TP3

Exercice 1

```
def racines(p,m):
    if not p.is_prime():
        print 'erreur: p non premier'
        return None
    if  not m in PositiveIntegers():
        print 'erreur: m non entier naturel'
        return None
    L= (p^m-1).divisors()
    return filter(lambda n: multiplicative_order(mod(p,n))== m, L)
```

```
p= 5
m= 6
p^m
```

```
15625
```

```
N = racines(p,m)
N
```

```
[7, 9, 14, 18, 21, 28, 36, 42, 56, 63, 72, 84, 93, 126, 168, 186,
217, 248, 252, 279, 372, 434, 504, 558, 651, 744, 868, 1116, 1302,
1736, 1953, 2232, 2604, 3906, 5208, 7812, 15624]
```

```
F = map(lambda n: euler_phi(n)//m, N)
F
```

```
[1, 1, 1, 1, 2, 2, 2, 2, 4, 6, 4, 4, 10, 6, 8, 10, 30, 20, 12, 30,
20, 30, 24, 30, 60, 40, 60, 60, 60, 120, 180, 120, 120, 180, 240,
360, 720]
```

```
add(F)     # nombre de corps finis distincts (mais tous isomorphes
entre eux) de cardinal 15625
```

```
2580
```

```
map(lambda n: euler_phi(n), N)
```

```
[6, 6, 6, 6, 12, 12, 12, 12, 24, 36, 24, 24, 60, 36, 48, 60, 180,
120, 72, 180, 120, 180, 144, 180, 360, 240, 360, 360, 360, 720,
1080, 720, 720, 1080, 1440, 2160, 4320]
```

```
A = PolynomialRing(ZZ,'X')
X = A.gen()
A
```

```
Univariate Polynomial Ring in X over Integer Ring
```

```
n=N[18]
n, F[18]
```

```
(252, 12)
```

```
Phi = cyclotomic_polynomial(n,X)
Phi
```

```
X^72 + X^66 - X^54 - X^48 + X^36 - X^24 - X^18 + X^6 + 1
```

```
d = quo(Phi.degree(),m)
d    # modulo p F se factorise en 12 facteurs tous de degré 6
```

```
12
```

```
Phibar = Phi.change_ring(GF(p))
Phibar,parent(Phibar)
```

```
(X^72 + X^66 + 4*X^54 + 4*X^48 + X^36 + 4*X^24 + 4*X^18 + X^6 + 1,
Univariate Polynomial Ring in X over Finite Field of size 5)
```

```
FF= Phibar.factor()
FF
```

```
(X^6 + 4*X^4 + X^3 + X^2 + 4) * (X^6 + 4*X^4 + 4*X^3 + X^2 + 4) *
(X^6 + X^5 + 4*X^3 + X + 4) * (X^6 + X^5 + 2*X^4 + 3*X^2 + X + 4)
(X^6 + X^5 + 2*X^4 + 4*X^3 + 3*X^2 + X + 4) * (X^6 + X^5 + 3*X^4 +
X^3 + 2*X^2 + X + 4) * (X^6 + 2*X^5 + 3*X^4 + 2*X^2 + 2*X + 4) *
(X^6 + 3*X^5 + 3*X^4 + 2*X^2 + 3*X + 4) * (X^6 + 4*X^5 + X^3 + 4*X
4) * (X^6 + 4*X^5 + 2*X^4 + 3*X^2 + 4*X + 4) * (X^6 + 4*X^5 + 2*X^
+ X^3 + 3*X^2 + 4*X + 4) * (X^6 + 4*X^5 + 3*X^4 + 4*X^3 + 2*X^2 +
4*X + 4)
```

```
LFF = [f[0]for f in list(FF)]
LFF
```

```
[X^6 + 4*X^4 + X^3 + X^2 + 4, X^6 + 4*X^4 + 4*X^3 + X^2 + 4, X^6 +
X^5 + 4*X^3 + X + 4, X^6 + X^5 + 2*X^4 + 3*X^2 + X + 4, X^6 + X^5
