Dokumentation FAE

Zahlenkonverter

Inhalt

1. Einleitung

Ziel des Projekts

Der Zahlenkonverter ist ein Programm, in dem der Nutzer eine eingegebene Zahl in verschiedene Zahlentypen umwandeln kann. Die Umwandlungen können von Dezimal zu Binär oder Hexadezimal, von Binär zu Dezimal oder Hexadezimal oder von Hexadezimal zu Dezimal oder Binär stattfinden.

Aufgabenstellung

Das Programm soll erst ausgeführt werden, wenn ein Anmeldevorgang mittels Datenbankabgleich erfolgreich ausgeführt wurde. Im Anschluss soll der Benutzer eine Umwandlungsart, sowie eine Zahl eingeben, welche dann vom Programm auf Fehler überprüft werden. Entweder wird eine erneute Eingabe verlangt oder die Umwandlung wird bei korrekter Eingabe durchgeführt und das Ergebnis ausgegeben.
Das Programm soll so lange laufen, bis der Nutzer es entweder durch eine Eingabe bei der Abfrage beendet oder durch das Drücken einer festgelegten Taste beendet wird. Vor dem Beenden des Programms erscheint immer eine Sicherheitsabfrage.
Das Programm soll optional auf einem Raspberry Pi installiert werden und dort Zugriff auf eine Datenbank haben, in der die Anmeldedaten der User gespeichert und abgeglichen werden.

2. Planung

Das Programm ist modular aufgebaut und in mehrere Dateien unterteilt, um Aufgaben klar zu trennen und die Wartung zu erleichtern:

Übersicht über die Programmstruktur

main.py:

Zentrale Steuerungseinheit, die Funktionen aus den Modulen importiert und den Ablauf orchestriert (z. B. Anmeldung, Konvertierung, Programmsteuerung).

login.py:

Verantwortlich für Benutzeranmeldung und -registrierung.
Nutzt Funktionen aus *authentifizierung.py* zur Verifizierung oder Erstellung neuer Benutzer.

authentifizierung.py:

Stellt die Authentifizierungslogik bereit (*db_connect()*, *einloggen()*, *registrieren()*).

zahlenkonverter.py:

Beinhaltet die Logik für Konvertierungsarten, Zahleneingabe und die eigentliche Umrechnung (z. B. *dez_zu_hex()* oder *bin_zu_dez()*).

beenden.py:

Enthält Funktionen zur Programmbedienung (*weitermachen()* und *sicherheitsabfrage()*).

Besonderheiten

if __name__ == "__main__":: Jede Datei enthält dieses Statement, um sicherzustellen, dass Funktionen nur in *main.py* ausgeführt werden, nicht beim Import.

Programmablauf

Start in *main.py*: Begrüßung und Anmeldung.
Anmeldung (*login.py*): Einloggen oder Registrierung.
Konvertierung (*zahlenkonverter.py*): Auswahl der Konvertierungsart, Zahleneingabe und Ergebnis.
Fortführung oder Beenden (*beenden.py*): Benutzer entscheidet über das Weiterführen oder Beenden des Programms.

3. Umsetzung

Das Programm wurde in Pycharm geschrieben und wird über XAMPP PHP myAdmin mit einer Datenbank verbunden.
Um das Programm auszuführen, ist die main.py Datei über das Terminal zu öffnen oder aufzurufen.

Beschreibung wichtiger Funktionen und Zwecke:

Die Beschreibungen sind nach den Dateinamen gegliedert, in denen sie vorkommen.

main.py

Die Datei *main.py* dient als zentrale Steuerungseinheit für das gesamte Programm. Sie verbindet die verschiedenen Module (login, zahlenkonverter und beenden) und sorgt für die

korrekte Ausführung der Funktionen in einer logischen Reihenfolge. Die Datei enthält die Hauptfunktion *main()*, die als Einstiegspunkt für das Programm dient und den gesamten Workflow, von der Anmeldung bis zur Zahleneingabe und -umwandlung, orchestriert.

