This is to solve the Sage portion of Chapter 5 of

"A journey through the realm of numbers:

from quadratic equations to quadratic reciprocity."

In [1]:
```
# Experimenting with the eulerphi-function.
```

In [2]:
```
def eulerphi(n):
    gcdlist = [gcd(k,n) for k in range(n)]
    return gcdlist.count(1)
```

In [3]:
```
for n in range(20):
    print(n, eulerphi(n))
```

```
0 0
1 1
2 1
3 2
4 2
5 4
6 2
7 6
8 4
9 6
10 4
11 10
12 4
13 12
14 6
15 8
16 8
17 16
18 6
19 18
```

In [4]:
```
all(gcd(m,n) > 1 or eulerphi(m*n) == eulerphi(m)*eulerphi(n)
    for m in range(1,n)
    for n in range(1,100))
```

Out[4]:
```
True
```

In [5]:
```
# We note that this is a slow code and using the built in
# euler_phi function runs much faster.
all(gcd(m,n) > 1 or euler_phi(m*n) == euler_phi(m)*euler_phi(n)
    for m in range(1,n)
    for n in range(1,100))
```

Out[5]:
```
True
```

In [6]:
```
# Counting irreducible quadratics.
```

In [7]:
```
def counting(p):
    Fpx = PolynomialRing(GF(p), 'x') # defines Fpx=F_p[x]
    x = Fpx.gen()          # this defines x as the variable in Fpx
    fulllist = [x^2+a*x+b for a in GF(p)
                          for b in GF(p)]
    primelist = [f for f in fulllist if is_prime(f)]
    return len(primelist)
```

In [8]:
```
counting(3)
```

Out[8]: 3

In [9]:
```
for p in range(1,20):
    if is_prime(p):
        print(p, counting(p))
```

```
2 1
3 3
5 10
7 21
11 55
13 78
17 136
19 171
```

In [10]:
```
# We may notice that in those examples the number
# of irreducible polynomials equals p(p-1)/2.
for p in range(1,50):
    if is_prime(p):
         print(p, p*(p-1)/2, counting(p))
```

```
2 1 1
3 3 3
5 10 10
7 21 21
11 55 55
13 78 78
17 136 136
19 171 171
23 253 253
29 406 406
31 465 465
37 666 666
41 820 820
43 903 903
47 1081 1081
```

In [11]:
```
# So let us verify this for all primes below 200.
all(p*(p-1)/2 == counting(p)
    for p in range(1,200) if is_prime(p))
```

Out[11]: True

In [12]:

```
# You could try to verify this also for all primes
# below 1000. However, this takes a very long time and it might
# be quicker to rigorously prove the claim for all p instead.
```

In [ ]:

```
```