Relations/Descriptor festlegen

class Deskriptor\_(object):

def \_\_init\_\_(self, name, BF, BS ,OB, UB,VB, SB ):

self.\_BF = Benutzt fuer

self.\_BS = Benutztes Synonym

self.\_OB = Oberbegriff

self.\_UB = Unterbegriff

self.\_VB = Verwandter Begriff

self.\_SB = Spitzenbegriff

thesaurus[deskriptor]=self

def set name(self):

def add\_BF

def add\_BS

def add\_OB

def add\_UB

def add\_VB

def add\_SB

import xml.etree.ElementTree as ET

+ tree = ET.parse('some.xml')

+ root = tree.getroot() #Wurzel-Element xml holen

+ if root != 'concepts':

+ print 'falsches Format'

+

+ else:

+

+ print root # Objekt an Speicherstelle

+ print root.tag

+ print root.attrib # Attribute, die zum Tag gehoeren

+ print root.getchildren() # alle Kinderknoten auf der naechsten Ebene

+ # Pruefen, ob es weitere Kinderknoten gibt:

+

+ if root.getchildren():

+ print "Es gibt weitere Kinderknoten"

+ else:

+ print "Keine weitere Kinderknoten vorhanden"

+ for child in root:

+ print child.tag, child.attrib, child.text

+

+ words = root.getchildren()

+

+ for descriptor in descriptoren:

+ discriptor.tag, discriptor, attrib

+

+ for descriptor in root.findall('descriptor'): #jedes "word finden"

+ rank = descriptor.find('rank').text # zu jedem word ein rank finden

+ name = descriptor.get('name')

+ print name, rank

+

+

* finde alle TT,NT,UF,US,RT,BT

def get\_all\_rel(deskriptorsaetze,deskriptor,rel):

if deskriptorsaetze[deskriptor]['BS'],[‘BF’],[‘VB’],[‘SB’], [‘OB’], [‘UB’], [‘VB’] != []:

return get\_rel(deskriptorsaetze,deskriptorsaetze[deskriptor]['BS'][0],rel],[‘BF’] [], rel,[‘VB’] [], rel,[‘SB’] [], rel, [‘OB’] [], rel, [‘UB’] [], rel, [‘VB’] [], rel)

else:

if deskriptorsaetze[deskriptor][rel] == []:

return None

else:

return deskriptorsaetze[deskriptor][rel]

ODER EINZELN:

def get\_rel(deskriptorsaetze,deskriptor,rel):

if deskriptorsaetze[deskriptor]['BS'] != []:

return get\_rel(deskriptorsaetze,deskriptorsaetze[deskriptor]['BS'][0],rel)

else:

if deskriptorsaetze[deskriptor][rel] == []:

return None

else:

return deskriptorsaetze[deskriptor][rel]

def get\_rel(deskriptorsaetze,deskriptor,rel):

if deskriptorsaetze[deskriptor]['BF'] != []:

return get\_rel(deskriptorsaetze,deskriptorsaetze[deskriptor]['BF'][0],rel)

else:

if deskriptorsaetze[deskriptor][rel] == []:

return None

else:

return deskriptorsaetze[deskriptor][rel]

…………

\* setze mit deutschen Deskriptoren gleich

deskriptorsaetze[deskriptor] = {'BF':UF, 'BS':USE, 'SB':TT, 'OB':BT, 'UB':NT, 'VB':RT}

ODER

Man könnte zu begin einenmal die class in deutsch wie englisch definieren.

class Deskriptor\_german(object):

def \_\_init\_\_(self, name, BF, BS ,OB, UB,VB, SB ):

self.\_BF = Benutzt fuer

self.\_BS = Benutztes Synonym

self.\_OB = Oberbegriff

self.\_UB = Unterbegriff

self.\_VB = Verwandter Begriff

self.\_SB = Spitzenbegriff

class Deskriptor\_english(object):

def \_\_init\_\_(self, name, BF, BS ,OB, UB,VB, SB ):

self.\_UF = Benutzt fuer

self.\_USE = Benutztes Synonym

self.\_BT = Oberbegriff

self.\_NT = Unterbegriff

self.\_RT = Verwandter Begriff

self.\_TT = Spitzenbegriff

\* intialisiere Deskriptoren (einzeln für jeden)

\* Packe Deskriptoren in ein Thesaurus Dict

for ds in [datei]:

dict\_thesaurus = {'BF':[], 'BS':[], 'SB':[], 'OB':[], 'UB':[], 'VB':[]}

deskriptor = ds.keys()[0]

for rel in ds[deskriptor].items():

dict[rel[0]]=rel[1]

deskriptorsatz\_hinzufuegen(deskriptorsaetze,deskriptor,dict['BF'],dict['BS'],dict['SB'],dict['OB'],dict['UB'],dict['VB'])

\* Import von weiteren Formaten (JSON, CSV)

+ ### Import von JSON

+ > import json

+

+ with open("import.json", "r") as json\_input:

+ data = json.load(json\_input)

+ print data, type(data)

+

+ new\_data = {"Nachname": "Meier", "Vorname": "Anton"} ""hier müssen wir noch die Relationen einsetzen""

+

+ with open("import.json", "w") as json\_output:

+ json.dump(new\_data, json\_output)

=> erfolgreich geladen

@@ -29,7 +73,24 @@ P.S.: Diese Datei ist eine Markdown-Datei. Mit anderen Worten sie besitzt eine s

\* zeige Deskiptor mit Relationen an

\* zeige einzelne Relationen an

+

+ > def find\_trace(deskriptorsaetze,deskriptor,trace):

+ if deskriptor:

+ trace.append(deskriptor)

+ find\_trace(deskriptorsaetze, get\_rel(deskriptorsaetze,deskriptor,'OB'),trace)

+ return trace

+

\* zeige Top-Term an

+

+ > def make\_tree(topterm,deskriptorsaetze,level=0):

+ if level in tree\_dict:

+ tree\_dict[level] = tree\_dict[level]+ " " + topterm

+ else:

+ tree\_dict[level] = topterm

+ if get\_rel(deskriptorsaetze,topterm,'UB'):

+ for term in get\_rel(deskriptorsaetze,topterm,'UB'):

+ if term:

+ make\_tree(term,deskriptorsaetze,level+1)