Struktur und Funktionsweise

Importierte Module

login: Beinhaltet die Funktionen zur Benutzeranmeldung und Registrierung (aus der Datei *login.py*).
zahlenkonverter: Enthält die Logik zur Konvertierung von Zahlen (aus der Datei *zahlenkonverter.py*).
beenden: Stellt Sicherheitsmechanismen zur Verfügung, z. B. die Funktionen *sicherheitsabfrage()* und *weitermachen()*.

`Funktion main()`

Die Hauptfunktion, die alle Programmteile in der richtigen Reihenfolge ausführt.

Ablauf

Begrüßung:
Zu Beginn wird eine Begrüßungsnachricht angezeigt:
********* SUPER DUPER ZAHLENKONVERTER *********
Benutzeranmeldung:
Aufruf der Funktion *anmeldung()* aus dem Modul login, um den Benutzer zu authentifizieren.
Konvertierungsprozess in einer Schleife:
Konvertierungsauswahl (*konvert_wahl()*): Der Benutzer wählt die Art der Konvertierung (z. B. Dezimal zu Binär).
Zahleneingabe (*zahlen_eingabe()*): Der Benutzer gibt die Zahl ein, die konvertiert werden soll.
Umwandlung (*konvertierung()*): Die eingegebene Zahl wird in das gewünschte Format konvertiert und das Ergebnis wird ausgegeben.
Fortsetzen oder Beenden (weitermachen()): Nach der Umwandlung wird der Benutzer gefragt, ob er eine weitere Zahl konvertieren möchte. Bei Eingabe von 'n' wird die Schleife beendet.
Fehlerbehandlung:
Sollte der Benutzer während des Programms Strg + C (KeyboardInterrupt) drücken, wird die Funktion *sicherheitsabfrage()* aus dem Modul beenden aufgerufen, um den abrupten Abbruch zu verhindern.

Typische Ausgabe

Begrüßung:

```
********* SUPER DUPER ZAHLENKONVERTER *********
```

Ablauf des Programms:

```
Anmeldung:
Möchtest du dich
a) mit vorhandenem User Anmelden
b) einen neuen User registrieren
>>>>> a
Einloggen erfolgreich.
```

Konvertierung:

```
Welche Konvertierung willst du vornehmen?
a) Dezimal in Hexadezimal
b) Dezimal in Binär
c) Hexadezimal in Dezimal
...
>>>>> a
Gib eine Zahl ein:
>>>>> 255
Das Ergebnis der Umwandlung ist:
>>>>> ff
```

Fortsetzen oder Beenden:

```
Möchtest du eine weitere Zahl eingeben? (j/n):
>>>>> n
Willst du das Programm wirklich beenden? (j / n)
>>>>> j
Das Programm wird beendet. Tschüss!
```

Besonderheiten und Sicherheitsmechanismen

Sicherheitsabfrage bei Programmabbruch:

Falls der Benutzer das Programm unerwartet abbrechen möchte (durch Strg + C), wird die Funktion *sicherheitsabfrage()* aufgerufen, um dies zu bestätigen.

Fehlerbehandlung bei ungültigen Eingaben:

Alle Benutzereingaben werden auf Gültigkeit geprüft, um Fehler oder unerwartetes Verhalten zu vermeiden.

Modularer Aufbau:

Die Datei vereint die Funktionen aus mehreren Modulen, was die Strukturierung und Wartung des Codes erleichtert.

login.py

Die Datei *login.py* dient als Einstiegspunkt für Benutzer, die sich authentifizieren möchten. Sie bietet zwei Hauptoptionen: Anmeldung mit einem bestehenden Benutzerkonto oder die Registrierung eines neuen Benutzers. Diese Datei greift auf Funktionen aus den Modulen authentifizierung und beenden zurück und stellt sicher, dass die Benutzerinteraktion reibungslos abläuft.

