BuzzwordBingo in der Kommandozeile

Yusuf Fuat Sarac, Niraj Rao, Duc Anh Hoang

STUDIENGANG WIRTSCHAFTSINFORMATIK (IBIS)
BETRIEBSSYSTEME & RECHNERNETZE

FRANKFURT UNIVERSITY OF APPLIED SCIENCE
NIEBELUNGENPLATZ 1
60318 FRANKFURT AM MAIN

Inhaltsverzeichnis

1 Einleitung

Die folgende Dokumentation zeigt sowohl die Entwicklung als auch die Implementierung eines Spiels (BuzzwordBingo), welches mit Python konzipiert wurde und vollständig in der Shell abläuft. Der Benutzer zwischen verschieden Spielmodi, Anzahl der Spieler und verschieden Feldgrößen auswählen. Im Einzelspielermodus wird das Spiel komplett innerhalb der Konsole gespielt, während im Mehrspielermodus, basierend auf der Anzahl der Spieler, entsprechende BuzzwordBingo-Karten in einer Textdatei gespeichert werden.

2 Konzept

In unserem Programm wird der Benutzer zunächst mit einer AscII - Logo herzlichst willkommen und gefragt, wie groß das Spiellayout sein solle. Hier kann man zwischen 3x3, 5x5 und 7x7 großen Feldern aussuchen. Bei den 5x5 und 7x7 Größen ist in der Mitte des Spielfeldes ein Joker verzeichnet. Der Spieler wird darauf hingewiesen, dass es nur diese drei Auswahlmöglichkeiten gibt, bei Fehleingabe wird das Spiel neugestartet.

Abbildung 1 - Willkommensbildschirm

Ist die passende Spielgröße ausgewählt, wird entschieden, ob im Einzelspielermodus oder Mehrspielermodus gespielt werden soll. Der Einzelspielermodus spielt sich komplett in der Shell ab. Ist das entsprechende Wort gefallen, muss der Spieler dieses in die Konsole eintippen, es wird anschließend zu einem „X“ umgewandelt. Hat man eine diagonale, horizontale oder vertikale Reihe an „X“, so gilt das Spiel als gewonnen, was durch eine Laufschrift signalisiert wird.

Im Mehrspielermodus werden so viele Buzzwordbingokarten in Textdateien gespeichert, wie die Anzahl der Spieler ausgewählt wurde. Diese Textdateien können anschließend im selben Ordner aufgerufen und ausgedruckt werden, in welches der Pythoncode gespeichert ist. Die Buzzwordbingokarten werden zufällig durch eine Liste von Wörtern, welche sich ebenfalls in einer Textdatei befinden, generiert.

```
Wählen sie zwischen Zwei Spiel Modi aus Tippen sie die jeweilige Zahl ein:
1: Einzelspieler
2: Mehrspieler
1
+---------------+---------------+-------------+-------------+-----------+
|       1       |       2       |      3      |      4      |     5     |
+---------------+---------------+-------------+-------------+-----------+
|     Fangen    |    Internet   |   Umsatz    |   Jetski    | Achtsam   |
|      Kiwi     | Saekularismus | Taxifahrer  | Luftschiff  |   Eule    |
| Kommunikation |   Unendlich   |      X      |  Zukunft    | Trinken   |
|    Klausur    |  Entwicklung  |   Chemie    | Unendlich   | Putzfrau  |
|    Schwach    |      Obst     |  Projekt    |  Berater    |  Schwer   |
+---------------+---------------+-------------+-------------+-----------+

Das Spiel beginnt jetzt!
```

