**1.**

Function listMaximum(list)

maximumNumber = 0

**For** number **in** list:

**If** number > maximumNumber:

maximumNumber = number

**Return** maximumNumber

**2.**

def create\_list():

  import random

  MAX\_LEN = 50

  my\_list = []

  for i in range(MAX\_LEN):

    my\_list.append(random.randint(1,99))

  return my\_list

print (create\_list())

<https://colab.research.google.com/drive/158UGGfURs3DMsEf7uo3VCdUGhCSyxzLW?usp=sharing>

**3.**

# Question 3

names = ['Jennifer', 'Albatross', 'Justin', 'Dave', 'Shankarnarayan', 'Ezra', 'Alice', 'Kwabena']

for name in names:

  length = str(len(name))

  print (name + " has " + length + " characters.")

<https://colab.research.google.com/drive/158UGGfURs3DMsEf7uo3VCdUGhCSyxzLW?usp=sharing>

**4.**

names = ['Jennifer', 'Albatross', 'Justin', 'Dave', 'Shankarnarayan', 'Ezra', 'Alice', 'Kwabena']

for name in names:

  nameLength = len(name)

  listLength = len(names)

  nameCompare = []

  newNames = []

  for i in range(listLength):

    otherName = len(names[i])

    if otherName <= nameLength:

      nameCompare.append(i)

  for l in nameCompare:

    newNames.append (names[l])

  numberOfNames = str(len(nameCompare))

  print (name + " has " + numberOfNames + " names that are shorter in length.")

  print (newNames)

<https://colab.research.google.com/drive/158UGGfURs3DMsEf7uo3VCdUGhCSyxzLW?usp=sharing>

**5. –** I cannot figure this one out.

names = ['Jennifer', 'Albatross', 'Justin', 'Dave', 'Shankarnarayan', 'Ezra', 'Alice', 'Kwabena']

n = 0

while n <= len(names):

  nameLength = len(name)

  listLength = len(names)

  nameCompare = []

  newNames = []

  i = 0

  while i <= listLength:

    otherName = len(names[i])

    if otherName <= nameLength:

      nameCompare.append(i)

    i = i + 1

  for l in nameCompare:

    newNames.append (names[l])

  numberOfNames = str(len(nameCompare))

  print (name + " has " + numberOfNames + " names that are shorter in length.")

  print (newNames)

  n = n + 1

<https://colab.research.google.com/drive/158UGGfURs3DMsEf7uo3VCdUGhCSyxzLW?usp=sharing>

6.

leaderboardList = []

def leaderboard(number):

  MAX\_LEN = 5

  if len(leaderboardList) < 5:

    leaderboardList.append(number)

  else:

    leaderboardList.sort()

    if number > leaderboardList[4]:

      print ("New High Score Created!!")

    if number > leaderboardList[0] and number <= leaderboardList[4]:

      print ("New Number Added to the List")

    leaderboardList.append(number)

    leaderboardList.sort()

    leaderboardList.reverse()

    leaderboardList.pop()

import time

import random

for i in range (10):

  time.sleep(3)

  number = random.randint (1,50)

  print ("New number generated:" + str(number))

  leaderboard(number)

  print (sorted(leaderboardList))

<https://colab.research.google.com/drive/158UGGfURs3DMsEf7uo3VCdUGhCSyxzLW?usp=sharing>

**Extra Credit 1.**

Bubble sort is way of comparing the numbers in a list sequentially to see each number is greater than the next, and if so, swap the numbers. This process continues through list until all the numbers are sorted. For example, in a list of 2, 6, 4, 9, and 1, the program initially compares the first number, 2, to the next number, 6. It recognizes that 2 is not greater than 6 and then repeats the for loop to examine if the next number, 6, is greater than the number to the right, 4. Because 6 is greater than 4, the positions of the numbers will be swapped. This process continues through the list until the final number comparisons, and reaches a sorted list of 1, 2, 4, 6, 9.

list1 = [2, 6, 4, 9, 1]

list2 = [3, 67, 9, 34, 2, 99]

def bubbleSort(list):

  for i in range(len(list)):

    for number in range(0, (len(list) - i - 1)):

      nextNumIndex = number + 1

      if list[number] > list[nextNumIndex]:

        (list[number], list[nextNumIndex]) = (list[nextNumIndex], list[number])

  print (list)

bubbleSort(list1)

bubbleSort(list2)

<https://colab.research.google.com/drive/158UGGfURs3DMsEf7uo3VCdUGhCSyxzLW?usp=sharing>

**Extra Credit 2.**

Stacks are a type of data structure that follow a linear and ordered procedure for storing the data, and replacing the data. Because of this ordered approach in how data is added and removed it is often referred to as First In Last Out or Last In First Out. The first data to enter the data structure remains the longest, and the most recent data in the structure is returned first. Common examples of stacks are the Back and Forward buttons in a web browser, and reversing the order of a string. The browser keeps record of the sites you visited most recently, but contains the data going back many sites. The back or forward button will take you to the most recently visited in the list. Reversing a string is done by taking the last letter to be typed, and working back to the next recent, and so on until the first letter is reached.

A queue is similar to a stack data structure in that the data is also stored linearly, but its retrieval is performed in the opposite manner – the data to enter the structure first is then the first to be returned (First In First Out). Two common examples of how queues are used in computer programs are print spoolers and an email queue. The first print jobs to enter the print spooler are the first to be submitted to the printer, and the first emails sent to the queue are the first to be processed

Source Code:

<https://colab.research.google.com/drive/158UGGfURs3DMsEf7uo3VCdUGhCSyxzLW?usp=sharing>