Techno India University,W.B.
Department of Computer Applications
Subject:-Tools and Techniques of Programming using Python
ASSIGNMENT-5
Topic:Python Data Structures

Last Date of Submission: 28.02.2023

Develop Python scripts to solve following problems:

1. Write a program that asks the user to enter a list of integers. Do the following:
   (a) Print the total number of items in the list.
   (b) Print the last item in the list.
   (c) Print the list in reverse order.
   (d) Print Yes if the list contains a 5 and No otherwise.
   (e) Print the number of fives in the list.
   (f) Remove the first and last items from the list, sort the remaining items, and print the result.
   (g) Print how many integers in the list are less than 5.
2. Write a program to generate a list of 20 random numbers between 1 and 100 and then
   (a) Print the list.
   (b) Print the average of the elements in the list.
   (c) Print the largest and smallest values in the list.
   (d) Print the second largest and second smallest entries in the list
   (e) Print how many even numbers are in the list.

3. Start with the list [7,9,11]. Do the following:
   (a) Set the second entry (index 1) to 17
   (b) Add 7, 5, and 9 to the end of the list
   (c) Remove the first entry from the list
   (d) Sort the list
   (e) Double the list
   (f) Insert 25 at index 3
   (g) Show the final list.
4. Develop a Python script that prompts the user to enter a list containing numbers between 11 and 22. Then replace all of the entries in the list that are greater than 19 with 19.
5. Develop a Python Script that prompts the user to enter a list of strings and then to create a new list that consists of those strings with their first characters removed.
6. Develop a Python Script to find and show the ASCII value of each character of a string.
7. Develop a Python Script to create the following lists:
   (a) A list consisting of the integers 0 through 59
   (b) A list containing the squares of the integers 1 through 49.
   (c) The list **['a','bb','ccc','dddd', ...]** that ends with 26 copies of the letter z.
8. Write a program that takes any two lists L and M of the same size and adds their elements together to form a new list N whose elements are sums of the corresponding elements in L and M. For instance, if L=[3,1,4] and M=[1,5,9], then N should equal [4,6,13].
9. Write a program that asks the user for an integer and creates a list that consists of the factors of that integer.
10. When playing games where you have to roll two dice, it is nice to know the odds of each roll. For instance, the odds of rolling a 12 are about 3%, and the odds of rolling a 7 are about 17%. You can

compute these mathematically, but if you don't know the math, you can write a program to do it. To do this, your program should simulate rolling two dice about 10,000 times and compute and print out the percentage of rolls that come out to be **2, 3, 4, ..., 12.**

11. Write a program that rotates the elements of a list so that the element at the first index moves to the second index, the element in the second index moves to the third index, etc., and the element in the last index moves to the first index.

12. Develop a python script to create the list below, which consists of ones separated by increasingly many zeroes. The last two ones in the list should be separated by ten zeroes.
[1,1,0,1,0,0,1,0,0,0,1,0,0,0,0,1,....]

13. Write a program that generates 100 random integers that are either 0 or 1 and then finds the longest run of zeros, the largest number of zeros in a row. For instance, the longest run of zeros in **[1,0,1,1,0,0,0,0,1,0,0] is 4.**

14. Write a program that removes any repeated items from a list so that each item appears at most once. For instance, the list **[1,1,2,3,4,3,0,0]** would become **[1,2,3,4,0].**

15. Write a program using the concept of list that asks the user to enter a length in feet. The program should then give the user the option to convert from feet into inches, yards, miles, millimeters, centimeters, meters, or kilometers. For instance, if the user enters a 1, then the program converts to inches, if they enter a 2, then the program converts to yards, etc.

16. There is a probably unbreakable cipher called a one-time pad. The way it works is as described below: you shift each character of the message by a random amount between 1 and 26 characters, wrapping around the alphabet if necessary. For instance, if the current character is y and the shift is 5, then the new character is d. Each character gets its own shift, so there needs to be as many random shifts as there are characters in the message. As an example, suppose the user enters secret. The program should generate a random

shift between 1 and 26 for each character. Suppose the randomly generated shifts are 1,3, 2, 10, 8 and 2. The encrypted message would then be **thebmv.**

(a) Write a program that asks the user for a message and encrypts the message using the one-time pad. First convert the string to lowercase. Any spaces and punctuation in the string should be left unchanged. For example, **Secret!!! becomes thebmv!!! using the shifts above**.

