TECH STARTER

Python Einführung - Textbasiertes Abenteuer "Die verlorene Schatzsuche"

1. Einführung

Entwickle ein textbasiertes Abenteuerspiel namens "Die verlorene Schatzsuche", bei dem der Spieler durch verschiedene Räume navigiert, Rätsel löst und letztendlich den Schatz findet. Das Ziel dieser Übung ist es, die Vorteile von Funktionen und das Auslagern von Code in externe Dateien kennenzulernen.

Anforderungen:

1. Funktionen verwenden:
- Erstelle Funktionen für die folgenden Aufgaben:
- `begrüßung()`: Begrüßt den Spieler und erklärt die Spielregeln.
  `raum_betreten(raum)`: Beschreibt den aktuellen Raum und die verfügbaren Aktionen.

- `rätsel_lösen(rätsel)`: Präsentiert ein Rätsel und überprüft die Antwort des Spielers.

- `spiel_beenden()`: Verabschiedet sich vom Spieler und beendet das Spiel.

2. Code auslagern:

- Lagere die Funktionen `rätsel_lösen(rätsel)` und `spiel_beenden()` in eine externe Python-Datei namens `spiel_utils.py` aus.

- Importiere diese Funktionen in dein Hauptprogramm.

3. Spielstruktur:

- Das Spiel besteht aus mindestens drei verschiedenen Räumen.

- Jeder Raum enthält ein einzigartiges Rätsel, das gelöst werden muss, um weiterzukommen.

- Der Spieler kann durch Eingabe von Befehlen wie "norden", "süden", "osten", "westen" navigieren.

- Wenn der Spieler den Schatz gefunden hat, soll das Spiel eine entsprechende Nachricht ausgeben und enden.

4. Interaktive Elemente:

- Implementiere eine einfache Eingabeaufforderung, die die Aktionen des Spielers entgegennimmt.

- Verarbeite die Eingaben und führe die entsprechenden Funktionen aus.

Optionale Erweiterungen:

- Füge ein Inventarsystem hinzu, in dem der Spieler Gegenstände sammeln und verwenden kann.

- Implementiere ein einfaches Punktesystem, das die Leistung des Spielers bewertet.

- Erstelle zusätzliche Räume und Rätsel, um das Spiel umfangreicher zu gestalten.

Hinweise:

- Achte auf eine saubere und übersichtliche Code-Struktur.

- Verwende aussagekräftige Funktions- und Variablennamen.

- Kommentiere deinen Code, um die Funktionsweise zu erklären.

- Teste dein Spiel ausgiebig, um sicherzustellen, dass alle Funktionen korrekt arbeiten.

Ziel der Übung:

Durch diese Aufgabe lernst du:

- Wie Funktionen dazu beitragen, deinen Code übersichtlicher und wiederverwendbar zu machen.

- Wie du Code in externe Dateien auslagern und in deinem Hauptprogramm verwenden kannst.

- Wie du ein interaktives Programm entwickelst, das Benutzereingaben verarbeitet.

Viel Spaß beim Programmieren und auf zur Schatzsuche!

2. Feature 1: Funktion zum Hinzufügen von Aufgaben

Erstelle einen Feature Branch „feature/add-task-function“ und erweitere dein Python-Skript um die Funktion `add_task`.

1. Nutze die `input`-Funktion, um den User nach einer Büroaufgabe zu fragen, z.B.: „Bitte gib eine Aufgabe ein, die in deiner Aufgabenliste hinzugefügt werden soll“.
2. Speichere die Eingabe in der Variablen `task`.
3. Füge die Aufgabe zur Liste hinzu.
4. Gib eine Meldung aus, dass die Aufgabe zur Liste hinzugefügt wurde.

Tipp: Du kannst auch ein Fälligkeitsdatum (z.B. mit `input`) abfragen und dieses der Aufgabe hinzufügen.

3. Feature 2: Funktion zum Anzeigen der Aufgabenliste

Erstelle einen Feature Branch „feature/add-show-tasklist-function“ und erweitere dein Python-Skript um die Funktion `show_tasklist`:

1. Prüfe, ob die Liste leer ist. Wenn ja, gib den Text „Deine Aufgabenliste ist leer“ aus.
2. Wenn die Liste nicht leer ist, drucke alle Aufgaben mit einer `for`-Schleife.
3. Gib auch das Fälligkeitsdatum (falls vorhanden) mit aus.

Tipp: Du kannst Aufgaben farblich markieren, z.B. alle Aufgaben, die innerhalb von 2 Tagen fällig sind, in Rot.

4. Feature 3: Hauptprogramm erstellen

Erstelle einen Feature Branch „feature/add-main-function“ und erweitere dein Python-Skript um die Funktion `main`:

1. Erstelle eine `while`-Schleife, die den User fragt, ob er eine Aufgabe hinzufügen, die Liste anzeigen oder das Programm beenden möchte.
2. Nutze eine `if/elif`-Bedingung, um die Eingabe des Users zu verarbeiten.

Tipp: Füge eine Option zum Entfernen von Aufgaben hinzu, um erledigte Aufgaben von der Liste zu streichen.

5. Feature 4: Programm automatisch starten

Füge am Ende deines Skripts folgenden Code ein, um das Programm automatisch zu starten, wenn es ausgeführt wird:

```
if __name__ == "__main__":
    main()
```

6. Erweiterungen (optional)

1. **Priorisierung von Aufgaben:** Füge eine Prioritätsstufe (hoch, mittel, niedrig) hinzu und sortiere die Liste nach Priorität.
2. **Exportfunktion:** Implementiere eine Funktion, um die Liste in eine CSV-Datei zu exportieren.
3. **Benachrichtigungen:** Implementiere eine einfache Benachrichtigung, wenn eine Aufgabe fällig ist.

7. Debugging-Tipps

Wenn Fehler auftreten, nutze `print`-Anweisungen, um Variablen oder Ergebnisse zu überprüfen. Zum Beispiel könntest du nach jeder User-Eingabe die Variable ausgeben, um sicherzustellen, dass der Wert korrekt ist.

```
print("Funktion wird aufgerufen")
```