A.2 Python-Spickzettel

Hier stehen alle Python-Befehle aus Kapitel 1 auf einen Blick. Die Spalte „Ergebnis“ zeigt, was herauskommt oder ausgegeben wird.

A.2.1 Ausgabe, Eingabe, Variablen

Was	Beispiel	Ergebnis
Text ausgeben	`print("Hallo")`	`Hallo`
mehrere Werte ausgeben	`print("Alter:", 15)`	`Alter: 15`
Variable anlegen (Zuweisung)	`alter = 15`	`alter` hat den Wert `15`
Variable verändern	`alter = alter + 1`	`alter` hat den Wert `16`
Text einlesen	`name = input("Name? ")`	`name` ist immer ein `str`
ganze Zahl einlesen	`zahl = int(input("Zahl? "))`	`zahl` ist ein `int`
Kommazahl einlesen	`x = float(input("Wert? "))`	`x` ist ein `float`
Datentyp anzeigen	`print(type(1.5))`	`<class 'float'>`
in Text umwandeln	`str(15)`	`"15"`
in Zahl umwandeln	`int("15")` / `float("2.5")`	`15` / `2.5`
runden	`round(3.14159, 2)`	`3.14`

A.2.2 Datentypen

Datentyp	Bedeutung	Beispiele
`int`	ganze Zahl	`15`, `-3`, `0`
`float`	Kommazahl (mit Punkt!)	`1.68`, `3.0`
`str`	Zeichenkette (Text)	`"Hallo"`, `"15"`
`bool`	Wahrheitswert	`True`, `False`

A.2.3 Operatoren

Art	Operatoren	Beispiel	Ergebnis
Grundrechenarten	`+ - * /`	`7 / 2`	`3.5`
ganzzahlige Division	`//`	`7 // 2`	`3`
Rest der Division	`%`	`7 % 2`	`1`
Potenz	`**`	`2 ** 3`	`8`
Vergleiche	`== != < <= > >=`	`3 != 4`	`True`
logisches UND	`and`	`True and False`	`False`
logisches ODER	`or`	`True or False`	`True`
logisches NICHT	`not`	`not True`	`False`
De Morgan	`not (A and B)`	wie `not A or not B`	
Kurzschreibweise	`+= -= *=`	`x += 2`	wie `x = x + 2`

A.2.4 Verzweigungen

```
if bedingung:
    anweisungen        # werden nur ausgeführt, wenn die Bedingung wahr ist
elif andere_bedingung:
    anweisungen        # sonst: wenn diese Bedingung wahr ist
else:
    anweisungen        # sonst: in allen übrigen Fällen
```

Doppelpunkt am Ende der Zeile, eingerückte Anweisungen (4 Leerzeichen). `elif` und `else` sind freiwillig.

Mehrfachauswahl (`CASE OF`)

```
match wert:
    case 1:
        anweisungen    # wenn wert == 1
    case 2:
        anweisungen    # wenn wert == 2
    case _:
        anweisungen    # Auffangfall: alle anderen Werte (wie else)
```

Nur für den Vergleich **eines** Werts mit **festen** Werten. Für Bereiche (`alter < 18`) nimmt man `if`. Ab Python 3.10.

A.2.5 Schleifen

Was	Beispiel	Ergebnis
Zählschleife	`for i in range(5):`	`i` = 0, 1, 2, 3, 4
mit Startwert	`for i in range(1, 5):`	`i` = 1, 2, 3, 4
mit Schrittweite	`for i in range(0, 10, 3):`	`i` = 0, 3, 6, 9
über Zeichen	`for z in "Hi":`	`z` = `"H"`, `"i"`
über Listenelemente	`for x in [4, 7]:`	`x` = 4, 7
bedingte Schleife	`while zahl > 0:`	wiederholt, solange die Bedingung wahr ist
Schleife verlassen	`break`	beendet die Schleife sofort

for oder while? Wenn du vorher weißt, wie oft wiederholt wird (oder über alle Elemente gehst): `for`. Wenn es von einer Bedingung abhängt, die sich erst während des Ablaufs entscheidet: `while`.

A.2.6 Zufall und Mathematik

Was	Beispiel	Ergebnis
Zufallszahl (nach `import random`)	`random.randint(1, 6)`	1, 2, 3, 4, 5 oder 6
Kreiszahl π (nach `import math`)	`math.pi`	`3.141592653589793`

`import random` bzw. `import math` steht einmal am Anfang des Programms.

A.2.7 Funktionen

```
1 def quadrat(x):           # Funktion mit dem Parameter x definieren
2     return x * x          # Rückgabewert
3
4 ergebnis = quadrat(4)     # Funktion aufrufen: ergebnis bekommt den Wert 16
5 print(quadrat(5))         # gibt 25 aus
```

A.2.8 Zeichenketten

Was	Beispiel (`s = "Hallo"`)	Ergebnis
Länge	`len(s)`	`5`
Zeichen an Index	`s[0]` / `s[4]`	`"H"` / `"o"`
letztes Zeichen	`s[-1]`	`"o"`
verketten	`s + " Welt"`	`"Hallo Welt"`
vervielfachen	`"ab" * 3`	`"ababab"`
enthalten?	`"ll" in s`	`True`
Groß-/Kleinbuchstaben	`s.upper()` / `s.lower()`	`"HALLO"` / `"hallo"`
nur Buchstaben / Ziffern?	`s.isalpha()` / `s.isdigit()`	`True` / `False`
f-String	`f"{s}, Tom!"`	`"Hallo, Tom!"`
Zeichencode	`ord("A")` / `chr(66)`	`65` / `"B"`

A.2.9 Listen

Was	Beispiel (`z = [4, 7, 1]`)	Ergebnis
Element lesen	`z[0]` / `z[-1]`	`4` / `1`
Element ändern	`z[1] = 9`	`z` ist `[4, 9, 1]`
Länge	`len(z)`	`3`
anhängen	`z.append(5)`	`z` ist `[4, 7, 1, 5]`
einfügen	`z.insert(0, 8)`	`z` ist `[8, 4, 7, 1]`
entfernen (Wert)	`z.remove(7)`	`z` ist `[4, 1]`
entfernen (letztes)	`z.pop()`	gibt `1` zurück, `z` ist `[4, 7]`
entfernen (Index)	`z.pop(0)`	gibt `4` zurück, `z` ist `[7, 1]`
enthalten? / Index suchen	`7 in z` / `z.index(7)`	`True` / `1`
Summe, Minimum, Maximum	`sum(z)`, `min(z)`, `max(z)`	12, 1, 7
sortiert	`sorted(z)`	`[1, 4, 7]`
leere Liste	`z = []`	`z` hat 0 Elemente

Jede Zeile der Tabelle geht von der ursprünglichen Liste `z = [4, 7, 1]` aus.