Euclid's Algorithm for finding the highest common factor of two positive integers:

`def hcf(a,b):`	The function is defined, taking two input values: a and b.
`    i=0`	The variable i is introduced to count the number of iterations required.
`    while a*b>0:`	A 'while' loop repeats till one or other of a & b is 0
`        if a>b:`	The 'if' statement checks which is the largest…
`            a,b=b,a`	…and redefines (swaps a and b) if required to ensure b>a
`        b=b-a`	b is now replaced by the difference b-a
`        i=i+1`	This counts as one more iteration, so increase i
`        print "a="+str(a)+",b="+str(b)+", i= "+str(i)`	Print information on the current values of a, b and i (while loop continues as long as a*b>0)
`    print "HCF="+str(a)`	Print the highest common factor (the current value of a)

The modulus function allows you to find the remainder when one positive integer is divided by another. In Python, this function can be called using a%b ('a modulo b', or 'the remainder when a is divided by b').

Eg: 38%5 returns 3 since $\frac{38}{5} = 7\frac{3}{5}$.

Can you incorporate this function to come up with a more efficient algorithm to find the highest common factor?

Hint: consider what the algorithm above does when a=100 and b=15, and how 100%15 could provide a short cut.

Solution:

`def hcf(a,b):`
`  while a*b>0:`
`  if a>b:`
`  a,b=b,a`
`  b=b-a`
`  print "HCF="+str(a)`
	Original algorithm (simplified form)
	Replaces the largest of a and b with the difference between a and b each time.
		`def hcf2(a,b):`
		`  while a*b>0:`
		`  if a>b:`
		`  a,b=b,a`
		`  b=b%a`
		`  print "HCF="+str(a)`
			Improved algorithm
			Replaces the largest of a and b with the remainder when the largest is divided by the smallest.

Note: this improved algorithm is capable of finding the HCF of any two numbers in at most 5 times the number of digits of the smallest of the two numbers.