Python – Spickzettel

Konsolen Ausgabe

Einfacher Text:

```
print("Text")
```

Formatierter Text:

```
x=5
print(f"Zahl: {x}")
```

Beispiele Formatierungen:

- {kommaZahl:.2f} → 2 Nachkommastellen fix
- {0.1234:.1%} → Als Prozent Anzeigen 12,3 %

Konsolen Eingabe

```
textEingabe=input("Frage")
```

Input liefert nur string / Text!
Um mit Eingabe rechnen zu können in Zahl umwandeln!
Eingabe mit Check und Umwandeln als Ganze Zahl:

```
eingabe = input("Zahl: ")

if eingabe.isdigit():
    zahl = int(eingabe)

else:
    print("Das war keine Zahl")
```

Umwandlungen

```
# String → Zahl
int("42")        # → 42
float("3.14")    # → 3.14

# Zahl → String
str(42)          # → '42'

# Zahl → Bool
bool(0)          # → False
bool(123)        # → True

# Runden
float(5)   # → 5.0
int(5.9)   # → 5   (Abschneiden, nicht Runden!)
round(5.9)       # → 6
round(3.14159, 2)# → 3.14

# Bytes <-> Strings
text = "Hallo"
b = text.encode("utf-8")    # → b'Hallo'
print(type(b))   # → <class 'bytes'>

# zurück zu String
text2 = b.decode("utf-8")   # → 'Hallo'
```

Zufall generieren

```
import random

# Zwischen 1 und 10 – Ganzzahl
zahlA=random.randint(1,10)

# Zwichen 0 und 1 – Kommazahl
zahlB=random.random()

# Zufällig auswählen
farben = ["rot", "blau", "grün", "gelb"]
zufaellige_farbe = random.choice(farben)
print(f"Zufällige Farbe: {zufaellige_farbe}")
```

Bedingungen

IF kann alleine stehen oder kombiniert werden mit elif (weiterer Fall) oder else (Alternative)

```
if x < 3:
  print("Kleiner als 3")
else:
  print("Größer oder gleich 3")
print("Der Teil wird immer ausgeführt")
```

Schleifen

While: Führe eingerückten Block solange (immer wieder) aus, solange die Bedingung wahr ist:

```
x=5
while x > 0:
  x-=1
  print(f"Wert: {x}")
```

Listen erstellen

```
# Leere Liste
zahlen = []

# Liste mit Werten
farben = ["rot", "grün", "blau"]

# Gemischte Datentypen
mix = [1, "zwei", 3.0, True]

# Liste aus einer Funktion
liste = list("Hallo")  # ['H', 'a', 'l', 'l', 'o']
```

Liste – Abrufen

```
farben = ["rot", "grün", "blau"]
print(farben[0])   # erstes Element → 'rot'
print(farben[-1])  # letztes Element → 'blau'
```

Liste – Hinzufügen | Löschen

```
farben.append("grün")      # am Ende hinzufügen
farben.insert(1, "weiß")   # an bestimmter Position
farben.remove("rot")       # nach Wert löschen
entfernt = farben.pop()    # letztes Element entfernen
```

Liste - Zählen

```
zahlen = [1, 2, 3, 2, 2]
print(len(zahlen))    # 5
print(zahlen.count(2)) # 3 (wie oft kommt 2 vor)
```

Liste – Durchlaufen

```
for farbe in ["rot", "grün", "blau"]:
    print(farbe)

# Mit Index
for i, farbe in enumerate(["rot", "grün", "blau"], start=1):
    print(f"{i}. {farbe}")
```

Liste – Schneiden

```
zahlen = [0, 1, 2, 3, 4, 5]

print(zahlen[2:5])  # [2, 3, 4]
print(zahlen[:3])   # [0, 1, 2]
print(zahlen[::2])  # [0, 2, 4] (Schrittweite 2)
print(zahlen[::-1]) # [5, 4, 3, 2, 1, 0] (umgedreht)
```

Liste – Sortieren

```
zahlen = [5, 2, 9, 1]
zahlen.sort()
print(zahlen) # [1, 2, 5, 9]
```

Liste – Kombinieren

```
a = [1, 2]
b = [3, 4]
c = a + b
print(c)    # [1, 2, 3, 4]
```

Liste – Prüfen und Suchen

```
farben = ["rot", "grün", "blau"]

if "grün" in farben:
   print("Grün ist dabei!")

print(farben.index("blau")) # Position → 2
```

Funktionen

```
# Definieren
def MeineFunktion():
   print("Funtion ohne Parameter")

def MeineParameterFunktion(parameterA):
   print(f"Hallo {parameterA}")

def MitRückgabe():
   return 123;

# Expliziter Typ
def rechnen(a:int,b:int):
   return a+b

# Aufrufen:
MeineFunktion()
MeineParameterFunktion("Carl")
print(MitRückgabe())
```

Klassen definieren

```
class MeinePersonKlasse:

   def __init__(self):
      self.Name="Carl"
      self.Alter=38

   def Greet(self):
      print(f"Hallo ich bin {self.Name} und bin {self.Alter} Jahre alt.")

variable=MeinePersonKlasse()
variable.Greet()
```

