SE 226 Advanced Programming Spring Project

Best Hotels for You

SILA ÇETİNKAYA

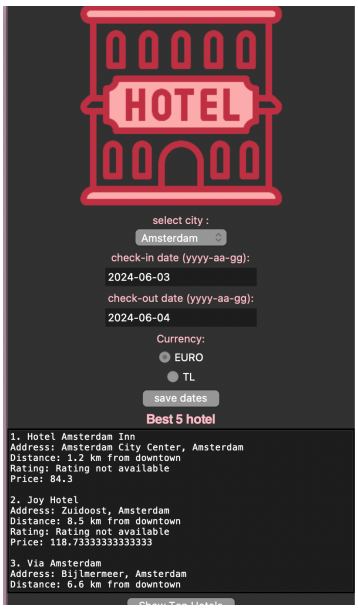
20210601016

Best Hotel for You

Creating a hotel listing program with a GUI was a truly enjoyable project for me. I faced some challenges along the way, particularly with web scraping, as I had no prior experience in this area. Adding the functionality to include hotel prices and convert them from euros to Turkish lira was indeed a challenging aspect of the project for me. This aspect proved to be quite demanding and required a significant amount of time and effort. However, overcoming these challenges was rewarding. Designing the GUI was a delightful part of the project, and I thoroughly enjoyed it. For the hotel information storage and display section, I mostly relied on course materials, which proved to be immensely helpful. Overall, despite the initial difficulties, the project was a valuable learning experience, and I gained new skills in both web scraping and GUI development.

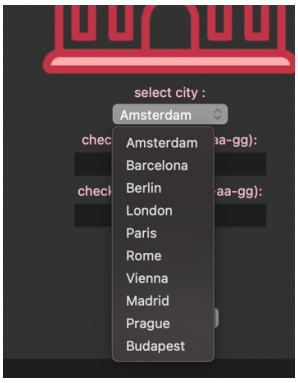
GUI Development Requirements

1-Use Tkinter or any other suitable GUI library to create the graphical interface for the program.
I used Tkinter in my project, and I found it very user-friendly.



2- Design and implement a dropdown list to allow users to select a city (at least 10 Europe cities)

I created an OptionMenu with 10 European cities. I defined a variable named "city_menu" and linked it to "selected_city". When the user selects a city, the "selected_city" variable will receive the name of that city.



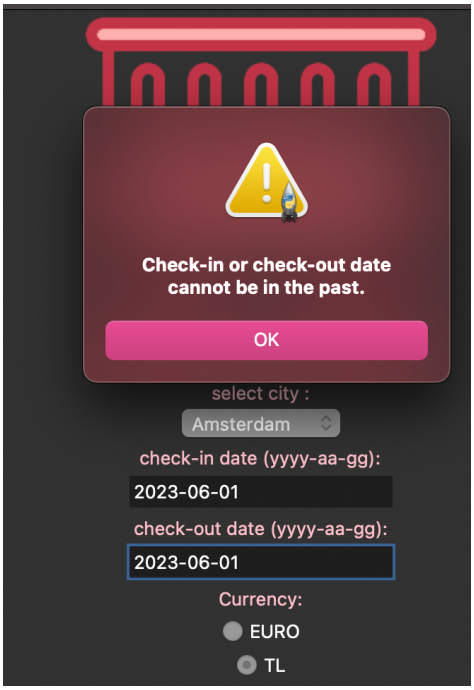

```python
# Şehir seçimi için dropdown menü
label = tk.Label(root, text="select city :",foreground="pink")
label.pack()

european_cities = [
    "Amsterdam", "Barcelona", "Berlin",     "London",
    "Paris", "Rome", "Vienna", "Madrid", "Prague", "Budapest"
]
selected_city = tk.StringVar(root)
selected_city.set(european_cities[0])

city_menu = tk.OptionMenu(root, selected_city, *european_cities)
city_menu.pack()
```

