12. Iterationen

Problem: Eine bestimmte Anzahl von Primzahlen (Liste von Primzahlen) soll multipliziert werden. Die Primzahlen sind in einer also in einer Liste gespeichert.

30030

$□$ Dieser Code ist schreibaufwendig und unübersichtlich
$□$ Umständlich, bei großen Listen

andere Lösung: Interationen

12.1 `for`-Schleife

30030

Syntax

```
1 for var in expresssion:
2         block
```

- Schlüsselworte sind `for`und `in`
- 1. Zeile heißt Schleifenkopf
- 2. Zeile Schleifenrumpf `block` mit einer oder mehreren Anweisungen
- Schleifenvariable `var` im Schleifenkopf
- Schleifeniteration ist ein Durchlauf (Ausführnung eines Schleifenrumpfs)

Beispiel:

```
HH
aa
ll
ll
```

12.2 range

- erzeugt eine Folge von Indices für die Schleifendurchläufe, genauer einen Iterator.
- `range(stop)` - ergibt 0, 1, …, stop -1
- `range(start, stop)` - ergibt start, start + 1, … , stop -1
- `range(start, stop, step)` - ergibt start, start +step, start+ 2* step, … , stop -1

Beispiele:

```
range(0, 5)
```

```
range(2, 30, 5)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

```
[2, 7, 12, 17, 22, 27]
```

```
3 **3 =  27
4 **3 =  64
```

12.3 `zip`

- Funktion
- nimmt eine oder mehrere Sequenzen und liefert eine “Liste” von Tupeln mit korrespndierenden Elementen
- erzeugt eigentlich keine Liste, sondern einen Iterator

Beipsiel

```
[('Rind', 'Kartoffeln'), ('Huhn', 'Pommes'), ('Hund', 'Spätzle')]
```

- mit `zip` können mehrere Sequenzen parallel durchlaufen werden
- bei unterschiedlicher Länge ist das Ergbnis so lang, wie die kürzere Sequenz.

Beispiel

```
r h 5
i u 6
n h 7
d n 8
```

12.4 `reversed`

- Funktion
- Durchlauf einer Sequenz in umgekehrter Richtung

Beispiel

```
o
l
l
a
h
```

12.5 Beispiel: Implemtierung einer Fakultätsfunktion

Fakultät ist wie folgt definiert:

$$0! = 1$$
$$(n+1)! = (n+1) \cdot n!$$

Aufgabe: Entwickle eine Funktion `fact`, welche die Fakulät einer ganzen Zahl berechnet.

1. Bezeichner und Datentypen
   - Die Eingabe ist `n: int` (mit n >= 0)
   - Die Ausgabe ist `int`
2. Funktionsgerüst `python` `def fact(n: int)-> int ##assume n>1`
   `# fill in` `return`
3. Beispiele

```
assert fact(0) == 1
assert fact(1) == 1
```