# !/usr/bin/env python3

import os from operator import itemgetter import notes\_dictionaries as notesDict

PATH\_TO\_USE = ".."

class FileLineWrapper(object):

def \_\_init\_\_(self, f):  
 self.f = f  
 self.line = 0  
  
def close(self):  
 return self.f.close()  
  
def readline(self):  
 self.line += 1  
 return self.f.readline()

def printOsTree(): for path, dirs, files in os.walk(PATH\_TO\_USE): print(path) for f in files: print('' + f)

def list\_files(): file\_paths = [] for root, directories, files in os.walk(PATH\_TO\_USE, followlinks=True): for filename in files: filepath = os.path.join(root, filename) if "pycache" not in filepath: file\_paths.append(filepath)

return file\_paths

def printlist(alist): for i in alist: print(i)

def list\_notes(): paternList = notesDict.magicFileExtentions fulllist = list\_files() alist = [] for i in fulllist: for extention in paternList: if extention in i: alist.append(i) return alist

def makeStringBeaultiful(string): st = "" st = string.replace("", "").strip() return st

## DONE Verify all super/subfolder for magicwords in files

## DONE Generate a dictionary with folder/file: magicwords

def pathContent\_dict(): noteslist = sorted(list\_notes()) magicTasksDisct = {} for key, content in notesDict.magicWords.items(): for i in noteslist: # f = FileLineWrapper() openlist = [] openlist = open(i).readlines() for line in openlist: if content in line: keyLineAndMagicWord = "[" + str(openlist.index(line) + 1) + "]" + makeStringBeaultiful(line) valuePath = i magicTasksDisct[keyLineAndMagicWord] = valuePath return magicTasksDisct

## Show TODOS list

def showPathContentDict\_list(): retlist = [] Ksize = 0 Vsize = 0 for k, v in pathContent\_dict().items(): if len(k) > Ksize: Ksize = len(k) if len(v) > Vsize: Vsize = len(v) - 7 for key, content in notesDict.magicWords.items(): retlist.append("" + content + ":" + "*"*  (Ksize + Vsize + 5))

for k, v in sorted(pathContent\_dict().items(), key=itemgetter(1)):  
 if content in k:  
 formatStr = '{0:' + str(Vsize) + '} ==> {1:10}'  
 retlist.append(formatStr.format(v.replace(".notes", "").replace("../", ""), k))  
 # retlist.append("\_" \* (Ksize + Vsize))  
 retlist.append("" + "\*" \* (Ksize + Vsize + 5) + "\n\n")  
return retlist

printlist(showPathContentDict\_list())

## TODO: implement export print file with date

## TODO: Generate a JSON file with all magicwords found