Q 1) – Avec des indices pythoniens, tout est plus simple :

$$F_n = (\omega_n^{k\ell})_{0\leqslant k,\ell<n}.$$

Le code le plus simple étant, de notre point de vue, le meilleur, on peut définir la matrice F_n de la manière suivante (une liste de lignes, qu'on convertit en tableau numpy au moment de renvoyer la valeur calculée).

```
import numpy as np
import numpy.linalg as alg

def F(n):
   omega = np.exp(2j*np.pi/n)
   M = []
   for k in range(n):
      ligne = [ omega**(k*ell) for ell in range(n) ]
      M.append(ligne)
   return np.array(M)
```

REMARQUE.— Si on veut limiter au strict minimum les opérations effectuées, on peut utiliser le code suivant.

```
def F(n):
   omega = np.exp(2j*np.pi/n)
   Fn = np.zeros((n,n), dtype=np.complex)
   facteur = 1
   for k in range(n):
      coeff = 1
      for ell in range(n):
         Fn[k, ell] = coeff
         coeff *= facteur
      facteur *= omega
   return Fn
```

Plus économique, c'est incontestable, mais moins évident et, de ce fait, moins sûr.

❧ Les calculs sont effectués en virgule flottante, il faut penser à arrondir les valeurs d'un coup d'œil pour connaître les matrices exactes.

$$F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad F_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & j & j^2 \\ 1 & j^2 & j \end{pmatrix} \quad F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Q 2) – Le produit $F_n.\overline{F_n}$ est facile à calculer.

```
def F_Fbarre(n):
   Fn = F(n)
   return np.dot(Fn, Fn.conjugate())
```

En arrondissant les résultats fournis, on conjecture que

$$\forall\, n \geqslant 2, \quad F_n.\overline{F_n} = nI_n.$$

Q 3) – Rien de plus compliqué avec l'inverse.

```
def F_inv(n):
   return alg.inv(F(n))
```

Rien de vraiment neuf par rapport à la question précédente non plus : on conjecture que

$$\forall\, n \geqslant 2, \quad F_n^{-1} = \frac{1}{n} \cdot \overline{F_n}.$$

Q 4) – Et toujours rien de compliqué pour le calcul des puissances : le module d'algèbre linéaire a déjà pensé à tout !

```
def Fnk(n, k):
    A = F(n)
    return alg.matrix_power(A, k)
```

Si on sait ce qu'on cherche, on trouve $F_2^4 = 4I_2$, $F_3^4 = 9I_3$, $F_4^4 = 16I_4$... et on conjecture que

$$\forall\, n \geqslant 2, \quad F_n^4 = n^2 I_n.$$

Si on ne connaît pas le résultat à établir, on aura peut-être la chance de commencer par calculer F_n^2 pour différentes valeurs de n avant de conjecturer que

$$\forall\, n \geqslant 2, \quad F_n^2 = \left(n\delta_{i+j\equiv 0\ [n]}\right)_{0\leqslant i,j<n} = \begin{pmatrix} n & & 0 \\ & \iddots & \\ 0 & & n \end{pmatrix} \in \mathfrak{M}_n(\mathbb{R}).$$

REMARQUE.— La matrice F_n intervient dans l'algorithme de la transformation de Fourier rapide (FFT). On pourra partir à la recherche du sujet EPITA 1999 pour trouver quelques précisions.