

Blatt 6

Vincent Kümmerle und Elvis Gnaglo

2. Dezember 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	<code>a[3]</code>
vorletztes	<code>a[-2]</code>
drittes bis drittletztes	<code>a[2:-2]</code>
jedes 2. ab dem 4.	<code>a[3::2]</code>
jedes 3. rückwärts ab dem vorletzten	<code>a[-2::-3]</code>
7. entfernen	<code>del a[6]</code>

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

3 Gerade / Ungerade

```
1 # 1. Funktion definieren
2 def is_even(number):
3     """Check, if a given number is even."""
4     assert isinstance(number, int) and number >= 1, "Die gegebene
5         Zahl ist keine positive, natürliche Zahl."
6     return number % 2 == 0
7
8 # 2. Eingabe definieren
9 eingabe_text = input("Bitte gib eine Zahl ein: ")
10
11 try:
12     zahl = int(eingabe_text)
13
14     if is_even(zahl):
15         print(f"Die Zahl {zahl} ist gerade.")
16     else:
17         print(f"Die Zahl {zahl} ist ungerade.")
18
19 except ValueError:
20     print("Das war keine gültige ganze Zahl!")
21 except AssertionError as e:
22     print(f"Fehler: {e}")
```

4 Summation

```
1 def get_summation_sq(number):
2     """Summiere Quadratzahlen bis N."""
3     assert isinstance(number, int) and number >= 1, \
4         "Die gegebene Zahl ist keine positive, natürliche Zahl."
5
6     result = 0
7     for k in range(1, number + 1):
8         result += k**2
9
10    return result
11
12 try:
13     eingabe_text = input("Gib eine natürliche Zahl N ein: ")
14     N = int(eingabe_text)
15
16     # Test: Vergleich zur bekannten Formel  $S(N) = N(N+1)(2N+1)/6$ 
17     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 = 1
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

```
1 def berechne_euler(genauigkeit):
2     """Berechnet die Eulersche Zahl e mittels der Potenzreihe.
3     Eingabe: Anzahl der Summanden (Iterationen), die berechnet werden
4     ."""
5     e_approx = 0
6     x = 1 # Mit x=1 wird e^1
7
8     for k in range(genauigkeit):
9         # Die Formel zur Berechnung lautet: (x^k)/k!
10        # Da x=1 gilt, ist der Zähler immer 1.
11        term = 1 / fak(k)
12        e_approx += term
13
14    return e_approx
15
16 try:
17     eingabe = input("Gib die Anzahl der Iterationen für die Näherung
18     ein: ")
19     n_iter = int(eingabe)
20
21     # e berechnen
22     e_wert = berechne_euler(n_iter)
23
24     print(f"Näherungswert für e nach {n_iter} Schritten: {e_wert}")
25
26 except ValueError:
27     print("Bitte gib eine gültige ganze Zahl ein!")
28
29 except AssertionError as e:
30     print(f"Fehler: {e}")
```