**Notizen zu Python**

• Keine Semikolons

• Keinen Variablentypen festlegen, sondern nur Name und Wert

Zu Strings

.upper()

Nur Großbuchstaben

.lower()

Nur Kleinbuchstaben

.islower()/ .isupper()

Kontrolle ob alle klein / groß ist gibt Boolean zurück

len(*string*)

Gibt die Länge eines Strings zurück

*String*[x]

Greift auf den x. Buchstaben des Strings zu (Start ab 0)

.index(“x“)

Gibt die Position des Beginns des Suchparameters wieder

.replace(“a“,“b“)

Ersetzt den Stringteil a durch b

Int(*Variable*)

Konvertiert den String zu einer int

Float(*Variable)*

Konvertiert den String zu einer float

Zu Zahlen

• Einfache Grundrechnung funktioniert auch im Print-Befehl ( ohne Anführungszeichen)

• x\*\*y x hoch y

• Für Ausgabe als String : str(*Variable*)

abs(*Variable*)

Gibt den Betrag zurück

pow(*Variable a, Variable b*)

Gibt Variable a hoch variable b zurück

Max(*Variable a, Variable b)*

Gibt die größere zurück

Min(*Variable a, Variable b)*

Gibt die kleinere zurück

Round(*Variable)*

Rundet die Variable normal

Mit Hilfe von

Form math import \*

Floor(*Variable*)

Rundet immer ab

Ceil(*Variable)*

Rundet immer auf

Sqrt(*Variable)*

Gibt die Wurzel zurück

**Interaktion mit Usern**

*Variable* = Input(“*Inputtext für den User*”) ist ein String bei default

**Lists**

*Listname* = [““,““,““] der Type ist egal mischbar

Ausgabe print(*Listname*[x]) Position wie bei Array

-1 = letztes Element

x:y bedeutet ab dem x. alle folgenden bis y

.extend(*List*)

Hängt eine weitere Liste an die Liste an

.append(*Variable*)

Hängt eine weitere Variable bzw einen weiteren Wert an

.insert(*Index*, *Variable)*

Fügt die Variable / den Wert an der Position des Indexes ein und verschiebt alle anderen einen nach hinten

.remove(*Listelement)*

Entfernt dieses Element aus der Liste

.clear()

Entleert die Liste

.pop()

Entfernt das letzte element

.index(*Listelement)*

Gibt den index dieses Elementes zurück error wenn es nicht vorhanden ist

.count(*Listelement*)

Gibt zurück wie oft ein Element in einer List eist

.sort()

Sortiert aufsteigend alphabetisch oder numerisch

.reverse()

Sortiert absteigend

.copy()

Kopiert die Liste

**Tupel**

Tupelname = (Wert1, Wert2,…)

• Tupels sind unveränderbar

• Ausgabe wie List

• List aus Tupels möglich Listname = [(Wert, Wert), (Wert, Wert)]

Werte nicht veränderbar aber das löschen von Wertesets oder Anhängen neuer etc. ist möglich

**Methoden**

Def *Methodenname* ():

Code der Funktion

Ggf. Return

alles was zur Methode gehört muss eingrückt / nach innen verschoben sein !!!

Parameter einfach in die Klammern ohne Type! Also Reihenfolge wichtig

Code nach return wird NICHT ausgeführt

Zum Aufrufen :

Ggf. Variable = Methodenname( ggf. Parameter )

**If-Abfragen**

If condition :

Code wenn condition true ist

Elif condition2 :

Code wenn 1 false aber 2 true ist

else:

Code wenn condition 1 und 2 false sind

**Für Combinationen**

Or mindestens eins ist true

And beides ist true

Not (Bedinung ) bei false true und andersherum

Vergleichs Operatoren wie in Java ( <, <=, >, >=, ==, !=)

**Dictionaries**

*Dictionaryname* = {

Key 1:value1, keys müssen einzigartig sein

key2: value2,

}

Ausgabe

Print( *Dictionaryname* [key]) gibt den Value zurück

Wenn key nicht vorhanden default = none

Default value setzten durch

Print( *Dictionaryname* [key], defaultvalue )

**While Schleife**

while condition:

Code in der While Schleife

alles was zur Schleife gehört muss eingerückt / nach innen verschoben sein !!!

**For Schleifen**

for x in collection :

Code der for schliefe

alles was zur Schleife gehört muss eingerückt / nach innen verschoben sein !!!

x ist “Zählervariable“(Type egal); Collection kann auch z.B. ein String sein

als Collection auch in range(x,y) möglich alles was >=x und <y ist

**Arrays**

*Arrayname =* [

[1,2,3],

[4,5,6],

[7,8,9],

[0]

]

- Ansteuerung wie in Java

- für einzelne Ansteuerung für alle verschachtelte for schleifen

**Comments**

#Comment text

Oder   
‘‘‘

Commenttext

‘‘‘

**Try- except Blogs**

try:

Try-Code

except Errortype1:

Fehlerhandling2 Code

Except Errortype 2 :

Fehlerhandling 2 Code

**Dateien Auslesen**

**Öffnen**

Filevariable = open(“*Dateiname bzw. Pfad”, “r”*) r = read; w= write; a =append = add at the end; r+= read and write

**Schließen**

Filevariable.close()

**Lesen**

.readable()

Kontrolliert ob die Datei lesbar ist

.read()

Liest alles aus der Datei aus

.readline()

Liest eine Ziele der Datei aus setzt Cursor zur nächsten Zeile

.readlines()

Liest alle Zeilen aus der Datei und speichert sie in ein Array

**In Dateien Schreiben**

Mode a zum Anfügen

Filevariable.write(““)

Mode w zum Schreiben überschreibt alles bisherige, erzeugt neues Dokument, wenn nicht vorhanden

Filevariable.write(““)

**Modules**

Import filename

**Zum Verwenden:**

filename.methode()

**Klassen und Objekt**

Class *classname*:

def \_\_init\_\_(self, eigenschaft1, eigenschaft2,…):

self.eigenschaft1= eigenschaft1

self.eigenschaft2 = eigenschaft2

für Methoden

def *Methodenname* (self)

Methodencode

Return

**Zum Erstellen eines Objektes**

from *classnamefile* import *classname*

object = classname (eigenschaft1, eigenschaft2)

**Vererbung**

from classfile import classname

class *newclassname* (classname):

classendefinition Methoden und Eigenschaften können überschieben werden