Report of the SE226 Project

In this project, only one class and multiple methods were utilized. Additionally, various libraries from Python were employed to meet the necessary requirements. The details of these class, methods, and libraries are as follows:

Web Scraping Requirements

I. Utilize the “requests” library to send HTTP requests and fetch the hotel information from Booking.com.The hotel information must contain:
o Hotel title
o Address
o Distance to city center/downtown
o Hotel rating
o Price
for the first 10 hotels that are listed in web page.

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

III. If any attribute/field does not hold a valid value assign “NOT GIVEN” to it.

```
def getHotelInformation(city,checkInDate,checkOutDate):
 url = f"https://www.booking.com/searchresults.html?ss={city}&checkin_year={checkInDate[:4]}&checkin_month={checkInDate[5:7]}&checkin_m
 headers = {
  'User-Agent': 'Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.64 Safari/537.36',
  'Accept-Language': 'en-US, en;q=0.5'
 }
 try:
  response = requests.get(url, headers=headers)
  response.raise_for_status()
  soup = BeautifulSoup(response.text, features: 'html.parser')

  hotels = soup.findAll( name: 'div', attrs: {'data-testid': 'property-card'})
  hotels_data = []



  for hotel in hotels[:10]:
   name_element = hotel.find('div', {'data-testid': 'title'})
   name = name_element.text.strip() if name_element else 'NOT GIVEN'

   address_element = hotel.find('span', {'data-testid': 'address'})
   address = address_element.text.strip() if address_element else 'NOT GIVEN'

   distance_element = hotel.find('span', {'data-testid': 'distance'})
   distance = distance_element.text.strip() if distance_element else 'NOT GIVEN'

   rating_element = hotel.find('div', {'data-testid': 'review-score'})
   rating = rating_element.text.strip()[:3] if rating_element else 'NOT GIVEN'

   price_element = hotel.find('span', {'data-testid': 'price-and-discounted-price'})
   price = price_element.text.strip()[3:] if price_element else 'NOT GIVEN'
```

In the code block above, information was requested from Booking.com by sending HTTP requests. The response from the server was stored in the response variable.Then parsed the HTML content of the response using BeautifulSoup, a Python library for parsing HTML and XML documents. The parsed content was stored in the soup variable, allowing easy navigation and data extraction from the HTML document.Finally, the first 10 hotels were taken from the list and the name, address, distance, score and price of each hotel were extracted. If the relevant HTML element was not found, 'NOT GIVEN' was assigned as the value. In summary, hotel information was retrieved from a web page and stored in a structured format.

Hotel Information Storage/Display Requirements

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

```
hotels_data.append({
 'name': name,
 'address': address,
 'distance': distance,
 'rating': rating,
 'price': price
})
```

A dictionary containing the retrieved hotel information was added to a previously created list named “hotels_data”. Each dictionary represents a single hotel and contains the keys 'name', 'address', 'distance', 'rating' and 'price' with their respective values.

II. Sort the data based on previously mentioned rules.

The function btSort sorts a list of hotel data based on their ratings in descending order. It handles cases where the rating is 'NOT GIVEN' by temporarily removing these entries and reappending them at the end of the sorted list.

```python
def btSort():
 global hotels_data
 notGivenList = []

 for i in range(len(hotels_data)):
   for j in range(len(hotels_data)-i-1):

    if hotels_data[j]['rating'] == "NOT GIVEN":
     notGivenList.append(hotels_data[j])
     hotels_data.pop(j)
     continue

    if hotels_data[j]['rating'] < hotels_data[j+1]['rating']:
     hotels_data[j], hotels_data[j+1] = hotels_data[j+1], hotels_data[j]

 hotels_data.extend(notGivenList)
```

