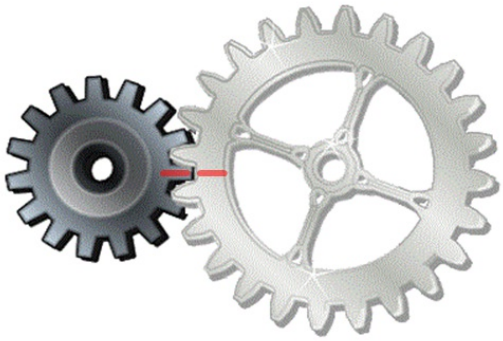
Gears

A machine has two gears. One of which has 15 teeth, the other has 24. In places the gears interlock, they are marked with a red line. What is the smallest amount of rotations the smaller gear has to make in order to get the markings next to each other?



The rotation can be calculated by dividing the lowest common multiple of the number of teeth of both gears by the number of teeth of the smallest gear. This is easily seen if we write out for every gear how many teeth pass the point where the gears interlock in one rotation:

> first gear: 15, 30, 45, 60, 75, 90, 105, **120**, 135, 150, 165, 180, 195, 210, 225, **240**, 255, 270, 285, …
> second gear: 24, 48, 72, 96, **120**, 144, 168, 192, 216, **240**, 264, 288, 312, 336, **360**, 384, 408, 432, 456, …

When the smallest gear has made eight full rotations, a total of 120 teeth have past the point where both gears interlock. This is the same number of teeth that passes the point when the largest gear has made 5 full rotations.

Preparation

Watch the video on test-driven development to learn to work with doctests. In doing so you will understand the two last statements in the skeleton of the source code that is given in this assignment.

Assignment

Below, we have already constructed the skeleton of the solution of this assignment. Your task is to implement the functions gcd, lcm and rotations, so that they calculate the following properties for two gears a and b: *i*) the greatest common denominator of the number of teeth, *ii*) the lowest common multiple of the number of teeth and *iii*) the minimum amount of rotations of the first gear after which the gears are back in their starting positions. The greatest common denominator is of course the largest integer that is a divisor of both a and b. The lowest common multiple is the lowest integer that is a multiple of both a and b.

```
def gcd(a, b):
    """
    Computes the greatest common divisor of the two integers a and b.
```

```
    >>> gcd(15, 24)
    3
    >>> gcd(252, 105)
    21
    """

    # calculation of greatest common divisor based on Euclid's algorithm

def lcm(a, b):

    """
    Computes the least common multiple of the two integers a and b.

    >>> lcm(15, 24)
    120
    >>> lcm(252, 105)
    1260
    """

    # calculation of least common multiple

def rotations(a, b):

    """
    Computes the smallest number of rotations of a gear having a teeth that
    puts gears with a and b teeth back into their initial position.

    >>> rotations(15, 24)
    8
    >>> rotations(252, 105)
    5
    """

    # calculation of number of rotations

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

The fractions module from the Python Standard Library contains a function gcd that prints the greatest common denominator of two integers. However, it is prohibited to use the fractions module, you are supposed to calculate the greatest common denominator yourself. Base your implementation of the function gcd on the **Euclidean algorithm**. The algorithm starts with a couple of integers and uses them to form a new pair that consists of the smallest of both numbers and the subtraction of the largest and the smallest number. This procedure repeats itself until the pair consists of two equal numbers. That number is the greatest common denominator of the original pair of numbers.

The basic principle that is used for this, is that the greatest common denominator doesn't change if the smallest number is subtracted of the largest number. Therefore, the greatest common denominator of 252 and 105 is also the greatest common denominator of 147 (= 252 - 105) and 105. Seeing as the largest number decreases, the repetition of this procedure results in smaller and smaller numbers, so that the repetition will stop sooner or later when both numbers are equal (if the procedure would be repeated once more after that, one of both numbers would become 0).

The lowest common multiple of two integers a and b, can be determined as the product of

both numbers, divided by their greatest common denominator.

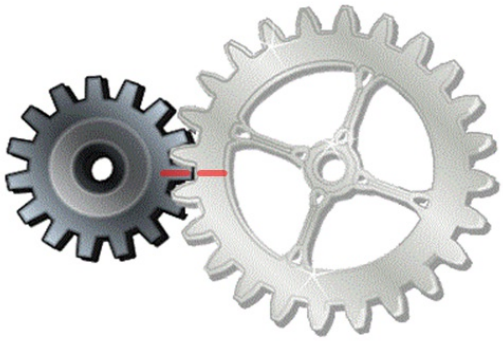
Example

