Kaprekar series

General information

The string method `join`

The string method `join` can be used to concatenate all string in an iterable object (e.g. a list) into a single string. This is done by concatenating all strings in the iterable object using a separator, which is the string on which the string method `join` is called.

```
>>> aList = ['a', 'b', 'c']
>>> ' '.join(aList)
'a b c'
>>> ''.join(aList)
'abc'
>>> '---'.join(aList)
'a---b---c'
>>> ' - '.join(aList)
'a - b - c'
```

Sorting lists

Python supports two ways to rearrange the elements of a list from the smallest to the largest. You can either call the list method `sort` on the list, or you can pass the list to the built-in function `sorted`. However, there is an important different between these two alternatives. The list method `sort` modifies the list *in place* (and does not return a new list), whereas the built-in function `sorted` returns a new list whose elements are sorted from the smallest to the largest.

```
>>> aList = [4, 2, 3, 1]
>>> aList.sort()
>>> aList
[1, 2, 3, 4]
>>>
>>> aList = [4, 2, 3, 1]
>>> sorted(aList)
[1, 2, 3, 4]
```

Remarks

Sort in descending order

By default the list method `sort` and the built-in function `sorted` sort the elements of a list in ascending order. Both function also have an optional parameter `reverse` to which the value `True` can be passed to have the elements arranged in descending order.

```
>>> aList = [1, 2, 3, 4]
>>> aList.sort(reverse=True)
>>> aList
[1, 2, 3, 4]
>>>
>>> aList = [4, 2, 3, 1]
```

```
>>> sorted(aList, reverse=True)
[1, 2, 3, 4]
```

Partitioning the phone book

General information

Check data types with `isinstance`

To check whether or not a given object `o` has a given data type `t`, you can use the built-in function `type(o)` to see if it returns the data type `t` for the object `o`. However, it is better (more pythonic) to use the built-in function `isinstance(o, t)` in this case. This function returns a Boolean value that indicates whether or not the object `o` has data type `t` or a date type that is derived from the data type `t`.

```
>>> type(3) == int
True
>>> isinstance(3.14, int)
False
>>> isinstance(3.14, float)
True
>>> isinstance([1, 2, 3], list)
True
```

Stable marriage

Specific information

In this assignment, the pseudocode of the algorithm is given. However, you still have to make some decisions about the data structures you will use to ease the implementation of the algorithm. With data structures, we mean the data types of the object that you'll use to represent the data needed by the algorithm.

The data you need to track for the algorithm are:

1. the woman each man is currently coupled with (with the option that a man is not coupled to a woman at this point in time)
2. the man each woman is currently coupled with (with the option that a woman is not coupled to a man at this point in time)
3. the woman in his list of preference a man has already been coupled with by the algorithm

Here are some options you might consider:

1. you can maintain a list that contains all men that are not coupled to a woman at this point in time; at the start of the algorithm, this list contains all men; each time a man is coupled to a woman, he is removed from the list; each time a man is decoupled from a woman, he is added to the list
2. you can maintain a list whose i-th position indicates which man women i coupled with at this point in time; if the i-th position contains the value `None`, this means that women i is not coupled to any man at this point in time
3. you can maintain a twodimensional grid of Boolean values (`True` or `False`), where position (i, j) indicates whether man i has already been coupled to woman j by the algorithm

4. as an alternative, you can also maintain a list whose i-th position indicates the number of woman man i has already been coupled with by the algorithm; at the start of the algorithm, none of the men has been coupled to any woman yet