1. def anagram(in\_string,in\_list):

in\_string\_list = list(in\_string.lower())

in\_string\_list.remove(' ')

not\_exist\_list = []

output = False

for item in in\_list:

for ele in item:

if ele in in\_string\_list:

in\_string\_list.remove(ele)

else:

not\_exist\_list.append(ele)

if len(in\_string\_list) == 0 and len(not\_exist\_list) == 0:

output = True

print(f'anagram{in\_string,in\_list} ➞ {output}')

anagram("Justin Bieber", ["injures", "ebb", "it"])

anagram("Natalie Portman", ["ornamental", "pita"])

anagram("Chris Pratt", ["chirps", "rat"])

anagram("Jeff Goldblum", ["jog", "meld", "bluffs"])

Output:anagram('Justin Bieber', ['injures', 'ebb', 'it']) ➞ True

anagram('Natalie Portman', ['ornamental', 'pita']) ➞ True

anagram('Chris Pratt', ['chirps', 'rat']) ➞ False

anagram('Jeff Goldblum', ['jog', 'meld', 'bluffs']) ➞ False

1. def leaderboards(in\_list):

temp\_dict = {}

out\_list = []

for ele in in\_list:

temp\_dict[(ele['reputation']\*2)+ele['score']] = ele

for ele in sorted(temp\_dict.keys(),reverse=True):

out\_list.append(temp\_dict[ele])

print(f'leaderboards({in\_list}) ➞ {out\_list}')

leaderboards([

{ "name": "a", "score": 100, "reputation": 20 },

{ "name": "b", "score": 90, "reputation": 40 },

{ "name": "c", "score": 115, "reputation": 30 },

])

Output:

leaderboards([{'name': 'a', 'score': 100, 'reputation': 20}, {'name': 'b', 'score': 90, 'reputation': 40}, {'name': 'c', 'score': 115, 'reputation': 30}]) ➞ [{'name': 'c', 'score': 115, 'reputation': 30}, {'name': 'b', 'score': 90, 'reputation': 40}, {'name': 'a', 'score': 100, 'reputation': 20}]

1. def hangman(in\_string,in\_list):

out\_string = ''

for ele in range(len(in\_string)):

if in\_string[ele].lower() in in\_list:

out\_string += in\_string[ele]

elif in\_string[ele] in '"! ':

out\_string += in\_string[ele]

else:

out\_string += '-'

print(f'hangman{in\_string,in\_list} ➞ {out\_string}')

hangman("helicopter", ["o", "e", "s"])

hangman("tree", ["r", "t", "e"])

hangman("Python rules", ["a", "n", "p", "r", "z"])

hangman("He\"s a very naughty boy!", ["e", "a", "y"])

Output:

hangman('helicopter', ['o', 'e', 's']) ➞ -e---o--e-

hangman('tree', ['r', 't', 'e']) ➞ tree

hangman('Python rules', ['a', 'n', 'p', 'r', 'z']) ➞ P----n r----

hangman('He"s a very naughty boy!', ['e', 'a', 'y']) ➞ -e"- a -e-y -a----y --y!

1. def max\_collatz(in\_num):

out\_list = []

out\_list.append(in\_num)

temp\_in\_num = in\_num

while True:

if temp\_in\_num%2 == 0:

temp\_in\_num /= 2

else:

temp\_in\_num = (temp\_in\_num\*3)+1

out\_list.append(int(temp\_in\_num))

if temp\_in\_num ==1:

break

x= str(out\_list)

print(f'max\_collatz({in\_num}) ➞ {max(out\_list)}')

max\_collatz(10)

max\_collatz(32)

max\_collatz(85)

Output:

max\_collatz(10) ➞ 16

max\_collatz(32) ➞ 32

max\_collatz(85) ➞ 256

1. def digit\_sort(in\_list):

max\_len = len(str(max(in\_list)))

output = []

for item in range(max\_len,0,-1):

temp = []

for ele in in\_list:

if len(str(ele)) == item:

temp.append(ele)

output.extend(sorted(temp))

print(f'digit\_sort({in\_list}) ➞ {output}')

digit\_sort([77, 23, 5, 7, 101])

digit\_sort([1, 5, 9, 2, 789, 563, 444])

digit\_sort([53219, 3772, 564, 32, 1])

Output:

digit\_sort([77, 23, 5, 7, 101]) ➞ [101, 23, 77, 5, 7]

digit\_sort([1, 5, 9, 2, 789, 563, 444]) ➞ [444, 563, 789, 1, 2, 5, 9]

digit\_sort([53219, 3772, 564, 32, 1]) ➞ [53219, 3772, 564, 32, 1]