Solutions,12feb,2022

1.Write a Python program to construct an infinite iterator that returns evenly spaced values starting with a specified number and step.

Solution:

```
>>> import itertools as it
>>> start = 10
>>> step = 1
>>> print("The starting number is ", start, >>> "and step is ",step)
>>> my_counter = it.count(start, step)
>>> # Following  loop will run for ever
>>> print("The said function print >>>never-ending items:")
>>> for i in my_counter:
>>>     print(i)
```

2.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)
```

3.Write a Python program to make an iterator that drops elements from the iterable as soon as an element is a positive number.

Solution:

```
>>> import itertools as it
>>> def drop_while(nums):
>>>     return it.dropwhile(lambda x : x < 0, >>> nums)
>>> nums = [-1,-2,-3,4,-10,2,0,5,12]
>>> print("Original list: ",nums)
>>> result = drop_while(nums)
>>> print("Drops elements from the >>>iterable when a positive number arises >>>
\n",list(result))
>>> #Alternate solution
>>> def negative_num(x):
>>>     return x < 0
>>> def drop_while(nums):
>>>     return it.dropwhile(negative_num, >>> nums)
>>> nums = [-1,-2,-3,4,-10,2,0,5,12]
>>> print("Original list: ",nums)
>>> result = drop_while(nums)
>>> print("Drops elements from the >>>iterable when a positive number arises >>>
\n",list(result))
```

4.Write a Python program to create an iterator that returns consecutive keys and groups from an iterable.

Solution:

```
>>> import itertools as it
>>> print("Iterate over characters of a >>>string and display\nconsecutive keys >>> and
groups from the iterable:")
>>> str1 = >>>'AAAAJJJJHHHHNWWWEERRRSSS>>> OOIIU'
>>> data_groupby = it.groupby(str1)
>>> for key, group in data_groupby:
>>>     print('Key:', key)
>>>     print('Group:', list(group))
```

```
>>> print("\nIterate over elements of a list >>>and display\nconsecutive keys and >>>
groups from the iterable:")
>>> str1 = >>>'AAAAJJJJHHHHNWWWEERRRSSS>>> OOIIU'
>>> str1 = [1,2,2,3,4,4,5,5,5,6,6,7,7,7,8]
>>> data_groupby = it.groupby(str1)
>>> for key, group in data_groupby:
>>>     print('Key:', key)
>>>     print('Group:', list(group))
```

5.Write a Python program to split an iterable and generate iterables specified number of times.

Solution:

```
>>> import itertools as it
>>> def tee_data(iter, n):
>>>     return it.tee(iter, n)
>>> #List
>>> result = tee_data(['A','B','C','D'], 5)
>>> print("Generate iterables specified number of times:")
>>> for i in result:
>>>     print(list(i))

#String
>>> result = tee_data("Python itertools", 4)
>>> print("\nGenerate iterables specified >>>number of times:")
>>> for i in result:
>>>     print(list(i))
```

6.Write a Python program to create an iterator to get specified number of permutations of elements.

Solution:

```
>>> import itertools as it
>>> def permutations_data(iter, length):
>>>     return it.permutations(iter, length)
#List
>>> result = >>>permutations_data(['A','B','C','D'], 3)
>>> print("\nIterator to get specified >>>number of permutations of elements:")
```

```
>>> for i in result:
>>>     print(i)

#String
>>> result = permutations_data("Python", 2)
>>> print("\nIterator to get specified >>>number of permutations of elements:")
>>> for i in result:
>>>     print(i)
```

7.Write a Python program to generate combinations of a given length of given iterable

Solution:

```
>>> import itertools as it
>>> def combinations_data(iter, length):
>>>     return it.combinations(iter, length)
>>> #List
>>> result = >>>combinations_data(['A','B','C','D'], 1)
>>> print("\nCombinations of an given iterable of length 1:")
>>> for i in result:
>>>     print(i)

#String
>>> result = combinations_data("Python", 1)
>>> print("\nCombinations of an given iterable of length 1:")
>>> for i in result:
>>>     print(i)

#List
>>> result = combinations_data(['A','B','C','D'], 2)
>>> print("\nCombinations of an given iterable of length 2:")
>>> for i in result:
>>>     print(i)

#String
>>> result = combinations_data("Python", 2)
>>> print("\nCombinations of an given iterable of length 2:")
>>> for i in result:
>>>     print(i)
```

8.Write a Python program to create Cartesian product of two or more given lists using itertools.

Solution:

```
>>> import itertools
>>> def cartesian_product(lists):
>>>    return list(itertools.product(*lists))
>>>
>>> ls = [[1,2],[3,4]]
>>> print("Original Lists:",ls)
>>> print("Cartesian product of the said lists: ",cartesian_product(ls))
>>> ls = [[1,2,3],[3,4,5]]
>>> print("\nOriginal Lists:",ls)
>>> print("Cartesian product of the said lists: ",cartesian_product(ls))
>>> ls = [[],[1,2,3]]
>>> print("\nOriginal Lists:",ls)
>>> print("Cartesian product of the said lists: ",cartesian_product(ls))
>>> ls = [[1,2],[]]
>>> print("\nOriginal Lists:",ls)
>>> print("Cartesian product of the said lists: ",cartesian_product(ls))
```

9.Write a Python program to chose specified number of colours from three different colours and generate all the combinations with repetitions.

Solution:

```
>>> from itertools import >>>combinations_with_replacement
>>>
>>> def combinations_colors(l, n):
>>>     return >>>combinations_with_replacement(l,n)
>>> l = ["Red","Green","Blue"]
>>> print("Original List: ",l)
>>> n=1
>>> print("\nn = 1")
>>> print(list(combinations_colors(l, n)))
>>> n=2
>>> print("\nn = 2")
>>> print(list(combinations_colors(l, n)))
>>> n=3
>>> print("\nn = 3")
>>> print(list(combinations_colors(l, n)))
```

10.Write a Python program to generate all possible permutations of n different objects.

Solution:

```
>>> import itertools
>>> def permutations_all(l):
>>>     for values in itertools.permutations(l):
>>>         print(values)

>>> permutations_all([1])
>>> print("\n")
>>> permutations_all([1,2])
>>> print("\n")
>>> permutations_all([1,2,3])
```