Für Variablen, Funktionen,

Module, Klassen... Namen

a...zA...Z_ gefolgt von a...zA...Z_0...9

□ Sonderzeichen sind zu vermeiden

python-Schlüsselwörter sind verboten □ GROß-/kleinschreibung spielt eine Rolle

```
Grund-Typen
integer, float, boolean, string, bytes
   int 783 0 -192
                             0b010 0o642 0xF3
binär oktal hexadezimal
                         -1.7e-6
float 9.23 0.0
                               ×10<sup>-6</sup>
 bool True False
   str "One\nTwo"
                               Multiline string:
        escaped new line
                                  """X\tY\tZ
                                  1\t2\t3"""
          'I<u>\</u>m'
          escaped '
                                    escaped tab
bytes b"toto\xfe\775"
              hexadezimal oktal
                                      🖠 unveränderbare
```

Identifizierer

```
geordnete Folgen, schneller indizierter Zugriff, Werte wiederholbar Container Typen
         list [1,5,9]
                           ["x",11,8.9]
                                                 ["mot"]
                                                                  []
                             11, "y", 7.4
                                                 ("mot",)
      ,tuple (1,5,9)
                                                                   ()
 * str bytes (geordnete Folge von chars / bytes)
■ key Container, a priori ohne Ordnung, schneller key Zugriff, jeder key ist einmalig b""
           dict {"key":"value"}
                                       dict(a=3,b=4,k="v")
                                                                  { }
key/value Verknüpfungen){1:"one",3:"three",2:"two",3.14:"π"}
           set {"key1", "key2"}
                                       {1,9,3,0}
                                                               set()
½ keys=hashbare Werte (Grundtypen, unveränderbare...) frozenset unveränderbares set
                                                                 empty
```

```
© a toto x7 y_max BigOne
      8 8y and for
                   Variablenzuweisung

    Zuweisung ⇔ Binden eines Namens an einen Wert

1) Auswertung des Wertes auf der rechten Seite
2) Zuweisung zum linken Namen
x=1.2+8+\sin(y)
a=b=c=0 Zuweisung zum selben Wert
y, z, r=9.2, -7.6, 0 Mehrfache Zuweisung
a, b=b, a Wertetausch
a, *b=seq \ Entpacken einer Folge in
*a, b=seq ∫ Element und Liste
                                          and
x+=3
           Inkrement \Leftrightarrow x=x+3
x - = 2
           Dekrement \Leftrightarrow x=x-2
                                          /=
x=None « undefniert » konstanter Wert
del x
           Entferne Name x aus dem Speicher
```

```
type (Ausdruck)
                                                                Konvertierungen
int ("15") \rightarrow 15
int("3f",16) \rightarrow 63
                               ganzzahlige Zahlenbasis kann im zweiten Parameter stehen
int (15.56) \rightarrow 15
                               schneide Nachkommastellen ab
float ("-11.24e8") \rightarrow -1124000000.0
round (15.56, 1) \rightarrow 15.6
                               runde auf eine Dezimale (0 Dezimalen → Ganzzahl)
bool (x) False für kein x, leeren Container x, None oder False x; True für andere x
str(x) → "..."
                 Zeichenkette von x für die Anzeige (Formatierung im Hintergrund)
chr(64) \rightarrow '@' \quad ord('@') \rightarrow 64
                                        code \leftrightarrow char
repr(x) → "..." Literal-Darstellung von x
bytes([72,9,64]) \rightarrow b'H\t@'
list("abc") \rightarrow ['a', 'b', 'c']
dict([(3,"three"),(1,"one")]) \rightarrow \{1:'one',3:'three'\}
set(["one", "two"]) → {'one', 'two'}
Trenner str und Folge von str → zusammengesetzter str
   ':'.join(['toto','12','pswd']) → 'toto:12:pswd'
str getrennt durch whitespaces → list von str
   "words with spaces".split() → ['words', 'with', 'spaces']
str getrennt durch Separator str → list von str
   "1,4,8,2".split(",") \rightarrow ['1','4','8','2']
Folge eines Typs → list eines anderen Typs (via comprehension list)
   [int(x) for x in ('1', '29', '-3')] \rightarrow [1,29,-3]
```

