In [6]:
```
#Dictionaries 4.1
#create dictionary called animals
animalSounds = {'cow':'muuuh', 'cat':'meoooow', 'lion':'roaaaar', 'dog':'wooo
f'} #KEY:VALUE
animalSounds['cat']
#NB with dictionaries comes first the KEY : then the VALUE
```

Out[6]:
```
'meoooow'
```

In [7]:
```
#Dictionaries 4.2
#add new entries to dictionary
animalSounds = {'cow':'muuuh', 'cat':'meoooow', 'lion':'roaaaar', 'dog':'wooo
f'}
animalSounds['chicken'] = 'piep'
print(animalSounds)
```

```
{'cow': 'muuuh', 'cat': 'meoooow', 'lion': 'roaaaar', 'dog': 'wooof', 'chicke
n': 'piep'}
```

In [18]:
```
#Sets
#5. Create two sets manually. One containing odd numbers from 1 to 19, the oth
er containing all numbers
#from 1 to 10, then: 5.1. Output one set which is a union of the two sets.
set1 = {1,3,5,7,9,11,13,15,17,19}
set2 = {1,2,3,4,5,6,7,8,9,10}
print("both sets")
print(set1, set2)
print()
#5.1 Output Union: both sets, duplicates showing only once
union = set1 | set2
print("union")
print(union)
print()
#5.2 Intersection: containing items appearing in both
intersection = set1 & set2
print("intersection")
print(intersection)
print()
#5.3 difference in sets, containing just those elements in one set
set1minus2 = set1 - set2
print("set 1 minus duplicates in set 2")
print(set1minus2)
print()
#5.4 symmetric difference, items appearing in only one set = all but the inter
sect
justonce = set1 ^ set2
print("symmetric difference")
print(justonce)
```

```
both sets
{1, 3, 5, 7, 9, 11, 13, 15, 17, 19} {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

union
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19}

intersection
{1, 3, 5, 7, 9}

set 1 minus duplicates in set 2
{11, 13, 15, 17, 19}

symmetric difference
{2, 4, 6, 8, 10, 11, 13, 15, 17, 19}
```

In [20]:
```
colors = ["green", "blue", "yellow", "purple"]
for c in colors: #for loop that prints all elements of array, can be c, w what
aver
    print(c)
```

```
green
blue
yellow
purple
```

In [21]:
```
colors = ["green", "blue", "yellow", "purple"]
for w in colors: #for loop to print all elements and count letters
    print(w, len(w))
```

```
green 5
blue 4
yellow 6
purple 6
```

In [13]:
```
#6. Write a For Loop that will iterate through a list which contains several c
olours as strings (all lowercase).
#The loop should capitalise the first letter each string, and output each colo
ur with the capitalised first letter.
colors = ["green", "blue", "yellow", "purple"]
[x.capitalize() for x in colors] #for loop that capitalizes all first letters
 in list
```

Out[13]:
```
['Green', 'Blue', 'Yellow', 'Purple']
```

In [33]:
```
#7. Write a series of statements that will remove any duplicates from the foll
owing list,
#and return them in order: 1,1,1,1,5,4,3,6,7,3,2,1.
listOfUnordered = [1,1,1,1,5,4,3,6,7,3,2,1]
listOfUnordered.sort()
list2 = dict.fromkeys(listOfUnordered).keys() #remove duplicates with dict.key
s
print(listOfUnordered)
print(list2)
```

```
[1, 1, 1, 1, 1, 2, 3, 3, 4, 5, 6, 7]
dict_keys([1, 2, 3, 4, 5, 6, 7])
```

In [40]:
```
#7. Write a series of statements that will remove any duplicates from the foll
owing list,
#and return them in order: 1,1,1,1,5,4,3,6,7,3,2,1. Note: use a For Loop for t
his.
listOfUnordered = [1,1,1,1,5,4,3,6,7,3,2,1]
orderedList = []
for i in range(len(listOfUnordered)):
    if listOfUnordered[i] not in orderedList:
        orderedList.append(listOfUnordered[i])
orderedList.sort()
print(orderedList)
```

```
[1, 2, 3, 4, 5, 6, 7]
```

In [36]:
```
#8. Take the sentence: “The quick brown fox jumps over the lazy dog.”, and:
#8.1. Write one statement which will reverse the order of all characters in th
e sentence
#as follows: 'god yzal eht revo spmuj xof nworb kciuq ehT'
#UNSING A LOOP
foxOld = "The quick brown fox jumps over the lazy dog."
def reverse(foxOld):
    newFox = ""
    for i in foxOld:
        newFox = i + newFox
    return newFox
print(foxOld)
print("now executing function 'reverse' USING A LOOP:")
print(reverse(foxOld))
```

```
The quick brown fox jumps over the lazy dog.
now executing function 'reverse' USING A LOOP:
.god yzal eht revo spmuj xof nworb kciuq ehT
```