Module

Um Funktionen, Klassen, etc. von DateinameA.py zu benutzen:

```
Import DateinameA

DateinameA.Funktion(.....)
DateinameA.Variable...

Oder: Direkt einbinden:

From DateinameA import Klasse1

Test=Klasse1(....)
```

Match – Case

```
def wochentag_aktion(tag):
   match tag:
      case "Montag":
         return "Neue Woche, los geht's!"
      case "Dienstag":
         return "Noch vier Tage bis Wochenende."
      case "Freitag":
         return "Fast geschafft!"
      case "Samstag" | "Sonntag":  # mehrere Werte
```

```
        return "Wochenende! 🎉"
    case _:  # Default-Fall
        return "Ungültiger Tag."

def beschreibe_wert(x):
  match x:
    case int() if x > 0:
      return "Positive ganze Zahl"
    case int() if x < 0:
      return "Negative ganze Zahl"
    case float():
      return "Kommazahl"
    case str():
      return f"Text: '{x}'"
    case _:
      return "Unbekannter Typ"
```

Mathematik

```
import math

print(math.sqrt(16))    # 4.0
print(math.pow(2, 3))   # 8.0
print(math.pi)        # 3.141592653589793
print(math.sin(math.pi/2))# 1.0
```

Datum und Uhr Zeit

```
from datetime import datetime, timedelta

jetzt = datetime.now()
print(jetzt.strftime("%d.%m.%Y %H:%M:%S"))

morgen = jetzt + timedelta(days=1)
print("Morgen:", morgen.date())
```

Zeit Messen

```
import time

start = time.time()
time.sleep(1.5)
print("Dauer:", time.time() - start, "Sekunden")
```

JSON

Geht auch mit Klassen / Objekten

```
import json

person = {"name": "Carl", "alter": 30}
text = json.dumps(person)
print(text)

back = json.loads(text)
print(back["name"])
```

Datei Schreiben

w: write (neu schreiben, löscht alte Inhalte)
a: append (anhängen, falls Datei existiert)

```
# Text in eine Datei schreiben (überschreibt
vorhandene Datei)

with open("beispiel.txt", "w", encoding="utf-8") as f:
  f.write("Hallo Welt!\n")
  f.write("Das ist eine zweite Zeile.\n")
```

Datei lesen

```
with open("beispiel.txt", "r", encoding="utf-8") as f:
  inhalt = f.read()

  print(inhalt)
```

Datei Prüfen und Löschen

```
import os

from pathlib import Path

if Path("beispiel.txt").exists():
  print("Datei vorhanden!")

else:
  print("Datei fehlt!")

os.remove("beispiel.txt")  # Datei löschen
```

CSV / Tabelle – Datei erstellen und lesen

```
import csv

daten = [

  ["Name", "Alter", "Land"],
  ["Anna", 25, "DE"],
  ["Bob", 30, "USA"]
]

with open("personen.csv", "w", newline="",
encoding="utf-8") as f:
  writer = csv.writer(f)
  writer.writerows(daten)

with open("personen.csv", "r", encoding="utf-8") as f:
  reader = csv.reader(f)
  for zeile in reader:
    print(zeile)
```

Turtle – Rechteck

```
import turtle

t = turtle.Turtle()

for i in range(4):
  t.forward(100)  # 100 Pixel vorwärts
  t.right(90)     # 90° Drehung nach rechts

turtle.done()
```

Turtle – Interaktiv

Mit den Pfeiltasten steuern

```
import turtle

t = turtle.Turtle()
t.speed(0)

def hoch():
  t.setheading(90)
  t.forward(20)

def runter():
  t.setheading(270)
  t.forward(20)

def links():
  t.setheading(180)
  t.forward(20)

def rechts():
  t.setheading(0)
  t.forward(20)

turtle.listen()

turtle.onkey(hoch, "Up")
turtle.onkey(runter, "Down")
turtle.onkey(links, "Left")
turtle.onkey(rechts, "Right")

turtle.done()
```

Turtle mit Text

```
import turtle

t = turtle.Turtle()

t.color("green")
t.pensize(3)
t.write("Hallo!", font=("Arial", 16, "bold"))

t.penup()
t.goto(0, -50)
t.pendown()

t.color("blue")
t.write("Willkommen zur Turtle-Grafik!", font=("Arial",
16, "bold"))

turtle.done()
```

Webserver Flesk

Installieren:

pip install flask

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def home():
    return "<h1>Hello from
Flask!</h1><a
href=\"greet/carl\">Test</a>"

@app.route("/greet/<name>")
def greet(name):
    return f"<h2>Hello,
{name}!</h2>"

if __name__ == "__main__":
    app.run(debug=True)
```

REST Webservice GET

Installieren:

pip install requests

```
import requests

# Beispiel-URL

url="https://restcountries.com/v3.1/
name/germany"

# GET-Request senden

response = requests.get(url)