2*X^4 + 4*X^3 + 3*X^2 + X + 4, X^6 + X^5 + 3*X^4 + X^3 + 2*X^2 + X
4, X^6 + 2*X^5 + 3*X^4 + 2*X^2 + 2*X + 4, X^6 + 3*X^5 + 3*X^4 +
2*X^2 + 3*X + 4, X^6 + 4*X^5 + X^3 + 4*X + 4, X^6 + 4*X^5 + 2*X^4
3*X^2 + 4*X + 4, X^6 + 4*X^5 + 2*X^4 + X^3 + 3*X^2 + 4*X + 4, X^6
4*X^5 + 3*X^4 + 4*X^3 + 2*X^2 + 4*X + 4]
```

```
P = LFF[3]
P    # on choisit l'un des facteurs irréductibles de F mo
```

```
X^6 + X^5 + 2*X^4 + 3*X^2 + X + 4
```

```
K = GF(p^m,name='x',modulus=P)
K
```

```
Finite Field in x of size 5^6
```

```
K.cardinality()
```

```
15625
```

```
K.characteristic()
```

```
5
```

```
K.degree()
```

```
6
```

```
K.polynomial()
```

```
x^6 + x^5 + 2*x^4 + 3*x^2 + x + 4
```

```
x = K.gen()
x^6
```

```
4*x^5 + 3*x^4 + 2*x^2 + 4*x + 1
```

```
x^252, x^126  # x est bien d'ordre 256
```

```
(1, 4)
```

```
d = n.isqrt()
d
```

```
15
```

```
n_sur_d = n//d
n_sur_d
```

```
16
```

```
U = [x^r for r in range(0,d)]
U
```

```
[1, x, x^2, x^3, x^4, x^5, 4*x^5 + 3*x^4 + 2*x^2 + 4*x + 1, 4*x^5
2*x^4 + 2*x^3 + 2*x^2 + 2*x + 4, 3*x^5 + 4*x^4 + 2*x^3 + 4, x^5 +
x^4 + x^2 + x + 3, 3*x^4 + x^3 + 3*x^2 + 2*x + 1, 3*x^5 + x^4 +
3*x^3 + 2*x^2 + x, 3*x^5 + 2*x^4 + 2*x^3 + 2*x^2 + 2*x + 3, 4*x^5
x^4 + 2*x^3 + 3*x^2 + 3, 2*x^5 + 4*x^4 + 3*x^3 + 3*x^2 + 4*x + 4]
```

```
y = x^(-d)
y
```

```
3*x^4 + x^2 + 4*x
```

```
V = [y^q for q in range(0,n_sur_d+1)]
V
```

```
[1, 3*x^4 + x^2 + 4*x, x^3 + 3*x^2 + 4*x + 2, x^5 + x^4 + 2*x^3 +
2*x^2 + 1, 3*x^3, 4*x^5 + 3*x^3 + 3*x^2 + 3*x + 1, x^5 + x^4 + x^3
x^2 + 2*x + 3, x^5 + x^4 + 3*x^2, x^5 + 2*x^4 + 3*x^2 + x + 4, x^5
4*x^4 + 2*x^3 + 3*x^2 + x + 2, x^5 + 3*x^4 + x^3 + 2*x^2 + 3*x + 2
3*x^4 + 2*x^3 + x^2 + x + 4, 2*x^5 + 2*x^4 + 2*x^2 + 2*x + 1, 2*x^
+ x^3 + x^2 + 3*x + 1, 3*x^5 + 3*x^4 + 4*x^2 + 1, 3*x^5 + 4*x^4 +
2*x + 2, 3*x^5 + 2*x^4 + 2*x^3 + 2*x^2 + 2*x + 3]
```

```
z1 =x^5 + 2*x^2 + x + 3
z1
```

```
x^5 + 2*x^2 + x + 3
```

```
for v in V:
    zz = z1*v
    if zz in U:
        r = U.index(zz)
        q = V.index(v)
        break
k= d*q +r
q,r,k, z1 == x^k
```

```
(8, 3, 123, True)
```

```
z2 = 3*x^4 + 3*x^3 + x^2 + 2*x + 4
z2
```

```
3*x^4 + 3*x^3 + x^2 + 2*x + 4
```

```
del r
for v in V:
    zz = z2*v
    if zz in U:
        r = U.index(zz)
```

```
        q = V.index(v)
        break
k= d*q +r
q,r,k
```

```
Traceback (click to the left of this block for traceback)
...
NameError: name 'r' is not defined
```