Python-projekt Bestellassistent / Chatbot

Von Benjamin Marx, 4. Klassenarbeit (23/24)

Aufgaben in den Mindestanforderungen

Die split() Funktion

Die Funktion `split(x)` kann in Python einen String an jeder Zeichenkette x teilen und die Bestandteile in eine Liste zusammenführen. Gibt man keinen Parameter an wird automatisch eine Leertaste „ “ genommen, was praktisch sein kann um einen Satz in Wörter zu teilen. Beispiel:

```
„Hallo und guten Tag Welt“.split()
>> [„Hallo“, „und“, „guten“, „Tag“, „Welt]
```

Erklärung Zeile 9 & 10

Sowohl in Zeile 9 als auch in Zeile 10 wird der sequentielle Datentyp `wort`, also die Liste der Wörter (Strings) des user-inputs, indiziert. In Zeile 9 wird das Element am Index 0 ausgelesen, und in Zeile 10 der Index 10.

So kann sichergestellt werden, dass das erste Wort des Benutzers (also die Anzahl, wenn seine Antwort richtig formatiert ist) in `anzahl` und das zweite Wort (also das Gericht) in `gericht` gespeichert wird.

Projektprotokoll

24.04.

Heute habe ich erst einmal nur mit den Projektideen herumprobiert. Ich habe mich dann für den Pizza-Chatbot entschieden, da das Projekt auf mich persönlich interessanter wirkt. Dann habe ich mir ein wenig die Struktur meines Programmes überlegt, weil es auf mich keinen besonders effektiven Eindruck machte einfach nur viele if-statements und Ausgaben aneinander zu reihen, wie im Orientierungsbeispiel vorgegeben. Dazu aber später mehr.

26.04.2024

Heute habe ich mir hauptsächlich vorgenommen die beiden in der Projektbeschreibung gestellten Aufgaben zu bearbeiten. Die Lösung ist oben zu finden. Außerdem habe ich am ersten Prototypen des geplanten Formats, welches ich später erkläre, gearbeitet.

03.05

Ich habe heute weiter an dem Prototypen gebastelt. Ich glaube, dass ich einen sinnvollen Plan habe, wohin es gehen soll, mit einer main-loop, in der immer erst die Antwort analysiert und ein passender Output herausgesucht wird, und einem Checkpoint-system für Ausgaben. Genaueres zu erklären ergibt aber vermutlich erst wirklich Sinn, wenn der Prototyp steht, was ich vorraussichtllich nächste Woche schaffe.

08.05

Heute habe ich einen ersten Prototypen fertiggestellt.

Die Idee ist, dass das Programm aufgebaut ist mehreren Phasen (also zum Beispiel „`greeting`“, „`ordering`“ und „`checkout`“) und der aktuelle Status, also die aktuelle Phase, in einer Variable gespeichert wird.

Zu jeder Phase gibt es eine (sortierte) Liste mit einzelnen Phrasen, die als Dictionary mit weiteren Informationen, wie der Name (um auf die Phrase zugreifen zu können) gespeichert sind. Teilweise können die Phrasen auch als Checkpoints fungieren, sodass ein Satz, der nur einmal ausgegeben werden soll, wie den Begrüßungssatz, im Dictionary abgehakt werden kann. So sehen die Outputs jetzt also vorläufig für die Begrüßung aus:

```
outputs = {
    "greeting": [{
        "name": "welcome",
        "checked": False,
        "phrases": [f"Hallo und willkommen bei der Pizzeria
{PIZZERIA_NAME}! Was kann ich für sie tun?"
         ]
    }, {
        "name": "name_q",
        "checked": False,
        "phrases": ["Okay. Was ist denn dein Name?", "Sehr gut. Aber wie
heißt du eigentlich?"]
    }, {
        "name": "name_a",
        "checked": False,
        "phrases": [f"Freut mich _user_name. Möchtest du etwas bestellen?"]

    }]
```

Nun gibt es eine mainloop, die immer nacheinander folgendes machst:

