Roman numerals

In ancient Rome a numeric system was used based on Roman numerals for the representation of integers. This numeric system is not a positional system, but an additive system in which the value of the integer is determined as the total sum of the digits used in the representation. The numbers one to ten can be expressed as Roman numerals as follows: I, II, III, IV, V, VI, VII, VIII, IX and X. Negative numbers and the zero value cannot be represented in the Roman numeric system. The Roman numeric system has largely become in disuse since the fourteenth century, in favor of the decimal numeric system that is based on Arab digits. However, Roman numeral are still in common use today in some applications, such as in numbering kings having the same name (e.g. Louis XIV of France), in numbering annual events and in numbering centuries in some countries (e.g. XIXe siècle).

In the Roman numeric system, integers are represented using symbols, the actual digits, where each symbol has a certain value independent of the position where it occurs in the integer representation. The symbols used in the Roman numeric system are:

symbol	value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

The order of the Roman digits in the integer representation is not random. The values of the individual digits are added, except if a symbol with a lower value is immediately followed by a symbol with a higher value: in that case, the lower value is subtracted instead of being added. The rules for representing integers using Roman numerals has been very loose in Ancient times. As such, in its credits the BBC used the typographically more elegant representation MIM for the year 1999, instead of MCMXCIX.

Assignment

PEP 313 was a proposal to extend the Python programming language with support for the representation of Roman numerals as a literal data type. This proposal was rejected by Guido Van Rossum, who created the Python language and is nicknamed the Benevolent Dictator for Life (BDFL). However, since it is possible to extend Python by defining new data types, it is possible that we create or own data type to represent Roman numerals in Python. In order to do so, you proceed as follows:

- Write a function roman2arab that converts a given string that only contains Roman digits (upper case or lower case) into the integer representation of the same number.
- Write a function arab2roman that converts a given integer into the string representation of the same number using Roman digits (upper case only). This representation should be determined by traversing the following table from left to right. As long as the remaining

integer value is larger then or equal to the current value from the table, the corresponding combination of Roman digits is appended to the Roman numeral. The current value is then subtracted from the integer value. As soon as the current value is lower then the remaining integer value, we traverse to the next position in the table.

M	CM	D	CD	C	XC	L	XL	X	IX	V	IV	I
1000	900	500	400	100	90	50	40	10	9	5	4	1

- Define a class Roman that can be used to represent Roman numerals in Python. New objects of this class can be initialized either by an integer value i (where $1 \leq i < 4000$) or by a string s that only contains Roman digits (upper case and/or lower case). All other values passed while initializing an object of the class should result in an AssertionError being thrown with the message invalid roman numeral. The conversion of a Roman numeral to a string — obtained using the built-in functions str() and repr() — should give the integer value represented in Roman digits (upper case, as generated by the function arab2roman) and the converstion to an integer — obtained using the built-in function int() — should give the integer value in Arab digits. Make sure that the sum, difference and product of two Roman numerals can be computed using the operators +, - and *. The evaluation of these operations should result in a new Roman numeral.

Example

```
>>> roman2arab('MDCCCCX')
1910

>>> arab2roman(1910)
'MCMX'

>>> int(Roman('MDCCCCX'))
1910
>>> int(Roman('MCMLIV'))
1954
>>> int(Roman('MCMXC'))
1990
>>> rsum = Roman('MDCCCCX') + Roman('MCMXC')
>>> isinstance(rsum, Roman)
True
>>> print(rsum)
MMMCM
>>> difference = Roman('MCMXC') - Roman('MDCCCCX')
>>> isinstance(difference, Roman)
True
>>> str(difference)
'LXXX'
>>> product = Roman('CMX') * Roman('III')
>>> isinstance(product, Roman)
True
>>> product
MMDCCXXX

>>> print(Roman('mdccccx'))
MCMX
>>> Roman(1234)
MCCXXXIV
>>> Roman(4321)
Traceback (most recent call last):
```

```
AssertionError: invalid roman numeral
>>> Roman('ABCDEF')
Traceback (most recent call last):
AssertionError: invalid roman numeral
>>> Roman(3.14)
Traceback (most recent call last):
AssertionError: invalid roman numeral
```

In het oude Rome gebruikte men een talstelsel gebaseerd op Romeinse cijfers voor het weergeven van natuurlijke getallen. Dit talstelsel is geen positiestelsel, maar een additief stelsel waarin de waarde van het voorgestelde getal bepaald wordt door het totaal van de samenstellende symbolen. De getallen één tot en met tien worden met Romeinse cijfers geschreven als: I, II, III, IV, V, VI, VII, VIII, IX en X. Negatieve getallen en het getal nul kunnen niet voorgesteld worden in Romeinse cijfers. Het talstelsel is sinds de veertiende eeuw grotendeels verlaten ten voordele van het decimaal talstelsel gebaseerd op Arabische cijfers. In sommige toepassingen zijn Romeinse cijfers echter nog steeds in gebruik, bijvoorbeeld bij de nummering van vorsten met dezelfde naam (bijv. Lodewijk XIV), bij de nummering van jaarlijkse events en bij de nummering van eeuwen in sommige landen (bijv. XIXe siècle).

