Python / Galgenmännchen

1. Galgenmännchen, Galgenraten – hangman-Spiel

Wir basteln uns das bekannte Spiel Galgenmännchen in Python.
Hier kommen bereits bekannte Python-Befehle zum Einsatz:

- Liste
- while-Schleife
- for-Schleife
- in
- Schleifen abbrechen mit break

Wir benötigen wieder für unser Spiel den Zufall, daher können wir gleich am Anfang unseres Codes das random-Modul importieren

```
import random
```

2. Listen

Natürlich benötigen wir noch Wörter, aus denen unser Programm dann per Zufall auswählen kann für das Ratespiel. Bisher haben wir Listen immer der folgenden Form aufgebaut:

```
woerter = ['Wort1', 'Wort2', 'Wort3']
```

Einfacher (vor allem für später, wenn die Wörter über eine Datei geladen werden) ist die Anweisung split(). Diese erzeugt aus einem String eine Liste.

```
woerter = 'Wort1 Wort2 Wort3'.split()
```

Dieser Aufruf von split() ohne einen Parameter erledigt für uns schon unsere Aufgabe. Der Vollständigkeithalber: als Parameter kann das verwendete Trennzeichen angegeben werden. Soll Beispielsweise anhand von „@“ getrennt werden, erhalten wir bei einer E-Mail-Adresse sehr schnell die Domain.

```
mailaufgeteilt = 'mailadresse@pyhton-lernen.de'.split('@')
```

Zusätzlich kann eine Anzahl mitgegeben werden, wie groß die erzeugte Liste werden soll.

```
beispiel = 'Wort1#Wort2#Wort3'.split('#', 2)
```

Einfach mal probieren.

3. Zufallswort

Zurück zu unserem Spiel. Zum Entwickeln nutzen wir übersichtliche 5 Wörter – es geht hier mehr um das Prinzip. Die Anzahl der Wörter kann später ja beliebig erweitert werden.

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
```

Wir benötigen nun aus unserer Liste ein zufälliges Wort. Dazu bietet uns die Anweisung random.choice() die schnellste Möglichkeit.
Das zufällig ausgewählte Wort speichern wir in der Variable ratewort und lassen es zur Kontrolle ausgeben. Bei jedem Programmaufruf sollte ein anderes kommen.

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
ratewort = random.choice(woerter)
print (ratewort)
```

4. Unterstriche

Wir wollen für die einzelnen Buchstaben des Wortes nur Unterstriche ausgegeben bekommen. Zusätzlich soll ein Leerzeichen nach dem Unterstrich kommen, damit es deutlich ist, wie viele Buchstaben es sind. Dazu entwickeln wir in 2 Schritten den entsprechenden Code. Erst das es wie gewünscht ausgegeben wird und dann packen wir diesen Code in eine Funktion.
Wir durchlaufen die einzelnen Buchstaben unserer Variablen „ratewort“ und lassen zum Testen jeden Buchstaben mit Leerzeichen danach ausgeben.
für jeden Buchstaben als Bildschirmausgabe einen Unterstrich mit Leerzeichen ausgeben:

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
ratewort = random.choice(woerter)
print (ratewort)
# zum Test nur!
for buchstaben in ratewort:
    print (buchstaben, " ")
```

Als Ergebnis erhalten wir die einzelnen Buchstaben jeweils in einer neuen Zeile.

```
hangman in Python


Katze
K
a
t
z
e
```

5. Ausgabe
Von unserem Leerzeichen sehen wir nicht mehr viel. In der alten Python2-Version konnte man noch mit einem Komma nach der print-Anweisung den Umbruch verhindern. Wir wollen aber nicht mit einer alten Version arbeiten! Schaut man sich den Befehlsaufbau von print() in Python3 an, können wir der Funktion print() noch weitere Parameter mitgeben wie beispielsweise end=' ' oder in unserem Fall ein Unterstrich:

```
for buchstaben in ratewort:
    print (buchstaben, " ", end='_')
```

Als Ausgabe erhalten wir in einer Zeile dann:

```
K _a _t _z _e _
```

Wenn wir nur die Unterstiche ohne Buchstaben wollen, dann können wir auch nur unseren Parameter „missbrauchen“.

