Python

Web-Automation mit Selenium

GUI mit Tkinter

http mit requests

Kommentare

```
#print("Hello, World!")
"""
This is a comment
written in
more than just one line
"""
```

if / else / ternary

```
if condition_1:
    Statement 1
    Statement 2
elif condition_2:
    Statement 3
    Statement 4
else:
    Statement 5
    Statement 6
```

https://thispointer.com/python-ifelifelse-statement/

```
value_1 if condition else value_2
```

https://thispointer.com/python-if-else-in-one-line-a-ternary-operator/

Switch/Case:
https://entwickler.de/python/tutorial-so-implementiert-man-ein-switch-case-statement-in-python/

Multiline if: https://stackoverflow.com/questions/181530/styling-multi-line-conditions-in-if-statements

Schleifen

```
x = 1
# Infinite While Loop
```

```
while True:
    print(x)
    # If x is 6, then break the loop
    if x == 6:
        break
    x += 1

sample_str = 'Sample Text'
# Iterate over all the characters in string
for elem in sample_str:
    # If char is not lower case then skip printing
    if elem.islower() == False:
        continue
    print(elem)

for i in range(first,last+1):
    print(i)
```

https://thispointer.com/python-programming/

Typumwandlung

integer

```
int(str, base=10)
```

```
value = '234'
# Convert string to integer
num = int(value)
value = '01110011'
num = int(value, base=2)
value = '0xFF11'
# Convert hex string to integer
num = int(value, base=16)
```

https://thispointer.com/python-string-to-int/

float

```
def is_float(value):
  try:
    float(value)
    return True
  except:
    return False

def string_to_number(str):
```

```
  if("." in str):
    try:
      res = float(str)
    except:
      res = str
  elif(str.isdigit()):
    res = int(str)
  else:
    res = str
  return(res)

a = "545.2222"
float(a)
545.22220000000004

# , in . (mit . als tausendtrenner)
# 1) Replace all commas for points:
value.replace(",", ".")
# 2)Remove all but the last point:
value.replace(".", "", value.count(".") -1)
```

https://stackoverflow.com/questions/7106417/convert-decimal-mark

Formatstrings

https://realpython.com/python-string-formatting/

```
f"var: {var}"
```

string-of-char:

```
"="*20
```

Listen

```
# List of string
listOfStrings = ['Hi' , 'hello', 'at', 'this', 'there', 'from']

if 'at' in listOfStrings :
    print("Yes, 'at' found in List : " , listOfStrings)

if listOfStrings.count('at') > 0 :
    print("Yes, 'at' found in List : " , listOfStrings)
```

Quelle:
https://thispointer.com/python-how-to-check-if-an-item-exists-in-list-search-by-value-or-condition/

Liste von Objekten

```
content = []
for i in find_all('name'):
    info = {
        "src":    i.get('src'),
        "height": i.get('height'),
        "width":  i.get('width'),
    }
    content.append(info)
```

dict

https://realpython.com/iterate-through-dictionary-python/#iterating-through-items

ini

```
import configparser

config = configparser.ConfigParser()
config.read('mysql.conf')
print(config['mysql']['host'].strip('\"')) # löscht umfassende "" bei
strings
```

json

https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/

grep

https://www.kite.com/python/answers/how-to-search-a-file-using-grep-in-python

exceptions

```
def divide(a,b):
    result=None
    try:
        result=a/b
    except ZeroDivisionError as e:
        print("do not divide through zero!")
    except Exception as e:
```

```
        print(type(e).__name__,e)
    else:
        print("success")
    finally:
        print("always executed (err/non-err)")
    return result
```

```
try:
    ...
except Exception as e:
   print(f"{type(e).__name__} at line {e.__traceback__.tb_lineno} of
{__file__}: {e}")
```

MySQL

```
import pymysql
import configparser

config = configparser.ConfigParser()
config.read('mysql.conf')

db = pymysql.connect(config['mysql']['host'].strip('\"'),
                     config['mysql']['user'].strip('\"'),
                     config['mysql']['password'].strip('\"'),
                     config['mysql']['database'].strip('\"') )
#cursor = db.cursor()
cursor = db.cursor(pymysql.cursors.DictCursor)
sql = "SELECT * FROM users WHERE Nachname = %(lastname)s"
lastname="Wunderlich"
try:
    # Execute the SQL command
    cursor.execute(sql,{"id":1,"lastname":lastname})
    print(cursor._last_executed)
    # Fetch all the rows in a list of lists or as list of dict (if using the
DictCursor).
    results = cursor.fetchall()
    for row in results:
        print(row)
except Exception as e:
    print("Exception: {}".format(type(e).__name__))
    print("Exception message: {}".format(e))

db.close()
```

