Blatt 2

Vorkurs Bauingenieurwesen - Informatik - 06.09.2023

Beispiel - Funktion

Die Funktion $f(x) = \frac{1}{2}x + \frac{1}{2}x^2$ kann in Python geschrieben werden als:

```
def f(x: float) -> float:
    return 0.5 * x + 0.5 * x ** 2
```

Die Variable `x` gibt es nur, solange das Ergebnis der Funktion berechnet wird.
In einer Funktion kann beliebiger Code ausgeführt werden:

```
def fakultaet(n: int) -> int:
    ergebnis = 1
    while n > 0:
        ergebnis *= n
        n -= 1
    return ergebnis #beende Funktion und gebe ergebnis zurück
    n = 7 #diese Zeile wird nie ausgeführt
```

Sobald eine Zeile ausgeführt wird, die mit `return` beginnt, stoppt die Funktion und gibt den Wert rechts von `return` als Ergebnis zurück.

Aufgabe 1

Die Funktion `len` gibt die Anzahl der Buchstaben in einem Wert vom Typ `str` an. Etwa wird `len("hallo")` zu `5` ausgewertet. Schreibe mithilfe von `len` eine Funktion `laengerer`, die den Längeren von zwei `str` zurückgibt.

```
def laengerer(a: str, b: str) -> str:
```

Beispiel: `laengerer("Hi", "Tschüss")` soll `"Tschüss"` zurückgeben.

Aufgabe 2

Schreibe eine Funktion `float_zu_str`, die eine Kommazahl in einen `str` umwandelt, allerdings soll für `0.0` oder Zahlen größer als 0 auch ein Plus mit zurückgegeben werden.

```
def float_zu_str(x: float) -> str:
```

Beispiel: `float_zu_str(3.0)` soll zu `"+ 3.0"` auswerten, `float_zu_str(-7.2)` zu `"- 7.2"`. Tipp: Nutze `str(x)` um aus `x` einen `str` zu machen, nutze `abs(x)` um den Betrag von `x` zu bekommen.

Aufgabe 3

Schreibe eine Funktion, die zwei natürliche Zahlen `a` und `b` übergeben bekommt und die Summe aller natürlicher Zahlen von `a` bis `b` berechnet, also

$$a + (a+1) + (a+2) + \cdots + (b-2) + (b-1) + b$$

```
def summe_von_bis(a: int, b: int) -> int:
```

Beispiel: `summe_von_bis(0, 100)` soll `5050` zurückgeben (siehe Blatt 1 Aufgabe 6).

Aufgabe 4

Möchte man den größten gemeinsamen Teiler von zwei Natürlichen Zahlen a und b finden, kann man den *Euklidischen Algorithmus* benutzen. Dieser arbeitet folgenderweise:

1. wiederhole Schritte 2 und 3 solange b nicht 0 ist:
2. bestimme den Rest von $\frac{a}{b}$ (also `a % b`)
3. ersetze a durch b und b durch den soeben bestimmten Rest
4. a ist nun die Lösung

Schreibe eine Funktion `ggt`, die auf die beschriebene Weise den größten gemeinsamen Teiler von a und b ausrechnet.

```
def ggt(a: int, b: int) -> int:
```

Beispiel: `ggt(12, 30)` wird zu `6` ausgewertet.

Aufgabe 5

Um ein Polynom eindeutig zu definieren, muss man nur die Faktoren (*Koeffizienten*) vor den x-Potenzen kennen und wissen, zu welcher x-Potenz jeder Faktor gehört. Etwa ist $g(x) = 3 - 2x + \frac{1}{10}x^3$ eindeutig durch die Liste `[3, -2, 0, 0.1]` definiert, wenn man sagt an Index `i` findet man in der Liste den Faktor vor x^i.

Schreibe eine Funktion `poly_zu_str`, die eine solche Liste in ihrer Interpretation als Polynom in einen `str` umwandelt, beispielsweise `poly_zu_str([3, -2, 0, 0.1])` in
`"3 - 2 x + 0 x^2 + 0.1 x^3"`. Tipp: nutze `float_zu_str` aus Aufgabe 3.

```
def poly_zu_str(poly: list[float]) -> str:
```

Aufgabe 6

Schreibe eine Funktion `funktionswert`, die ein übergebenes Polynom (repräsentiert als Liste) an einem übergebenen Punkt `x` auswertet.

```
def funktionswert(poly: list[float], x: float) -> float:
```

Beispiel: `funktionswert([3, -2, 0, 0.1], 0)` soll den Wert `0.0` zurückgeben,
`funktionswert([3, -2, 0, 0.1], 1)` soll den Wert `1.1` zurückgeben.

Aufgabe 7

Schreibe eine Funktion `ableitung`, die ein Polynom als Liste übergeben bekommt und die Ableitung zurückgibt.

```
def ableitung(poly: list[float]) -> list[float]:
```

Beispiel: `ableitung([3, -2, 0, 0.1])` wird `[-2, 0, 0.3]`.
Tipp: Die Ableitung von

$$a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_nx^n$$

ist

$$a_1 + 2a_2x + 3a_3x^2 + 4a_4x^3 + \cdots + na_nx^{n-1}$$

Aufgabe 8

Schreibe eine Funktion `integral`, die ein Polynom als Liste übergeben bekommt und das unbestimmte Integral zurückgibt. Die Integrationskonstante im Ergebnis soll als 0 gewählt werden.

```
def integral(poly: list[float]) -> list[float]:
```

Beispiel: `integral([-2, 0, 0.3])` wird `[0, -2, 0, 0.1]`.