Leap Fehlermeldungen

**Lexikalischer Fehler**

**LEX001 – Unerwartetes Zeichen**

❌ **Fehler:** Unerwartetes Zeichen '@' in Zeile 4.  
💡 **Erklärung:** Das Zeichen '@' ist in Leap nicht erlaubt.  
🛠️ **Lösung:** Entferne das Zeichen oder ersetze es durch ein gültiges Symbol.

**LEX002 – Unvollständige Zeichenkette**

❌ **Fehler:** Zeichenkette wurde nicht geschlossen.  
💡 **Erklärung:** Ein Text beginnt mit " oder ', wurde aber nicht korrekt beendet.  
🛠️ **Lösung:** Füge am Ende der Zeichenkette das fehlende Anführungszeichen hinzu.

**LEX003 – Ungültiger Escape-Code**

❌ **Fehler:** Ungültige Escape-Sequenz „\q“ in Zeichenkette.  
💡 **Erklärung:** In Zeichenketten dürfen nur bestimmte Escape-Sequenzen wie \n, \t, \" verwendet werden.  
🛠️ **Lösung:** Ersetze „\q“ durch eine gültige Sequenz oder entferne den Backslash.

**LEX004 – Ungültige Zahl**

❌ **Fehler:** Zahl kann nicht interpretiert werden (12.3.4).  
💡 **Erklärung:** Zahlen dürfen nur einen Dezimalpunkt enthalten.  
🛠️ **Lösung:** Korrigiere die Zahl, z. B. 12.34.

**LEX005 – Zu langer Identifier**

⚠️ **Fehler:** Der Variablenname ist zu lang.  
💡 **Erklärung:** Variablennamen dürfen maximal 128 Zeichen lang sein.  
🛠️ **Lösung:** Verwende einen kürzeren und aussagekräftigen Namen.

**LEX006 – Kommentar nicht geschlossen**

❌ **Fehler:** Mehrzeiliger Kommentar wurde nicht beendet.  
💡 **Erklärung:** Ein Kommentar, der mit /\* beginnt, muss mit \*/ enden.  
🛠️ **Lösung:** Füge das fehlende \*/ am Ende des Kommentars hinzu.

**Syntaxische Fehler**

**SYN001 – Erwartetes Token fehlt**

❌ **Fehler:** Fehlende schließende Klammer ) in Zeile 6.  
💡 **Erklärung:** Jede öffnende Klammer ( benötigt eine passende schließende ).  
🛠️ **Lösung:** Überprüfe deine Klammern und füge die fehlende hinzu.

**SYN002 – Unerwartetes Token**

❌ **Fehler:** Unerwartetes Schlüsselwort 'else' nach Anweisung.  
💡 **Erklärung:** 'else' darf nur direkt nach einer if-Bedingung stehen.  
🛠️ **Lösung:** Entferne das 'else' oder überprüfe die vorherige if-Anweisung.

**SYN003 – Block nicht geschlossen**

❌ **Fehler:** Block wurde nicht beendet (fehlendes end).  
💡 **Erklärung:** In Leap müssen alle Blöcke mit begin ... end abgeschlossen werden.  
🛠️ **Lösung:** Füge am Ende des Blocks end hinzu.

**SYN004 – Ungültige Anweisung**

❌ **Fehler:** Ungültige Zuweisung – links muss eine Variable stehen.  
💡 **Erklärung:** Du kannst einem Wert keinen neuen Wert zuweisen (5 = x ist ungültig).  
🛠️ **Lösung:** Schreibe x = 5 statt 5 = x.

**SYN005 – Falsche Reihenfolge von Tokens**

❌ **Fehler:** Unerwartete Reihenfolge von Symbolen in Zeile 8.  
💡 **Erklärung:** Der Ausdruck folgt keiner gültigen Syntax.  
🛠️ **Lösung:** Überprüfe die Reihenfolge – z. B. sollte var name = x heißen, nicht var = x name.

**SYN006 – Doppelte Deklaration im selben Block**

❌ **Fehler:** Variable 'x' wurde mehrfach deklariert.  
💡 **Erklärung:** Innerhalb eines Blocks darf eine Variable nur einmal angelegt werden.  
🛠️ **Lösung:** Entferne die zweite Deklaration oder verwende einen anderen Namen.

**SYN007 – Fehlendes Trennzeichen**

❌ **Fehler:** Es fehlt ein Komma zwischen Argumenten.  
💡 **Erklärung:** Leap erwartet zwischen mehreren Werten oder Parametern ein Komma.  
🛠️ **Lösung:** Schreibe a, b statt a b.

**Semantische Fehler**

**SEM001 – Variable nicht definiert**

❌ **Fehler:** Variable 'x' wurde verwendet, aber nie deklariert.  
💡 **Erklärung:** Du versuchst, eine Variable zu verwenden, die nicht existiert.  
🛠️ **Lösung:** Deklariere die Variable zuerst mit let x = ....