vorherige Script Instanz beenden

```
import os
```

```
import psutil
#import time

application="python"
scriptfile=os.path.basename(__file__)

print(f"own pid ({scriptfile}):{os.getpid()}")
for p in psutil.process_iter():
#    print(p.pid,p.name(),p.cmdline())
    if p.name().startswith(application):
        print("Python: ",p.pid,p.name(),p.cmdline())
        if p.pid == os.getpid():
            print("own process")
        else:
            print("other process...")
            if scriptfile in p.cmdline():
                print("kill it...")
                p.terminate()

#while True:
#    time.sleep(5)

print("exit")
```

time

```
import time

start = time.time()
#do something
end = time.time()
print(round((end - start) * 1000,2),"ms")
```

Datetime

formatstring parsen (3.6) und zeitzone umrechnen:

```
from datetime import datetime,timezone

def parseisodt(d):
    dtfmt='%Y-%m-%dT%H:%M:%S%z'
    if ":" == d[-3:-2]:
        d = d[:-3]+d[-2:]
    dt=datetime.strptime(d,dtfmt)
    return dt

d=parseisodt('2021-03-12T08:03:00+01:00')
```

```
d=d.astimezone(tz=timezone.utc)
print(d)
```

ab 3.7 gibt es dann fromisoformat

regex

https://docs.python.org/3/library/re.html

```
re.sub()
re.match()
```

```
# ersetzen mit backref benötigt r-string (r'')
>>> import re
>>> s="2020-10-11"
>>> re.sub('^(\d+)-(\d+)-(\d+)$',r'\3.\2.\1',s)
'11.10.2020'
```

```
REGEXP_TEST = re.compile(r'''
(?P<test>[0-9A-Z]{3}) # name=test, 3 chars of [0-9A-Z]
''', re.VERBOSE)

def test(txt):
    match = REGEXP_TEST.search(txt)
    if match:
        print(match.group("test"))
        return match.groupdict()
    else:
        return false
```

externe Anwendung

```
import subprocess
subprocess.call(["ls", "-l"])
```

OOP

Beispiel-Klasse Fifo

```
class Fifo:
    def __init__(self,liste=[]):
        self.__liste=list(liste)#wichtig, damit leere liste erzeugt wird!!!

    def put_in(self,element):
        self.__liste.append(element)
```

```
    def take_out(self):
        if len(self.__liste)>0:
            return self.__liste.pop(0)
        else:
            print("Liste leer!")

    def interactive(self):
        while True:
            s=input("bitte Element eingeben (beenden mit leer, rausnehmen
mit ?):")
            if s=='':
                break
            elif s=='?':
                print(self.take_out())
            else:
                self.put_in(s)

    def getElements(self):
        return self.__liste
    def __str__(self): #print(obj)
        return f"Ich bin eine {self.__class__.__name__} {self.__liste}"

    def __add__(self,other_fifo):
        if isinstance(other_fifo,Fifo):
            return Fifo(self.getElements() + other_fifo.getElements())
        else:
            print(f"Addition abgelehnt, element ist ein
{other_fifo.__class__} ")

l=Fifo()
l.put_in("test1")
l.put_in("test2")
#print(l)
l.take_out()
print(l)
#l.interactive()

l2=Fifo()
print("l2,sollte leer sein:",l2)
l2.put_in("test3")
l2.put_in("test4")
```

Modularisierung

Import: https://www.kite.com/python/answers/how-to-import-a-class-from-another-file-in-python

to import a class apibase from file apibase.py

```
from apibase import apibase
```

to import all from myhelper.py:

```
from myhelper import *
```

dict anzeigen

```
import json
def formatdict(d):
    #obj=json.loads(json_string)
    formatted_str = json.dumps(d, indent=2,default=str)
    return formatted_str
```

Pfad des Scriptes

```
import os.path
p=os.path.dirname(os.path.abspath(__file__))
```

nano

für Tab2Spaces + 4 Zeicheneinrückung verwende ich einen alias (~/.bashrc)

```
alias pynano="nano -ET4"
```

python2 venv

https://stackoverflow.com/questions/1534210/use-different-python-version-with-virtualenv/11301911#11301911

```
wget https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tgz
tar -xzf Python-2.7.18.tgz
mkdir ~/.localpython
./configure --prefix=$HOME/.localpython
make
make install
cd ~/.localpython
bin/python -m ensurepip --upgrade #install 2.7 pip
#install python3 virtualenv
pip install virtualenv
python3 -m virtualenv ~/env27 -p $HOME/.localpython/bin/python2.7
source ~/env27/bin/activate
```

```
python -V
```