Zusammenfassung Inf Prog

Operatoren:
Arithmetisch:
- **%** gibt den Restwert bei einer Division aus 15%2 = 1,
- **//** gibt an wie oft der Wert2 in Wert1 Platz hat 15//7 = 2,
- ****** um eine Zahl hochzustellen z.B 2**2 =2^2
- **+**, **-**, *****, **/** klassische Mathematische Operatoren
- **math.sqrt** = Wurzel (Am Anfang muss math importiert werden: import math)

Logische Op:
- **and**, **or**, **not**, **xor**

Relational:
- **==**, **!=** ungleich, **>**, **<**, **=>**, **<=**

math.factorial(x) # Retourniert die Fakultät als ein Integer. Behebt Betragsfehler, wenn x kein Integral oder negativ ist. 5! = 120

Konvertierung:
- **int** 2, **float** 2.0 #unveränderbar
- **bool** true or false #unveränderbar
- **string** “Text“**,** “42“ #unveränderbar
- **tuple** = (1,2,3) # Tuple werden in runden Klammern geschrieben und sind unveränderbare Listen
- **list** = [1,2,3,4] #Listen sind veränderbar und sie können verschachtelt werden
- **Listen sind Sequenzen**

#Thema Listen
liste = ["Sahra", "Anna","Debora","Sina"]
print(liste[0]) # Das gibt den Wert [Sahra] raus.
Print(liste[0][0]) #Gibt den ersten Wert aus [Sahra] und das 0 Element [S] vom ersten Wert
liste[0] = "E" #löscht den vorhandenen Wert [Sahra] und überschreibt diesen mit [E]

[:] #Notation um eine Kopie von einer Liste zu erstellen

---

l = [1,2,3]
l.append(4) #Mit diesem Befehl kann man dynamisch Werte einfügen (l steht für den Listennamen)
line = "" Dieser Befehl sorg dafür, dass die Werte alle auf der Selben Zeile ausgegeben werden und nicht untereinander

---

import random (Es werden willkürlich Zahlen in der gewählten Spektrum generiert)

```
for i in range (2):
    x = int(random.uniform(0,8))
    y = int(random.uniform(0,8))
```

random.randrange(len(restaurants)) (Es können auch andere Werte willkürlich ausgewählt werden nicht nur int Werte)

- Append and delete items
  - `x.append(5)` `# appends a five to the end`
  - `del(x[4])` `# removes the 5th item`

```
x[-1]          # last item in the list
x[-2:]         # last two items in the list
x[:-2]         # everything except the last 2

x[::-1]        # to reverse the list
```

```
 7  print(list(range(5)))    #[0, 1, 2, 3, 4]
 8  print(list(range(5,10))) #[5, 6, 7, 8, 9]
 9  print(list(range(5,12,3))) #[5, 8, 11]
10  print(list(range(5,0))) #[]
11  print(list(range(5,0,-1))) #[5, 4, 3, 2, 1]
12
13  for e in range(5):
14      print("Hallo", e)
15      #Hallo 0
16      #Hallo 1
17      #Hallo 2
18      #Hallo 3
19      #Hallo 4
20
21  print("abcdef"[:5]) #abcde
22  print(len("abcdef")) #6
23
24  text = "a,b,c,d,e"
25  n_text = text.split(",")  #['a', 'b', 'c', 'd', 'e']
```

Add `mu` as a parameter to `get_braking_distance()`

zh aw

In the function definition, add `mu` as parameter and remove the local variable

```
def get_braking_distance(v0, mu):
    '''
    Calculates braking distance [m] with mu and v0 [m/s]
    '''
    g = 9.81  # gravitational acceleration
    return 0.5 * v0**2 / (mu*g)
```

Test the new function:

```
velocity = 30  # m/s
friction = 0.2
distance = get_braking_distance(velocity, friction)
print("The braking distance for v0 = ", velocity, end='')
print(" and friction mu = ", friction, end='')
print(" is", distance, "m")
```

- Numbers:
  - `int` (42), `float` (42.0)
- Boolean:
  - `bool` (truth values: either `True` or `False`)
- Strings:
  - `str`
    - `"This is a string"`; `'This is also a string'`
    - `'42'`; `"42.0"`
- Collections:
  - `Tuple` (→ see V09)
    - `(42, 42.0)`
  - `List` (→ see also V09)
    - `[42, 42.0, "This is a String"]`
  - `dictionary` (→ see V11)
    - {"first": 42, "second": 42.0}

