Blatt 1

Vorkurs Bauingenieurwesen - Informatik - 01.03.24

Aufgabe 1

Erstelle eine neue Textdatei `blatt1.py` mit dem Inhalt

```
def mal_3(x: int) -> int:
    return 3 * x

print("test:")
print(mal_3(2))
print(mal_3(3))
```

und führe das Programm mit IDLE aus. Nutze danach die python-Konsole (den "Taschenrechnermodus") um `mal_3("he")` auszurechen. Warum funktioniert die Funktion auch hier? Wie könne der Mal-Operator ($*$) definiert sein, wenn ein Faktor eine ganze Zahl (e.g. 3) und der andere Faktor eine Zeichenkette ist?

Die folgenden Aufgaben können in die selbe Datei geschrieben werden.

Aufgabe 2

Schreibe eine Funktion `fahrenheit_zu_celsius`, die das Argument als Temperatur in Grad Fahrenheit interpretiert und die äquivalente Celsiustemperatur zurück gibt.

```
def fahrenheit_zu_celsius(fahrenheit: float) -> float:
```

Beispiele: `fahrenheit_zu_celsius(0)` soll etwa `-17.777` ergeben. `fahrenheit_zu_celsius(32)` soll `0.0` ergeben. Die Umrechnungsformel ist $\{t\}_{^\circ\mathrm{C}} = (\{t\}_{^\circ\mathrm{F}} - 32) \cdot \frac{5}{9}$.

Aufgabe 3

Schreibe eine Funktion `rankine_zu_fahrenheit`, die das Argument als Temperatur in Grad Rankine interpretiert und die äquivalente Fahrenheittemperatur zurück gibt.

```
def rankine_zu_fahrenheit(rankine: float) -> float:
```

Beispiele: `rankine_zu_fahrenheit(0)` soll `-459.67` ergeben. `rankine_zu_fahrenheit(1337)` soll `877.33` ergeben. Die Umrechnungsformel ist $\{t\}_{^\circ\mathrm{F}} = \{t\}_{^\circ\mathrm{Ra}} - 459.67$.

Aufgabe 4

Schreibe eine Funktion `rankine_zu_celsius`, die das Argument als Temperatur in Grad Rankine interpretiert und die äquivalente Celsiustemperatur zurück gibt.

```
def rankine_zu_celsius(rankine: float) -> float:
```

Beispiele: `rankine_zu_celsius(0)` soll ca. `-273.15` ergeben. `rankine_zu_celsius(1000)` soll ca. `282.406` ergeben.
Tipp: Benutze die Funktionen der vorherigen Aufgaben.

Aufgabe 5

Schreibe eine Funktion `sterne`, die eine Natürliche Zahl n übergeben bekommt und eine Zeichenkette besthened aus n Sternen (`*`) zurückgibt.

```
def sterne(n: int) -> str:
```

Der Ausdruck `sterne(4)` soll etwa zu `"****"` ausgewertet werden. Benutze die `sterne`-Funktion, um die Funktion `sterne_im_dreieck` zu implementieren. `sterne_im_dreieck` soll eine Natürliche Zahl m übergeben bekommen und m Zeilen mit Sternen auf der Konsole ausgeben. Die erste Zeile soll m Sterne lang sein, die zweite Zeile $m-1$ Sterne und so weiter, bis die letzte Zeile nur noch einen enzelnen Stern lang ist.

```
def sterne_im_dreieck(m: int):
```

Als Beispiel soll der Ausdruck `sterne_im_dreieck(4)` die folgende Ausgabe produzieren:

```
****
***
**
*
```

Randbemerkung: `sterne_im_dreieck` ist die erste python-Funktion dieses Aufgabenblattes, die keine Funktion im mathematischen Sinn ist.

Aufgabe 6

Schreibe eine Funktion `zahlenquadrate`, die eine natürliche Zahl $n \in \{0, \ldots, 9\}$ übergeben bekommt und in der Konsole ein "Bild"mit $2n+1$ Zeilen und $2n+1$ Ziffern pro Zeile zeichnet.

```
def zahlenquadrate(n: int):
```

Das Bild soll dabei aus n geschachtelten Quadraten bestehen. Das äußerste ist gezeichet mit der Ziffer n und hat Kantenlänge $2n+1$. Das nächstinnere Quadrat hat Kantenlänge $2(n-1)+1 = 2n-1$ und besteht aus dem Zeichen $n-1$. Das Muster setzt sich fort bis das innerste Quadrat nur noch aus dem einzigen zeichen `0` besteht. Folgendes Bild soll Ergebnis des Aufrufes `zahlenquadrate(3)` sein:

```
3 3 3 3 3 3 3
3 2 2 2 2 2 3
3 2 1 1 1 2 3
3 2 1 0 1 2 3
3 2 1 1 1 2 3
3 2 2 2 2 2 3
3 3 3 3 3 3 3
```

Tipp: der folgende Code gibt die Ziffer `7` in der Konsole aus. Statt des standardmäßig folgenden Zeilenumbruchs wird nach `7` aber nur ein einzelnes Leerzeichen ausgegeben.

```
m = 7
print(m, end=" ")
```

Aufgabe 7

Ähnlich wie die `zahlenquadrate`- Funktion soll nun die Funktion `zahlenrauten` geschachtelte Rauten anzeigen.

```
def zahlenrauten(n: int):
```

Beilspiel für `zahlenrauten(2)`:

```
    2
  2 1 2
2 1 0 1 2
  2 1 2
    2
```