Abbildung 2 – Einzelspielermodus 5x5 Feld

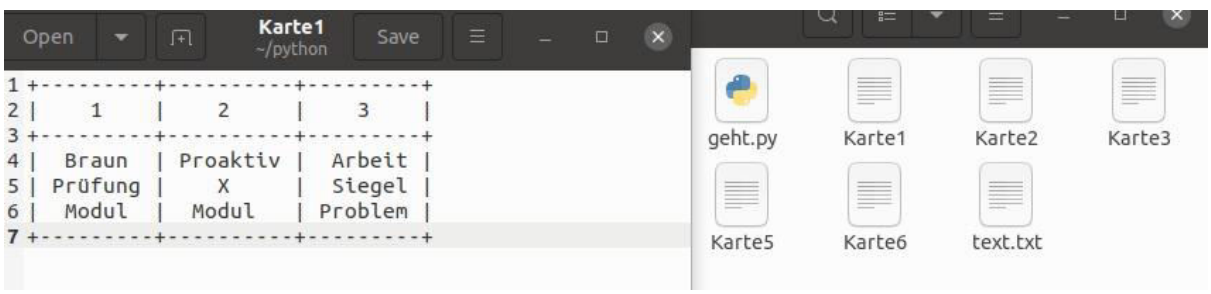

Abbildung 3 – Mehrspielermodus 3x3 Feld

Haben sie eine diagonale, horizontale oder vertikale Linie an markierten Wörtern, so gilt das Spiel als gewonnen. Ihr Sieg wird durch ein Pop-up Fenster angezeigt. Es wird ebenfalls in der Konsole durch eine lilafarbene Schrift signalisiert, dass sie das Spiel gewonnen haben.

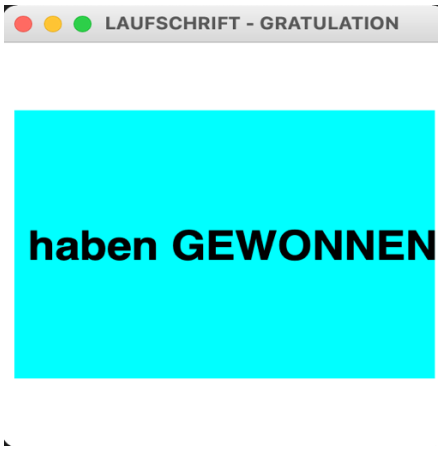

Abbildung 4 - Laufschrift

3 Implementierung

Zunächst mussten erstmal alle Bausteine zusammengesucht werden, die für den Aufbau des Spieles benötigt werden. Diese Bausteine wurden anschließend in Methoden bzw. Funktionen übersetzt. Nach diesem Prinzip wurden nach und nach Funktionen geschrieben und hinzugefügt. Der Einfachheit Haber haben wir keine Mainfunktion geschrieben, stattdessen haben wir einen Hauptteil, welcher durch Kommentare gekennzeichnet wurde. In diesem Hauptteil kamen letztlich alle Funktionen zusammen.

3.1 Funktionen

Um die Struktur des Codes besser zu verstehen, werden im Folgenden alle Methoden genauer beschrieben. Anschließend wird der Hauptteil erläutert, um die Zusammenhänge zu verdeutlichen. Die Funktionen werden von oben bis unten erklärt.

Karte_erstellen (höhe, breite, liste):

In dieser Funktion wird die BuzzwordBingokarte erstellt. Als Übergabeparameter gelten die Variablen höhe, breite und liste. Höhe und breite werden durch einen Userinput definiert, während ‚liste' die möglichen Wörter für das Spiel in eine Textdatei ausliest und diese anschließend in der Variable ‚liste' speichert.
Die erste For-Schleife hat eine range von 0 bis ‚höhe', was die Größe des Spielfeldes bestimmt. Danach wird die Karte definiert, indem sie die Funktion random.sample verwendet, um aus der ‚liste' zufällige Wörter auswählt. Diese Wörter werden aufgrund des „.sample" auch nur einmal ausgewählt. Dadurch entstehen keine Doppelungen in den Spielkarten. Die ‚breite' gibt die Anzahl der Zeilen an, für die jeweils ein Wort zufällig generiert wird.
Die zweite For-Schleife ist für den Joker in der Mitte bei den 5x5 und 7x7 Feldern verantwortlich. Die ‚Zahl' gibt an, um welche Zeile es sich handelt, anschließend wird an dieser Stelle mit dem Index gearbeitet. [0] [2] = „X" bedeutet, dass am Index [2] eine „X" geschrieben wird. Die ‚neue' Karte wird am Ende der Funktion zurückgegeben.

