Aufgabe 1

```
31
```

Aufgabe 2

```
70
```

Aufgabe 3

```
165
```

Aufgabe 4

```
[3, 20]
```

Aufgabe 5

```
314.0
```

Aufgabe 6

```
-1.5
```

Aufgabe 7

```
73
```

Aufgabe 8

```
2
```

Aufgabe 9

```
-1
```

Aufgabe 10

```
None
```

Aufgabe 11

```
14
```

Aufgabe 12

```
7
```

Aufgabe 13

```
14
```

Aufgabe 14

```
geht nicht
```

Aufgabe 15

```
80
```

Aufgabe 16

```
17
```

zur Erinnerung: `M = [[2, 1], [3, 10]]` ist eine Matrix (Liste von Listen), auf deren Elemente man mit einem Doppelindex zugreift:

`M[0][0]`: Nimm das Element in M mit dem Index 0 und dort das Element mit dem Index 0; also 2.

`M[1][1]`: Nimm das Element in M mit dem Index 1 und dort das Element mit dem Index 1; also 10.

`M[0][1]`: Nimm das Element in M mit dem Index 0 und dort das Element mit dem Index 1; also 1.

`M[1][0]`: Nimm das Element in M mit dem Index 1 und dort das Element mit dem Index 0; also 3.

Aufgabe 17

```
def change(x):
    a = x

a = 1
change(2)
print(a)
```

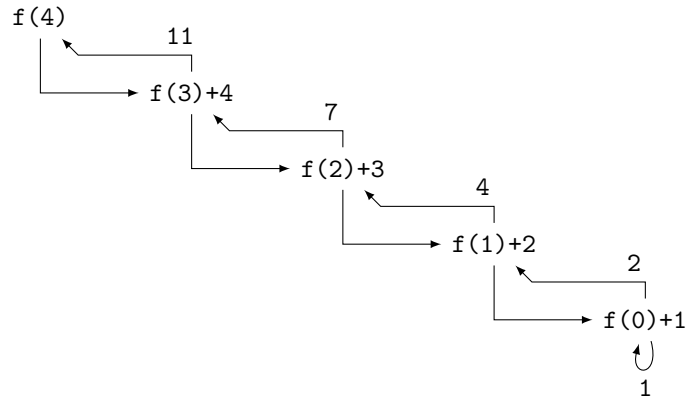
```
1
```

Im Gegensatz zu Listen (und anderen zusammengesetzten Objekten) wird bei den Funktionsparametern von „einfachen“ Objekten wie Zahlen, Zeichenketten und Wahrheitswerten, eine „echte“ Kopie erstellt und den lokalen Variablen zugewiesen. Diese lokalen Werte sind nur während der Laufzeit von `change(...)` gültig. Danach wird werden die lokalen Werte gelöscht und der ursprüngliche Kontext wieder hergestellt.

Aufgabe 18

```
11
```

graphische Darstellung der Rekursion:



Aufgabe 19

```
def rechteckUmfang(a, b):
    u = 2*(a + b)
    return u

print(rechteckUmfang(3, 4))
print(rechteckUmfang(15, 15))
print(rechteckUmfang(0, 6))
print(rechteckUmfang(3.2, 5.6))
```

Aufgabe 20

```
def inchToCm(x):
    return x * 2.54

def poundToKg(x):
    return x * 0.45359237

def momentToSec(x):
    return x * 1.5
```

Aufgabe 21

```
PI = 3.14159

def umfang(r):
    return 2 * PI* r

def inhalt(r):
    return PI * r * r
```

Aufgabe 22

```
def getX(P):
    return P[0]

def getY(P):
    return P[1]

def add(P,Q):
    return [P[0]+Q[0], P[1]+Q[1]]

def sub(P,Q):
    return [P[0]-Q[0], P[1]-Q[1]]

A = [3, 7]
B = [-1, 1]

print(getX(A))
print(getX(B))
print(add(A,B))
print(sub(A,B))
```

Aufgabe 23

```
def solve(a, b, c):
    D = b*b-4*a*c
    if D < 0:
        return []
    elif D == 0:
        return [-b/(2*a)]
    else:
        x1 = (-b + D**0.5)/(2*a)
        x2 = (-b - D**0.5)/(2*a)
        return [x1, x2]

# Tests
print(solve(2, 7, -4))
print(solve(1, 3, 7))
print(solve(1, 6, 9))
print(solve(16, 0, -9))
```

Aufgabe 24

```
def ggT(a,b):
    if b == 0:
        return a
    while b != 0:
        a, b = b, a % b
    return a

print(ggT(5168,2413))
print(ggT(71,0))
print(ggT(0,222))
```