Struktur und Funktionsweise

Importierte Module

authentifizierung: Dieses Modul enthält die Logik für das Einloggen (*einloggen()*) und Registrieren (*registrieren()*), sowie die Funktion *db_connect()* für den Datenbankzugriff.
beenden: Dieses Modul wird für die Handhabung eines sicheren Programmabbruchs verwendet. Es enthält unter anderem die Funktion *sicherheitsabfrage()*.

Funktion anmeldung()

Die Funktion *anmeldung()* ist der Kern der Datei und bietet dem Benutzer eine Auswahl zwischen:

1. **Anmeldung** (*anmeldewahl = 'a'*): Führt den Benutzer durch den Login-Prozess mithilfe der Funktion *einloggen()* aus dem Modul authentifizierung.
2. **Registrierung** (*anmeldewahl = 'b'*): Ermöglicht es dem Benutzer, ein neues Konto zu erstellen, indem die Funktion *registrieren()* aus dem Modul authentifizierung aufgerufen wird.

Ablauf der Funktion:

1. **Benutzereingabe:** Es wird eine Auswahl zwischen Anmeldung und Registrierung abgefragt. Die Eingabe wird in Kleinbuchstaben (*lower()*) umgewandelt, um Eingabefehler zu minimieren.
2. **Datenbankverbindung:** Vor jedem Aufruf von *einloggen()* oder *registrieren()* wird die Datenbankverbindung mithilfe von *db_connect()* hergestellt.
3. **Fehlerbehandlung:** Falls der Benutzer die Tastenkombination Strg + C (KeyboardInterrupt) drückt, wird eine Sicherheitsabfrage ausgelöst, um den unerwarteten Abbruch des Programms zu verhindern.

Typische Ausgabe

```
Möchtest du dich
a) mit vorhandenem User Anmelden
b) einen neuen User registrieren
>>>>> a
```

authentifizierung.py

Die Datei *authentifizierung.py* stellt eine Verbindung zur angelegten Datenbank her und ist für den Einlogg- oder Registrierungsvorgang vorhanden. Dafür werden die Funktionen db_connect, einloggen und registrieren verwendet.

Struktur und Funktionsweise

Importierte Module

mysql.connector: Bibliothek zur Verbindung und Interaktion mit einer MySQL-Datenbank.
string: Wird verwendet, um Sonderzeichen für Passwortkriterien zu prüfen.

Funktionen

1. db_connect()

Stellt die Verbindung zur MySQL-Datenbank zahlenkonverter her.
Verwendete Parameter:
host="localhost": IP-Adresse des Hosts (aktuell zu einer lokalen Datenbank).
user="root", password="": Benutzername und Passwort für den Datenbankzugriff.
database="zahlenkonverter": Name der Datenbank.
Rückgabe: Eine aktive Verbindung zur Datenbank.

2. einloggen()

Ermöglicht Benutzern, sich mit einem bestehenden Konto anzumelden.
Benutzer gibt Username und Passwort ein.
Die Funktion überprüft die Eingabe über einen SQL-Query (*SELECT password FROM usersneu WHERE username = %s*).
Bei korrekter Eingabe wird der Benutzer erfolgreich angemeldet, ansonsten erfolgt eine Fehlermeldung.

3. registrieren()

Ermöglicht die Erstellung eines neuen Benutzerkontos.

Benutzer wählt einen Benutzernamen. Die Funktion prüft über einen SQL-Query (*SELECT username FROM usersneu WHERE username = %s*), ob der Name verfügbar ist.
Benutzer erstellt ein Passwort, das folgenden Kriterien entsprechen muss:
Mindestens 8 Zeichen lang. Enthält mindestens eine Zahl. Enthält mindestens ein Sonderzeichen.
Das Passwort wird bestätigt, bevor Benutzername und Passwort in der Datenbank gespeichert werden (*INSERT INTO usersneu (username, password) VALUES (%s, %s)*).
Fehlerbehandlung:
Doppelte Benutzernamen lösen eine Fehlermeldung aus.
Ungültige Passwörter werden abgelehnt.