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

First, the hotels_data list containing hotel information dictionaries is converted to a Pandas DataFrame.A DataFrame is a tabular data structure similar to a spreadsheet, with rows and columns. Each dictionary in hotels_data becomes a row in the DataFrame, and the keys of the dictionaries become the column names.DataFrame hotels are then saved in a CSV file named 'test_hotels.csv'. Below, the code is followed by an example.

```python
hotels = pd.DataFrame(hotels_data)
hotels.head()
hotels.to_csv( path_or_buf: 'test_hotels.csv', header=True, index=False)
```

```
name,address,distance,rating,price
Hôtel Sunny,"5th arr., Paris",2.2 km from downtown,7.6,"25,240"
Meliá Paris Vendôme,"1st arr., Paris",2.2 km from downtown,8.1,"87,330"
Kyriad Hotel XIII Italie Gobelins,"13th arr., Paris",2.6 km from downtown,8.0,"28,559"
Hotel Restaurant Au Boeuf Couronné,"19th arr., Paris",4.5 km from downtown,8.4,"29,224"
Hotel Best Western Anjou Lafayette,"9th arr., Paris",2.2 km from downtown,8.2,"30,949"
Hotel le 18 Paris,"18th arr., Paris",4.5 km from downtown,7.7,"25,555"
NH Paris Gare de l'Est,"10th arr., Paris",2.2 km from downtown,7.7,"30,826"
Hotel des Bains,"14th arr., Paris",2.5 km from downtown,7.6,"27,369"
Hôtel Ambassadeur,"17th arr., Paris",4.3 km from downtown,7.6,"22,749"
Hôtel Quartier Latin,"5th arr., Paris",1 km from downtown,8.2,"35,999"
```

GUI Development Requirements

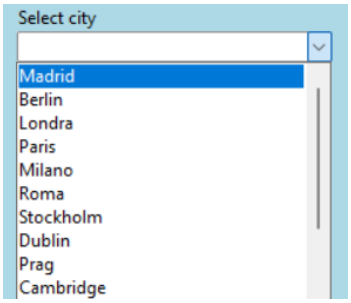
I. Use Tkinter or any other suitable GUI library to create the graphical interface for the program.

```
window = Tk()
window.title("Booking Reservation Panel")
window.geometry("1200x600")
window.resizable(width=False, height=False)
window.config(bg='#add8e6')
```

The above code uses the Tkinter library to create a booking panel. The purpose of the code is to provide a graphical user interface through which users can make reservations. The main window is created, the title, dimensions and background color are set. This basic structure provides a foundation for other components (buttons, input boxes, labels, etc.) that will be added to the interface later.

II. Design and implement a dropdown list to allow users to select a city.

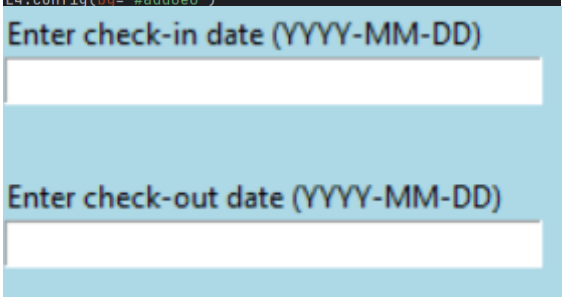
```
E1 = ttk.Combobox(window, width=32, state='readonly')
E1['values'] = ('Madrid','Berlin','Londra','Paris','Milano','Roma','Stockholm',
E1.place(x=67, y=45)
```



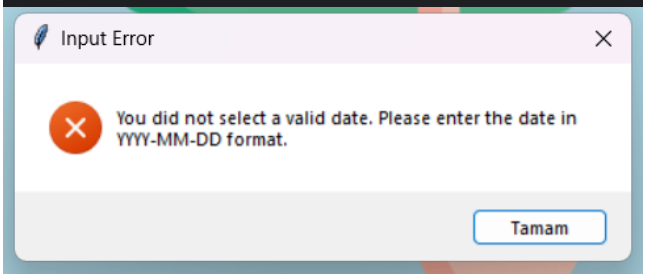
As in the code above and the example on the side, a dropdownlist was created and 14 cities were added for the user to select. Additionally, with the "state=readonly" code, it was ensured that input cannot be entered without making any selection.