Immutable	Mutable
bool	
int	
float	
str	
complex	
tuple	**list**
	dict
frozenset	**set**
bytes	*bytearray*
NoneType	

Hier siehst du wie sich der Boolean Wert (True or False) verhaltet:

- Can be operated on with keywords: `not`, `and`, `or`
- Is often the result of relational operators (comparisons, e.g. `>`, `==`, `!=`, (`xor`))

```
>>> x = True
>>> y = False
>>> x and y
False
>>> x or y
True
>>> not x
False
>>> not y
True
>>> type(x)
bool
```

x	y	x and y	x or y	x ^ y	not x
T	T	T	T	F	F
T	F	F	T	T	F
F	T	F	T	T	T
F	F	F	F	F	T

```
print(3 > 2)                        # True
print((2*3) + 4 != 2*3 + 4)         # False
print(True or False)                # True
print(7 or 0)                       # 7
print(True and "OK" or "KO")        # OK
print(False and "OK" or "KO")       # KO
```

Observations
- `and` and `or` perform Boolean logic, **but**…
- They do **not** return Boolean values
- Instead, they **return one of the actual values** they are comparing
  ➔ see Appendix

Pack variables into tuples

```
x, y = 2, 3
z = (x, y)
```

Unpack tuples into variables

```
x, y = z  # same x = 2, y = 3
y, x = z  # switched x = 3, y = 2
```

```
try:
    # Code that can fail (few code!)

except IOError [as <detail>]:
    # What we do if there is an IOError
    # The variable detail contains more information

except SomeOtherError [as <detail>]:
    # What we do if there is SomeOtherError

else:
    # Code if there is no Error

finally:
    # Always executed
```

```
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or
modulo by zero
>>> l[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'l' is not defined
>>> l = []
>>> l[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

For reading lines from a file, you can loop over the file object. This is memory efficient, fast, and leads to simple code:

```
>>>
```

```
>>> for line in f:
...     print(line, end='')
...
This is the first line of the file.
Second line of the file
```

```
import os

#Dateipfad abrufen
cwd = os.getcwd()

old_directory = os.path.join(cwd, "music_extension_example")
print(old_directory)

files = os.listdir(old_directory)
print(files)
counter = 0

#Dateien umbennen
for e in files:
    counter += 1

    old_name = e
    new_name = "song" + str(counter) + ".mp3"

    old_path = os.path.join(old_directory, e)
    new_path = os.path.join(old_directory, new_name)

    os.rename(old_path, new_path)
```

Was gibt das folgende Programm auf der Konsole aus:

```
s = "*"
print(s * 3, s, len(s))
```

Antwort: ****1 ✖

Richtig: *** * 1

```python
#Unterrichtsübung
try:
    s = input("Geben Sie ein Text ein:")
    word = s.split(" ")

    f = open("text2.txt", "w")
    for e in word:
        f.write(e)
        f.write("\n")
    f.close()

    f = open("text3.txt", "r") #aktuell extra auf eine falsche text date
    s = ""
    for line in f:
        line = line[:-1] #hier wird das letzt Zeichen aus line gelöscht,
        s += line + " "
    print(s)
    f.close()

except FileNotFoundError:
    print("Diese Datei kann nicht geöffnet werden!")
```

```python
f = open("test.txt", "w")
f.write("Das ist ein FZG.\n\nHier geht es weiter..")
f.close()

f = open("test.txt", "r")
for line in f:
    print(line, end="")
    #durch print wird immer ein Zeilenumbruch generiert, durch end="" nehme
f.close()

print()

import os

cwd = os.getcwd()
print(cwd + "/" + "txt/")

new_directory = os.path.join(cwd, "txt")
print(new_directory)

files = os.listdir(cwd)
print(files)
```

```python
import os
cwd = os.getcwd() #get current directory
old_directory = os.path.join(cwd, "neuer Ordner") #gesuchter Ordner anhänge
files = os.listdir(old_directory) #Liste mit allen File Names im Ordner

file_txt = open("text.txt", "w") #w, r, a --> Mode
#--> () ist das EOF End of File, readlines for mor than one
old_directory = file_txt.read() #ganzes file as one string
file_txt.write("Any Text") #schreibt irgend ein Text in das File
file_txt.close() #File immer wieder schliessen
```

```python
def read(a):
    if a > 1:
        return 2 * read (a-1)
    else:
        return 1

x = read(5)
print(x) #x=16
```