```
for buchstaben in ratewort:
    print ("", end='_ ')
```

Wir erhalten dann unsere gewünschte Ausgabe:

```
_ _ _ _ _
```

Im folgenden Schritt werden wir 2 Fliegen mit einer Klappe erschlagen.

6. Speichern von erratenen Buchstaben
Wir benötigen eine Möglichkeit, bereits erratene Buchstaben zu speichern und dann zusammen mit den nicht erratenen (sprich die Unterstriche) auszugeben.
Legen wir dazu eine weitere Liste an mit dem Namen erraten. Diese wird am Anfang unseres Codes gemacht, damit alle Variablen und Listen-Bezeichnungen sofort sichtbar sind:

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
erraten = []
ratewort = random.choice(woerter)
```


Bisher ist unsere Liste mit Leer. Also füllen wir diese vor unserer ersten Raterunde mit Unterstrichen. Dazu durchlaufen wir mit der for-Anweisung unser durch Zufall bestimmtes ratewort und werden für jeden einzelnen Buchstaben unsere Liste erraten mit einem Eintrag „_“ ergänzen:

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
erraten = []
ratewort = random.choice(woerter)
for buchstaben in ratewort:
    erraten.append('_')
```

Jetzt können wir übergehen, dass der Nutzer seine Eingaben machen kann.

7. Nutzereingabe in Schleife

Da unser Spiel auf jeden Fall über mehrere Runden läuft, benötigen wir eine Schleife. Diese haben wir bereits im Kapitel zum Chatbot (https://www.python-lernen.de/chatbot-programmieren.htm) kennen gelernt. Der Nutzer hat auch immer die Möglichkeit das Spiel vorzeitig zu beenden. Wieder brauchen wir eine Variable mit dem Namen nutzereingabe, die wir am Anfang festlegen. Und nun kommt unsere while-Schleife, bis entweder das Spiel gewonnen oder verloren ist oder der Spieler abbricht.

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
erraten = []
nutzereingabe = ""
ratewort = random.choice(woerter)
for buchstaben in ratewort:
    erraten.append('_')
while nutzereingabe != "bye":
    nutzereingabe = input("Ihr Vorschlag: ")
```

Der Nutzer kann nun unendlich lange Vorschläge machen – noch wird weder etwas kontrolliert noch ausgegeben.
Lassen wir vor der Nutzereingabe unseren bisherigen erraten Stand ausgeben (Unterstriche und erratene Buchstaben) – also vor der ersten Runde nur Unterstriche.

```
while nutzereingabe != "bye":
    for ausgabe in erraten:
        print (ausgabe, end=' ')
    print()
    nutzereingabe = input("Ihr Vorschlag: ")
```

Das print() benötigen wir, damit der Text „Ihr Vorschlag:“ in der nächsten Zeile kommt.
Bisher unsere Bildschirmausgabe:

```
hangman in Python

_ _ _ _

Ihr Vorschlag:
```

8. Kontrolle ob Buchstabe von Nutzereingabe in Ratewort vorkommt

Unser Ratewort lassen wir nun Buchstabe für Buchstabe über eine for-Schleife durchlaufen und überprüfen, ob der Spieler-Buchstabe vorkommt:

```
    nutzereingabe = input("Ihr Vorschlag: ")
    for buchstaben in ratewort:
        if buchstaben == nutzereingabe:
            print ("Treffer")
```

Jetzt kann schon getestet werden und wir erhalten bei Übereinstimmung als Bildschirmausgabe „Treffer“. Jetzt ist es wichtig, einfach auch Großschreibung zu testen bzw. was passiert, wenn mehr als 1 Buchstaben durch den Spieler eingegeben wird.

