Cracking Codes with Python (updated to 6th printing)

Page 175: The line:

"In this example, the 8-square rod is the longest rod that can fit evenly into 24 and 32."

should now read:

"In this example, the 8-square rod is the longest rod that can fit evenly into 24 and 16."

Page 209: Line 14 of the *simpleSubcipher.py* program which reads:

```
if keyIsValid(myKey):
```

should now read:

```
if not keyIsValid(myKey):
```

Page 212: Line 14 of the *simpleSubcipher.py* program which reads:

```
if keyIsValid(myKey):
```

should now read:

```
if not keyIsValid(myKey):
```

Page 281: Line 23 of the *vigenereDictionaryHacker.py* program which reads:

```
for word in lines:
```

should now read:

```
for word in words:
```

Page 325: The code for *primeNum.py* that reads:

```
83.
84.     # See if any of the low prime numbers can divide num:
85.     for prime in LOW_PRIMES:
86.         if (num % prime == 0):
87.             return False
88.
```

should now read:

```
83.     # See if any of the low prime numbers can divide num:
84.     for prime in LOW_PRIMES:
85.         if (num == prime):
86.             return True
```

```
87.         if (num % prime == 0):
88.             return False
```

Page 333: The section of text starting from the second paragraph that reads:

Line 85 loops through each of the prime numbers in the `LOW_PRIMES` list:

```
84.     # See if any of the low prime numbers can divide num:
85.     for prime in LOW_PRIMES:
86.         if (num % prime == 0):
87.             return False
```

The integer in `num` is modded by each prime number using the mod operator on line 86, and if the result evaluates to `0`, we know that `prime` divides `num` so `num` is not prime. In that case, line 87 returns `False`.

Those are the two quick tests we'll perform to determine whether a number is prime. If the execution continues past line 87, the `rabinMiller()` function checks `num`'s primality.

should now read:

Line 84 loops through each of the prime numbers in the `LOW_PRIMES` list:

```
83. # See if any of the low prime numbers can divide num:
84. for prime in LOW_PRIMES:
85. if (num == prime):
86. return True
87. if (num % prime == 0):
88. return False
```

If the integer in `num` is the same as `prime`, then obviously `num` must be a prime number and line 86 returns `True`.

The integer in `num` is modded by each prime number using the mod operator on line 87, and if the result evaluates to `0`, we know that prime divides `num` so `num` is not prime. In that case, line 88 returns `False`.

Those are the three quick tests we'll perform to determine whether a number is prime. If the execution continues past line 88, the `rabinMiller()` function checks `num`'s primality.

Page 341: The code line that reads:

```
64. print('The private key is a %s and a %s digit number.' %
     (len(str(publicKey[0])), len(str(publicKey[1]))))
```

should now read:

```
64. print('The private key is a %s and a %s digit number.' %
      (len(str(privateKey[0])), len(str(privateKey[1]))))
```