Lokale und globale Variablen (1)

- Variablen, auf die innerhalb einer Funktion schreibend zugegriffen werden, sind **lokal** in der Funktion. Sie können nur in der Funktion verwendet werden, in der sie definiert werden. Beispiel

```
def funct():
    w = "lokale Variable"
    print(w)                    # das geht, Ausgabe: lokale Variable

funct()
print(w)                        # das geht nicht
```

- Gibt es also innerhalb einer Funktion eine Zuweisung an eine Variable, ist sie automatisch lokal (mit einer Ausnahme, die auf einer der folgenden Folien beschrieben wird.)

Lokale und globale Variablen (2)

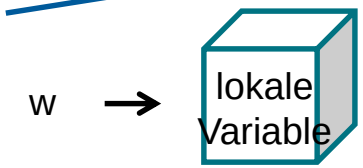
- Variablen, auf die außerhalb von Funktionen schreibend zugegriffen werden, heißen **globale** Variablen. Globale Variablen können aber auch innerhalb von Funktionen lesend verwendet werden. Beispiel:

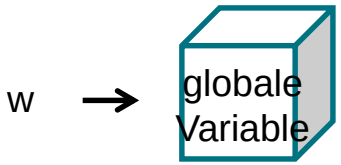
```
def funct():
    print(w)                    # das geht, Ausgabe: globale Variable

w = "globale Variable"
funct()
print(w)                        # das geht auch, Ausgabe: globale Variable
```

Lokale und globale Variablen (3)

- Es lässt sich in großen Programmen nicht vermeiden, dass lokale Variablen in verschiedenen Funktionen den selben Namen haben. Das geht aber problemlos.
- Was ist aber, wenn es eine globale Variable mit dem selben Namen einer lokalen Variable gibt? Beispiel:

```
def funct():
    w = "lokale Variable"
    print(w)          # Ausgabe: lokale Variable

w = "globale Variable"
funct()
print(w)              # Ausgabe: globale Variable
```





Auch das geht also problemlos.

Was passiert in diesem Beispiel?

- In Zeile 4 wird eine Zeichenkette globale Variable angelegt und w zeigt darauf. Danach wird die Funktion aufgerufen.
- Beim Aufruf von funct wird eine Zeichenkette lokale Variable angelegt und w zeigt darauf. Diese Referenz ist aber eine andere Referenz als die von oben. Man könnte sich vorstellen, dass diese Referenz intern einen anderen Namen hat, z.B. funct_w. Die Zeichenkette wird ausgegeben. Beim Ende der Funktion werden die Zeichenkette und die Referenz darauf wieder gelöscht.
- In der letzten Zeile wird die Zeichenkette, auf die w zeigt, ausgegeben. Die Ausgabe ist globale Variable, die andere Zeichenkette gibt es ja gar nicht mehr.

Lokale und globale Variablen (4)

- Achten Sie darauf, dass eine lokale Variable nicht lesend verwendet wird, bevor sie einen Wert erhalten hat. Beispiel:

```
def funct():
    print(w)                # Fehler, die lokale Variable w hat noch keinen Wert
    w = "lokale Variable"
    print(w)                # das geht, Ausgabe: lokale Variable

w = "globale Variable"
funct()
print(w)                    # das geht sowieso, Ausgabe: globale Variable
```

Lokale und globale Variablen (5)

- Wie wir gesehen haben, kann man in einer Funktion lesend auf eine globale Variable zugreifen, falls es keine gleichnamige lokale Variable gibt.
- Wenn man in einer Funktion auf eine globale Variable schreibend zugreifen muss, muss man die Variable global deklarieren. Es gibt aber dann keine gleichnamige lokale Variable. Beispiel:

```
def funct():
    global w                    # w ist jetzt keine lokale Variable mehr
    print(w)                    # das geht, Ausgabe: globale Variable
    w = "keine lokale Variable"
    print(w)                    # das geht, Ausgabe: keine lokale Variable

w = "globale Variable"
funct()
print(w)                        # das geht sowieso, Ausgabe: keine lokale Variable
```

- 🕮 Kapitel 14.8 in (Klein 2018)

Lokale und globale Variablen (6)

Aufgaben

1. Was passiert bei der Ausführung folgender Skripten? Mit roter Schriftfarbe sind die Änderungen gegenüber dem letzten Beispiel gekennzeichnet.

```
def funct():
    w = "lokale Variable"
    global w
    print(w)
    w = "keine lokale Variable"
    print(w)

w = "globale Variable"
funct()
print(w)
```

```
def funct():
    print(w)
    global w
    print(w)
    w = "keine lokale Variable"
    print(w)

w = "globale Variable"
funct()
print(w)
```

Überlegen Sie sich die Lösung zuerst, ohne die Skripten auszuführen. Zur Kontrolle können Sie die Skripten starten und Ihre Antworten überprüfen.

