Funktionen

Definieren von Funktionen

Funktionen in Python sind definierte Codebereiche, die aber erst ausgeführt werden, wenn sie aufgerufen wurden. In diesem Fall hier gibt die Funktion mit dem Befehl `return` ein Ergebnis zurück. Die Zahlen `a` und `b`, die addiert werden sollen, werden als Parameter übergeben. Diese werden in den runden Klammern der Funktion bekanntgegeben (initialisiert). Die Funktionen sind mit Blöcken in Scratch vergleichbar.

```
# Hier stehen die Imports, wenn sie benötigt werden.
import math # Wird allerdings für die grundlegegenden Funktionen nicht benötigt.


# Definition der Addition.
def add(a,b):
  return a+b


# a = int(input("Erste Zahl eingeben: ")) die Funktion int() wird benötigt, um aus der Eingabe
eine Zahl zu machen. Wenn du keine Zahl eingibst, dann wird hier ein Fehler ausgegeben.
# b = int(input("Zweite Zahl eingeben: "))
ergebnis = add(1,2) # Das Ergebnis wird in einer Variablen gespeichert.
print("a+b ergeben: "+ str(ergebnis)) # Das Ergebnis wir auf der Konsole ausgegeben.
```

Wie du sehen kannst, ist der Befehl `return` eingerückt. Dies muss auch sein, sonst funktioniert das Programm nicht. Probier es ruhig aus. In Python werden mit den Einrückungen Codeblöcke, die zusammengehören, markiert. Andere Programmiersprachen benutzen hierfür z. B. Klammern. Entferne die Kommentarzeichen der beiden Zeilen beginnend mit a und b. Schreibe das Programm so um, dass nun die Zahlen verwendet werden, die du in der Eingabe eingibst.

Aufgabe

a) Jetzt schreibe selber eine Funktion, die a-b rechnet.

b) Schreibe auch hier das Programm so um, dass du die Zahlen eingeben kannst.

```
ergebnis = sub(1,2) # Das Ergebnis wird in einer Variablen gespeichert.
print("1-2 ergeben: "+ str(ergebnis)) # Das Ergebnis wir auf der Konsole ausgegeben.
```

Reihenfolge von Code

Code, der in einer Python-Datei steht wird in der Reihenfolge ausgeführt, in der er steht. Code, der in einem `def`-Block steht, wird erst dann ausgeführt, wenn er im Programm aufgerufen wird. Daher müssen die Definitionen auch vor dem Code stehen, der ihn aufruft. Korrigiere den nachfolgenden Code:

(Die print-Funktion in diesem Beispiel hat eine andere Formatierung. Du kannst frei wählen, welche du benutzt.)

```
zahl =  input("Bitte gib eine Zahl zwischen 1 und 10 ein.")
print(number(zahl))

def number(z):
    return f"Das ist deine Zahl: {z}"
```

Rückgabewerte von Funktionen

Funktionen müssen nicht unbedingt Werte mit `return` zurückgeben. Sie können auch einfach nur einen Teil der Programmfunktion übernehmen. Hier gibt die Funktion nur den Willkommensgruß nach dem Start des Programms aus.

```
def hello():
    print("Willkommen bei meinem tollen Programm")

print("Programm startet ...")
hello()
```

Parameter

Wie man hier sieht, wurde der Funktion auch kein Parameter übergeben. Jetzt wollen wir die Funktion etwas intelligenter gestalten. Dazu benötigen wir eine Liste mit zugelassenen Usern für dieses Programm (siehe Variablen)

```
def hello(u):
    users = ["Mr J", "Gamer32", "NoClue01"]
    if u in users:
        print("Willkommen bei meinem tollen Programm")
    else:
        print("Du darfst hier nicht sein. ")

print("Programm startet ...")
user = input("Bitte gib deinen Benutzernamen ein.")
hello(user)
```

Natürlich könnten wir auch mit einem Rückgabewert arbeiten, damit das Programm auch erfährt, ob die Anmeldung funktioniert hat oder nicht.

```
def hello(u):
    users = ["Mr J", "Gamer32", "NoClue01"]
    if u in users:
        return True
    else:
        return False

print("Programm startet ...")
user = input("Bitte gib deinen Benutzernamen ein.")
if hello(user):
    print("Willkommen bei meinem tollen Programm")
else:
    print("Du darfst hier nicht sein. ")
    exit(1) # Dieser Befehl würde das Programm an dieser Stelle mit dem Fehlercode 1 beenden.
```

Sinnvolle Namen

Es ist wichtig, Funktionen und Variablen sinnvolle Namen zu geben. Die Funktion `hello()` hat keinen sinnvollen Namen. Man muss sich überlegen, welche Funktion im Programm die `Funktion()` haben soll. In diesem Fall soll überprüft werden, ob der User im System bekannt ist. Ein Name wie `is_user()` oder `valid_user()` wären hier sinnvoller. Ändere daher den Namen `hello()` in etwas sinnvolles und sofge dafür, dass das Programm noch funktioniert.

Revision #2
Created 24 November 2025 09:21:11 by Marcus Jacobs
Updated 24 November 2025 10:01:20 by Marcus Jacobs