**Table of Contents**

[1. file Matching 3](#_Toc483980637)

# File Matching

|  |
| --- |
|  |
| import collections  import copy  #-----------------------------------------------------------------------------#  # Stores the contents of the file 'filename' in a dictionary reference by  # 'file\_dict'. The file's format (All the field names in the file) is stored  # in 'layout'.  #  # The 'file\_dict' does not contain fields that are Specified as 'N' in the  # Layout file.  #-----------------------------------------------------------------------------#  def process\_data\_file(filename: str,  layout: dict,  file\_dict: dict) -> None:    *""" Process the given file and build a dictionary """*    with open(filename) as file:  for line in file:  fields = line.strip(*'\n'*).split(*','*)    # Create a dictionary with Header and Data  temp\_dict = dict(zip(header, fields))    # If a field is not needed to compared, Delete its entry from  # the dictionary.    for field in layout.keys():  if layout[field][*'match\_flag'*] == *'N'*:  del temp\_dict[field]    # Gather the data stored in Key field(s) and store in a list.  kv\_list = []    for k in file\_key:  kv\_list.append(temp\_dict[k])  del temp\_dict[k]    # Convert the Key data to a tuple  key = tuple(kv\_list)    file\_dict[key] = copy.deepcopy(temp\_dict)    return  #-----------------------------------------------------------------------------#  # Each line in the layout file must have at least 2 entries:  # 1. Field name  # 2. Whether the field has to be matched or not (Y/N)  # 3. Whether it is a key field or not (KEY1, KEY2, ...)  #  # Sample File:  # ACCOUNT\_NO Y KEY1  # TRAN\_ID Y KEY2  # TRAN\_DETAILS N  # AMT Y  #-----------------------------------------------------------------------------#  def process\_layout\_file(filename: str,  header: list,  layout: dict,  key\_list: list) -> None:  *""" """*  with open(filename) as layout\_file:  for line in layout\_file:  (field, match\_flag, \*key\_flag) = line.strip(*'\n'*).split()    # header contains all the field names that are specified  # in the layout file.  header.append(field)    match\_ind = (*'match\_flag'*, match\_flag.upper())    # If the field is a key field, store it.  if len(key\_flag) == 1:  key\_ind = (*'key\_flag'*, key\_flag[0])  else:  key\_ind = (*'key\_flag'*, *'null'*)    temp\_dict = dict([match\_ind, key\_ind])    # 'layout' is a map, storing each field's properties  layout[field] = temp\_dict    for key in layout.keys():  if layout[key][*'key\_flag'*] != *'null'*:  key\_fields[layout[key][*'key\_flag'*]] = key    # Sort the dictionary, so that, the Key indicators KEY1, KEY2, ...  # will be in order.    ordered\_keys = collections.OrderedDict(sorted(key\_fields.items()))    [key\_list.append(v) for k, v in ordered\_keys.items()]    return  #-----------------------------------------------------------------------------#  #-----------------------------------------------------------------------------#  def compare\_data(source: dict,  target: dict,  fields\_to\_be\_compared: list) -> None:  *"""*  *Compares Source and Target dictionaries and logs the differences in a file*  *"""*    log = open(*"file\_comparision.log"*, *'a'*)    # Identify the keys that exist in Source only  source\_only\_keys = [key for key in source.keys() if key not in target.keys()]    print(*'Following keys exist in Source file only:'*, file=log)  print(*'-----------------------------------------'*, file=log)    if len(source\_only\_keys) == 0:  print(*'%s'* % *'None'*, file=log)  else:  for key in source\_only\_keys:  print(*'%s'* % str(key), file=log)    print(*' '*, file=log)  # Identify the keys that exist in Target only  target\_only\_keys = [key for key in target.keys() if key not in source.keys()]    print(*'Following keys exist in Target file only: '*, file=log)  print(*'------------------------------------------'*, file=log)    if len(target\_only\_keys) == 0:  print(*'%s'* % *'None'*, file=log)  else:  for key in target\_only\_keys:  print(*'%s'* % str(key), file=log)    print(*' '*, file=log)    print(*'Data Mismatches:'*, file=log)  print(*'----------------'*, file=log)    # For keys that exist in both the files, match the Data fields and  # identify the differences.    for key in source.keys():  field\_flag = 0    if key in target.keys():  for field in fields\_to\_be\_compared:  if source[key][field] != target[key][field]:  if field\_flag == 0:  print(*'Key : %-50s Field: %-20s Base Data: %-30s new Data: %-30s'* \  % (key, field, source[key][field], target[key][field]), file=log)  field\_flag = 1  else:  print(*' %-50s Field: %-20s Base Data: %-30s new Data: %-30s'* \  % (*' '*, field, source[key][field], target[key][field]), file=log)  log.close()  return  #-----------------------------------------------------------------------------#  # Main Section of the program.  #-----------------------------------------------------------------------------#  def main():  pass    if \_\_name\_\_ == *'\_\_main\_\_'*:  source\_dict = {}  target\_dict = {}  header = []  layout = {}  key\_fields = {}  key\_list = []  fields\_to\_be\_compared = []    # Process Layout file and Identify file structure. Also, Identify what  # fields need to be compared; and the File's Key fields.  process\_layout\_file(*'layout.txt'*, header, layout, key\_list)    # This is the key that uniquely identifies a record  file\_key = tuple(key\_list)  # Gather the fields that need to be compared between the files  fields\_to\_be\_compared = [field for field in layout.keys()  if layout[field][*'match\_flag'*] == *'Y'*  and layout[field][*'key\_flag'*] == *'null'*]    print(*'Header :'*, header)  print(*'File Key :'*, file\_key)  print(*'Fields to be compared:'*, fields\_to\_be\_compared)    process\_data\_file(*'File1.txt'*, layout, source\_dict)    process\_data\_file(*'File2.txt'*, layout, target\_dict)    compare\_data(source\_dict, target\_dict, fields\_to\_be\_compared)    print(*'Execution Completed.'*) |