Karte_Ausgeben (Karte):

Im If-else Anweisungsblock werden basierend auf der ‚höhe' und ‚breite' des Spielfeldes die Spaltennummerierung in dem Importieren Modul ‚Prettytable' beschriftet. In der For-Schleife wird das „Prettytable" gefüllt mit den Worten von der vorherigen, definierten Karte (in Funktion Karte_erstellen). Anschließend wird die Karte mittels einer Printanweisung ausgegeben.

gezogenes_wort (Karte, liste):

Diese Funktion ist nur für den Einzelspielermodus ausgelegt. Hier tippt man das Wort in die Konsole ein, welches gefallen ist bzw. welches man markieren möchte. Das eingegebene Wort wird anschließend zu einem „X" verwandelt, um die Markierung anzuzeigen. „y" ist hier die Indexvariable, welche die Position für das „X" bestimmt.

Gleicht der Spielerinput (wort_eingabe) nämlich „wort.casefold()“, so wird jede Zeile durchsucht bis die elif - Methode erfüllt wird.
Casefold() flexibilisiert die Eingabe für den User, sodass bei falscher Groß -und Kleinschreibung kein Fehler angezeigt wird. Als kleines Extra haben wir noch einen Soundeffekt eingefügt, bei jedem richtigen Wort, welches eingetippt wird. Lassen sie sich überraschen.

Prüfen (Karte, höhe, breite):

In dieser Funktion werden die Gewinnmöglichkeiten überprüft. Man kann auf drei verschieden Arten gewinnen (diagonal, horizontal, vertikal). Es wird anfangs „sieg = False“ gesetzt, da die Gewinnprüfung nur Sinn macht, wenn man das Spiel noch nicht gewonnen hat.
Der horizontale Gewinn wird überprüft, indem mit der ersten For – Schleife (Z.66) jede Zeile einzeln durchgegangen wird. Abhängig von der Größe des Spielfeldes (zweite For-Schleife) wird jede Zeile nach dem „X“ gesucht. Wenn ein „X“ gefunden wird, inkrementiert der Counter um eins. Ist der Counter so hoch wie die Höhe des Spielfeldes, weiß man, dass eine ganze Zeile aus „X“ besteht.
Der vertikale Gewinn wird überprüft, indem die For – Schleifen des horizontalen Gewinns einfach vertauscht werden. Dadurch werden nach dem „X“ spaltenweise gesucht. Auch hier gewinnt man, wenn Counter == int(höhe) ist.
Beim diagonalen Gewinn wird neben dem Counter noch ein „y“ definiert. Dieses „y“ hat anfangs den Wert „0“ dh. in der If – Anweisung (Z.92) wird geprüft, ob an der Stelle [0][0] (oben links) ein X ist. Ist dies der Fall wird y um eins erhöht und die for Schleife (Z.91) läuft nochmal ab. Diesmal wird die zweite Zeile durchgegangen mit dem Index [0][1] (die nächste diagonale Stelle). Ist hier wieder ein „X“ erhöht sich y wieder um eins usw. Der Counter und y erhöhen sich immer gleichzeitig. Ist der Counter == int(höhe) so hat man den diagonalen Verlauf von links oben nach rechts unten überprüft und das Spiel ist gewonnen.
Die For – Schleife (Z.100) überprüft den diagonalen Gewinn von rechts oben nach links unten. Das „y=int(höhe))“ überprüft das „X“ als erstes oben rechts. Ist an dieser ein „X“ wird das y um eins vermindert. Die For – Schleife (Z.103) geht anschließend zur nächsten Zeile. Mit dem neuen y – Wert wird die nächste diagonale Zeile nach dem „X“ überprüft. Jedes Mal wenn das ‚y‘ um eins dekrementiert, inkrementiert der

Counter um eins. Die Gewinnbestätigung ist hier genauso wie in den anderen Gewinnprüfungen.

restart (): & restarthöhebreite():

Diese beiden Methoden fangen Fehlinputs des Users auf und starten das Spiel aufgrund dessen neu. Die Module ‚**os**' und ‚**sys**' mussten hier importiert werden, damit die Neustartmethode funktioniert. Das ‚os' - Modul wird verwendet, um den Path unserer Pythondatei zu lokalisieren, um diesen dann neuzustarten. Sys.executable stellt den vollen Pfad zum Python Intepreter auf, was danach in der eckigen Klammer beschrieben wird, während **sys.argv** den aktuellen Dateinamen überprüft.
Die Restartmethoden werden an den Stellen des Codes implementiert, an denen der User mögliche Fehlinputs eintippen kann. Diese werden im späteren Verlauf des Codes weiter erläutert.

