Blatt 6

Vincent Kümmerle und Elvis Gnaglo

29. November 2025

1 Listen

Für die Liste a = [2, "d", 5, 8, 233, "dx", 54, "we", "g", ..., 72, 23, "g"] sind die Zugriff Ausdrücke in Tabelle 1 aufgeführt. Sie lassen sich durch print(a[x]) überprüfen.

Tabelle 1: Zugriff auf verschiedene Listenelemente in Python.

Element	Ausdruck
viertes	a[3]
vorletztes	a[-2]
drittes bis drittletztes	a[2:-2]
jedes 2. ab dem 4.	a[3::2]
jedes 3. rückwärts ab dem vorletzten	a[-2::-3]
7. entfernen	del a[6]

2 Datentypen und Ausdrücke

Die erwarteten Ergebnisse sind mit Begründungen in Tabelle 2 aufgelistet.

Tabelle 2: Erwartete Ergebnisse für verschiedene Ausdrücke in Python.

Ausdruck	Ergebnis	Begründung
3 + 5	8	Ganzzahlsummation: int + int = int
3 + 5.0	8.0	Typkonvertierung: Addition mit float ergibt float
“3” + “5”	“35”	Zeichenketten werden aneinander gehängt (Konkatenation)
“3” * 5	“33333”	String wird fünfmal wiederholt
3 // 2	1	Ganzzahldivision und Ergebnis wird abgerundet
3 / 2	1.5	Normale Division ergibt float
int(2.71828)	2	int() schneidet alle Nachkommastellen ab
round(2.71828)	3	round() rundet auf ganzzahliges Ergebnis
“hallo” + “Welt”	“halloWelt”	Strings werden aneinander gehängt

3 Gerade / Ungerade

```python
# 1. Funktion definieren
def is_even(number):
    """Check, if a given number is even."""
    assert isinstance(number, int) and number >= 1, "Die gegebene
       Zahl ist keine positive, natürliche Zahl."
    return number % 2 == 0

# 2. Eingabe definieren
eingabe_text = input("Bitte gib eine Zahl ein: ")

try:
    zahl = int(eingabe_text)

    if is_even(zahl):
        print(f"Die Zahl {zahl} ist gerade.")
    else:
        print(f"Die Zahl {zahl} ist ungerade.")

except ValueError:
    print("Das war keine gültige ganze Zahl!")
except AssertionError as e:
    print(f"Fehler: {e}")
```

4 Summation

```python
def get_summation_sq(number):
    """Summiere Quadratzahlen bis N."""
    assert isinstance(number, int) and number >= 1, \
        "Die gegebene Zahl ist keine positive, natürliche Zahl."

    result = 0
    for k in range(1, number + 1):
        result += k**2

    return result

try:
    eingabe_text = input("Gib eine natürliche Zahl N ein: ")
    N = int(eingabe_text)

    # Test: Vergleich zur bekannten Formel S(N) = N(N+1)(2N+1)/6
    assert get_summation_sq(N) == N*(N+1)*(2*N+1)//6
```

```
18
19      # Ausgabe
20      result = get_summation_sq(N)
21      print("Die Summe der Quadratzahlen bis", N, "ist:", result)
22
23  except ValueError:
24      print("Das war keine gültige ganze Zahl!")
25  except AssertionError as e:
26      print(f"Fehler: {e}")
```

5 Fakultät

```
1   def fak(n):
2       """Berechne das Produkt aller natürlichen Zahlen von 1 bis n."""
3       assert isinstance(n, int) and n >= 0, \
4           "Die gegebene Zahl ist keine positive, natürliche Zahl."
5
6       result = 0
7       for k in range(1, n + 1):
8           result *= k
9       return result
10
11
12  try:
13      eingabe = input("Gib eine natürliche Zahl n ein: ")
14      n = int(eingabe)
15
16      # Fakultät berechnen
17      result = fak(n)
18
19      print(f"Die Fakultät von {n} ist: {result}")
20
21  except ValueError:
22      print("Das war keine gültige ganze Zahl!")
23  except AssertionError as e:
24      print(f"Fehler: {e}")
```

5.1 Eulersche Zahl

```python
def berechne_euler(genauigkeit):
    """Berechnet die Eulersche Zahl e mittels der Potenzreihe.
    Eingabe: Anzahl der Summanden (Iterationen), die berechnet werden."""
    e_approx = 0
    x = 1  # Mit x=1 wird e^1

    for k in range(genauigkeit):
        # Die Formel zur Berechnung lautet:(x^k)/k!
        # Da x=1 gilt, ist der Zähler immer 1.
        term = 1 / fak(k)
        e_approx += term

    return e_approx

try:
    eingabe = input("Gib die Anzahl der Iterationen für die Näherung ein: ")
    n_iter = int(eingabe)

    # e berechnen
    e_wert = berechne_euler(n_iter)

    print(f"Näherungswert für e nach {n_iter} Schritten: {e_wert}")

except ValueError:
    print("Bitte gib eine gültige ganze Zahl ein!")
except AssertionError as e:
    print(f"Fehler: {e}")
```