17.9.24

immer bei 0 anfangen (i)
festlegen ob "≧" oder nur ">"

18.9.24

Kinemetrie

- was wird gemessen → messe ich das, was ich haben will?
  ↳ wo will ich hin?

→ Kamerabasierte Kinemetrie
  ↳ infrarot Kameras, senden Licht aus
  Kameraabbildungsverhalten bestimmen
eigentlich gewollt:
  Rekonstruktion 3D - Punkt → mind. 2 Kameras
  ↳ viele Ungenauigkeiten
  + Körper deckt Marker ab
  ↳ deswegen mehrere Kameras

23.9

REPL = Read - Eval - Print - Loop' => interaktive Konsole, wo Ergebnisse von Befehlen direkt angezeigt werden können
→ mit REPL Aktionen durchführen, die auch in Codedatei möglich sind
- Ausführen & Auswerten von Anweisungen
- Deklarieren von Variablen & Funktionen
- Verwenden integriertes Hilfe : help([object]) / help() = interaktive Hilfekonsole => q zum Beenden

Variablen = Grundlegende Bausteine von Programmen, speichern Daten im Arbeitsspeicher
? Typen von Variablen → 'string' / 'integer'
! Hinzufügen von .0 am Ende von 1 = großer Unterschied bei Behandlung des Werts
Ganze Zahlen sind eine Teilmenge von Gleitkommazahlen, boolesche Werte (true/false) sind eine Teilmenge von ganzen Zahlen
'string' ist ein grundlegender Datentyp, der zum Speichern & Verwalten von Text verwendet wird
'input' gibt den eingegebenen Wert immer als Zeichenfolge (Text) zurück
Über int() & str() Werte umwandeln
! ';' signalisiert, dass viele Codezeilen vorhanden sind

Datentypen:

type	BESCHREIBUNG	Beispiel(e)
Numerischer Typ	Zahl, mit oder ohne Dezimalstellen	`int, float, complex, no = 3`
Texttyp	Zeichenfolge	`str = "a literal string"`
Boolescher Typ	Boolean	`continue = True`

Operatoren: Ausführung von verschiedenen Vorgängen für Variablen

Operator	Beispiel
=	`x = 2` x enthält jetzt „2".
+=	`x += 2` x wird um 2 erhöht. Wenn sie zuvor „2" enthielt, hat sie jetzt den Wert „4".
-=	`x -= 2` x wird um 2 verringert. Wenn sie zuvor „2" enthielt, hat sie jetzt den Wert „0".
/=	`x /= 2` x wird durch 2 dividiert. Wenn sie zuvor „2" enthielt, hat sie jetzt den Wert „1".
*=	`x *= 2` x wird mit 2 multipliziert. Wenn sie zuvor „2" enthielt, hat sie jetzt den Wert „4".

Boolean operators

- bedingte Logik → if
↳ Code nur ausführen, wenn Bedingung erfüllt (True)
! In Python werden None & 0 auch als False interpretiert
→ Text einer if-Anweisung ist eingerückt

Wenn Testausdruck False => Code in if-Anweisung übersprungen
Alles, was in der gleichen Einzugsebene ist, wird im selben Codeblock ausgeführt
„or" => gesamter Ausdruck True, wenn ein Teilausdruck True ist
„and" => beide Bedingungen im Testausdruck müssen wahr sein, damit Gesamtausgabe „True" ist

Im Folgenden finden Sie einige wichtige Informationen zur bedingten Logik:

- Boolesche Typen können einen von zwei Werten aufweisen: `True` oder `False`.
- `if`-, `else`- und `elif`-Anweisungen können zum Ausführen von Code unter bestimmten Bedingungen verwendet werden.
- `and`- und `or`-Operatoren können verwendet werden, um Testausdrücke zu kombinieren und komplexere Bedingungen zu erstellen.

Zeichenfolge

Vorgänge mit Zeichenfolgen erzeugen immer neue Zeichenfolgen
↳ Python verändert aber keine Zeichenfolge! => neue Variable erzeugen
Zeichenfolgemethoden sind Bestandteil des str-Typs
.title() Methode → gibt mit großen Anfangsbuchstaben zurück & kann mit Zeichenfolge direkt verwendet werden