1. Den aktuellen input des Menschen in einer zur aktuellen Phase angepassten Funktion analysiert und den Namen von passenden nächsten Antwortphrasen in der Liste `allowed_outputs` (sodass die wichtigeren/passenderen Antworten weiter vorne stehen) speichert.
   Hier soll später hauptsächlich mit if und else Statements geprüft werden, wie der Bot antworten sollt.
2. Einen auf der `allowed_outputs`-Liste basierenden Satz heraussuchen, den der Bot ausgibt. Dafür werden die gespeicherten Namen in `allowed_outputs` durchgegangen. Wenn dann das zugehörige Dictionary noch nicht abgehakt ist, soll aus der Phrasenliste ein zufällig gewählter Satz ausgegeben werden.
   Die Funktion, welche das in der Begrüßungsphase macht:

   ```
   # Funktionen für Ausgaben des Bots
   def greetings():
       """Begrüßung"""
       global outputs, allowed_outputs, last_output, user_name
       for dic in outputs["greeting"]:
           if dic["checked"] == False:
               if dic["name"] in allowed_outputs:
                   dic["checked"] = True
                   last_output = dic["name"]
                   return random.choice(dic["phrases"])

       return "Ich habe keine Antwort für Sie, tut mir leid."
   ```

   Die Idee, dass ein zufälliger Satz ausgewählt werden soll, habe ich, damit ich die Möglichkeit habe ein wenig Variation in den Bot hereinzubringen, auch wenn ich vermutlich meistens nur einen Satz in die Liste tun werde.

3. Den ausgewählten Satz ausgeben und auf eine Antwort warten.

15.05

Heute habe ich die Grundfunktionen für die Funktionsweise weitergeschrieben oder erstellt. Darunter:

- `show_menu()` - eine Funktion, die das Menü geordnet mit print-Statements ausgibt. Das Menü wird am Anfang als Dictionary gespeichert. Es sieht (beispielhafte verkürzte Version) so aus: `menu = {'pizzas': {'margherita': 8.50, 'salami': 9.00 }, 'drinks': { 'cola': 2.50, 'wasser': 1.50}}`
- `add_to_cart(item, quantity)` - eine Funktion, die etwas einer bestimmten Anzahl in den Warenkorb hinzufügt. Der Warenkorb ist ein Dictionary , bei dem zu jeder enthaltenden Menüauswahl als Schlüssel die Quantität enthalten ist.
- `remove_from_cart(item, quantity)` - dasselbe, nur anders herum.
- `view_cart()` - Eine Funktion, die den Warenkorb organisiert mit print-Statements ausgibt.
- `get_item_price(item)` - eine Funktion, die in dem Menü-dictionary den Preis eines bestimmten Inhalts heraussucht und zurückgibt.
- `Calculate_total()` - Eine Funktion, die für jedes Gericht im Warenkorb den Preis heraussucht und die Summe ausgibt.

17.05

Heute habe ich eigentlich nur zwei Sachen am Code umgestellt, von denen ich denke, dass sie so langfristig besser funktionieren:

1. Beim Menü-Dictionary steht der Preis jetzt nicht einfach so als value, sondern als value in einem neuen genesteten Dictionary. Ein anderer Wert in diesem Dictionary ist auch eine Zutatenliste, die ich vielleicht später benutzen möchte.
2. Zuvor habe ich im `outputs` – Dictionary auf Variablen und Konstanten mit f-Strings zugegriffen. Da das `outputs` – Dictionary allerdings am Anfang definiert wird, wird immer der Status der jeweiligen Variable bei der Definition zugegriffen. Konkret heißt das, dass bei dynamischen Variablen immer der Wert,den die Variable bei ihrer Definition hatte, ausgegeben wird.
   Gelöst habe ich das, indem ich bei statt f-strings einfach Schlüsselwörter, wie „`_user_name`“ verwende, und diese dann immer mithilfe der Funktion `replace_vars()` kurz vor der Ausgabe in der mainloop durch den aktuellen Wert der zugehörigen Variable ersetze.

