Spickzettel Python

Blöcke	Untergeordnete Blöcke werden durch Einrückung gekennzeichnet.

Kommentare	`# einzeiliger Kommentar`
	
	`"""`
	`mehrzeiliger`
	`Kommentar`
	`"""`

Primitive Datentypen	`bool` Wahrheitswert (Werte: `True`/`False`) `int` Ganzzahl `float` Kommazahl (notiert in der Form 1.0f) `double` Kommazahl `str` Zeichenkette, z. B. `"Hallo"` weitere `long, complex, …`

Vereinbarung und Zuweisung bei primitiven Datentypen	`x = 6`
	Optional können in Python Datentypen bei der Initialisierung in der folgenden Form angegeben werden.
	`x:int = 6`
	Dies wird aber nicht auf Korrektheit überprüft.

Klasse mit Vereinbarung von Attributen und Konstruktor	`class Haus:`
	`  def __init__(self, strasseNeu, hNr):`
	`  self.strasse = strasseNeu`
	`  self.hausnummer = hNr`
	
	Auch bei Parametern kann optional der Typ angegeben werden:
	`  def Haus(self, strasseNeu: str, hNr: int):`

Vererbung	`class Reihenhaus(Haus):` 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):` `  self.Gehen(-schritte)`

Methode mit Rückgabewert	`def Quadrieren(zahl):`
	`  return zahl*zahl`
	
	Auch hier ist optional die Typangabe möglich:
	`def Quadrieren(zahl: int) -> int:`

Vergleichsoperatoren	== ist gleich != ungleich > größer >= größer gleich < kleiner <= kleiner gleich
Logische Operatoren	not nicht and und or oder

Bedingte Anweisung (bei einseitiger bedingter Anweisung entfällt der else-Teil)	`if farbe == "rot":`
	`  self.anzahlRot = self.anzahlRot + 1`
	`else:`
	`  self.anzahlSchwarz = self.anzahlSchwarz + 1`

Mehrfachauswahl durch Reihung mehrerer bedingter Anweisungen	`if farbe == "rot":`
	`  self.anzahlRot = self.anzahlRot + 1`
	`elif farbe == "blau":`
	`  self.anzahlBlau = self.anzahlBlau + 1`
	`else:`
	`  self.anzahlGelb = self.anzahlGelb + 1`

Zählwiederholung	`for i in range (0, 9):`
	`  self.Gehen(i*50)`
	`  self.Drehen(90)`

Wiederholung mit Anfangsbedingung	`while not self.BeruehrtObjekt():` `  self.Gehen(10)`
Konsolenausgabe	`print("Hallo Welt")`