Ergebnis:
Eine Eingabe von mehr als einem Buchstaben durch den Spieler läuft ins Leere. Dann hat der Spieler Pech gehabt – damit können wir leben (zumal der Spieler auch „bye“ zum Spielabbruch eingeben können muss).
Großschreibung ist ein Problem – sowohl bei Spielereingabe wie auch beim zu erratendem Wort. Also vergleichen wir immer nur Kleinbuchstaben und wir sind aus dem Problem raus.

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
erraten = []
nutzereingabe = ""
ratewort = random.choice(woerter)
for buchstaben in ratewort:
    erraten.append('_')
while nutzereingabe != "bye":
    for ausgabe in erraten:
        print (ausgabe, end=' ')
    print()
    nutzereingabe = input("Ihr Vorschlag: ")
    for buchstaben in ratewort:
        if buchstaben.lower() == nutzereingabe.lower():
            print ("Treffer")
```

9. Richtiges Raten eines Buchstabens speichern
Zusätzlich zu der Bildschirmausgabe „Treffer“ soll der korrekte Buchstabe in unserer Liste erraten an den entsprechenden Positionen gespeichert werden.
Dazu brauchen wir noch eine Hilfsvariable „x“ um die Position des aktuellen Buchstabens in der Liste zu kennen.
Diese Hilfsvariable wird bei jedem Durchlauf hochgezählt und bei einem Treffer wird der Buchstabe in unserer Liste erraten an der entsprechenden Position eingetragen:

```
    x = 0
    for buchstaben in ratewort:
        if buchstaben.lower() == nutzereingabe.lower():
            print ("Treffer")
            erraten[x] = buchstaben
        x += 1
```

10. Spiel gewonnen?
Jetzt wollen wir kontrollieren, ob das Spiel gewonnen wurde. Dazu dürfen in unserer Liste erraten keine Unterstriche mehr vorhanden sein. Diese Kontrolle läuft bei Listen sehr einfach über die Python-Anweisung in.

```
    # Kontrolle ob gewonnen
    if '_' in erraten:
        print('Noch nicht gewonnen')
    else:
        print("gewonnen")
```

Damit das Python-Programm bei gewonnenem Spiel beendet wird, müssen wir nur unsere Schleife vorzeitig beenden. Das können wir über die Anweisung break erzwingen.

```
import random
print("hangman in Python")
woerter = 'Hund Katze Maus Haus raus'.split()
erraten = []
nutzereingabe = ""
ratewort = random.choice(woerter)
for buchstaben in ratewort:
    erraten.append('_')
while nutzereingabe != "bye":
    for ausgabe in erraten:
        print (ausgabe, end=' ')
    print()
    nutzereingabe = input("Ihr Vorschlag: ")
    x = 0
```



```
    for buchstaben in ratewort:
        if buchstaben.lower() == nutzereingabe.lower():
            print ("Treffer")
            erraten[x] = buchstaben
        x += 1
    print()
    # Kontrolle ob gewonnen
    if '_' in erraten:
        print('Noch nicht gewonnen')
    else:
        print("gewonnen")
        break
```

11. Spiel verloren?

Bisher können wir unendlich lange raten und habe nicht das Spiel verloren. Daher sollten wir die Bedingung, wann das Spiel verloren ist einbauen. Verloren ist, wenn man 7-mal einen falschen Tipp als Spiel abgibt. Diese Zahl führen wir in der Variabel fehlversuche mit. Auch diese wird am Anfang festgelegt.

Innerhalb der Kontrolle, ob gewonnen nehmen wir die Kontrolle auf, ob verloren. Die Ausgabe „Noch nicht gewonnen“ werden deaktiviert.

```
    # Kontrolle ob gewonnen
    if '_' in erraten:
        # print('Noch nicht gewonnen')
        fehlversuche -= 1
        if fehlversuche  == 0:
            print("Schade - verloren!")
            break
    else:
        print("gewonnen")
        break
```

