Lektion 4a: Einführung in Python mit Turtles

```
from turtle import *

def mystery():
    fillcolor("Green")
    begin_fill()
    pencolor("Red")
    forward (100)
    right (90)
    forward (100)
    right (90)
    forward (100)
    right (90)
    forward (100)
    end_fill()

mystery()
```

Aufgabe 1: Beantworten Sie diese Fragen

1. Was bedeutet die 90 in right(90)?

---

2. Was bewirkt „forward(100)”?

---

3. Was passiert, wenn Sie „endfill()” an eine frühere Stelle im Code verschieben?

---

4. Was passiert, wenn Sie alle Zeilen mit right(90) in right(45) ändern? Probieren Sie es aus und sehen Sie selbst!

---

5. Mit welchen Befehlen kann man Ihrer Meinung nach zurück und nach links gehen?

---

6. Wofür wird das Wort „def” verwendet?

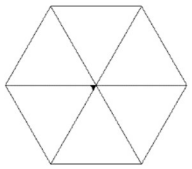
---

7. Was passiert, wenn eine der Zeilen innerhalb der Funktion nicht eingerückt ist? Probieren Sie es aus und sehen Sie selbst.

Aufgabe 2: Erstellen Sie weitere Formen

Zeichnen Sie mit turtles (Hinweis: Löschen Sie ihren Code nicht, sondern kommentieren Sie diesen ggf. aus oder nutzen Sie clear())

1. Benennen Sie die Funktion „mystery" in „square()" um und ändern Sie
   a. die Farbe der Linien
   b. die Füllfarbe
   c. die Länge der Linien
2. Erstellen Sie eine neue Funktion namens triangle() – Sie müssen den Winkel zwischen den Linien ändern
3. Verwenden Sie penup() und pendown(), um ein Quadrat und ein Dreieck mit einem Abstand dazwischen zu zeichnen
4. Verwenden Sie die Funktion circle(), um zwei Kreise zu zeichnen, einen innerhalb des anderen.
5. Zeichnen Sie die Form auf der rechten Seite.



Übersicht Turtle-Befehle

forward() oder fd	Vorwärts gehen	**circle()**	Zeichne einen Kreis
backward()	Zurückgehen	**clear()**	Zeichnung löschen
right()	Nach rechts drehen	**setposition (x,y)**	Bewege dich zur Position x y
left()	Nach links drehen	**pensize()**	Breite des Pinsels/Stifts festlegen
penup()	Heben Sie den Stift an, damit Sie die Turtle bewegen können, ohne zu zeichnen	**shape(„turtle")**	Ändert die Form in einer Turtle
pendown()	Legt den Stift wieder auf den Bildschirm, sodass Sie zeichnen können.	**shape(„circle")**	Ändern Sie die Form in einen Kreis
fillcolor(„Brown")	Ändert die Füllfarbe in Braun (oder eine andere Farbe)	**stamp()**	Hinterlässt einen Abdruck der Form auf dem Bildschirm
begin_fill()	Beginnt mit dem Füllen der Form	**pencolor(„Red")**	Ändert die Stiftfarbe zu Rot (oder einer anderen Farbe)
end_fill()	Beendet die Füllsequenz	**write()**	Schreibt einen Text.

Lektion 4b: Iteration (for-Schleife)

```
def square():
    for counter in range(4):
        forward(100)
        right(90)


square()
left(45)
square()
```

Übungen for-Schleife mit turtles

1. Ändern Sie Ihre Dreiecksfunktion so, dass sie eine for-Schleife verwendet
2. Schreiben Sie eine Funktion pentagon(), um ein Fünfeck (5 Seiten) mit einer Seitenlänge von 200 zu erstellen
3. Schreiben Sie eine Funktion hexagon(), um ein blaues Sechseck (6 Seiten) mit einer Seitenlänge von 50 zu erstellen.

Erweiterungsübung

Erstellen Sie eine Funktion envelope(), die einen Umschlag zeichnet, und überprüfen Sie, ob sie funktioniert (Hinweis: Sie ruft square() und triangle() auf).

Schlüsselbegriffe – Lektion 4: Iteration (for-Schleife)

Begriff	Bedeutung	Wie man es in Python macht

Iteration	Eine Schleife. Es gibt zwei Arten von Schleifen: for-Schleifen (mit einer festgelegten Anzahl von Wiederholungen) und bedingte Schleifen oder while-Schleifen.	`for counter in range(4):`
		`  forward(50)`
		`  right(90)`

Wichtige Punkte zur for-Schleife

1. Die for-Schleife ist eine bestimmte Art von **Schleife**. Die for-Schleife ist eine Kontrollstruktur in der Informatik, die es ermöglicht, einen **Codeblock wiederholt auszuführen**. Sie wird besonders häufig eingesetzt, um über eine Sequenz von Elementen, wie zum Beispiel eine Zahlenreihe, zu iterieren (=wiederholen). Sie verwendet einen **Zähler**, der bei jeder Iteration der Schleife einen Wert annimmt.

2. Die Funktion **range()** gibt an, wie oft die Anweisung wiederholt wird. Beispielsweise bedeutet range(4) eine viermalige Wiederholung und range(10) eine zehnmalige Wiederholung. Tatsächlich generiert range(4) die vier Zahlen 0, 1, 2 und 3, und der Zähler nimmt nacheinander jeden Wert an. Da es vier Zahlen gibt, die den Wert annehmen können, gibt es vier Wiederholungen.

Vertiefung: Die for-Schleife mit einer Zahlenfolge benutzen

Die for-Schleife funktioniert mit Hilfe der range-Funktion auch mit Zahlenbereichen.Nehme die Zahl 2 fünfmal mit der Zahl 3 Mal.

```
zahl=2
for i in range (0,5):
    zahl=zahl*3
print(„Das Ergebnis lautet“,zahl)
```

- **WICHTIG: Die Range-Funktion inkludiert nur den unteren Wert, nicht den oberen.**
- **Die Schleife läuft 5 Durchläufe: den 0., den 1. , den 2., den 3. und den 4.**

Beispielprogramme:

Aufgabe: Schreiben Sie ein Programm, dass die Zahlen 1 bis 10 ausgibt:

Aufgabe: Schreiben Sie ein Programm, dass die Zahlen 1 bis 100 addiert

Weitere Möglichkeiten mit Zahlenfolgen:
Schrittweite festlegen:
range (3,10,2) → Erzeugt eine Folge von 3 bis 10 mit der Schrittweite 2, d.h. 3,5,7,9
Rückwärts zählen:
range (10,0,-1) → Erzeugt eine Zahlenfolge von 10 bis 0 mit einer Schrittweite von -1, d.h. 10,9,8,7,6,5,4,3,2,1

Übungsaufgaben zu for-Schleifen

1. Quadratzahlen
   Schreiben Sie ein Programm, dass die Quadratzahl zu allen Zahlen von 0 bis 10 ausgibt.

2. Summe aus Zahlen
   Ein Programm soll vom Benutzer erfragen, wie viele Zahlen er zusammenzählen möchte. Im Anschluss wird vom Anwender genau so oft wie angegeben, eine Zahl abgefragt und zu einer Summe gezählt.

3. Summe aus positiven Zahlen
   Wir möchten wieder ein Programm, das vom Nutzer einliest, wie viele Zahlen er addieren möchte. Allerdings sollen nun nur die positiven Eingaben in die Summe mit aufgenommen werden.