(b) Write a program to decrypt a string encrypted as above.
The reason it is called a one-time-pad is that the list of random shifts should only be used once. It becomes easily breakable if the same random shifts are used for more than one message. Moreover, it is only probably unbreakable if the random numbers are truly random, and the numbers generated by **randint** are not truly random. For this problem, just use **randint**.

17. Write a program that asks the user to enter some text and then counts how many articles are there in the text. Articles are the words 'a', 'an', and 'the'.

18. Write a program that allows the user to enter five numbers (read as strings).
Create a string that consists of the user's numbers separated by plus signs. For instance, if the user enters 2, 5, 11, 33, and 55, then the string should be **'2+5+11+33+55'.**

(a) Ask the user to enter a sentence and print out the third word of the sentence.

(b) Ask the user to enter a sentence and print out every third word of the sentence.

19. write a program that asks the user to enter a sentence and then randomly rearranges the words of the sentence. Don't worry about getting punctuation or capitalization correct. (b) Do the above problem, but now make sure that the sentence starts with a capital, that the original first word is not capitalized if it comes in the middle of the sentence, and that the period is in the right place.

20. Write a simple quote-of-the-day program. The program should contain a list of quotes, and when the user runs the program, a randomly selected quote should be printed.

21. Write a simple lottery drawing program. The lottery drawing should consist of six different numbers between 1 and 48.

22. Write a program that estimates the average number of drawings it takes before the user's numbers are picked in a lottery that consists of correctly picking six different numbers that are between 1 and 10. To do this, run a loop 1000 times that randomly generates a set of user numbers and simulates drawings until the user's numbers are drawn. Find the average number of drawings needed over the 1000 times the loop runs.

23. Write a program that simulates drawing names out of a hat. In this drawing, the number of hat entries each person gets may vary. Allow the user to input a list of names and a list of how many entries each person has in the drawing, and print out who wins the drawing.

24. Write a simple quiz game that has a list of ten questions and a list of answers to those questions. The game should give the player four randomly selected questions to answer. It should ask the questions one-by-one, and tell the player whether they got the question right or wrong. At the end it should print out how many out of four they got right.

25. Write a censoring program. Allow the user to enter some text and your program should print out the text with all the curse words starred out. The number of stars should match the length of the curse word. For the purposes of this program, just use the"curse" words darn, dang, freakin, heck, and shoot. Sample output is below:
Enter some text: Oh shoot, I thought I had the dang problem figured out. Darn it. Oh well, it was a heck of a freakin try.
Oh ***, I thought I had the **** problem figured out. **** it. Oh well, it was a **** of a ****** try.**

26. Use the choice method to create a random anagram of a string.

27. Write a program that gets a string from the user containing a potential telephone number. The program should print Valid if it decides the phone number is a real phone number, and Invalid otherwise. A phone number is considered valid as long as it is written in the form abc-def-hijk or 1-abc-def-hijk. The dashes must be included, the phone number should contain only numbers and dashes, and the number of digits in each group must be correct. Test your program with the output shown below.
Enter a phone number: 1-301-447-5820
Valid
Enter a phone number: 301-447-5820
Valid
Enter a phone number: 301-4477-5820
Invalid
Enter a phone number: 3X1-447-5820
Invalid
Enter a phone number: 3014475820
Invalid

28. Let L be a list of strings. Write list comprehensions that create new lists from L for each of the following:
    (a) A list that consists of the strings of s with their first characters removed
    (b) A list of the lengths of the strings of s
    (c) A list that consists of only those strings of s that are at least three characters long

29. Use a list comprehension to produce a list that consists of all palindromic numbers between 100 and 1000.

30. Use a list comprehension to create the list below, which consists of ones separated by increasingly many zeroes. The last two ones in the list should be separated by ten zeroes. [1,1,0,1,0,0,1,0,0,0,1,0,0,0,0,1,....]