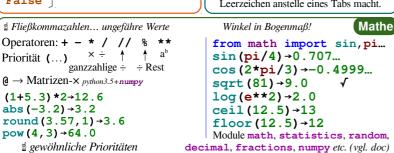
```
Für Listen, Tupel, Zeichenketten, bytes...
                   -5
                                  -3
                                         -2
                                                 -1
                                                         Anzahl der Elemente
                          -4
 negativer Index
                    0
                           1
                                   2
                                          3
  positiver Index
                                                          len (1st) \rightarrow 5
          lst=[10,
                         20,
                                  30;
                                         40
                                                 50]
                                                          positiver slice
                  0
                                      3
                                             4
                                                            (hier von 0 bis 4)
                 -5
  negativer slice
Zugriff auf Sub-Folgen via 1st [start slice : end slice : Schrittweite]
lst[:-1] \rightarrow [10,20,30,40] lst[::-1] \rightarrow [50,40,30,20,10] lst[1:3] \rightarrow [20,30]
lst[1:-1] \rightarrow [20, 30, 40]
                                   lst[::-2] \rightarrow [50, 30, 10]
```

Indizierung von Folgen und Containern Individueller Zugriff auf Elemente via 1st [index] 1st[0]→10 ⇒ erstes Element lst[1]→20 $lst[-1] \rightarrow 50$ ⇒ letztes Element $lst[-2] \rightarrow 40$ Bei veränderbaren Folgen (list): Entfernen mit del 1st[3] und Verändern durch Zuweisung 1st [4]=25 $lst[:3] \rightarrow [10, 20, 30]$ $lst[-3:-1] \rightarrow [30,40]$ $lst[3:] \rightarrow [40,50]$

 $lst[::2] \rightarrow [10, 30, 50]$ lst $[:] \rightarrow [10, 20, 30, 40, 50]$ flache Kopie der Folge Ohne angegebenen slice \rightarrow von Anfang / bis zum Ende. Bei veränderbaren Folgen (1ist): Entfernen mit del 1st[3:5] und Verändern durch Zuweisung 1st[1:4]=[15,25]

```
Boolesche Logik
 Komparatoren: < > <= >= !=
 (boolesche Resultate)
                     ≤ ≥ =
 a and b Logisches Und Beide
                             gleichzeitig
a or b Logisches Oder Eines oder beide
2 Falle: and und or geben einen Wert von
a oder von b zurück (shortcut evaluation).
⇒ stelle sicher, dass a und b boolesch sind.
              Logisches Nicht
not a
True
               Wahr- und Falsch-Konstanten
False
```

```
Ausdrucksblöcke
Eltern-Ausdruck
  Befehlsblock 1...
   Eltern-Ausdruck:
      Befehlsblock 2...
Nächster Befehl nach Block 1
 🛮 stelle den Editor so ein, dass er 4
 Leerzeichen anstelle eines Tabs macht.
```



module truc⇔file truc.py | Import von Modulen/Namen from monmod import nom1, nom2 as fct →direkter Zugriff auf Namen, umbenennen mit as import monmod → Zugriff über monmod.nom1 ... Module und Pakete werden im python-Pfad gesucht (vgl. sys.path)



elif... und nur einem finalen else benutzt werden Nur der Block mit der ersten wahren Bedingung wird ausgeführt. mit einer Variablen x: if bool(x) ==True: ⇔ if x:

if bool(x) ==False: ⇔ if not x:

raise AusnKlasse(...)

→ Normaler Ausführungsblock

except AusnKlasse as e:

Fehlerbehandlungsblock

Einen Fehler signalisieren:

Fehlerbehandlung:

try:



fehlerhafte normale raise X(fehlerhafte Ausführung Ausführung 🖠 finally startet den Block für die Endverarbeitung in allen Fällen.