**SEM002 – Mehrfache Definition**

❌ **Fehler:** 'x' wurde bereits definiert.  
💡 **Erklärung:** Gleicher Variablenname im selben Bereich ist nicht erlaubt.  
🛠️ **Lösung:** Verwende einen anderen Namen oder lösche die doppelte Definition.

**SEM003 – Typkonflikt**

❌ **Fehler:** Typkonflikt: Du versuchst, Text und Zahl zu kombinieren.  
💡 **Erklärung:** "Text" + 5 ist nicht erlaubt, da Typen unterschiedlich sind.  
🛠️ **Lösung:** Konvertiere Werte oder überprüfe den Ausdruck.

**SEM004 – Funktion nicht definiert**

❌ **Fehler:** Funktion 'doSomething' existiert nicht.  
💡 **Erklärung:** Du rufst eine Funktion auf, die nie erstellt wurde.  
🛠️ **Lösung:** Definiere die Funktion mit func doSomething() ... end.

**SEM005 – Falsche Anzahl an Parametern**

❌ **Fehler:** Funktion 'sum' erwartet 2 Argumente, aber 1 wurde übergeben.  
💡 **Erklärung:** Beim Funktionsaufruf fehlt oder überschreitet ein Argument.  
🛠️ **Lösung:** Überprüfe die Funktionsdefinition und passe den Aufruf an.

**SEM006 – Ungültiger Rückgabewert**

❌ **Fehler:** Funktion gibt keinen Wert zurück, obwohl einer erwartet wird.  
💡 **Erklärung:** Eine Funktion mit Rückgabewert muss immer return verwenden.  
🛠️ **Lösung:** Füge am Ende return <wert> hinzu.

**SEM007 – Zugriff auf ungültigen Index**

❌ **Fehler:** Index 10 liegt außerhalb der Liste mit 5 Elementen.  
💡 **Erklärung:** Du greifst auf ein Element zu, das nicht existiert.  
🛠️ **Lösung:** Überprüfe die Listenlänge oder nutze Bedingungen zur Kontrolle.

**SEM008 – Ungültiger Datentyp für Operation**

❌ **Fehler:** Bedingung erwartet einen booleschen Wert, erhielt aber einen Text.  
💡 **Erklärung:** In if-Bedingungen müssen logische Werte stehen (true oder false).  
🛠️ **Lösung:** Vergleiche den Wert mit etwas, z. B. if (text == "ok").

**SEM009 – Verwendung vor Initialisierung**

❌ **Fehler:** Variable 'a' wird verwendet, bevor ihr ein Wert zugewiesen wurde.  
💡 **Erklärung:** Variablen sind zu Beginn leer.  
🛠️ **Lösung:** Weise der Variable zuerst einen Wert zu.

**SEM010 – Konstantenänderung**

❌ **Fehler:** Konstante 'PI' kann nicht verändert werden.  
💡 **Erklärung:** const-Werte sind fest und dürfen nicht überschrieben werden.  
🛠️ **Lösung:** Verwende eine neue Variable, wenn du den Wert ändern willst.

**Runtime Error**

**RUN001 – Division durch Null**

❌ **Fehler:** Division durch Null ist nicht erlaubt.  
💡 **Erklärung:** Mathematisch undefiniert.  
🛠️ **Lösung:** Prüfe, ob der Divisor != 0 ist, bevor du teilst.

**RUN002 – Stack Overflow**

❌ **Fehler:** Zu tiefe Rekursion oder unendlicher Funktionsaufruf.  
💡 **Erklärung:** Die Funktion ruft sich immer wieder selbst auf.  
🛠️ **Lösung:** Überprüfe deine Abbruchbedingungen.

**RUN003 – Nullzugriff**

❌ **Fehler:** Zugriff auf null ist nicht erlaubt.  
💡 **Erklärung:** Du versuchst, auf ein Objekt zuzugreifen, das nicht existiert.  
🛠️ **Lösung:** Überprüfe, ob die Variable null ist, bevor du sie verwendest.

**RUN004 – Datei nicht gefunden**

❌ **Fehler:** Datei „data.txt“ konnte nicht geöffnet werden.  
💡 **Erklärung:** Die Datei existiert nicht oder der Pfad ist falsch.  
🛠️ **Lösung:** Überprüfe den Dateipfad und die Schreibweise.

**RUN005 – Ungültiger Pfad / Ressource**

❌ **Fehler:** Zugriff auf Ressource außerhalb erlaubtem Bereich.  
💡 **Erklärung:** Sicherheitsverletzung – Leap schützt vor unautorisiertem Zugriff.  
🛠️ **Lösung:** Verwende gültige, erlaubte Pfade.

**RUN006 – Speicherlimit überschritten**

❌ **Fehler:** Das Programm hat zu viel Speicher verbraucht.  
💡 **Erklärung:** Wahrscheinlich durch unendliche Schleife oder zu große Datenmenge.  
🛠️ **Lösung:** Reduziere die Datenmenge oder verbessere Schleifenlogik.

