Dynamic List processing

Write the code for the following activities. Put all code into one file. You can comment sections out as you complete and test them so you don’t have to keep running the same code.

1. Prompt the user for a number which will represent the number of items in a list. Then use a for loop to add that many integers to the list. For example, if the use enters 3, the for loop should get 3 integers from the user and load them into the list. Then display the list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Number of items  List of integers | def add\_numbers\_to\_list1(number\_list):  x = int(input("How many items are in the list? "))  for \_ in range(x):  i = int(input("Please enter a number: "))  number\_list.append(i)  return number\_list | List of entered numbers  List of integers |
|  |  |  |

1. Insert the score of 99 at position 1 within the list. Display the updated list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List of numbers  Score to insert (99) | Def add numbers to list2 (number list):  Number list.insert (0,99) | Updated list with 99 at position 1. |

1. Replace the value of 99 with the value 100. Display the updated list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List of numbers  value of 99 with the value 100 | Def add numbers to list3 (number list):  Number list [0] = 100 | Updated list with the value 99 replaced by 100. |

1. Create a second list with the values 500, 600, 700, 800, 900. Display this list. Extend the first list with this second list. Display the first list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List of numbers | Def add number to list4 (number list)  Number list. extend(number list 2) |  |
| Values 500, 600, 700, 800, 900 |  |  |
|  |  |  |

1. Remove the value 800 from the first list. Display the first list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List of numbers | Def add number to list5 (number list):  Number list.remove(800) | Display the first list without 800. |
|  |  |  |

1. Remove the third item from the first list. Display the first list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List of numbers | Def add number to the list6(number list) | The updated first list. |
|  | Number list. remove (number list [2]) |  |

1. Create a list of grades: grades =["A", "B", "C", "A", "A", "C"]

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List grades ["A", "B", "C", "A", "A", "C"] |  |  |
|  |  |  |

1. Display a count of the number of A grades.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List grades ["A", "B", "C", "A", "A", "C"] | Def count a grade a(grades\_list):  print("There are", grades\_list.count("A"), "A grades.") | count of the number of A grades. |
|  |  |  |

1. Display the index (position) of the first B grade.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List grades | Def count a gradeb (grades\_list):  print("There are", grades\_list.count("B"), "B grades.") | count of the number of B grades. |
|  |  |  |

1. Look for grade of F in the grades list. Display a message that F is not in the list. (Do not let the code generate an error).

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| List grades | Def check f grade (grades list)  If “F” in grades list:  Print (“There is F in the list.”)  Else: print (“Grade F is not in the list.”) | count of the number of f grades. |
|  |  |  |

1. Clear (but do not delete) the second list of integers. Display the list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Number list  Second list | Def add number to list11 (number list)  Number list2.clear() | Display the empty second list. |
|  |  |  |
|  |  |  |

1. Delete the second list. Try to display it. (should get an error because the list no longer exists.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Number list  Second list | def delete second list():  del second\_list ] | error |
|  | Print number list 2 |  |

1. Create a list of players in this order (“Rizzo”, “Davis”, “Baez”, “Happ”, “Bryan”)

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Players (“Rizzo”, “Davis”, “Baez”, “Happ”, “Bryan”) |  |  |
|  |  |  |

1. Sort the list of players. Display the sorted list.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Players (“Rizzo”, “Davis”, “Baez”, “Happ”, “Bryan”) | def player sort(players):  players.sort()  for i in players:  print(i) | Display the sorted players list. |
|  |  |  |

1. Make a copy of the list of players called players2. Display players2.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Players (“Rizzo”, “Davis”, “Baez”, “Happ”, “Bryan”) | Def copyplayers(players,players2): | copied list players2. |
|  | y=players.copy()  players2.append(y)  return players2  for y in players2:  print(y) |  |
|  |  |  |
|  |  |  |

1. Reverse the order of players2. Display players, then display players2.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Players (“Rizzo”, “Davis”, “Baez”, “Happ”, “Bryan”) | def backwards(players,players2):  players.sort()  for t in players:  print(t)  z=players.copy()  players2.append(z)  w=reversed(players)  print(list(w)) | Reverse order of players2 |
|  |  |  |
|  |  |  |
|  |  |  |