22.05

Heute habe ich zuerst Punkt 2 von gestern weitergeführt. Denn statt in all meinen Funktionen, wie beispielsweise `Show_menu()` den Output sofort zu Printen, kann ich diesen ja auch mit `return` zurückgeben. Dadurch kann ich auch diese Funktionen in `replace_vars()` verwenden, wodurch ich die Funktionen direkt mit den Output-Sätzen verbinden kann und nicht mehr woanders aufrufen muss. Ich muss in `outputs` also nur noch mit einem Schlüsselwort auf die Funktion verweisen.

Zudem habe ich mit der zweiten Phase, also „`ordering`", angefangen, indem ich entsprechende Phrasen in `outputs` hinzugefügt, und die Funktion `process_input_ordering()` erstellt habe.

05.06

Heute habe ich vor allem „`ordering`" weiter gemacht, indem ich mehr Möglichkeiten hinzugefügt habe. Darunter habe ich auch die Möglichkeit hinzugefügt etwas zu bestellen, indem man einfach nur ein gültiges Gericht, und eine Nummer schreibt. Durch viel spliten, cutten und replacen am Input-String funktioniert dies nun auch unabhängig davon ob der User „3xsalami", „3 Salami", oder „3malSalami" schreibt zuverlässig.

Dafür musste ich auch die Funktion `show_last_order()` schreiben, die auf eine zwischengespeicherte Version der letzten Bestellung zugreift, und sie aus- beziehungsweise zurückgibt

07.06

Heute habe ich viele kleine Sachen umgesetzt, repariert und neuformatiert. Darunter zählt unter anderem die Möglichkeit Phrasen auch nach dem abhaken noch einmal zu benutzen, oder die Zusammenführung der Funktionen `greetings(), ordering()` in `find_output()`, da sie nahezu identisch waren.

12.06

Heute habe ich implementiert, das man seinen Warenkorb in sofern bearbeiten kann, als das man eine bestimmte Anzahl von einem bestimmten Gericht entfernen kann.

Funktionieren tut das grundlegend ähnlich wie beim Bestellen: Die Funktion `analyse_edit_cart()` schneidet alles unnütze Weg, schaut ob eine gültige mit einem „." markierte Nummer enthalten ist, schaut ob das Schlüsselwort „lösch" oder „löschen" enthalten ist und löscht die richtige Anzahl.

14.06

Heute habe ich eine Möglichkeit eingebaut auch mit Wörtern (also zum Beispiel „zwei“) statt nur mit Zahlen die Anzahl beim Bestellen beziehungsweise Warenkorb-bearbeiten zu interagieren. Dafür habe ich mithilfe von einem LLM ein Dictionary geschrieben, was zu allen Wörtern eins bis fünfundzwanzig die passenden Zahlen beinhaltet. Nun ersetzte ich diese Wörter einfach durch die Zahlen, sodass ich am eigentlichen Algorithmus nichts wirklich ändern muss.

Zudem habe ich das Menü erweitert. Außerdem habe ich die Möglichkeit eingebaut, dass man mit dem Schlüsselwort „Hilfe“ oder „Info“ eine Erklärung bekommt, indem ich diese Erklärung in `menu` eingebaut habe.

15.06

Heute habe ich das komplette Ende geschrieben. Die Funktionsweise ist nichts Besonderes und gleich wie auch schon bei der Einleitung oder der Bestell-Phase, aber nun kann man bestellen und auch noch Trinkgeld auf den Gesamtpreis legen, indem man einfach einen Prozentsatz eingibt.

16.06

Heute habe ich nur noch die Möglichkeit eingebaut, dass man automatisch ab bestimmten am Anfang definierten Preisen einen bestimmten Rabatt bekommt. Dafür habe ich die Funktionen `calculate_sale()` und `show_calculate_sale()` geschrieben. Zudem habe ich noch dafür gesorgt, dass eine default Antwort zurück kommen kann, die an den jeweiligen Verlauf, beziehungsweise die letzte Bot-Nachricht angepasst ist, indem ich diese in `outputs` teilweise hinzugefügt habe.