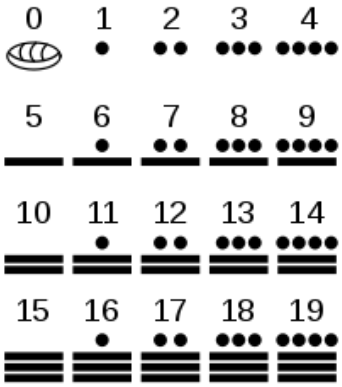
Maya numbers

In the beginning of our era, the Maya settled in Central-America. They were in many ways the most developed and fascinating civilization of their time. They may not have known wheels or draught animals, they possessed the art of weaving, architecture and pottery like no other. However, it was their accomplishments on the level of mathematics and astrology that were breath-taking. While Europe was dragging through the dark Middle Ages, the Maya calculated the sun year of 365,242 days (according to modern calculations 365,242198 days) and the moon cycle of 29,5302 days (29,53059 days according to modern calculations). Such staggeringly accurate calculations would barely have been possible if it hadn't been for the powerful scale of the Maya.

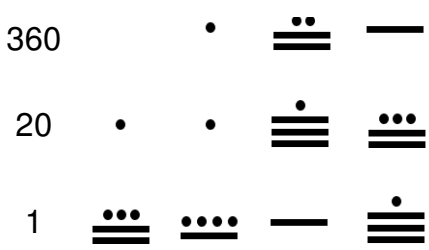
Maya priests and astronomers used a numerical system with base 20. The digits (0-19) were represented with three symbols: zero (shape of a shell), one (a full stop) and five (a horizontal line). This way, nineteen (19) was written as four full stops horizontally next to each other, on top of three horizontal stripes that were stacked onto each other. The concepts 'digit' and 'zero' were fairly unusual for that time, and completely unknown in Europe.



The twenty Maya digits.

For Maya numbers consisting of multiple Maya digits — i.e. with a value larger than 19 — the maya digits are written on top of each other. Here, the digit with the highest value is on top. The number thirty-three, for example, is written as a full stop (upper Maya digit), with under that three full stops on top of 2 horizontal stripes (lower Maya digit). The first full stop thus represents "one twinty" or 1×20, where three full stops and two horizontal lines (so 13) are added. As a consequence, you get $(1 \times 20) + 13 = 33$. Oddly enough, the digit on the third position of the Maya scale (counted from bottom to top) does not have value $20 \times 20 = 400$, but $18 \times 20 = 360$. Probably because 360 is about equal to the number of days in a calendar year. All maya digits above the third position (counted from the bottom) again have a normal value according to the 20-count scale, so for the fourth digit $360 \times 20 = 7200$, for the fifth digit $7200 \times 20 = 144000$, and so on. This way, the table underneath indicates for example that $12 \times 360 + 16 \times 20 + 5 = 4645$ and that $9 \times 7200 + 5 \times 360 + 13 \times 20 + 16 = 66876$.

Unit	**33**	**389**	**4645**	**66876**
7200				



Assignment

For this assignment we represent Maya numbers and Maya digits as strings. The Maya digit with value zero is noted with the letter `S`, and the other digits as a sequence of full stops (`.`; value 1) and hyphens (`-`; value 5), where the full stops are always in front of the hyphens. The Maya digits of a Maya numbers are written from left to right one after another — separated by a space — instead of from top to bottom. This way, the Maya number with value 66876 is represented as the string "`....- - ...-- .---`".

Expand the computer language Python with support for Maya numbers. To do so, you define a class `Maya` of which new objects can be initialized based on the string representation of a Maya number. This class must support the conversion of Maya numbers to strings (built-in functions `str()` and `repr()`) and to integers (built-in function `int()`). Make sure that the sum, the difference, the product and the whole division of two Maya numbers can be calculated using the operators `+`, `-`, `*`, `//` and `%`. The evaluation of these calculations must result in a new Maya number. As soon as an invalid maya number is made, either by initialization or by a calculation, the class must raise an `AssertionError` with the string `invalid maya number`.

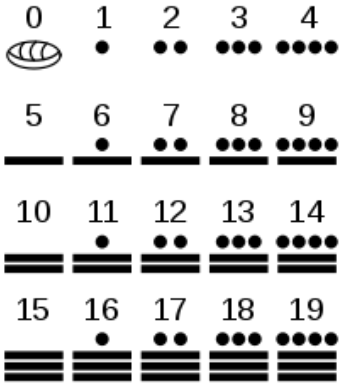
Example

```
>>> a = Maya('..')
>>> b = Maya('. ...--')
>>> c = a + b
>>> c
Maya('. ---')
>>> print(c)
. ---
>>> int(c)
35
>>> a - b
Traceback (most recent call last):
AssertionError: invalid maya number
>>> b - a
Maya('. .--')
>>> a * b
Maya('... .-')
>>> int(a * b)
66
>>> b // a
Maya('.---')
>>> b % a
Maya('.')

>>> Maya(11)
Traceback (most recent call last):
AssertionError: invalid maya number
```

```
>>> Maya('xxx')
Traceback (most recent call last):
AssertionError: invalid maya number

>>> Maya('SS ..--')
Traceback (most recent call last):
AssertionError: invalid maya number

>>> Maya('.....--')
Traceback (most recent call last):
AssertionError: invalid maya number

>>> Maya('..-- .- ...--- ...---')
Traceback (most recent call last):
AssertionError: invalid maya number
```

In het begin van onze jaartelling vestigden de Maya zich in Centraal-Amerika. Ze waren in vele opzichten veruit de meest ontwikkelde en fascinerende beschaving van hun tijd. Wielen en trekdieren waren hen dan wel onbekend, toch beheersten ze als geen ander de kunst van het weven, de architectuur en het pottenbakken. Maar echt adembenemend waren hun verwezenlijkingen op het vlak van wiskunde en sterrenkunde. Terwijl Europa zich door de donkere Middeleeuwen aan het slepen was, berekenden de Maya het zonnejaar op 365,242 dagen (volgens moderne berekeningen 365,242198 dagen) en de maancyclus op 29,5302 dagen (volgens moderne berekening 29,53059 dagen). Zo'n verbluffend accurate berekeningen waren nauwelijks mogelijk geweest zonder het krachtige talstelsel dat de Maya hanteerden.