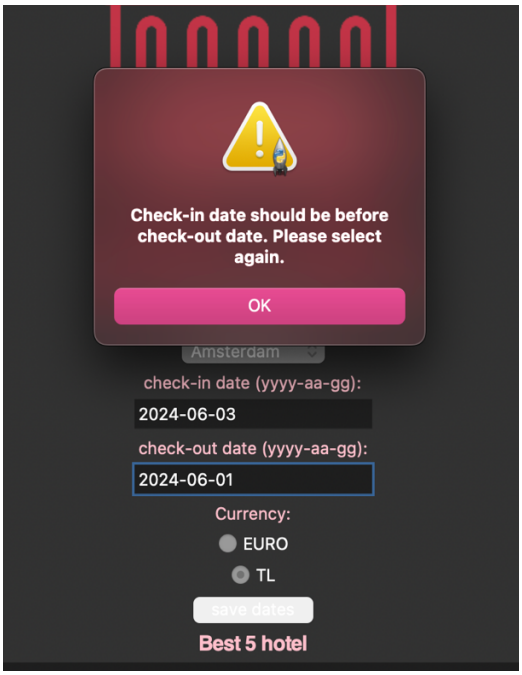
3- Design and implement a GUI item to allow users to enter check-in and check-out dates

When creating check-in and check-out dates, there are a few important points to consider. The first is to ensure that the date selected is not in the past. Secondly, the check-in date should be before the check-out date. To address these concerns, I created a function called "validate_dates".

```python
def validate_dates():
    try:
        check_in_date = check_in_entry.get()
        check_out_date = check_out_entry.get()
        today = datetime.now().strftime("%Y-%m-%d")

        if check_in_date < today or check_out_date < today:
            messagebox.showwarning("Invalid Date", "Check-in or check-out date cannot be in the past.")
            return False

        if check_in_date >= check_out_date:
            messagebox.showwarning("Invalid Date", "Check-in date should be before check-out date. Please select again.")
            return False

        return True

    except ValueError:
        messagebox.showwarning("Invalid Date", "Please select the dates correctly.")
        return False
```

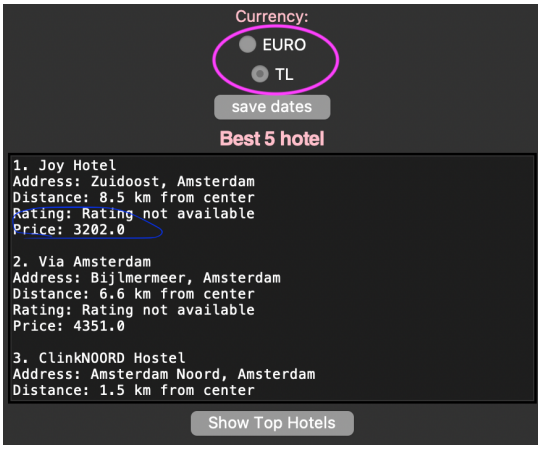
```python
# Giriş ve çıkış tarihleri için giriş kutuları
check_in_label = tk.Label(root, text="check-in date (yyyy-aa-gg):",foreground="pink")
check_in_label.pack()
check_in_entry = tk.Entry(root)
check_in_entry.pack()

check_out_label = tk.Label(root, text="check-out date (yyyy-aa-gg):",foreground="pink")
check_out_label.pack()
check_out_entry = tk.Entry(root)
check_out_entry.pack()
```

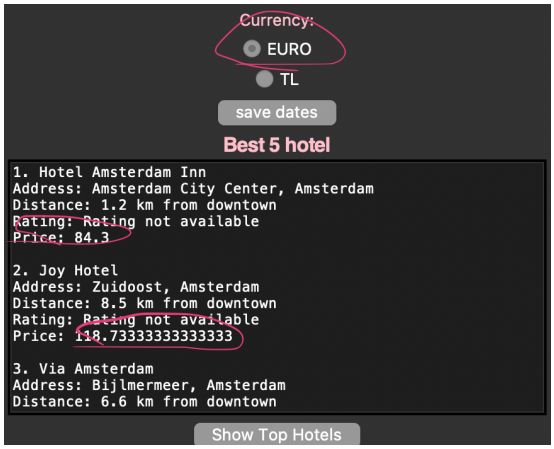




4- Create a radio-buton to show the hotel prices in Euro or TL (Use conversion 1Euro =30TL)