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

```
E2 = Entry(window, width=35)
E2.place(x=67, y=110)

L3 = Label(window, text="Enter check-out date (YYYY-MM-DD)")
L3.place(x=65, y=155)
L3.config(bg='#add8e6')

E3 = Entry(window, width=35)
E3.place(x=67, y=175)

L4 = Label(window, text="Select people")
L4.place(x=65,y=218)
L4.config(bg='#add8e6')
```




```
def validate_date(date_text):
 try:
  datetime.strptime(date_text, __format: '%Y-%m-%d')
  return True
 except ValueError:
  return False
```



Entries that allow users to enter entry and exit dates were created and their locations were determined.Additionally, a method was created to prevent incorrect entries.This method is used to validate date inputs from users. By using the `strptime` function from the `datetime` module, it checks whether a date string conforms to the specified format. If the date format is correct, it returns `True`; otherwise, it returns `False`.

IV. Create a radio-button to show the hotel prices in Euro or TL.

```
R1 = Radiobutton(window, text="TL", variable=var, value=1,command=convertEuroToTl)
R1.place(x=65, y=322)
R1.config(bg='#add8e6')

R2 = Radiobutton(window, text="EURO", variable=var, value=2,command=convertTlToEuro)
R2.place(x=105, y=322)
R2.config(bg='#add8e6')
```

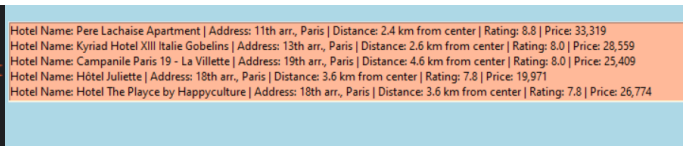

```
def convertTlToEuro():
 global Lb1
 global hotels_data
 global controlConvert

 if(controlConvert ==0):
  controlConvert = 1

  for hotel in hotels_data:
   currentValue = 0.0
   currentValue = float(hotel['price'].replace(',', '.')) / 30

   if currentValue < 1:
    currentValue *= 1000
    intValue = int(currentValue)
    hotel['price'] = str(intValue)
   else:
    formattedValue = format(currentValue,".3f")
    hotel['price'] = str(formattedValue).replace( __old: '.',  __new: ',')

a = 0
for hotel in hotels_data:
 text = f"Hotel Name: {hotel['name']} | Address: {hotel['address']} | Dist
 Lb1.insert(a,  *elements: text)
 a += 1
```

```
def convertEuroToTl():
 global controlConvert

 if(controlConvert == 1):
  controlConvert = 0

  for hotel in hotels_data:
   currentValue = 0.0
   currentValue = int(hotel['price'].replace(',', '')) * 30
   roundedValue = round(currentValue)
   strValue = str(roundedValue)
   if len(strValue) == 5:
    formattedValue = strValue[:2] + "," + strValue[2:]
    hotel['price'] = formattedValue
   else:
    formattedValue = strValue[:3] + "," + strValue[3:]
    hotel['price'] = formattedValue

a = 0
for hotel in hotels_data:
 text = f"Hotel Name: {hotel['name']} | Address: {hotel['address'
 Lb1.insert(a,  *elements: text)
 a += 1
```

First of all, 2 radio buttons were created and their positions were determined. Then, the functions of these buttons were determined to enable the conversion process. With the "controlConvert" variable, functions are prevented from being called repeatedly when a button is pressed repeatedly. The currency was converted from euro to TL or from TL to euro by making the necessary variable conversions within the functions. Finally, the updated list was rewritten to the listbox.

V. Create an area in the GUI to display top 5 hotels.

```
Lb1 = Listbox(window)
Lb1.place(x=65, y=370, width=1070, height
Lb1.config(bg='#ffb999')
```

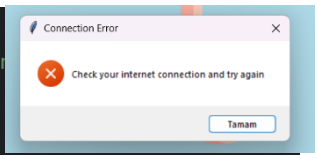


A listbox was created and its location was determined. As shown in the example, the top 5 of the 10 hotels in the pre-sorted list were printed into the listbox.

Additional Considerations

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

```
except requests.ConnectionError:
 messagebox.showerror( title: 'Connection Error',  message: 'Check your internet co
 return []
```



After requesting data from the site and using try in the code block received, in order to prevent the program from crashing in cases where there is no internet connection, connection errors were prevented with this except block and a message was sent to the user providing information about it.

While doing these, many problems have been solved and the requirements that have been completed as solved are above. There are no requirements left unfulfilled and the project has been completed.