Mayapriesters en sterrenkundigen gebruikten een talstelsel met basis 20. Daarin worden de cijfers (0-19) voorgesteld aan de hand van drie symbolen: nul (vorm van een schelp), één (een punt) en vijf (een horizontale streep). Zo wordt negentien (19) bijvoorbeeld geschreven als vier punten horizontaal achter elkaar, boven drie horizontale strepen die bovenop elkaar gestapeld worden. De concepten 'cijfer' en 'nul' waren vrij ongebruikelijk voor die tijd, en nog totaal onbekend in Europa.



De twintig mayacijfers.

Voor mayagetallen bestaande uit meerdere mayacijfers — dus getallen met waarde groter dan 19 — worden de mayacijfers verticaal bovenop elkaar geschreven. Hierbij staat het cijfer met de hoogste getalwaarde bovenaan. Het getal drieëndertig wordt bijvoorbeeld geschreven als een punt (bovenste mayacijfer), met daaronder drie punten bovenop 2 horizontale strepen (onderste mayacijfer). Het eerste punt stelt dus "één twintig" of 1×20 voor, waarbij drie punten en twee horizontale strepen (dus 13) opgeteld worden. Bijgevolg krijg je $(1 \times 20) + 13 = 33$. Vreemd genoeg heeft in het Maya-talstelsel het cijfer op de derde positie (geteld vanaf onder) niet als getalwaarde $20 \times 20 = 400$, maar $18 \times 20 = 360$. Vermoedelijk omdat 360

ongeveer gelijk is aan het aantal dagen in een kalenderjaar. Alle mayacijfers boven de derde positie (geteld vanaf onder) hebben terug een normale getalwaarde volgens het 20-tallig stelsel, dus voor het vierde cijfer $360 \times 20 = 7200$, voor het vijfde cijfer $7200 \times 20 = 144000$, enzoverder. Op die manier geeft onderstaande tabel bijvoorbeeld aan dat $12 \times 360 + 16 \times 20 + 5 = 4645$ en dat $9 \times 7200 + 5 \times 360 + 13 \times 20 + 16 = 66876$.

eenheid	33	389	4645	66876
7200				
360				
20				
1				

Opgave

Bij deze opgave stellen we mayagetallen en mayacijfers voor als strings. Het mayacijfer met waarde nul wordt genoteerd met de letter `S`, en de andere cijfers als een reeks punten (`.`; waarde 1) en koppeltekens (`-`; waarde 5), waarbij de punten altijd voor de koppeltekens staan. De mayacijfers van een mayagetal worden van links naar rechts achter elkaar geschreven — gescheiden door een spatie — in plaats van van boven naar onder. Op die manier wordt het mayagetal met getalwaarde 66876 dus voorgesteld als de string "`....- - ...-- .---`".

Breid de programmeertaal Python uit met ondersteuning voor mayagetallen. Hiervoor definieer je een klasse `Maya` waarvan nieuwe objecten kunnen geïnitialiseerd worden op basis van de stringvoorstelling van een mayagetal. Deze klasse moet ondersteuning bieden voor de conversie van mayagetallen naar strings (ingebouwde functies `str()` en `repr()`) en naar integers (ingebouwde functie `int()`). Zorg ervoor dat de som, het verschil, het product, de gehele deling, en de rest na gehele deling van twee mayagetallen kan berekend worden aan de hand van de operatoren `+`, `-`, `*`, `//` en `%`. De evaluatie van deze bewerkingen moet resulteren in een nieuw mayagetal. Van zodra er een ongeldig mayagetal wordt aangemaakt, hetzij bij initialisatie, hetzij bij een berekening, moet de klasse een `AssertionError` opwerpen met de string `ongeldig mayagetal`.

Voorbeeld

```
>>> a = Maya('..')
>>> b = Maya('. ...--')
>>> c = a + b
>>> c
Maya('. ---')
>>> print(c)
. ---
>>> int(c)
35
>>> a - b
Traceback (most recent call last):
AssertionError: ongeldig mayagetal
```

```
>>> b - a
Maya('. .--')
>>> a * b
Maya('... .-')
>>> int(a * b)
66
>>> b // a
Maya('.---')
>>> b % a
Maya('.')

>>> Maya(11)
Traceback (most recent call last):
AssertionError: ongeldig mayagetal

>>> Maya('xxx')
Traceback (most recent call last):
AssertionError: ongeldig mayagetal

>>> Maya('SS ..--')
Traceback (most recent call last):
AssertionError: ongeldig mayagetal

>>> Maya('.....--')
Traceback (most recent call last):
AssertionError: ongeldig mayagetal

>>> Maya('..-- .- ...--- ...---')
Traceback (most recent call last):
AssertionError: ongeldig mayagetal
```