Übung 2 - Verzweigungen und Listen

Programmiervorkurs 2025

1 Variablen - Zuweisungen

Ergänzt im folgenden Listing die fehlenden Variablenzuweisungen, so dass keine Fehler auftreten und am Ende die drei Zeilen `True`, `5` und `"abc"` in der Konsole ausgegeben werden.

```
variable1 = # hier Zuweisung 1 einsetzen
variable2 = 3
variable3 = # hier Zuweisung 2 einsetzen
variable2 = variable2 + variable1
if variable3 > 5:
    variable2 = 0
else:
    variable2 *= variable3
variable4 = # hier Zuweisung 3 einsetzen
variable5 = # hier Zuweisung 4 einsetzen
variable5 = variable5 + variable4
variable6 = # hier Zuweisung 5 einsetzen
variable7 = variable6 > variable2 * 2
variable5 = variable5 + "c"
print(variable7)
print(variable2)
print(variable5)
```

2 Fehler

2.1 Fehlertypen

Ordne die folgenden drei Fehlertypen den folgenden Ausdrücken zu:

1) SyntaxError	a) true	i) Syntaktischer Fehler
2) ZeroDivisionError	b) 4 +* 5	ii) Lexikalischer Fehler
3) NameError	c) 2 // 0	iii) Semantischer Fehler

2.2 Fehlersuche

Gegeben sind die folgenden Code-Zeilen. Führe die Programme aus und beschreibe, warum jeweils ein Fehler auftritt.

```
prinnt("Dillgurkensuppe gibt's heut' Mittag!")
print(Lorem ipsum !)
print("Alter: " + 18)
print("Hey guys, did you know ?)
```

3 Logische Operatoren

Es seien die folgenden Variablen deklariert und initialisiert:

```
x = 6
y = 7
```

```
z = 0
a = False
```

Welchen Wert enthält die Variable `b` jeweils nach Ausführung der folgenden Anweisungen?

1. `b = x > 5 or y < 7 and z != 0`
2. `b = x * y != y * x and x + z == 0`
3. `b = not (x == y)and z <= 0`
4. `b = x >= 11 or x < 9 and not(y == 2)and x + y * z > 0 or a`
5. `b = z != z or not a and x - y * z <= 0`
6. `b = not a and z < y - x`

4 Eingabe & Verzweigungen

4.1 Variablen einlesen

Im Rahmen der Vorlesung wurde die Funktion `input()` vorgestellt, welche es euch erlaubt, auf einfache Art und Weise Benutzereingaben von der Konsole einzulesen. Schreibt jetzt ein Programm, welches mit Hilfe dieser Funktion nacheinander folgende Eingaben einliest und in Variablen speichert:

- Euren Namen als Zeichenkette
- Euer Geburtsdatum als drei separate, ganze Zahlen: Tag, Monat, Jahr

Insgesamt sollen also vier Eingaben verarbeitet werden. Achtet dabei darauf, dass euer Programm die erwarteten Eingaben textuell klar strukturiert (Stichwort: Bedienerfreundlichkeit). Nachdem die letzte Eingabe verarbeitet ist, soll folgende Zeichenkette ausgegeben werden (wobei natürlich die Platzhalter zu ersetzen sind):

Ausgabe: `"<euer Name> hat am <tt.mm.jjjj> Geburtstag."`

Info

Nutzt für die Ausgabe der Variablen format-Strings (Syntax: `f'{var1} ist kleiner als {var2}'`) und keine String-Konkatentation (+-Operator)

4.2 Teilbarkeit

1. Frage den Nutzer nach zwei ganze Zahlen (a, b).
2. Ist eine Zahl ein Vielfaches der anderen, gib aus, welche ein Vielfaches von welcher ist.
3. Andernfalls gib aus, dass es keine Vielfachen gibt.

Beispiel:

- a=5, b=15 $\Longrightarrow$ 15 ist ein Vielfaches von 5
- a=5, b=17 $\Longrightarrow$ Kein Vielfaches

Info

- Überlege dir vorab alle Fälle, die eintreten können
- Teilbarkeit kann man mit `%` prüfen

4.3 Vergleiche

1. Frage den Nutzer nach drei Ganzzahlen.
2. Gib aus, wie viele Werte gleich sind.

Beispiel:

- a=1, b=1, c=1 $\Longrightarrow$ Drei gleiche Werte
- a=1, b=2, c=1 $\Longrightarrow$ Zwei gleiche Werte
- a=1, b=2, c=3 $\Longrightarrow$ Alle verschieden

5 Listen

5.1 Definition & Indizes

Beginne ein neues Programm und gehe dabei davon aus, dass die Liste `liste` genau 8 Elemente enthält. Euer Programm soll nun die Ergebnisse der folgenden Rechnungen ausgeben:

1. Die Summe der Zahlen an den Indizes 1, 2 und 7.
2. Das Produkt jeder zweiten Zahl (beginnend mit der zweiten).
3. Die erste Zahl mal 300.

Für diese Aufgabe musst du weder Schleifen noch Slicing verwenden. Probiere dein Programm aus, indem du die folgenden Listen einsetzt und die Ausgaben mit denen von eurem Programm vergleichst.

- Für `liste = [1,1,2,3,5,8,13,21]` ist das 24, 504 und 300
- Für `liste = [-1,1,-1,1,-1,1,-1,1]` ist das 1, 1 und -300

5.2 Listenlänge

In der Vorlesung wurde `len(liste)` eingeführt um die Länge einer Liste zu bestimmen. Überlegt euch warum der Ausdruck `liste[len(liste)]` immer zu einem Fehler führt.

Schreibt nun ein Programm, dass für beliebige Listen alle Elemente der Reihe nach ausgibt. Ihr könnt dafür eine `while`-Schleife verwenden.

Info

`print(liste)` und `str(liste)` ist natürlich nicht erlaubt.

5.3 Slicing

5.3.1 Wissensabfrage

Slicing ermöglicht es, nur einen Teil einer Liste zu verwenden. So gilt etwa `[1,2,3,4][1:3] == [2,3]`. Ein Slicing-Ausdruck besteht aus drei Komponenten `[a:b:c]`, die auch weg- oder leer gelassen werden können. Wofür steht jedes einzelne Argument? Welche Werte werden für die einzelnen Argumente verwendet, wenn diese leer gelassen werden?

5.3.2 Grundlagen

Finde Slicing Ausdrücke für die folgenden Listenoperationen zur Liste `lst = [8, 1, 3, -4, 5, 0, 2, 11]`

1. Die gesamte Liste
2. Die gesamte Liste rückwärts
3. Jedes zweite Element beginnend mit dem zweiten Element
4. Nur das erste Element
5. Das erste Element und jedes dritte danach mit Indizes kleiner 7
6. Elemente an Index 6, 5, 4 und 3

5.3.3 Text-Slicing

Slicing funktioniert auch mit Text. Dabei wird der Text wie eine Liste von Buchstaben behandelt. `"Lorem ipsum!"[:5]` ergibt beispielsweise den String `"Lorem"`. Finde einen Ausdruck mithilfe von Slicing, welcher ein Palindrom erkennt. *Ein Palindrom ist ein Wort, das vorwärts und rückwärts identisch ist*.