Lokale und globale Variablen (7)

Aufgaben

2. Welche Konsolausgabe hat das folgende Skript?

```
def tier(s):
    biene = "tom"
    print(biene, "und", s, "sind Tiere")
    insekt(biene)

def insekt(s):
    print(biene, "und", s, "sind auch Insekten")

biene = "maja"
tier(biene)
print(biene, "ist eine Biene")
```

Überlegen Sie sich die Lösung zuerst, ohne das Skript auszuführen. Zur Kontrolle können Sie das Skript starten und vergleichen.

Lokale und globale Variablen (8)

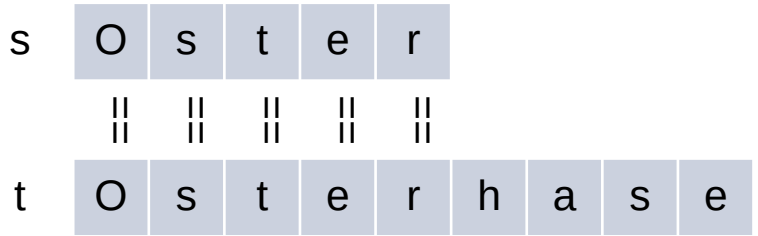
Aufgaben

Die Verarbeitung von Zeichenketten kommt in der Praxis häufig vor. Deshalb weitere Übungsaufgaben dazu.

3. Ein Präfix einer Zeichenkette ist ein Anfangsstück der Zeichenkette. Beispielsweise ist "Oster" ein Präfix von "Osterhase". Die leere Zeichenkette "" ist Präfix jeder Zeichenkette.

   Schreiben Sie eine Funktion is_prefix(s, t), die überprüft, ob die Zeichenkette s ein Präfix der Zeichenkette t ist. Der Rückgabewert ist ein boolescher Wert.

   Die Funktion soll in einer for-Schleife das erste Zeichen von s und t, das zweite Zeichen von s und t, … vergleichen.



Lokale und globale Variablen (9)

Aufgaben

Einige Testfälle:

```
prefix("Oster", "Osterhase")
prefix("Ostern", "Osterhase")
prefix("Osterhasen", "Osterhase")
prefix("", "Osterhase")
```

Python hat auch eine „eingebaute“ Funktion zur Präfix-Prüfung: `t.startswith(s)`

Eine weitere einfache Möglichkeit ist die Verwendung von Slicing: `s == t[0:len(s)]`

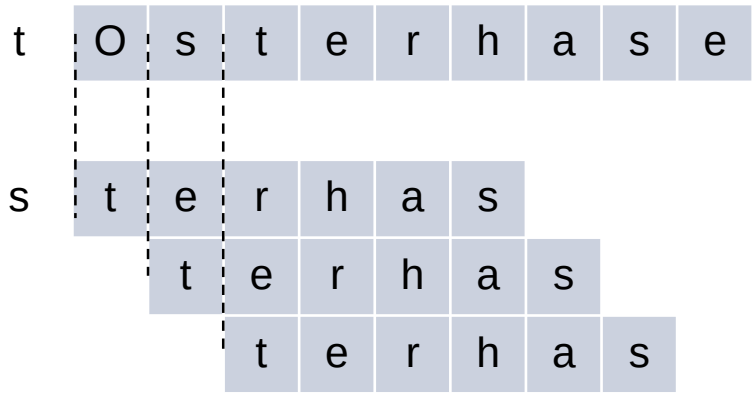
Es soll aber weder startswith noch Slicing in dieser Funktion verwendet werden.

Lokale und globale Variablen (10)

Aufgaben

4. Schreiben Sie eine Funktion is_substring(s, t), die überprüft, ob die Zeichenkette s in der Zeichenkette t zusammenhängend vorkommt. Der Rückgabewert ist ein boolescher Wert. Beispielsweise ist "terhas" ein Substring von "Osterhase". Die leere Zeichenkette "" ist Substring jeder Zeichenkette.

   Die Funktion soll in einer for-Schleife prüfen, ob s ein Präfix von t[0:], t[1:], … ist. Der Default für die obere Grenze beim Slicing ist len(t), d.h. t[1:] ist gleichbedeutend mit t[1:len(t)].



   Auch für Substring gibt es ein einfache eingebaute Funktion in Python. Welche ist das?