In het talstelsel met Romeinse cijfers worden getallen genoteerd met symbolen, de eigenlijke cijfers, waarvan elk een bepaalde waarde heeft die onafhankelijk is van de positie die het cijfer in het getal inneemt. De Romeinse cijfers zijn:

symbool	**waarde**
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

De volgorde van de Romeinse cijfers in een getal is niet willekeurig. De waarden van de losse cijfers worden bij elkaar opgeteld, behalve als een lager cijfer vóór een hoger cijfer staat: in dat geval wordt het lagere cijfer er van afgetrokken. De regels voor het schrijven van getallen met Romeinse cijfers lijken in de Oudheid zeer los te zijn geweest. Zo gebruikte de BBC voor de aftiteling in het jaar 1999 het typografisch elegante jaartal MIM in plaats van MCMXCIX.

Opgave

PEP 313 was een voorstel om de Python programmeertaal uit te breiden met ondersteuning voor Romeinse getallen. Dit voorstel werd echter niet goedgekeurd. Omdat Python toelaat om nieuwe gegevenstypes te definiëren, is het echter mogelijk om zelf een gegevenstype voor Romeinse getallen toe te voegen aan Python. Hiervoor ga je als volgt te werk:

- Schrijf een functie romeins2arabisch die een gegeven string die enkel bestaat uit Romeinse cijfers (hoofdletters of kleine letters) omzet naar een integervoorstelling van hetzelfde getal.
- Schrijf een functie arabisch2romeins die een gegeven integer omzet naar de stringvoorstelling van dezelfde waarde in Romeinse cijfers (enkel hoofdletters). Deze omzetting kan

gebeuren door onderstaande tabel van links naar rechts te doorlopen. Zolang de resterende integerwaarde groter of gelijk is aan de getalwaarde uit de tabel, voeg je de combinatie van Romeinse cijfers op de corresponderende positie achteraan toe aan het Romeins getal. De integerwaarde wordt daarna verlaagd met de getalwaarde. Van zodra de resterende integerwaarde kleiner is dan de getalwaarde uit de tabel, spring je één positie naar rechts in de tabel.

M	CM	D	CD	C	XC	L	XL	X	IX	V	IV	I
1000	900	500	400	100	90	50	40	10	9	5	4	1

- Definieer een klasse Romeins waarmee Romeinse getallen kunnen voorgesteld worden in Python. Nieuwe objecten van deze klasse kunnen zowel geïnitialiseerd worden met een integerwaarde i (waarvoor $1 \leq i < 4000$) of met een string s die enkel bestaat uit Romeinse cijfers (hoofdletters of kleine letters). Voor andere initialisatiewaarden moet de klasse een AssertionError opwerpen met de string ongeldig romeins cijfer. De conversie van een Romeins getal naar een string — bekomen met de ingebouwde functies str() en repr() — geeft de getalwaarde in Romeinse cijfers (hoofdletters, zoals gegenereerd door de functie arabisch2romeins) en de conversie naar een integer — bekomen met de ingebouwde functie int() — geeft de getalwaarde in Arabische cijfers. Zorg ervoor dat de som, het verschil en het product van twee Romeinse getallen kan berekend worden aan de hand van de operatoren +, - en *. De evaluatie van deze bewerkingen moet resulteren in een nieuw Romeins getal.

Voorbeeld

```
>>> romeins2arabisch('MDCCCCX')
1910

>>> arabisch2romeins(1910)
'MCMX'

>>> int(Romeins('MDCCCCX'))
1910
>>> int(Romeins('MCMLIV'))
1954
>>> int(Romeins('MCMXC'))
1990
>>> som = Romeins('MDCCCCX') + Romeins('MCMXC')
>>> isinstance(som, Romeins)
True
>>> print(som)
MMMCM
>>> verschil = Romeins('MCMXC') - Romeins('MDCCCCX')
>>> isinstance(verschil, Romeins)
True
>>> str(verschil)
'LXXX'
>>> product = Romeins('CMX') * Romeins('III')
>>> isinstance(product, Romeins)
True
>>> product
MMDCCXXX

>>> print(Romeins('mdccccx'))
MCMX
>>> Romeins(1234)
```

```
MCCXXXIV
>>> Romeins(4321)
Traceback (most recent call last):
AssertionError: ongeldig romeins cijfer
>>> Romeins('ABCDEF')
Traceback (most recent call last):
AssertionError: ongeldig romeins cijfer
>>> Romeins(3.14)
Traceback (most recent call last):
AssertionError: ongeldig romeins cijfer
```