Permutation of digits

Preparation

The iteratorprotocol in Python prescribes that iterator objects must support the two following methods:

iterator.__iter__()

> Prints the iterator object itself. This method is required, so that both compound data types and iterators can be used in for and in statements.

iterator.__next__()

> Prints the next element from that iteration. If no elements are left, a StopIteration exception should be shown.

The example below illustrates how the class Reverse can be initialized with a sequence object (e.g. a string or a list). This class is an iterator that can be used to run through the elements in a sequence object in reverse order.

```
class Reverse:

    """
    >>> for element in Reverse('python'):
    ...     print(element, end='')
    nohtyp
    >>> for element in Reverse([1, 2, 3, 4]):
    ...     print(element, end='')
    4321
    """

    def __init__(self, container):

        self.container = container
        self.index = len(container)

    def __iter__(self):

        return self

    def __next__(self):

        if self.index > 0:
            self.index -= 1
            return self.container[self.index]
        else:
            raise StopIteration
```

Problem

Find for a given integer the next number that is larger and can be formed with the same digits. Suppose that the number given is 38276, then 38627 is the first number that is larger and can be formed by the digits 2, 3, 6, 7 and 8.

Suppose we start from the number 135798642, we can find the first following number that is larger and is formed with the same digits using the following algorithm. Split the first number in two parts, where the right part consists of the longest possible non-increasing sequence of digits one after another of the given number: 1357 98642. In other words, there is no digit in the right part that is smaller than the digit that follows. However, two consecutive digits can be equal. If the first part does not contain any digits — or in other words, if the given number is already a non-increasing sequence of digits — no larger number formed with the same digits can be found.

After that, search in the second part for the position of the smallest digit that is larger than the last digit of the first part (leftmost position if this smallest digits occurs more than once in the second part), and change the positions of these two digits. In our example, 7 is the last digit of the first part, and the first following larger digit of the second part is 8 on the second position. After changing places, we get 1358 97642. Lastly, we inverse the order of the digits in the second part and we put both parts back together:

Assignment

Write a class DigitPermutation of which the objects are initialized with a given integer. Objects of this class are iterators that follow the Python iterator protocol. For every iteration, the first integer is printed that is larger than the digit before and can be written with the same digit combination.

Example

```
>>> iter = DigitPermutation(135798642)
>>> next(iter)
135824679
>>> next(iter)
135824697

>>> iter = DigitPermutation(123)
>>> for first in iter:
...     print(first)
132
213
231
312
321
```

Voorbereiding

Het iteratorprotocol van Python schrijft voor dat iteratorobjecten de volgende twee methoden moeten ondersteunen:

iterator.__iter__()

> Geeft het iteratorobject zelf terug. Deze methode is vereist, zodat zowel samengestelde gegevenstypes als iteratoren kunnen gebruikt worden binnen for en in statements.

iterator.__next__()

> Geeft het volgende element terug uit de iteratie. Indien er geen elementen meer zijn, dan moet een StopIteration uitzondering opgeworpen worden.

Onderstaand voorbeeld illustreert hoe de klasse Omgekeerd kan geïnitialiseerd worden met een sequentie-object (bijvoorbeeld een string of een lijst). Deze klasse is een iterator die kan gebruikt worden om de elementen van het sequentie-object in omgekeerde volgorde te doorlopen.

```
class Omgekeerd:

    """
    >>> for element in Omgekeerd('python'):
    ...     print(element, end='')
    nohtyp
    >>> for element in Omgekeerd([1, 2, 3, 4]):
    ...     print(element, end='')
    4321
    """

    def __init__(self, container):

        self.container = container
        self.index = len(container)

    def __iter__(self):

        return self

    def __next__(self):

        if self.index > 0:
            self.index -= 1
            return self.container[self.index]
        else:
            raise StopIteration
```

Probleemomschrijving

Vind voor een gegeven natuurlijk getal het eerstvolgende getal dat groter is en kan gevormd worden met dezelfde cijfers. Stel dat bijvoorbeeld het getal 38276 gegeven is, dan is 38627 het eerste getal dat groter is en kan gevormd worden met de cijfers 2, 3, 6, 7 en 8.

Stel dat we vertrekken van het getal 135798642, dan kunnen we het eerstvolgende getal dat groter is en dezelfde cijfers gebruikt vinden aan de hand van het volgende algoritme. Splits eerst het gegeven getal in twee delen, waarbij het rechter deel bestaat uit de langst mogelijke niet-stijgende reeks cijfers achteraan het gegeven getal: 1357 98642. Met andere woorden, in de rechter cijferreeks is geen enkel cijfer kleiner dan het volgende cijfer. Twee opeenvolgende cijfers kunnen echter wel gelijk zijn. Indien het eerste deel geen cijfers bevat — of met andere woorden als het gegeven getal zelf reeds een niet-stijgende reeks van cijfers is — dan kan geen groter getal meer gevormd worden met dezelfde cijfers.

Zoek daarna in het tweede deel naar de positie van het kleinste cijfer dat groter is dan het laatste cijfer van het eerste deel (meest linkse positie indien dit kleinste cijfer meerdere keren voorkomt in het tweede deel), en wissel deze twee cijfers om. In ons voorbeeld is 7 het laatste cijfer van het eerste deel, en het eerstvolgende grotere cijfer in het tweede deel is het cijfer 8 op de tweede positie. Na omwisseling bekomen we dus 1358 97642. Tenslotte keren we de volgorde van de cijfers in het tweede deel om, en voegen we de twee delen terug samen: 1358 24679.

Opgave

Schrijf een klasse Cijferpermutatie waarvan de objecten geïnitialiseerd worden met een gegeven natuurlijk getal. Objecten van deze klasse zijn iteratoren die het iteratorprotocol van Python volgen. Bij elke iteratie wordt het eerstvolgende natuurlijke getal teruggegeven dat groter is dan het vorige getal en kan geschreven worden met dezelfde combinatie van cijfers.

Voorbeeld

```
>>> iter = Cijferpermutatie(135798642)
>>> next(iter)
135824679
>>> next(iter)
135824697

>>> iter = Cijferpermutatie(123)
>>> for volgende in iter:
...     print(volgende)
132
213
231
312
321
```