Typische Ausgabe

Login-Versuch:

```
Username: user123
Passwort: geheim123
Einloggen erfolgreich.
Du besitzt nun das unglaubliche Privileg, Zahlen umzurechnen.
```

Registrierung:

```
Usernamen erstellen: neueruser
Passwort erstellen: Mein@Passwort1
Passwort bestätigen: Mein@Passwort1
Registrierung erfolgreich.
```

zahlenkonverter.py

Die Datei konvertierung.py bildet den Kern des Programms, indem sie die Auswahl der gewünschten Zahlenumwandlung, die Eingabe und Validierung der Zahlen sowie die eigentliche Konvertierung implementiert. Sie enthält alle Mechanismen, um zwischen verschiedenen Zahlensystemen (Dezimal, Hexadezimal, Binär) zu wechseln.

Struktur und Funktion

Importierte Module

main: Enthält die Funktion sicherheitsabfrage, die für die sichere Beendigung des Programms verwendet wird.
Funktionen

1. konvert_wahl()

Fragt den Benutzer, welche Konvertierung durchgeführt werden soll, und gibt die Auswahl in einer standardisierten Form zurück.

Ablauf:
Zeigt eine Liste der verfügbaren Konvertierungen an:

a) Dezimal in Hexadezimal
b) Dezimal in Binär
c) Hexadezimal in Dezimal
d) Hexadezimal in Binär
e) Binär in Dezimal
f) Binär in Hexadezimal

Nimmt die Benutzereingabe entgegen und überprüft, ob die Eingabe gültig ist.
Bei gültiger Eingabe:
Gibt die gewählte Umwandlungsoption (z. B. dez_hex) zurück.
Bei ungültiger Eingabe:
Gibt eine Fehlermeldung aus und wiederholt die Eingabeaufforderung.

2. zahlen_eingabe(umwandlung)

Nimmt die Zahl entgegen, die umgewandelt werden soll, und validiert die Eingabe basierend auf dem gewählten Umwandlungstyp.
Ablauf:
Dezimal in Hexadezimal/Binär (dez_hex, dez_bin):
Akzeptiert nur positive ganze Zahlen.
Hexadezimal in Dezimal/Binär (hex_dez, hex_bin):
Akzeptiert nur Zeichen, die einer Hexadezimalzahl entsprechen (0-9, A-F).
Binär in Dezimal/Hexadezimal (bin_dez, bin_hex):
Akzeptiert nur Zeichen, die einer Binärzahl entsprechen (0 oder 1).
Gibt die validierte Zahl zurück oder fordert den Benutzer bei ungültiger Eingabe zu einer erneuten Eingabe auf.

3. Umwandlungsfunktionen

Führen die spezifischen Zahlenkonvertierungen durch:
dez_zu_hex(zahl_eingabe): Konvertiert eine Dezimalzahl in Hexadezimal.
dez_zu_bin(zahl_eingabe): Konvertiert eine Dezimalzahl in Binär.
hex_zu_dez(zahl_eingabe): Konvertiert eine Hexadezimalzahl in Dezimal.
hex_zu_bin(zahl_eingabe): Konvertiert eine Hexadezimalzahl in Binär.
bin_zu_dez(zahl_eingabe): Konvertiert eine Binärzahl in Dezimal.
bin_zu_hex(zahl_eingabe): Konvertiert eine Binärzahl in Hexadezimal.
Diese Funktionen sind in einem Dictionary (*umwandlungs_funktionen*) organisiert, sodass sie basierend auf der Auswahl des Benutzers dynamisch aufgerufen werden können.

4. konvertierung(umwandlung, zahl)

Führt die gewählte Umwandlung durch und gibt das Ergebnis aus.