# Überprüfen, ob der Request
erfolgreich war

if response.status_code == 200:
    # Antwort als JSON ausgeben
    data = response.json()
    print("Erfolg! Daten
empfangen:")
    print(data)

else:
    print(f"Fehler:
{response.status_code}")
```

TKINTER – GUI – Textbox und MessageBox

TKinter ist eine Standard GUI Library die mit Python mitgeliefert wird. Es gibt auch andere, die müssen aber extra installiert werden.

BEISPIEL mit KI generiert

```
import tkinter as tk
from tkinter import messagebox

# Hauptfenster erstellen
root = tk.Tk()
root.title("Mein erstes GUI")
root.geometry("300x150")

# Funktion, die beim Klick auf den Button ausgeführt wird
def hallo_sagen():
    name = name_entry.get()
    if name:
        messagebox.showinfo("Hallo!", f"Hallo {name} 👋")
    else:
        messagebox.showwarning("Achtung", "Bitte gib deinen Namen ein!")

# Label
tk.Label(root, text="Wie heißt du?").pack(pady=5)

# Eingabefeld
name_entry = tk.Entry(root)
name_entry.pack(pady=5)

# Button
tk.Button(root, text="Sag Hallo", command=hallo_sagen).pack(pady=10)

# GUI starten
root.mainloop()
```

TKINTER – GUI – ToDo App

BEISPIEL mit KI generiert

```
import tkinter as tk
from tkinter import messagebox

# Hauptfenster erstellen

root = tk.Tk()
root.title("ToDo Liste 📝")
root.geometry("400x400")

# ----------------------------

#  Globale Variablen

# ----------------------------

todos = []  # Liste für ToDos (jede Aufgabe = [text, checkbox_var])

def todo_hinzufuegen():
    """Fügt ein neues ToDo zur Liste hinzu."""
    text = eingabe_feld.get().strip()  # Text aus Eingabefeld lesen
    if text == "":
        messagebox.showwarning("Achtung", "Bitte gib eine Aufgabe ein!")
        return
    # Variable für den Checkbutton (abgehakt oder nicht)
    var = tk.BooleanVar()
    # Frame für eine einzelne Aufgabe (damit Layout sauber bleibt)
    frame = tk.Frame(todo_liste_frame)
    # Checkbox + Label erstellen
    cb = tk.Checkbutton(frame, text=text, variable=var)
    cb.pack(side="left", anchor="w")
```

```
    # Frame anzeigen
    frame.pack(fill="x", pady=2, padx=5)
    # Aufgabe zur Liste hinzufügen
    todos.append((frame, var, text))
    # Eingabefeld leeren
    eingabe_feld.delete(0, tk.END)


def erledigte_loeschen():
    """Entfernt alle abgehakten Aufgaben."""
    geloescht = 0
    for (frame, var, text) in todos[:]:  # Kopie der Liste (weil wir sie verändern)
        if var.get():  # Wenn Checkbox aktiviert
            frame.destroy()  # GUI-Element entfernen
            todos.remove((frame, var, text))
            geloescht += 1
    if geloescht == 0:
        messagebox.showinfo("Info", "Keine erledigten Aufgaben zum Löschen.")
    else:
        messagebox.showinfo("Erledigt", f"{geloescht} Aufgaben gelöscht.")

#  GUI Layout
# Eingabebereich

eingabe_frame = tk.Frame(root)
eingabe_frame.pack(pady=10)

# Eingabefeld für neue Aufgaben

eingabe_feld = tk.Entry(eingabe_frame, width=30)
eingabe_feld.pack(side="left", padx=5)


# Button: Aufgabe hinzufügen
hinzufuegen_btn = tk.Button(eingabe_frame, text="➕ Hinzufügen", command=todo_hinzufuegen)
hinzufuegen_btn.pack(side="left")

# Bereich für ToDo-Liste
todo_liste_frame = tk.Frame(root)
todo_liste_frame.pack(pady=10, fill="both", expand=True)

# Scrollbar hinzufügen
scrollbar = tk.Scrollbar(todo_liste_frame)
scrollbar.pack(side="right", fill="y")

# Canvas, um Scrollen zu ermöglichen
canvas = tk.Canvas(todo_liste_frame, yscrollcommand=scrollbar.set)
canvas.pack(side="left", fill="both", expand=True)
scrollbar.config(command=canvas.yview)

# Frame im Canvas, in dem die Aufgaben erscheinen
inner_frame = tk.Frame(canvas)
canvas.create_window((0, 0), window=inner_frame, anchor="nw")

# Update Scrollregion, wenn neue Elemente hinzukommen
def update_scrollregion(event):
    canvas.configure(scrollregion=canvas.bbox("all"))

inner_frame.bind("<Configure>", update_scrollregion)

# Für einfache Version nehmen wir direkt dieses Frame:

todo_liste_frame = inner_frame

# Button: erledigte Aufgaben löschen
loeschen_btn = tk.Button(root, text="🗑️ Erledigte löschen", command=erledigte_loeschen)
loeschen_btn.pack(pady=10)

#  Start der Anwendung

root.mainloop()
```