Solutions,26jan,2022

1. Write a Python program to create a deque and append few elements to the left and right, then remove some elements from the left, right sides and reverse the deque.

Solution:

```
>>> import collections
>>> # Create a deque
>>>deque_colors=collections.deque(["Red","Green","White"])
>>> print(deque_colors)
>>> # Append to the left
>>> print("\nAdding to the left: ")
>>> deque_colors.appendleft("Pink")
>>> print(deque_colors)
>>> # Append to the right
>>> print("\nAdding to the right: ")
>>> deque_colors.append("Orange")
>>> print(deque_colors)
>>> # Remove from the right
>>> print("\nRemoving from the right: ")
>>> deque_colors.pop()
>>> print(deque_colors)
>>> # Remove from the left
>>> print("\nRemoving from the left: ")
>>> deque_colors.popleft()
>>> print(deque_colors)
>>> # Reverse the dequeue
>>> print("\nReversing the deque: ")
>>> deque_colors.reverse()
>>> print(deque_colors)
```

2.Write a Python program to create a deque from an existing iterable object.

Solution:

```
>>> import collections
>>> even_nums = (2, 4, 6)
>>> print("Original tuple:")
>>> print(even_nums)
```

```
>>> print(type(even_nums))
>>> even_nums_deque = collections.deque(even_nums)
>>> print("\nOriginal deque:")
>>> print(even_nums_deque)
>>> even_nums_deque.append(8)
>>> even_nums_deque.append(10)
>>> even_nums_deque.append(12)
>>> even_nums_deque.appendleft(2)
>>> print("New deque from an existing iterable object:")
>>> print(even_nums_deque)
>>> print(type(even_nums_deque))
```

3. Write a Python program to find the item with maximum frequency in a given list. Go to the editor
Sample Output:
Original list:
[2, 3, 8, 4, 7, 9, 8, 2, 6, 5, 1, 6, 1, 2, 3, 2, 4, 6, 9, 1, 2]

Solution:

```
>>> from collections import defaultdict
>>> def max_occurrences(nums):
>>>     dict = defaultdict(int)
>>>     for i in nums:
>>>         dict[i] += 1
>>>     result = max(dict.items(), key=lambda x: x[1])
>>>     return result
>>> nums = [2,3,8,4,7,9,8,2,6,5,1,6,1,2,3,2,4,6,9,1,2]
>>> print ("Original list:")
>>> print(nums)
>>> print("\nItem with maximum frequency of the said list:")
>>> print(max_occurrences(nums))
```

4.Write a Python program to break a given list of integers into sets of a given positive number. Return true or false.

Solution:

```
>>> import collections as clt
>>> def check_break_list(nums, n):
>>>     coll_data = clt.Counter(nums)
>>>     for x in sorted(coll_data.keys()):
>>>         for index in range(1, n):
```

```
>>>             coll_data[x+index] = coll_data[x+index]  - coll_data[x]
>>>             if coll_data[x+index] < 0:
>>>                 return False
>>>     return True
>>>
>>> nums = [1,2,3,4,5,6,7,8]
>>> n = 4
>>> print("Original list:",nums)
>>> print("Number to devide the said list:",n)
>>> print(check_break_list(nums, n))
>>> nums = [1,2,3,4,5,6,7,8]
>>> n = 3
>>> print("\nOriginal list:",nums)
>>> print("Number to divide the said list:",n)
>>> print(check_break_list(nums, n))
```

5.Write a Python program to group a sequence of key-value pairs into a dictionary of lists

Solution:

```
>>> from collections import defaultdict
>>> class_roll = [('v', 1), ('vi', 2), ('v', 3), ('vi', 4), ('vii', 1)]
>>> d = defaultdict(list)
>>> for k, v in class_roll:
>>>     d[k].append(v)
>>> print(sorted(d.items()))
```

6.Write a Python program to compare two unordered lists (not sets).

Solution:

```
>>> from collections import Counter
>>> def compare_lists(x, y):
>>>     return Counter(x) == Counter(y)
>>> n1 = [20, 10, 30, 10, 20, 30]
>>> n2 = [30, 20, 10, 30, 20, 50]
>>> print(compare_lists(n1,n2))
```

7.Write a Python program to remove the intersection of a 2nd set from the 1st set.

Solution:

```
>>>sn1 = {1,2,3,4,5}
>>> sn2 = {4,5,6,7,8}
>>> print("Original sets:")
>>> print(sn1)
>>> print(sn2)
>>> print("\nRemove the intersection of a 2nd set from the 1st set using
difference_update():")
>>> sn1.difference_update(sn2)
>>> print("sn1: ",sn1)
>>> print("sn2: ",sn2)
>>> sn1 = {1,2,3,4,5}
>>> sn2 = {4,5,6,7,8}
>>> print("\nRemove the intersection of a 2nd set from the 1st set using = operator:")
>>> sn1=sn2
>>> print("sn1: ",sn1)
>>> print("sn2: ",sn2)
```

8.write a program to print today's date ?

```
>>> # importing date class from datetime module

>>> from datetime import date


>>> # creating the date object of today's date

>>> todays_date = date.today()

>>> # printing todays date

>>> print("Current date: ", todays_date)
>>> #fetching the current year, month and day of today

>>> print("Current year:", todays_date.year)

>>> print("Current month:", todays_date.month)

>>> print("Current day:", todays_date.day)
```

9.Write a Python program that accept name of given subject and marks. Input number of subjects in first line and subject name, marks separated by a space in next line. Print subject name and marks in order of its first occurrence.

Solution:

```
>>> import collections, re
>>> n = int(input("Number of subjects: "))
>>> item_order = collections.OrderedDict()
>>> for i in range(n):
>>>    sub_marks_list = re.split(r'(\d+)$',input("Input Subject name and marks: ").strip())
>>>    subject_name = sub_marks_list[0]
>>>    item_price = int(sub_marks_list[1])
>>>    if subject_name not in item_order:
>>> item_order[subject_name]=item_price
>>>    else:
>>>        >>>item_order[subject_name]=item_order[subject_name]+item_price

>>> for i in item_order:
>>>    print(i+str(item_order[i]))
```

10.Write a Python program to generate an infinite cycle of elements from an iterable.
Note: Iterable should be a list or a string or a dictionary, etc.

Solution:

```
>>> import itertools as it
>>> def cycle_data(iter):
>>>     return it.cycle(iter)
>>> # Following  loops will run for ever      >>> #List
>>> result = cycle_data(['A','B','C','D'])
>>> print("The said function print never-ending items:")
>>> for i in result:
>>>     print(i)
>>>
>>> #String
>>> result = cycle_data('Python itertools')
>>> print("The said function print never-ending items:")
>>> for i in result:
>>>     print(i)
```