Putzlocher slated into German

```
Iterative Schleifenanweisung
                                         Bedingte Schleifenanweisung | Befehlsblock wird für jedes
   Befehlsblock wird ausgeführt,
                                                                                Element eines Containers oder Iterators ausgeführt.
Endlosschleifen
   solange wie Bedingung wahr ist.
                                                                                                                                      nächstes
      while logische Bedingung:
                                                                Schleifenkontrolle
                                                                                            for var in Folge:
                                                                                                                                           Ende
                                                                     direktes Verlassen
            Befehlsblock
                                                         break
                                                                                                  Befehlsblock
                                                         continue nächste Iteration
                                                                                                                                                Angewohnheit: Ändere nicht die Schleifenvariable
                                                                                         Geht über die Werte einer Folge:
  s = 0 \ Initialisierungen vor der Schleife
                                                              ₫ else Block für
  i = 1 Bedingung mit mindestens einem Variablenwert
                                                               normales Beenden
                                                                                        s = "Ein Text" | Initialisierungen vor der Schleife
vermeide
                                                               Algorithmus:
                                                                                        cnt = 0
   while i <= 100:
                                                                                          Schleifenzähler, Zuweisung erfolgt durch for-Befehl
                                                                    i = 100
                                                                         i^2
                                                                                        for c in s:
if c == "e":
        s = s + i**2
        i = i + 1
                          🖠 verändere die Variable der Bedingung!
                                                                                                                                Algorithmus:
  print("sum:",s)
                                                                                                   cnt = cnt + 1
                                                                                                                                Zähle die Anzahl
                                                                                        print("gefunden", cnt, "'e'")
                                                                                                                               der e in der
                                                                   Anzeige
                                                                                Schleife über dict/set ⇔ Schleife über Folge der keys Zeichenkette s.
 print("v=", 3, "cm : ", x, ", ", y+4)
                                                                                Nutze slices um über eine Sub-Folge zu iterieren.
                                                                                Schleife über die Indizes einer Folge:
      Anzuzeigende Elemenge: Literalwerte, Variablen, Ausdrücke
                                                                                Verändere Element beim Index.
 print Optionen:
                                                                                □ Greife auf benachbarte Elemente zu. (vor oder zurück)
 □ sep="<sup>^</sup>"
                           Trennzeichen, Standard: Leerzeichen
                                                                                lst = [11, 18, 9, 12, 23, 4, 17]
 □ end="\n"
                           Ende der Anzeige, Standard: neue Zeile
                                                                                lost = []
 □ file=sys.stdout Ausgabe in Datei, Standard: standard out
                                                                                                                          Algorithmus:
                                                                                for idx in range(len(lst)):
                                                                                     val = lst[idx]
                                                                                                                          Begrenze Werte
 s = input("Anweisungen:")
                                                                                     if val > 15:
                                                                                                                          größer als 15 auf 15,
                                                                                          lost.append(val)
                                                                                                                          rette verlorene Werte
    input liefert immer eine Zeichenkette zurück, konvertiere in den
                                                                                lst[idx] = 15
print("modif:",lst,"-lost:",lost)
                                                                                                                                                 gute
        nötigen Datentyp! (vgl. Konvertierungen auf der anderen Seite).
len (c) → Anzahl Elemente Generische Operationen auf Containern
                                                                                Schleife gleichzeitig über Indizes und Werte einer Folge:
min(c) max(c) sum(c)
                                            Note: For dictionaries and sets, these
                                                                                for idx,val in enumerate(lst):
