A). Write a program in your favorite language, to sort and filter a list of company names.

* Program should accept company names and return a filtered list of names based on input.
* Filtering can be partial or by complete names.
* Filtering should not be case sensitive.
* Ignore white spaces while filtering.
* Program allows for sorting the list of company names in alphabetical and reverse alphabetical order.
* Return your solution.

Solution: --  
---------------------  
(Complete code from local machine IDE)

# importing argparse lib for reading users input in a beautiful way

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument("namesList", nargs="+", type=str, help="List of unordered company names")

parser.add\_argument("--order", default='False', choices=['False', 'True'], help= "Order for ascending or descending")

args = parser.parse\_args()

def compNames\_filter(sequence, reverse=False):

  '''

  Input:

    sequence: list of unordered company names

    reverse: [default is False: ascending] ascending or descending alphabetical order

  Output:

    returns the sorted filtered list of company names

  '''

  # order False is for ascending order

  order = reverse

  if len(sequence) <=1:

    return sequence

  else:

    # declaring pivot point as median

    pivot = sequence.pop()

  left\_lower\_seq = []

  right\_high\_seq = []

  # iterating over each element in sequence

  for item in sequence:

    if item.lower().strip() > pivot.lower().strip():

      right\_high\_seq.append(item)

    else:

      left\_lower\_seq.append(item)

  if order == 'False':

    return compNames\_filter(left\_lower\_seq, order) + [pivot] + compNames\_filter(right\_high\_seq, order)

  else:

    return compNames\_filter(right\_high\_seq, order) + [pivot] + compNames\_filter(left\_lower\_seq, order)

if \_\_name\_\_ == '\_\_main\_\_':

  compList = (args.namesList)

  reverse = str(args.order)

  # print(reverse)

  print(compNames\_filter(compList, reverse))

B). Create and document test cases to test your program.

Answer :-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S#** | **Scenario** | **Input Data** | **Output** | **Result** |
| 1 | Case of names in output list should be retained. | ['openText', 'Google', 'OpenText'] | ['Google', 'openText', 'OpenText'] | PASS |
| 2 | Filtering should not be case sensitive | ['Facebook', 'OpenText', 'amazon'] | ['amazon', 'Facebook', 'OpenText'] | PASS |
| 3 | Whitespace should not be considered in filtering as priority | ['abctext', ' abcText ', ' Zebra', ' Xmas', 'amazon'] | ['abctext', ' abcText ', 'amazon', ' Xmas', ' Zebra'] | PASS |
| 4 | Should support for ascending order of names | ['abcText', 'zcdText', 'xcbText' , 'gdhtext'] | ['abcText', 'gdhtext', 'xcbText', 'zcdText'] | PASS |
| 5 | Should support for descending order of names | ['abcText', 'zcdText', 'xcbText' , 'gdhtext'] | ['zcdText', 'xcbText', 'gdhtext', 'abcText'] | PASS |
| 6 | Names with characters other than alphabets such as '$#!&\_' | ['abcText', 'zcdText', 'xcbText' , 'abc\_Text' , 'gdhtext' , '$%!&&Text'] | ['$%!&&Text', 'abc\_Text', 'abcText', 'gdhtext', 'xcbText', 'zcdText'] | PASS |
| 7 | Names with characters other than alphabets such as numbers | ['abcText', 'zcdText', 'xcbText' , 'abc\_Text', 'abc12\_Text', 'gdhtext'] | ['abc12\_Text', 'abc\_Text', 'abcText', 'gdhtext', 'xcbText', 'zcdText'] | PASS |