# Spickzettel Python

| | |
|---|---|
| Blöcke | Untergeordnete Blöcke werden durch Einrückung gekennzeichnet. |
| Kommentare | `# einzeiliger Kommentar`<br><br>`"""`<br>`mehrzeiliger`<br>`Kommentar`<br>`"""` |
| Primitive Datentypen | `bool` Wahrheitswert (Werte: `True`/`False`)<br>`int` Ganzzahl<br>`float` Kommazahl (notiert in der Form 1.0f)<br>`double` Kommazahl<br>`str` Zeichenkette, z. B. `"Hallo"`<br>weitere `long, complex, …` |
| Vereinbarung und Zuweisung bei primitiven Datentypen | `x = 6`<br>Optional können in Python Datentypen bei der Initialisierung in der folgenden Form angegeben werden.<br>`x:int = 6`<br>Dies wird aber nicht auf Korrektheit überprüft. |
| Klasse mit Vereinbarung von Attributen und Konstruktor | `class Haus:`<br>`  def Haus(self, strasseNeu, hNr):`<br>`  self.strasse = strasseNeu`<br>`  self.hausnummer = hNr`<br><br>Auch bei Parametern kann optional der Typ angegeben werden:<br>`  def Haus(self, strasseNeu: str, hNr: int):` |
| Vererbung | `class Reihenhaus(Haus):`<br><br>Der Aufruf des Oberklassenkonstruktors erfolgt mit `super().__init__(...)`, anderer Methoden mit `super().Methodenname(...)` |
| Objekterzeugung | `h1 = Haus("Schulstr. ", 8)` |
| Referenz auf das ausführende Objekt | `self` |
| Leere Referenz | None |
| Methode ohne Rückgabewert | `def RueckwaertsGehen(self, schritte):`<br>`  self.Gehen(-schritte)` |
| Methode mit Rückgabewert | `def Quadrieren(zahl):`<br>`  return zahl*zahl`<br><br>Auch hier ist optional die Typangabe möglich:<br>`def Quadrieren(zahl: int) -> int:` |

| | |
|---|---|
| Vergleichsoperatoren | == ist gleich<br>!= ungleich<br>> größer<br>>= größer gleich<br>< kleiner<br><= kleiner gleich |
| Logische Operatoren | not nicht<br>and und<br>or oder |
| Bedingte Anweisung (bei einseitiger bedingter Anweisung entfällt der else-Teil) | `if farbe == "rot":`<br>`  self.anzahlRot = self.anzahlRot + 1`<br>`else:`<br>`  self.anzahlSchwarz = self.anzahlSchwarz + 1` |
| Mehrfachauswahl durch Reihung mehrerer bedingter Anweisungen | `if farbe == "rot":`<br>`  self.anzahlRot = self.anzahlRot + 1`<br>`elif farbe == "blau":`<br>`  self.anzahlBlau = self.anzahlBlau + 1`<br>`else:`<br>`  self.anzahlGelb = self.anzahlGelb + 1` |
| Zählwiederholung | `for i in range (0, 9):`<br>`  self.Gehen(i*50)`<br>`  self.Drehen(90)` |
| Wiederholung mit Anfangsbedingung | `while not self.BeruehrtObjekt():`<br>`  self.Gehen(10)` |
| Konsolenausgabe | `print("Hallo Welt")` |