31. Let L=[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]. Use a list comprehension to produce a list of the gaps between consecutive entries in L. Then find the maximum gap size and the percentage of gaps that have size 2.

32. Write a program that finds the average of all of the entries in a 4×4 list of integers.

33. Write a program that creates a 10×10 list of random integers between 1 and 100. Then do the following:

    (a) Print the list.

    (b) Find the largest value in the third row.

    (c) Find the smallest value in the sixth column.

34. Write a program that creates and prints an 8×8 list whose entries alternate between 1 and 2 in a checkerboard pattern, starting with 1 in the upper left corner.

35. Write a program that checks to see if a 4×4 list is a magic square. In a magic square, every row, column, and the two diagonals add up to the same value.

36. Write a program that asks the user to enter a length. The program should ask them what unit the length is in and what unit they would like to convert it to. The possible units are inches, yards, miles, millimeters, centimeters, meters, and kilometers. While this can be done with 25 if statements, it is shorter and easier to add on to if you use a two-dimensional list of conversions, so please use lists for this problem.

37. The following is useful as part of a program to play Battleship. Suppose you have a 5×5 list that consists of zeroes and ones. Ask the user to enter a row and a column. If the entry in the list at that row and column is a one, the program should print Hit and otherwise it should print Miss. 38. This exercise is useful in creating a Memory game. Randomly generate a 6×6 list of assorted characters such that there are exactly two of each character. An example is shown below:
    @ 5 # A A ! 5 0 b @ $ z $ N x ! N z 0 - + # b : - : + c c x

38. The following is useful in implementing computer players in a number of different games. Write a program that creates a 5×5 list

consisting of zeroes and ones. Your program should then pick a random location in the list that contains a zero and change it to a one. If all the entries are one, the program should say so. [Hint: one way to do this is to create a new list whose items are the coordinates of all the ones in the list and use the choice method to randomly select one. Use a two-element list to represent a set of coordinates.]

39. Here is an old puzzle question you can solve with a computer program. There is only one five-digit number n that is such that every one of the following ten numbers shares exactly one digit in common in the same position as n. Find n.
01265,12171,23257,34548,45970,56236,67324,78084,89872,99414

40. We usually refer to the entries of a two-dimensional list by their row and column, like below on the left. Another way is shown below on the right. **(0,0) (0,1) (0,2) 0 1 2 (1,0) (1,1) (1,2) 3 4 5 (2,0) (2,1) (2,2) 6 7 8**

    - Write some code that translates from the left representation to the right one. The // and % operators will be useful. Be sure your code works for arrays of any size.
    - Write some code that translates from the right representation to the left one.

41. Write a program that repeatedly asks the user to enter product names and prices. Store all of these in a dictionary whose keys are the product names and whose values are the prices. When the user is done entering products and prices, allow them to repeatedly enter a product name and print the corresponding price or a message if the product is not in the dictionary.

42. Using the dictionary created in the previous problem, allow the user to enter a dollar amount and print out all the products whose price is less than that amount.
    For this problem, use the dictionary from the beginning of this chapter whose keys are month names and whose values are the number of days in the corresponding months.

- Ask the user to enter a month name and use the dictionary to tell them how many days are in the month. (b) Print out all of the keys in alphabetical order.
- Print out all of the months with 31 days.
- Print out the (key-value) pairs sorted by the number of days in each month
- Modify the program from part (a) and the dictionary so that the user does not have to know how to spell the month name exactly. That is, all they have to do is spell the first three letters of the month name correctly.

43. Write a program that uses a dictionary that contains ten user names and passwords. The program should ask the user to enter their username and password. If the username is not in the dictionary, the program should indicate that the person is not a valid user of the system. If the username is in the dictionary, but the user does not enter the right password, the program should say that the password is invalid. If the password is correct, then the program should tell the user that they are now logged in to the system.

44. Repeatedly ask the user to enter a team name and the how many games the team won and how many they lost. Store this information in a dictionary where the keys are the team-names and the values are lists of the form **[wins, losses].**

    - Using the dictionary created above, allow the user to enter a team name and print out the team's winning percentage.
    - Using the dictionary, create a list whose entries are the number of wins of each team.
    - Using the dictionary, create a list of all those teams that have winning records.