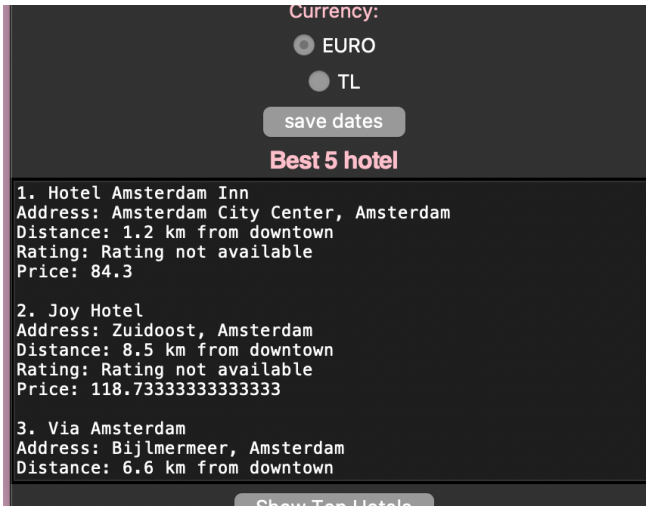
I created 2 radio button to select currency for the hotel price. The radio buttons let the user choose between "EURO" and "TL" (Turkish Lira).






```
check_out_entry.pack()
# Para birimi seçimi için radyo düğmeleri
currency_label = tk.Label(root, text="Currency:", foreground="pink")
currency_label.pack()

currency = tk.StringVar()
currency.set("TL")  # Default selection

euro_radio_button = tk.Radiobutton(root, text="EURO", variable=currency, value="EURO")
euro_radio_button.pack()

tl_radio_button = tk.Radiobutton(root, text="TL", variable=currency, value="TL")
tl_radio_button.pack()
```

5- Create an area in the GUI to display top 5 hotels (all aIributes will be presented).




```
# En iyi 5 otel
title_label = tk.Label(root, text="EN İYİ 5 HOTEL", font=("Helvetica", 16, "bold"),foreground="pink")
title_label.pack()

# En iyi 5 oteli göstermek için metin alanı
top_hotels_text = tk.Text(root, height=15, width=60)
top_hotels_text.pack()

# Otelleri göstermek için buton
display_button = tk.Button(root, text="Show Top Hotels", command=lambda: fetch_hotels(selected_city.get(), check_in_entry.get(), check_out_entry.get(), currency.get())
display_button.pack()
```

6- The sorted hotel list (all data aIributes) will be saved in a text/csv file (myhotels.txt or myhotels.csv).

```
        #otel bilgileri
        top_hotels_data.append({
            'title': name,
            'address': address if address else "Address not available",
            'distance': distance if distance else "Distance not available",
            'rating': rating if rating else "Rating not available",
            'price': price if price else "NOT GIVEN"
        })

        # loop 5 otelde durduran
        if len(top_hotels_data) >= 5:
            break
    #data kaydetme kısmı
    hotels = pd.DataFrame(top_hotels_data)
    hotels.head()
    hotels.to_csv('test_hotels.csv', header=True, index=False)
```

Web Scrapping Requirements

1-Utilize the “requests” library to send HTTP requests and fetch the hotel information from Booking.com. The hotel information must contain o Hotel title ,Address ,Distancet to city center,hotel rating ,price for the first 10 hotels that are listed in web page.

response = requests.get(url, headers=headers)

```
url = get_url(city, check_in, check_out)
headers = {
    'User-Agent': 'Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/
    'Accept-Language': 'en-US, en;q=0.5'
}
response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, 'html.parser')
hotels = soup.findAll('div', {'data-testid': 'property-card'})
```

2- Use the “beautifulsoup” library to parse the HTML content and extract the necessary data for the given user query.

