W2 Dictionary

Takeaway: to check if items are unique, use set()

def unique\_souvenir\_counts(souvenirs):  
 collec = {}  
 for sou in souvenirs:  
  
 if sou in collec:  
 collec[sou] += 1  
  
 else:  
 collec[sou] = 1  
 frequencies = list(collec.values())  
  
 if len(frequencies) == len(set(frequencies)):  
 return True  
 else:  
 return False

def prioritize\_observations(observed\_species, priority\_species):

# loop through the priority\_species

ob\_list = []

non\_list = []

ob\_list = [species for species in observed\_species if species in priority\_species]

non\_list = [species for species in observed\_species if species not in priority\_species]

ob\_list = sorted(ob\_list, key=priority\_species.index)

non\_list = sorted(non\_list, reverse = False)

ob\_list.extend(non\_list)

return ob\_list

def max\_species\_copies(raised\_species, target\_species):

# create a dictionary to count raised\_species

species\_count = {}

for species in raised\_species:

if species in species\_count:

species\_count[species] += 1

else:

species\_count[species] = 1

# count species in target\_species

target\_count = {}

for species in target\_species:

if species in target\_count:

target\_count[species] += 1

else:

target\_count[species] = 1

# find the min of target\_species and return it

min\_target = float('inf')

for species, item in target\_count.items():

if species not in species\_count::

return 0

target = species\_count[species]//target\_count[species]

if target < min\_target:

min\_target = target

return min\_target

def count\_unique\_species(ecosystem\_data):

# loop through the string and check if the char is numeric

clean = ""

for i in ecosystem\_data:

if i.isnumeric():

clean += i

else:

clean += " "

# strip white space

clean\_new = [int(i) for i in clean.split() if i.strip()]

# use set() and return the len

return len(set(clean\_new))

Takeaway: 1. Use string v.s. use a list. A list separate strings by a comma. 2. List comprehension: expression for I in iterable if condition 3. .split() and .strip() are string methods

def num\_equiv\_species\_pairs(species\_pairs):

# initialize an empty dictionary

# loop through each pair in the list, use set() to convert pair and add to the dictionary

unique\_species= {}

for species in species\_pairs:

species = tuple(sorted(species))

if species in unique\_species:

unique\_species[species]+=1

else:

unique\_species[species]=1

# remove keys whose value is one

unique\_species = {key:value for key, value in unique\_species.items() if value > 1}

# calculate with c\*(c-1)//2, add up numbers and return

num = 0

for i in unique\_species.values():

num += i\*(i-1)//2

return num

Key takeway: 1. The key in a dictionary must be unhashable (immutable), so can’t be a set, could be a tuple. 2. When looping through species\_pairs, each species is a list, using set() doesn’t help, should use sort before adding each to a dictionary. 3. Dictionary comprehension: {key: value for key, value in enumerate(iterable) if condition}

W3

Takeaway: fill from the back of a preallocated list. \*\*List doesn’t have .appendright()

Result = [0] \* len(engagements)

Position = -1

def engagement\_boost(engagements):

pt1 = 0

pt2 = len(engagements) - 1

# set position pointer

order = [0] \* len(engagements)

position = -1

# two pointers: compare the abs value of elements at the front and end,

while pt1 <= pt2:

if abs(engagements[pt1]) < abs(engagements[pt2]):

num = engagements[pt2]\*engagements[pt2]

order[position] = num

position -= 1

pt2 -=1

elif abs(engagements[pt1]) > abs(engagements[pt2]):

num = engagements[pt1]\*engagements[pt1]

order[position] = num

position -= 1

pt1 +=1

else:

num = engagements[pt1]\*engagements[pt1]

order[position] = num

position -= 1

pt1 += 1

pt2 -= 1

return order

Key: k is working on individual alphabet. And will add alphabet as separate element in a list [‘h’, ‘e’, ‘l’]. Think about word as a unit.

for j in lst:

for k in j[::-1]:

slicing i[::-1] reverse alphabet within a word

new\_lst = []

for i in lst:

i = i[::-1]

new\_lst.append(i)