Ablauf:

Ruft die passende Funktion aus *umwandlungs_funktionen* auf, basierend auf der Auswahl (*umwandlung*).

Gibt das Ergebnis der Konvertierung aus.

Bei unbekannten Umwandlungstypen wird eine Fehlermeldung ausgegeben.

Typische Ausgabe

Auswahl der Konvertierung:

```
Welche Konvertierung willst du vornehmen?
a) Dezimal in Hexadezimal
b) Dezimal in Binär
c) Hexadezimal in Dezimal
d) Hexadezimal in Binär
e) Binär in Dezimal
f) Binär in Hexadezimal
>>>>> b
```

Umwandlung von Dezimal in Binär.

```
Zahleneingabe (gültig):
Gib eine Zahl ein:
>>>>> 42
Umwandlung der Zahl 42
```

Konvertierungsergebnis:

```
Das Ergebnis der Umwandlung ist:
>>>>> 101010
```

Ungültige Eingabe:

```
Gib eine Zahl ein:
>>>>> ABC
Gib eine ganze, positive Zahl ein.
```

beenden.py

Die Datei beenden.py enthält Mechanismen, um das Programm entweder fortzuführen oder sicher zu beenden. Sie stellt sicher, dass der Benutzer bewusst entscheidet, ob er das Programm beendet oder weitere Zahlen eingeben möchte. Dabei werden Benutzerfehler durch Sicherheitsabfragen abgefangen, um unabsichtliche Programmabbrüche zu vermeiden.

Struktur und Funktionsweise

Importierte Module

sys: Wird verwendet, um das Programm direkt zu beenden (*sys.exit()*).

Funktionen

1. weitermachen()

Diese Funktion fragt den Benutzer, ob er eine weitere Zahl eingeben möchte oder das Programm beenden will.
Ablauf:
Benutzer gibt j (*weitermachen*) oder n (*beenden*) ein.
Bei Eingabe von j:
Eine Bestätigung wird ausgegeben: "Weiter geht die wilde Sause!"
Die Funktion gibt True zurück, um die Fortsetzung des Programms zu signalisieren.
Bei Eingabe von n:
Die Funktion ruft *sicherheitsabfrage()* auf, um zu bestätigen, ob das Programm tatsächlich beendet werden soll.
Gibt False zurück, wenn die Sicherheitsabfrage erfolgreich ist.
Bei ungültiger Eingabe:
Es wird eine Fehlermeldung ausgegeben und das Programm beendet (*sys.exit(0)*).

2. sicherheitsabfrage()

Diese Funktion stellt sicher, dass der Benutzer das Programm nicht versehentlich beendet.
Ablauf:
Benutzer gibt j (beenden) oder n (fortsetzen) ein.
Bei Eingabe von j:
Eine Abschiedsnachricht wird ausgegeben: "Das Programm wird beendet. Tschüss!"
Das Programm wird beendet (*sys.exit(0)*).
Bei Eingabe von n:
Eine Bestätigung wird ausgegeben: "Toll, einfach toll, dass du dich für den super duper Zahlenkonverter entscheidest!"
Die Funktion kehrt zur Hauptausführung zurück.
Bei ungültiger Eingabe:
Eine Fehlermeldung wird ausgegeben, und der Benutzer wird erneut zur Eingabe aufgefordert.