sorted(c) → list sortierte Kopie
                                             operations use keys.
                                                                                range ([Start,] Ende [,Schrittweite]) Ganzzahlige Folgen
val in c → boolesch, Zugehörigkeitsoperator in (Fehlen: not in)
enumerate (c) → Iterator über (Index, Wert)
                                                                                2 Start normal 0, Ende nicht in Folge enthalten, Schrittweite normal 1
zip (c1, c2...) → Iterator über Tupel, die c, Elemente beim gleichen Index haben
                                                                                range (5) \rightarrow 0 1 2 3 4
                                                                                                             range (2, 12, 3) \rightarrow 25811
all (c) → True wenn alle c Elemente zu wahr ausgewertet werden, sonst False
                                                                                range (3,8) \rightarrow 34567
                                                                                                             range (20, 5, -5) \rightarrow 20 15 10
any (C) -> True wenn mindestens ein Element von c als wahr ausgewertet wird, sonst False
                                                                                range (len (seq)) \rightarrow Folge der Indizes der Werte in seq
Speziell für Container geordneter Folgen (Listen, Tupel, Zeichenketten, bytes...)
                                                                                🛮 range liefert eine unveränderbare Folge ganzer Zahlen, wie man sie braucht.
reversed (c) → umgekehrter Iter. c*5→ vervielfachen c+c2→ zusammensetzen
                                                                                                                    Definition von Funktionen
                                                                                Name der Funktion (identifier)
c.index (val) \rightarrow Position
                                  c. count (val) \rightarrow Vorkommen zählen
                                                                                             benannte Parameter
import copy
copy.copy(c) → flache Kopie des Containers
                                                                                 def fct(x, y, z):
                                                                                                                                         fct
copy.deepcopy (c) → tiefe Kopie des Containers
                                                                                        """Dokumentation"""
                                                Operationen auf Listen
                                                                                        # Befehlsblock, Berechnung von res, etc.
return res ← Ergebnis des Funktionsaufrufs, wenn die
                              füge Wert am Ende an
lst.append(val)
                                                                                                           Funktion nichts zurückgibt: return None
lst.extend(seq)
                              füge Folge von Elementen am Ende an
                                                                                 Parameter und alle
lst.insert(idx, val)
                              füge Element beim Index ein
                                                                                 Variablen dieses Blocks existieren nur in diesem Block und während des
lst.remove(val)
                              entferne erstes Element mit Wert val
                                                                                 Funktionsaufrufs. (Denke an eine "black box".)
1st . pop ([idx]) \rightarrow Wert
                              entferne & liefere Element bei idx (Standard: Letztes)
                                                                                 Fortgeschritten: def fct(x,y,z,*args,a=3,b=5,**kwargs):
lst.sort() lst.reverse() sortiere / kehre Liste um vor Ort
                                                                                   *args variable Anzahl von Argumenten (→tuple), Standardwerte,
                                                                                   **kwargs variable Anzahl benannter Argumente (→dict)
   Operationen auf Dictionaries
                                               Operationen auf Mengen
                                                                                   = fct(3, i+2, 2*i)
                                                                                                                              Funktionsaufruf
                      d.clear()
d[key] = Wert
                                         Operatoren:
                                                                                  Speicher des
                                                                                                      ein Argument pro
                      del d[key]
                                           | → Vereinigung (vertikaler Balken)
d[key] \rightarrow Wert
                                                                                  Rückgabewerts
                                                                                                      Parameter
d. update (d2) { ändere/ergänze Verknüpfungen
                                            → Schnittmenge
                                                                                                              Fortgeschritten: fct()
                                                                                                                                            fct

    - ^ Mengen-Differenz

                                                                                d Das ist die Benutzung
d.keys()
d.values()
d.items()

Verknüpfungen

→aufzählbare Ansichten
keys/Werte/Assoziationen
                                                                                einer Funktion mit
                                           < <= > >= → Einschließungsrelationen
                                                                                                              *sequence
                                                                                Klammern (Aufruf).
                                         Operatoren existieren auch als Methoden.
                                         s.update(s2) s.copy()