**RUN007 – Timeout / Endlosschleife**

❌ **Fehler:** Programm wurde abgebrochen – Laufzeitlimit überschritten.  
💡 **Erklärung:** Das Programm lief zu lange ohne Ende.  
🛠️ **Lösung:** Überprüfe Schleifenbedingungen und rekursive Aufrufe.

**RUN008 – Typumwandlung fehlgeschlagen**

❌ **Fehler:** Wert „abc“ kann nicht in Typ int umgewandelt werden.  
💡 **Erklärung:** Die Konvertierung ist ungültig.  
🛠️ **Lösung:** Stelle sicher, dass der Wert umwandelbar ist oder prüfe den Typ vorher.

**RUN009 – Funktion ohne Rückgabewert aufgerufen**

❌ **Fehler:** Aufruf erwartet Rückgabewert, aber Funktion liefert keinen.  
💡 **Erklärung:** Du nutzt das Ergebnis einer Funktion, die void ist.  
🛠️ **Lösung:** Verwende die Funktion ohne Zuweisung oder füge return hinzu.

**RUN010 – Externe Ressource gesperrt**

❌ **Fehler:** Zugriff auf Datei/Socket verweigert.  
💡 **Erklärung:** Die Ressource ist durch ein anderes Programm belegt.  
🛠️ **Lösung:** Schließe andere Programme oder prüfe Zugriffsrechte.

**Systemfehler**

**SYS001 – Modul nicht gefunden**

❌ **Fehler:** Modul „core.math“ konnte nicht geladen werden.  
💡 **Erklärung:** Das Modul existiert nicht oder ist nicht installiert.  
🛠️ **Lösung:** Überprüfe den Modulnamen oder installiere es erneut.

**SYS002 – Version inkompatibel**

❌ **Fehler:** Der Code verwendet eine ältere Leap-Version.  
💡 **Erklärung:** Die Syntax ist veraltet oder nicht mehr unterstützt.  
🛠️ **Lösung:** Aktualisiere den Code auf die aktuelle Sprachversion.

**SYS003 – Interner Interpreterfehler**

❌ **Fehler:** Unerwarteter interner Fehler im Interpreter.  
💡 **Erklärung:** Dies ist ein Bug in Leap selbst, kein Fehler deines Codes.  
🛠️ **Lösung:** Melde den Fehler mit Code und Stacktrace an die Entwickler.

**SYS004 – Fehlende Berechtigung**

❌ **Fehler:** Zugriff verweigert – fehlende Berechtigung.  
💡 **Erklärung:** Das Programm darf auf diese Datei oder Ressource nicht zugreifen.  
🛠️ **Lösung:** Ändere die Dateiberechtigungen oder führe das Programm als Admin aus.

**SYS005 – Speicherzugriff verletzt**

❌ **Fehler:** Ungültiger Speicherzugriff erkannt.  
💡 **Erklärung:** Das Programm hat versucht, auf verbotenen Speicher zuzugreifen.  
🛠️ **Lösung:** Überprüfe Zeiger- oder Referenzoperationen.

**SYS006 – Initialisierung fehlgeschlagen**

❌ **Fehler:** Laufzeitumgebung konnte nicht gestartet werden.  
💡 **Erklärung:** Fehlende Systemkomponenten oder falsche Konfiguration.  
🛠️ **Lösung:** Überprüfe Installation und Konfiguration.

**SYS007 – Fehlerhafte Systembibliothek**

❌ **Fehler:** Bibliothek beschädigt oder inkompatibel.  
💡 **Erklärung:** Eine benötigte Datei ist defekt oder nicht kompatibel mit Leap.  
🛠️ **Lösung:** Ersetze oder reinstalliere die betroffene Bibliothek.

**Warnungen**

**WRN001 – Variable nie verwendet**

⚠️ **Hinweis:** Die Variable 'x' wurde nie verwendet.  
💡 **Tipp:** Entferne ungenutzte Variablen, um Code übersichtlicher zu halten.

**WRN002 – Bedingung immer wahr/falsch**

⚠️ **Hinweis:** Die Bedingung if (true) ist immer wahr.  
💡 **Tipp:** Entferne überflüssige Bedingungen.

**WRN003 – Schatten einer Variable**

⚠️ **Hinweis:** Lokale Variable 'x' verdeckt globale Variable gleichen Namens.  
💡 **Tipp:** Verwende eindeutige Namen, um Verwechslungen zu vermeiden.

**WRN004 – Unreachable Code**

⚠️ **Hinweis:** Dieser Codeabschnitt wird nie ausgeführt.  
💡 **Tipp:** Entferne Anweisungen nach return, break oder exit.

**WRN005 – Veraltete Syntax**

⚠️ **Hinweis:** Diese Schreibweise ist veraltet und wird in künftigen Versionen entfernt.  
💡 **Tipp:** Aktualisiere deinen Code auf die neue Syntax.