Typische Ausgabe

Fortsetzung des Programms:

```
Möchtest du eine weitere Zahl eingeben? (j/n):
>>>>> j
Weiter geht die wilde Sause!
```

Programm beenden:

```
Möchtest du eine weitere Zahl eingeben? (j/n):
>>>>> n
Willst du das Programm wirklich beenden? (j / n)
>>>>> j
Das Programm wird beendet. Tschüss!
```

Ungültige Eingabe:

```
Möchtest du eine weitere Zahl eingeben? (j/n):
>>>>> x
Ungültige Eingabe! Das Programm wird beendet.
```

4. Testung des Programms

Das Programm wurde während der gesamten Entwicklung regelmäßig getestet, sowohl in der PyCharm-Konsole als auch im Windows-Terminal. Dabei wurden insbesondere die Benutzerinteraktionen und die Konvertierungsfunktionen überprüft.
Es wurde darauf geachtet, dass:

- Alle Konvertierungsfunktionen korrekt arbeiten und die richtigen Ergebnisse liefern.
- Die Eingabe von Benutzern fehlerfrei verarbeitet wird (z. B. falsche Zahleneingabe oder ungültige Passwörter).
- Das Programm auf ungültige Benutzereingaben reagiert und entsprechende Fehlermeldungen ausgibt.
- Das sichere Beenden und Fortfahren des Programms bei Unterbrechung durch den Benutzer (z. B. durch Ctrl+C) gewährleistet ist.
- Durch diese kontinuierliche Überprüfung wurde sichergestellt, dass alle Funktionen zuverlässig und stabil laufen.

Aufgetretener Fehler:

- Kurz vor Abgabe des Programmes trat ein wiederkehrender Fehler bei Beendigung des Programmes auf, der bis zur Abgabe leider nicht behoben werden konnte. Wält man bei der sicherheitsabfrage ‘n’ als Antwort aus (Programm soll nicht beendet werden), beendet das Programm trotzdem. Dieser Fehler muss noch behoben werden.

5. Fazit

Wie lief das Projekt und was hat gut funktioniert?

Zu Beginn des Projekts war ich überwältigt, da mir sowohl das konkrete Vorgehen als auch einige technische Aspekte noch unklar waren. Es fiel mir schwer, die Aufgabenstellung vollständig zu verstehen, und es gab Momente, in denen ich an meinem Wissen zweifelte. Der Schlüssel zum Erfolg war, Schritt für Schritt an das Projekt heranzugehen und mit meiner Teamkollegin die Arbeit zu teilen. Das hat mir geholfen, nicht zu verzweifeln und mir eine klare Struktur zu erarbeiten.

Als das Programm funktionierte, war die Zusammenarbeit mit meiner Kollegin besonders hilfreich, um die einzelnen Teile zusammenzuführen. Rückgriff auf Online-Ressourcen und KI (wie ChatGPT) hat mich bei der Lösung von Problemen unterstützt, ohne jedoch den Code einfach zu übernehmen – ich habe stets darauf geachtet, die Konzepte zu verstehen und Lösungen eigenständig umzusetzen.

Was hätte besser gemacht werden können?

Im Rückblick hätte ich zu Beginn mehr Zeit in die Planung und das Konzept des Programms investieren können. Das hätte geholfen, unnötige Anpassungen und Neuschreibungen zu vermeiden. Einige meiner anfänglichen Ansätze mussten aufgrund technischer Hürden oder falscher Annahmen verworfen werden. Trotzdem sehe ich das als Teil meines Lernprozesses.
Des Weiteren hat die Ausführung des Programms mit Datenbankintegration auf dem Raspberry Pie nicht fehlerfrei funktioniert. Wir haben die main.py Datei zwar zum Laufen gebracht, doch trotz korrekter Eingabe des Datenbank-Servers, Nutzers und Passwort konnte eine Anmeldung nicht ausgeführt werden. Die Fehlermeldung deutet darauf hin, dass es Probleme bei der Verbindung zur Datenbank und der Nutzerrechte gibt.

Wie könnte man das Programm erweitern?

- GUI: Eine grafische Benutzeroberfläche würde das Programm benutzerfreundlicher und ansprechender gestalten.
- Benutzerbereich: Funktionen zum Bearbeiten und Löschen von Benutzerprofilen sowie zum Speichern von Umrechnungen wären nützlich.
- Mehrsprachigkeit: Eine mehrsprachige Benutzeroberfläche könnte internationale Nutzer ansprechen.