45. Repeatedly ask the user to enter game scores in a format like team 1 score 1-team 2 score 2. Store this information in a dictionary where the keys are the team names and the values are lists of the form **[wins, losses].**

46. Create a 5×5 list of numbers. Then write a program that creates a dictionary whose keys are the numbers and whose values are the how many times the number occurs. Then print the three most common numbers.

47. Using the card dictionary from earlier in this chapter, create a simple card game that deals two players three cards each. The player with the highest card wins. If there is a tie, then compare the second highest card and, if necessary, the third highest. If all three cards have the same value, then the game is a draw.

48. Using the card dictionary from earlier in the chapter, deal out three cards. Determine the following:

    - If the three cards form a flush (all of the same suit)
    - If there is a three-of-a-kind (all of the same value)
    - If there is a pair, but not three-of-a-kind
    - If the three cards form a straight (all in a row, like **(2, 3, 4) or (10, Jack, Queen)**)

49. Using the card dictionary from earlier in the chapter run a Monte Carlo simulation to estimate the probability of being dealt a flush in a five card hand..

50. we met the substitution cipher. This cipher replaces every letter with a different letter. For instance every a might be replaced with an e, every b might be replaced with an a, etc. Write a program that asks the user to enter two strings. Then determine if the second string could be an encoded version of the first one with a substitution cipher. For instance, CXYZ is not an encoded version of BOOK because O got mapped to two separate letters. Also,CXXK is not an encoded version of BOOK, because Kgot mapped to itself. On the other hand, CXXZ would be an encoding of BOOK.

51. Below are the notes used in music:
    C C# D D# E F F# G G# A A# B
    The notes for the C major chord are C,E,G.A mathematical way to get this is that E is 4 steps past C and G is 7 steps past C. This works for any base. For example, the notes for D major are D, F#, A.

We can represent the major chord steps as a list with two elements: [4,7]. The corresponding lists for some other chord types are shown below: Minor [3,7] Dominant seventh [4,7,10] Augmented fifth [4,8] Minor seventh [3,7,10] Minor fifth [4,6] Major seventh [4,7,11] Major sixth [4,7,9] Diminished seventh [3,6,10] Minor sixth [3,7,9] Write a program that asks the user for the key and the chord type and prints out the notes of the chord. Use a dictionary whose keys are the (musical) keys and whose values are the lists of steps.

52. Suppose you are given the following list of strings: **L = ['aabaabac', 'cabaabca', 'aaabbcba', 'aabacbab', 'acababba']**
Patterns like this show up in many places, including DNA sequencing. The user has a string of their own with only some letters filled in and the rest as asterisks. An example is **a***a****.** The user would like to know which of the strings in the list fit with their pattern. In the example just given, the matching strings are the first and fourth. One way to solve this problem is to create a dictionary whose keys are the indices in the user's string of the non-asterisk characters and whose values are those characters.
Write a program implementing this approach(or some other approach)to find the strings that match a user-entered string.

53. Dictionaries provide a convenient way to store structured data. Here is an example dictionary: d=[{'name':'Todd', 'phone':'555-1414', 'email':'todd@mail.net'}, {'name':'Helga', 'phone':'555-1618', 'email':'helga@mail.net'}, {'name':'Princess', 'phone':'555-3141', 'email':''}, {'name':'LJ', 'phone':'555-2718', 'email':'lj@mail.net'}]
Write a program that reads through any dictionary like this and prints the following:

    (a) All the users whose phone number ends in an 8

    (b) All the users that don't have an email address listed

54. Use a dictionary whose keys are the names of the time zones and whose values are offsets from the Eastern time zone. Write a program that converts a time from one time zone to another. The user enters the time in the usual Indian way, such as 3:48pm or 11:26am. The first time zone the user enters is that of the original time and

the second is the desired time zone. The possible time zones are Eastern, Central, Mountain, or Pacific.
Time: 11:48pm Starting zone: Pacific Ending zone: Eastern 2:48am

55. Write a program that converts decimal numbers into Roman numerals

56. Generalize the **Tic-tac-toe game to an m,n,k-game** in which two players alternate placing stones of their own color on an m×n board, with the goal of getting k of their own color in a row.