Beenden():

Diese Methode beendet das Programm, sobald der User das Wort „exit" eintippt. Nachdem das Spiel einmal begonnen hat bzw. sobald der Spielprozess am Laufen ist, kann man jederzeit die Beenden-Methode aufrufen, um das Spiel zu beenden.

Hauptteil des Codes

Nachdem alle grundlegenden Methoden zum Aufbau des Spiels geschrieben wurden, beginnt unser Hauptteil des Codes. Zunächst wird der Spieler mit einem schönen Willkommenslogo in ASCII-zeichen begrüßt. Anschließend wird die Textdatei, in der sich die Wörter für das Spiel befinden, ausgelesen (Zeile 140-143).

Daraufhin werden jeweils in zwei separaten while – Schleifen nach der Größe des Spielfeldes und dem Spielmodus gefragt. In diesen While-Schleifen befindet sich ein try – except Block, worin die Restartmethoden implementiert wurden, um jegliche Fehleingaben des Spielers aufzufangen. Gibt der Spieler beispielsweise eine falsche Zahl oder einen Buchstaben/ Sonderzeichen ein, so wird auf den Fehler aufmerksam gemacht und das Spiel wird neugestartet. (Zeile 145 -173)

Gibt der User nun beim Spielmodus die Nummer eins an, so wird der Einzelspielermodus gestartet. Durch die Liste ‚Spielfeld' und der Methode Karte_erstellen wird innerhalb einer For-Schleife das passende Spiellayout als Karte definiert/ausgegeben.

Gibt der User beim Spielmodus die Nummer zwei an, wird hingegen der Mehrspielermodus gestartet. Es wird nach der Anzahl der Spieler gefragt, welcher durch einen Userinput bestätigt wird. Auch hier befindet sich eine While-Schleife mit einem try-except-Block, um Fehleingaben (zb. Sonderzeichen, Buchstaben bzw. alles außer Int - Werte) zu fangen (Zeile 186 – 193). In der For-Schleife werden abhängig von der Anzahl der Spieler (anzahl_spieler) entsprechend viele Karten erzeugt, mittels der Karte_erstellen Funktion. Darunter wird im if-else – Block das Prettytable für die ausgewählte Spielfeldgröße generiert. (Zeile 196 – 203)

Die nächste For-Schleife ist dafür verantwortlich, die Karten zu erzeugen und in einer Textdatei zu speichern. Dies geschieht durch ‚f.write', die Karten werden hiermit beschriften.

```
229     for x in range(0,int(anzahl_spieler)):
230         f = open("Karte"+str(x+1), 'w')#karten erzeugt
231         Karte=spielfeld[x]
232 
233         for Zahl in Karte:
234 
235             t.add_row(Karte[Zahl][0])
236         f.write(str(t))
237 
238         for Zahl in Karte:
239             for Zahl in range(0,1):
240                 t.del_row(Zahl)
241 
242             f.close
```

```
#Einzelspielermodus Gewinnbekanntgabe mit Detail(wie viele Wörter gezogen wurden)
sieg = False

for x in range(0,int(anzahl_spieler)):

        sieg=Prüfen(spielfeld[x],int(höhe),int(breite))
        words_till_win = 0
        wort_eingabe = input("Bitte geben sie das Wort ein was sie Markiert haben wollen" + "\n")
        while not sieg and wort_eingabe != "exit":
            wort_eingabe = gezogenes_wort(spielfeld[x], liste, wort_eingabe)
            words_till_win += 1
            #Solange das Spiel noch nicht gewonnen ist, wird jedes eingegebene Wort mitgezählt

            print(f"\nGeschriebenes Wort: {wort_eingabe}.")
            print(f"Anzahl der eingegeben Wörtern: {words_till_win}.\n")
            Karte_Ausgeben(spielfeld[x])

            #Gewinn wird überprüft, wenn True --> Laufschrift
            sieg=Prüfen(spielfeld[x],int(höhe),int(breite))
```

Abbildung 6

Dieser Teil des Codes überprüft bei allen Spielfeldgrößen, die Anzahl der Wörter im Einzelspielermodus, die eingetippt wurden, bis das Spiel gewonnen wird. Die While – Schleife (Z. 251) sagt aus, dass alle Wörter außer „exit“ mitgezählt werden. Da „exit“ für das Beenden des Programms zuständig ist, darf diese nicht mit in den Spielverlauf als Wort integriert werden.