In [34]:
```
#8. Take the sentence: “The quick brown fox jumps over the lazy dog.”, and:
#8.1. Write one statement which will reverse the order of all characters in th
e sentence
#as follows: 'god yzal eht revo spmuj xof nworb kciuq ehT'
#USING RECURSION
foxOld = "The quick brown fox jumps over the lazy dog."
def reverse(foxOld):
    if len(foxOld) == 0:
        return foxOld
    else:
        return reverse(foxOld[1:]) + foxOld[0]
print(foxOld)
print("now executing function 'reverse' USING RECURSION:")
print(reverse(foxOld))
```

```
The quick brown fox jumps over the lazy dog.
now executing function 'reverse' USING RECURSION:
.god yzal eht revo spmuj xof nworb kciuq ehT
```

In [10]:
```
#8. Take the sentence: “The quick brown fox jumps over the lazy dog.”, and:
#8.1. Write one statement which will reverse the order of all characters in th
e sentence
#as follows: 'god yzal eht revo spmuj xof nworb kciuq ehT'
#USING EXTENDED SLICE SYNTAX
foxOld = "The quick brown fox jumps over the lazy dog."
def reverse(foxOld):
    foxOld = foxOld[::-1]
    return foxOld
print(foxOld)
print("now executing function 'reverse' USING EXTENDED SLICE SYNTAX:")
print(reverse(foxOld))
```

```
The quick brown fox jumps over the lazy dog.
now executing function 'reverse' USING EXTENDED SLICE SYNTAX:
.god yzal eht revo spmuj xof nworb kciuq ehT
```

In [8]:
```
#8.2. Write a series of statements that will reverse the order of the words.
#Hint: use the string.split() function to separate the sentence into words.
foxOld = "The quick brown fox jumps over the lazy dog."
def reverseWords(foxOld):
    inputWords = foxOld.split(" ")
    inputWords=inputWords[-1::-1]
    output = ' '.join(inputWords)
    return output
print(reverseWords(foxOld))
```

```
dog. lazy the over jumps fox brown quick The
```

In [38]:
```
# 8.3. Write a series of statements that will take all of the characters in th
e original sentence, remove any duplicates,
# and return them in descending (Z -> A) order.
#STEPHANIES CODE
foxOld = "The quick brown fox jumps over the lazy dog."
def takeCharacters(x):
    return list(dict.fromkeys(x))

newList = takeCharacters(foxOld.upper().replace(" ", " "))
newList.sort(reverse=True)
print(newList)
```

```
['Z', 'Y', 'X', 'W', 'V', 'U', 'T', 'S', 'R', 'Q', 'P', 'O', 'N', 'M', 'L',
'K', 'J', 'I', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A', '.', ' ']
```

In [11]:
```
# 8.3. Write a series of statements that will take all of the characters in th
e original sentence, remove any duplicates,
# and return them in descending (Z -> A) order.
#GILLIAN's CODE
foxOld = "The quick brown fox jumps over the lazy dog."
upperFox = foxOld.upper()
list1 = []
for x in range (0,len(foxOld)):
    y = upperFox[x]
    if y not in list1:
        list1.append(y)
        list1.sort(reverse=True)
print(foxOld)
print(upperFox)
print(list1)
```

```
The quick brown fox jumps over the lazy dog.
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
['Z', 'Y', 'X', 'W', 'V', 'U', 'T', 'S', 'R', 'Q', 'P', 'O', 'N', 'M', 'L',
'K', 'J', 'I', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A', '.', ' ']
```

In [32]:
```
# 8.4. Finally, take the list of characters you generated, and output another
 list with all vowels capitalised.
# Hint: you may wish to use the “enumerate” function on your list.
print(list1)
# Python code to count and display number of vowels. Simply using for and comp
aring it with a  string containg all vowels
def Check_Vow(string, vowels):
    final = [each for each in string if each in vowels]
    print(len(final))
    print(final)
# Driver Code
string = list1
vowels = "AeEeIiOoUu"
Check_Vow(string, vowels);
# Driver 2
string = "The quick brown fox jumps over the lazy dog."
vowels = "AeEeIiOoUu"
Check_Vow(string, vowels);
```

```
['Z', 'Y', 'X', 'W', 'V', 'U', 'T', 'S', 'R', 'Q', 'P', 'O', 'N', 'M', 'L',
'K', 'J', 'I', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A', '.', ' ']
5
['U', 'O', 'I', 'E', 'A']
10
['e', 'u', 'i', 'o', 'o', 'u', 'o', 'e', 'e', 'o']
```

In [23]:
```
# 8.4 # Count vowels in a different way. Using dictionary
def Check_Vow(string, vowels):
    # casefold has been used to ignore cases
    string = string.casefold()
    # Forms a dictionary with key as a vowel and the value as 0
    count = {}.fromkeys(vowels, 0)
    # To count the vowels
    for character in string:
        if character in count:
            count[character] += 1
    return count
# Driver Code
vowels = 'aeiou'
string = 'The quick brown fox jumps over the lazy dog.'
print (Check_Vow(string, vowels))
```

```
{'a': 1, 'e': 3, 'i': 1, 'o': 4, 'u': 2}
```