Programming Problem 2: Primality testing
## Primality Testing

Given a positive integer (> 1), determine whether it is a prime number or not.

### Examples:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>prime</td>
</tr>
<tr>
<td>2001</td>
<td>composite</td>
</tr>
<tr>
<td>987654321</td>
<td>composite</td>
</tr>
</tbody>
</table>
Why do we care?

- Our digital life depends on the security of information that we send over the internet.
- Security of information is made possible by encryption methods.
- One of the most well known encryption methods is the RSA algorithm (R = Ron Rivest, S = Adi Shamir, and A = Leonard Adleman).
- The first step of this algorithm is to find two large primes $p$ and $q$ and compute their product $n = p \times q$.
- “Large” here could mean 300 digits or so.
Generate all “candidate” factors of n, namely 2, 3, ..., n-1

For each generated “candidate” factor, check if n is evenly divisible by the factor (i.e., the remainder is 0).

If a “candidate” factor is found to be a real factor, then n is composite.

If no “candidate” factor is found to be a real factor, then n is a prime.
Algorithm in pseudocode

1. Input n
2. For each factor = 2, 3, ..., n-1 do the following
   3. if n is evenly divisible by factor then
      4. remember that n is a composite
   5. If we have detected that n is a composite
      6. output that n is a composite
   7. Otherwise output that n is a prime
number = int(raw_input("Enter a positive integer: "))

factor = 2
isPrime = True
while(factor <= number - 1):
    if(number % factor == 0):
        isPrime = False
        factor = factor + 1
    else:
        factor = factor + 1

if(isPrime):
    print number, "is prime"
else:
    print number, "is composite"