```
>>> gcd(15, 24)
3
>>> gcd(252, 105)
21

>>> lcm(15, 24)
120
>>> lcm(252, 105)
1260

>>> rotations(15, 24)
8
>>> rotations(252, 105)
5
```

In een machine zitten twee tandwielen. Eén tandwiel heeft 15 tanden en het andere 24 tanden. Op de plaatsen waar de tandwielen in elkaar grijpen, hebben we de tandwielen gemarkeerd met een rode lijn. Wat is het kleinste aantal omwentelingen van het eerste tandwiel dat de markeerlijnen van beide tandwielen terug naast elkaar zet?



Dit aantal omwentelingen kan berekend worden door het kleinste gemene veelvoud van het aantal tanden van beide tandwielen te delen door het aantal tanden van het eerste tandwiel. Dit is makkelijk in te zien als we voor elk tandwiel uitschrijven hoeveel tanden er bij elke volledige omwenteling voorbij het punt gekomen zijn waar de tandwielen in elkaar grijpen:

> eerste tandwiel: 15, 30, 45, 60, 75, 90, 105, **120**, 135, 150, 165, 180, 195, 210, 225, **240**, 255, 270, 285, …
> tweede tandwiel: 24, 48, 72, 96, **120**, 144, 168, 192, 216, **240**, 264, 288, 312, 336, **360**, 384, 408, 432, 456, …

Als het eerste tandwiel acht volledige omwentelingen heeft gemaakt, dan zijn er in totaal 120 tanden voorbij het punt gekomen zijn waar de tandwielen in elkaar grijpen. Dit is hetzelfde aantal tanden dat voorbij dat punt gekomen is als het tweede tandwiel 5 volledige omwentelingen gemaakt heeft.

Voorbereiding

Bekijk de video over test-driven development om te leren werken met doctests. Hierdoor zal je de laatste twee statements begrijpen in het skelet van de broncode dat wordt meegegeven met deze

opgave.

Opgave

We hebben hieronder reeds het skelet opgemaakt voor de oplossing van deze opgave. Je taak bestaat erin om de functies ggd, kgv en omwentelingen te implementeren, zodat ze voor twee tandwielen met a en b tanden respectievelijk de volgende eigenschappen berekenen: *i*) de grootste gemene deler van het aantal tanden, *ii*) het kleinste gemene veelvoud van het aantal tanden en *iii*) het minimaal aantal omwentelingen van het eerste tandwiel waarna de tandwielen terug in de begintoestand staan. De grootste gemene deler is het grootste natuurlijke getal dat zowel een deler is van a als van b. Het kleinste gemene veelvoud is het kleinste natuurlijke getal dat zowel een veelvoud is van a als van b.

```
def ggd(a, b):

    """
    Berekent de grootste gemene deler van twee natuurlijke getallen a en b.

    >>> ggd(15, 24)
    3
    >>> ggd(252, 105)
    21
    """

    # berekening grootste gemene deler op basis van het algoritme van Euclides

def kgv(a, b):

    """
    Berekent het kleinste gemene veelvoud van twee natuurlijke getallen a en b.

    >>> kgv(15, 24)
    120
    >>> kgv(252, 105)
    1260
    """

    # berekening kleinste gemene veelvoud

def omwentelingen(a, b):

    """
    Berekent het kleinste aantal omwentelingen van een tandwiel met a tanden dat
    tandwielen met a en b tanden terug in hun beginpositie brengt.

    >>> omwentelingen(15, 24)
    8
    >>> omwentelingen(252, 105)
    5
    """

    # berekening aantal omwentelingen

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

De fractions module uit de Python Standard Library bevat een functie gcd die de grootste gemene deler van twee natuurlijke getallen teruggeeft. Voor deze opgave mag je de fractions module echter niet gebruiken, en is het de bedoeling dat je grootste gemene deler zelf berekent. Baseer je implementatie van de functie ggd op het **algoritme van Euclides**. Het algoritme start met een paar natuurlijke getallen, en vormt daarmee een nieuw paar dat bestaat uit het kleinste van de twee getallen, en het verschil tussen het grootste en het kleinste getal. Deze procedure herhaalt zich totdat het paar bestaat uit twee gelijke getallen. Dat getal is de grootste gemene deler van het oorspronkelijke paar natuurlijke getallen.

Het basisprincipe dat hierbij gehanteerd wordt, is dat de grootste gemene deler niet verandert als het kleinste van het grootste getal afgetrokken wordt. Zo is de grootste gemene deler van 252 en 105 ook de grootste gemene deler van 147 (= 252 - 105) en 105. Aangezien grootste getal verlaagd wordt, resulteert het herhaald uitvoeren van deze procedure in steeds kleinere getallen, zodat de herhaling vroeger of later zal stoppen wanneer de twee getallen gelijk zijn (als de procedure daarna nog een keer zou herhaald worden, dan zou één van beide getallen nul worden).

Het kleinste gemene veelvoud van twee natuurlijke getallen a en b kan bepaald worden als het product van beide getallen, gedeeld door hun grootste gemene deler.

Voorbeeld

```
>>> ggd(15, 24)
3
>>> ggd(252, 105)
21

>>> kgv(15, 24)
120
>>> kgv(252, 105)
1260

>>> omwentelingen(15, 24)
8
>>> omwentelingen(252, 105)
5
```