1. **Questions préliminaires**
   (a) `int_to_car(21)` est de type chaine de caractère (`str`).
   Sa valeur est `'V'`.
   (b) `car_to_int("F")` est de type entier (`int`).
   Sa valeur est `5`.
2. **Chiffrement de Vigenère**
   (a) Fonction `duplique(ch, n)` :

```
def duplique(ch, n):
    return ch * n
```

   (b) Fonction `car_to_listofint` :

```
def car_to_listofint(ch):
    L = []
    for c in ch:
        L.append(char_to_int(c))
    return L
```

   (c) Fonction `somme` :

```
def somme(L1,L2):
    n = len(L1)
    L = []
    for k in range(n):
        L.append((L1[k] + L2[k]) % 26)
    return L
```

   (d) Fonction `listofint_to_car` :

```
def listofint_to_car(L):
    ch = ""
    for n in L:
        ch += int_to_char(n)
    return ch
```

   (e) Fonction `chiffrementVigenere` :

```
def chiffrementVigenere(text_clair,cle):
    CLE = duplique(cle, len(text_clair) // len(cle))
    L1 = car_to_listofint(text_clair)
    L1 = car_to_listofint(CLE)
    L = somme(L1,L2)
    return listofint_to_car(L)
```

3. **Déchiffrement**
   (a) Fonction `difference` :

```
def difference(L1, L2):
    n = len(L1)
    L = []
    for k in range(n):
        L.append((L1[k] - L2[k]) % 26)
    return L
```

   (b) Fonction `dechiffrementVigenere` :

```
def dechiffrementVigenere(texte_chiffre, cle):
    CLE = duplique(cle, len(texte_chiffre) // len(cle))
    L1 = car_to_listofint(texte_chiffre)
    L2 = car_to_listofint(CLE)
    L = difference(L1, L2)
    texte_clair = listofint_to_car(L)
    return texte_clair
```

4. **Cryptanalyse par analyse fréquentielle**
   (a) Aucune (erreur d'énoncé) ; une fonction renvoyant l'indice de coincidence est :

```
def indice_coincidence(ch1, ch2):
    compteur = 0
    n1 = len(ch1)
    n2 = len(ch2)
    i = 0
```

```
    while i < n1 and i < n2:
        if ch1[i] == ch2[i]:
            compteur = compteur + 1
            i = i+1
    return compteur/i
```

(b) Fonction `test_Friedman` :

```
def test_Friedmann(ch):
    n = len(ch):
    for d in range(1,n):
        if indice_coincidence(ch, ch[d:]) >= 0.07:
            return d
```

(c) Fonction `lettre_plus_freq` :

```
def lettre_plus_freq(ch):
    nbr_max = 0
    lettre = ch[0]
    for car in ch:
        nbr = ch.count(car)
        if nbr > nbr_max :
                nbr_max = nbr
                lettre = car
    return lettre
```

(d) Fonction `determine_cle` (code complété) :

```
def determine_cle(txt_chiffre):
    """renvoie la cle du texte chiffre
       par la methode de Vigenere"""
    # calcul de la longueur de la cle
    d = test_Friedman(txt_chiffre)
    # determination de chaque caractere de la cle
    cle = ""
    for k in range(d):
        car = lettre_plus_freq(txt_chiffre[k::d])
        cle = cle + int_to_car((car_to_int(car)-4)%26)
    return cle
```

(e) Fonction `cryptanalyse_Vigenere` :

```
def cryptanalyse_Vigenere(txt_chiffre):
    cle = determine_cle(txt_chiffre)
    txt_clair = dechiffrementVigenere(txt_chiffre, cle)
    return txt_clair
```