###Laufschrift### Zeile 309 – 362

In diesem Abschnitt wird eine Laufschrift erstellt, sobald der Spieler gewonnen hat. Durch das Modul Tkinter öffnet sich am Ende des Spiels ein separates Fenster, in welches dem Spieler gratuliert wird. Eine Laufschrift mit dem Titel „GLÜCKWUNSCH! Sie haben gewonnen!“ wird von rechts nach links ausgegeben.
In diesem Codeblock werden Details wie die Größe des Fensters, Schriftart, Schriftfarbe, Laufschrift-Richtung, Laufschrift-Geschwindigkeit, Hintergrundfarbe etc. definiert. Weiterhin wird ein auch noch durch einen Sound signalisiert, dass man gewonnen hat. Nachdem der Sound abgeklungen ist, erscheint das Pop-up Fenster mit der Laufschrift.

4 Probleme/Ergebnis

Der Anfang war ein relativ steiniger Weg, da man sich erstmal komplett in Python einarbeiten musste. Die Vorkenntnisse in Java waren auf jeden Fall vorteilhaft, jedoch hat es trotz dessen ein wenig Zeit gekostet die Syntax von Python zu verstehen und sich anzueignen. Auch hätte es uns viel Zeit erspart, wenn wir von Anfang an alle mit dem gleichen Betriebssystem programmiert hätten. Oft haben wir erlebt, dass manche Module/Bibliotheken auf dem einen Betriebssystem bereits vorinstalliert waren und bei anderen nicht. So haben wir uns immer gefragt, warum der Code bei dem einen läuft und bei dem anderen nicht, obwohl der Code eins zu eins identisch war. Hätten wir beispielsweise alle von Beginn an auf Linux den Code geschrieben, wäre das Problem vermutlich nicht aufgetreten.
Weiterhin musste man sich auch lernen, wie Spiele im Allgemeinen programmiert werden. So haben wir uns erstmal darüber informiert, welche grundlegenden Bausteine für ein Spiel benötigt werden.
Wir hatten besonders Schwierigkeiten damit eine übersichtliche Tabelle zu erstellen, welche als Spielkarte verwendet wird. Dieses Problem haben wir mit „Prettytable“ lösen können. Fehlinputs vom User aufzufangen war anfangs auch schwer zu implementieren. Denn um das Spiel neuzustarten, mussten wir uns erst mit neuen Modulen (sys, os) vertraut machen. Da dies uns fremd war, hat es auch etwas Zeit gekostet zu verstehen, dass man die Module erst installieren musste.
Die gefallenen Wörter entsprechend im Einzelspielermodus mit der Tastatur bzw. der Maus zu markieren, war auch eine schwere Aufgabe zu bewältigen. Wir wussten anfangs nicht, wie man eine Maus bzw. Tastatureingaben (links, rechts, oben, unten) implementiert. So haben wir uns als Lösung dazu entschieden das entsprechende Wort durch die Tastatur einzutippen, welches dann markiert wird.
Die grafische Oberfläche haben wir simpel gehalten mit ASCII Zeichen. Als das Programm fertiggeschrieben war, konnten wir uns jedoch mit dem Resultat nicht ganz zufriedengeben. Obwohl das Spiel komplett in der Shell lief und auch durchaus alle Anforderungen aus unserer Sicht erfüllt wurden, wollten wir das Spiel mit der Bibliothek curses verbinden. Dadurch hätten wir eine schönere grafische Oberfläche gestalten können. Jedoch war es doch zeitintensiver als gedacht, „curses“ zu lernen, da man unseren jetzigen Code doch massiv umschreiben musste. Wir haben das

Menu prorammieren können, haben es jedoch zeitlich nicht mehr geschafft den ganzen Code umzusetzen.