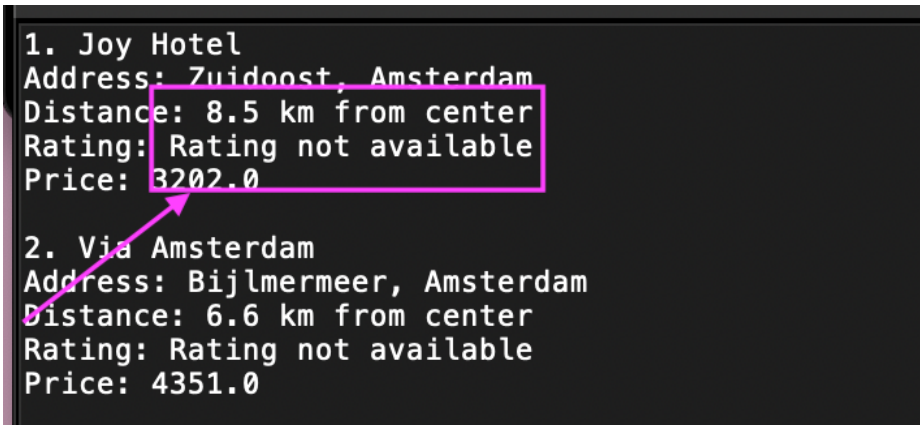
```
response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, 'html.parser')
hotels = soup.findAll('div', {'data-testid': 'property-card'})

top_hotels_data = []  # Clear existing data

for hotel in hotels:
    # add hotel name
    name_element = hotel.find('div', {'data-testid': 'title'})
    name = name_element.text.strip()
    # Extract the hotel address
    address_element = hotel.find('span', {'data-testid': 'address'})
    address = address_element.text.strip() if address_element else None
    # add distance
    distance_element = hotel.find('span', {'data-testid': 'distance'})
    distance = distance_element.text.strip() if distance_element else None
    # add rating
    rating_element = hotel.find('span', {'class': 'a3332d346a'})
    rating = rating_element.text.strip() if rating_element else None
    # add hotel price
    price_element = hotel.find('span', {'data-testid': 'price-and-discounted-price'})
    price_str = price_element.text.strip() if price_element else "NOT GIVEN"
```

->soup = BeautifulSoup(response.text, 'html.parser')

-> hotels = soup.findAll('div', {'data-testid': 'property-card'})

3- If any aIribute/field does not hold a valid value assign “NOT GIVEN” to it.

```
    for hotel in hotels:
        # add hotel name
        name_element = hotel.find('div', {'data-testid': 'title'})
        name = name_element.text.strip()
        # Extract the hotel address
        address_element = hotel.find('span', {'data-testid': 'address'})
        address = address_element.text.strip() if address_element else None
        # add distance
        distance_element = hotel.find('span', {'data-testid': 'distance'})
        distance = distance_element.text.strip() if distance_element else None
        # add rating
        rating_element = hotel.find('span', {'class': 'a3332d346a'})
        rating = rating_element.text.strip() if rating_element else None
        # add hotel price
        price_element = hotel.find('span', {'data-testid': 'price-and-discounted-price'})
        price_str = price_element.text.strip() if price_element else "NOT GIVEN"
```



Hotel Information Storage/Display Requirements

1-Utilize Python data structures, such as lists and dictionaries, to store the hotel data scraped from the internet.

```
# Create dictionary for cities and city_codes
cities = {
    "Rome": "-126693",
    "Madrid": "-390625",
    "Prague": "-553173",
    "Vienna": "-1995499",
    "Amsterdam": "-2140479",
    "Barcelona": "-372490",
    "Berlin": "-1746443",
    "London": "-2601889",
    "Paris": "-1456928",
    "Budapest": "850553"
}

top_hotels_data = []  # Define as a global variable
```

2- Sort the data based on previously mentioned rules (price or rating).

```
    # sort by order
    top_hotels_data.sort(key=lambda x: x['price'])
```

3- Store the hotel information (all aIributes and all hotels) in a text or csv file.

```
    #data kaydetme kısmı
    hotels = pd.DataFrame(top_hotels_data)
    hotels.head()
    hotels.to_csv('test_hotels.csv', header=True, index=False)
```

Additional-Provide error handling mechanisms to handle situations where the internet connection is unavailable, or the data cannot be retrieved.

```
        display_top_hotels()

    except requests.exceptions.RequestException as e:
        messagebox.showerror("Error", f"An error occurred while fetching hotel information: {str(e)}")

    except Exception as e:
        messagebox.showerror("Error", f"An error occurred while fetching hotel information: {str(e)}")
```