <td style="height: 21px; width: 216.398px;">&ndash; &franken</td>
10    </tr>
11    <tr style="height: 13px;">
12    <td style="height: 13px; width: 216.398px;">Thuyring 19</td>
13    </tr>
14    <tr style="height: 29px;">
15    <td style="width: 340.602px; height: 29px;">{{address_entry}}</td>
16    <td style="width: 216.398px; height: 29px;"></td>
17    </tr>
18    <tr style="height: 21px;">
19    <td style="height: 21px; width: 340.602px;">&reg;chungsdatum: <strong>{{datum_entry}}</strong></td>
20    <td style="height: 21px; width: 216.398px;">12101 Berlin</td>
21    </tr>
22    <tr style="height: 21px;">
23    <td style="height: 21px; width: 340.602px;">&reg;chungsnummer: <strong>{{rechnung_entry}}</strong></td>
24    <td style="height: 21px; width: 216.398px;"><a href="mailto:info@osteopathie-chiropraktik.com">info@osteopathie-chiropraktik.com</a></td>
25    </tr>
26    <tr style="height: 21px;">
27    <td style="height: 21px; width: 340.602px;">&reg;agnose: {{diagnose_entry}}</td>
28    <td style="height: 21px; width: 216.398px;"></td>
29    </tr>
30    </tbody>
31    </table>'''
32
33    # Öffne eine HTML-Tabelle für die Leistungen und Preise
34    html_code += '<table class="line-items-container" style="width: 526px; height: 84px;" height="81">\n'
35    html_code += '<thead>\n'
36    html_code += '<tr style="height: 21px;">\n'
37    html_code += '<th class="heading-quantity" style="width: 10.0078px; height: 21px;">&nbsp;</th>\n'
38    html_code += '<th class="heading-description" style="width: 131.914px; height: 21px;">Datum</th>\n'
39    html_code += '<th style="width: 172.484px; height: 21px;"><strong>Leistung</strong></th>\n'
40    html_code += '<th class="heading-price" style="width: 10.0078px; height: 21px;">&nbsp;</th>\n'
41    html_code += '<th class="heading-subtotal" style="width: 200.586px; height: 21px;">Preis</th>\n'
42    html_code += '</thead>\n'
43
44    for entry in leistungen_und_preise:
45        html_code += '<tr>\n'
46        html_code += '<td style="width: 10.0078px;">&nbsp;</td>\n'
47        html_code += '<td style="width: 131.914px;">{{datum}}</td>\n'
48        html_code += '<td style="width: 172.484px;">{{leistung}}</td>\n'
49        html_code += '<td style="width: 10.0078px;">&nbsp;</td>\n'
50        html_code += '<td style="width: 200.586px;">{{preis}}</td>\n'
51        html_code += '</tr>\n'
52
53    html_code += '</table>'
54
55    return html_code
56

```

make_html() → for entry in leistungen_und_pre...


```
ostheopathy_4.py create_html.py x
46 # Schleife durch die Liste der Leistungen und Preise
47 for entry in leistungen_und_preise:
48     html_code += '<tr style="height: 21px;">\n'
49     html_code += '<td style="width: 10.0078px; height: 21px;">&nbsp;</td>\n'
50     html_code += '<td style="width: 131.914px; height: 21px;">{{bh_tag_entry}}</td>\n'
51     html_code += '<td style="width: 172.484px; height: 21px;">' + entry["leistung"] + '</td>\n'
52     html_code += '<td class="right" style="width: 10.0078px; height: 21px;">&nbsp;</td>\n'
53     html_code += '<td class="bold" style="width: 200.586px; height: 21px;">' + entry["preis"] + '</td>\n'
54     html_code += '</tr>\n'
55
56 # SchlieÙe die HTML-Tabelle
57 html_code += '</tbody>\n'
58 html_code += '</table>\n'
59
60 html_code += '''</table>
61 <table class="line-items-container has-bottom-border" style="width: 469px; height: 82px;">
62 <thead>
63 <tr style="height: 21px;">
64 <th style="width: 135.578px; height: 21px;">Bankdetails</th>
65 <th style="width: 195.836px; height: 21px;">Steuernummer</th>
66 <th style="width: 136.586px; height: 21px;">Total Due</th>
67 </tr>
68 </thead>
69 <tbody>
70 <tr style="height: 61px;">
71 <td class="payment-info" style="width: 135.578px; height: 61px;">
72 <div>IBAN: <strong>123567744</strong></div>
73 <div>BIC: <strong>120000547</strong></div>
74 </td>
75 <td class="large" style="width: 195.836px; height: 61px;">21/291/XXXX</td>
76 <td class="large total" style="width: 136.586px; height: 61px;">{{total}}</td>
77 </tr>
78 </tbody>
79 </table>
80 <div class="footer">
81 <div class="footer-info">Bitte zahlen Sie den Rechnungsbetrag innerhalb von 30 Tagen.</div>
82 <div class="footer-info">Vielen Dank!</div>
83 </div>'''
84
85 # Dateiname für die HTML-Datei
86 html_file_name = 'rechnung.html'
87
88 make_html() > for entry in leistungen_und_pre...
```

```
ostheopathy_4.py create_html.py x
57 html_code += '</tbody>\n'
58 html_code += '</table>\n'
59
60 html_code += '''</table>
61 <table class="line-items-container has-bottom-border" style="width: 469px; height: 82px;">
62 <thead>
63 <tr style="height: 21px;">
64 <th style="width: 135.578px; height: 21px;">Bankdetails</th>
65 <th style="width: 195.836px; height: 21px;">Steuernummer</th>
66 <th style="width: 136.586px; height: 21px;">Total Due</th>
67 </tr>
68 </thead>
69 <tbody>
70 <tr style="height: 61px;">
71 <td class="payment-info" style="width: 135.578px; height: 61px;">
72 <div>IBAN: <strong>123567744</strong></div>
73 <div>BIC: <strong>120000547</strong></div>
74 </td>
75 <td class="large" style="width: 195.836px; height: 61px;">21/291/XXXX</td>
76 <td class="large total" style="width: 136.586px; height: 61px;">{{total}}</td>
77 </tr>
78 </tbody>
79 </table>
80 <div class="footer">
81 <div class="footer-info">Bitte zahlen Sie den Rechnungsbetrag innerhalb von 30 Tagen.</div>
82 <div class="footer-info">Vielen Dank!</div>
83 </div>'''
84
85 # Dateiname für die HTML-Datei
86 html_file_name = 'rechnung.html'
87
88 # Öffne die HTML-Datei im Schreibmodus und schreibe den HTML-Code hinein
89 with open(html_file_name, 'w') as html_file:
90     html_file.write(html_code)
91
92 print(f'Die HTML-Datei "{html_file_name}" wurde erfolgreich erstellt.')
93
94 make_html() > for entry in leistungen_und_pre...
```