d.pop(key[,default]) \rightarrow Wert
                                                                                s.startswith (prefix[,start[,ende]]) Zeichenketten-Operationen
d.popitem() → (key, Wert)
d.get(key[,default]) → Wert
                                         s.add(key) s.remove(key)
                                                                                s.endswith(suffix[,start[,ende]]) s.strip([Zeichen])
                                         s.discard(key) s.clear()
                                         s.pop()
                                                                                s.count(sub[,start[,ende]]) s.partition(sep) → (vorher,sep,nachher)
d. setdefault (key[,default]) → Wert
                                                                                s.index(sub[,start[,ende]]) s.find(sub[,start[,ende]])
                                                                   Dateien
 Daten permanent auf Platte speichern und auslesen
                                                                                s.is...() priift auf Zeichenkategorien (z.B. s.isalpha())
                                                                                              s.lower()
                                                                                                              s.title() s.swapcase()
     f = open("file.txt", "w", encoding="utf8")
                                                                                s.casefold() s.capitalize() s.center([Breite, Füllzeichen])
Datei-Variable Namé der Datei
                                 Öffnungsmodus
                                                                                s.ljust([Breite,Füllz.] s.rjust([Breite,Füllz.]) s.zfill([Breite])
                                                          Kodierung der
                                 " 'r' nur lesen (read)
für Operationen auf der Platte
                                                          Zeichen für
                                                                                s.encode (Kodierung)
                                                                                                         s.split([sep]) s.join(Folge)
                                 □ 'w' lesen&schreiben
               (+Pfad...)
                                                           Textdateien:
                                                                                  Formatierungsanweisungen zu formatierende Werte Formatierung
vgl.Module os, os.path und pathlib - ...'+' 'x'
                                                'b' 't' latin1
                                                                                 "Muster{} {} {}".format(x,y,r)—
schreiben
                                 ½ Lese leere Zeichenkette am Ende der Datei
                                                                                 " { Auswahl : Formatierung ! Umwandlung } "
f.write("coucou")
                                f.read([n])
                                                    → nächste Zeichen
                                                                                 □ Auswahl :
                                     wenn n nicht festgelegt, lese bis zum Ende!
                                                                                                            "{:+2.3f}".format(45.72793)
f.writelines (Liste von Zeilen)
                                f.readlines([n]) \rightarrow list der nächste Zeilen f.readline() \rightarrow nächste Zeile
                                                                                   2
                                                                                                            →'+45.728'
                                                                                                            "{1:>10s}".format(8, "toto")

→' toto'
                                                                                   nom
                                f.readline()
                                                                                   0.nom
        🕯 Standardmäßig Textmodus t (liest/schreibt str), möglich binärer
                                                                                   4 [key]
                                                                                                            "{x!r}".format(x="I'm")
        Modus b (liest/schreibt bytes). Konvertiere in den nötigen Typ!
                                                                                   0[2]
                                                                                                            \rightarrow '"I\'m"'
f.close()
                    □ Formatierung :
                                  f.truncate ([Größe]) Größe ändern
f.flush() Schreibe den Cache
                                                                                 Füllzeichen Ausrichtung Vorzeichen min.Breite Präzision~max.Breite Typ
Lesen/Schreiben erfolgt sequentiell in die Datei, änderbar mit:
                                                                                 <> ^= + - space
                                                                                                        0 am Anfang zum Auffüllen mit 0
f.tell() \rightarrow aktuelle Position
                                   f.seek (Position[, Ursprung])
                                                                                 Ganze Zahl: b binär, c Zeichen, d Dezimale (std.), o oktal, x or X hexa...
 Weit verbreitet: Öffnen mit einem geschützten
                                              with open (...) as f:
                                                                                 Fließkomma: e or E Exponentiell, f or F Fixpunkt, g or G passend,
Block (automatisches Schließen) und
                                                 for line in f
                                                                                 Zeichenkette: s ...
                                                                                                                                  % Prozent
 Leseschleife über die Zeilen einer Textdatei:
                                                    # Bearbeitung der line
                                                                                 □ Umwandlung : s (lesbarer Text) oder r (Literal-Repräsentation)
```