```python
from numpy import array

# Fonction qui vérifie si le message contient uniquement des caractères valides
def msgvalide(msg):
    i = 0
    cvalide=" '"  # Caractères autorisés : espace et apostrophe
    # Vérifie qu'il n'y a pas deux espaces consécutifs
    valide =msg!=""  and msg.find("  ") == -1 and msg[0]!=" " and msg[len(msg)-1]!=" "

    # Vérifie chaque caractère du message
    while i < len(msg) and valide:
        # Le caractère est valide s'il est une lettre OU fait partie des caractères spéciaux autorisés
        valide = ((msg[i].upper() >= "A" and msg[i].upper() <= "Z") or cvalide.find(msg[i]) != -1)
        i = i + 1
    return valide

# Fonction qui demande un message valide à l'utilisateur
def saisir_message():
    msg = input("donner le message à crypter")
    # Redemande tant que le message n'est pas valide
    while(msgvalide(msg) == False):
        msg = input("donner le message à crypter")
    return msg

# Fonction qui compte le nombre de mots dans le message
def nbmot(msg):
    nb = 1
    # Compte les espaces pour déterminer le nombre de mots
    for i in range(len(msg)):
        if msg[i] == " ":
            nb = nb + 1
    return nb  # Nombre de mots = nombre d'espaces + 1

# Fonction qui extrait le premier mot d'un message
def construiremot(msg):
    mot = ""
    i = 0
    # Lit les caractères jusqu'à rencontrer un espace ou la fin de la chaîne
    while i < len(msg) and msg[i] != " ":
        mot = mot + msg[i]
        i = i + 1
    return mot

# Fonction qui répartit le message dans un tableau de mots
def repartir(msg, t, n):
    for i in range(0, n):
        mot = construiremot(msg)  # Extrait le mot suivant
        t[i] = mot                 # Stocke dans le tableau
        # Supprime le mot extrait + l'espace du message
        msg = msg[len(mot) + 1:len(msg)]

# Fonction qui trouve la longueur maximale parmi tous les mots
def longmot(t, n):
    lmax = len(t[0])
    for i in range(n):
        if len(t[i]) > lmax:
            lmax = len(t[i])
    return lmax

# Fonction qui complète un mot avec des étoiles jusqu'à la longueur désirée
def etoile(mot, lmax):
    while(len(mot) < lmax):
        mot = mot + "*"  # Ajoute une étoile à la fin
    return mot

# Fonction qui complète tous les mots avec des étoiles pour qu'ils aient la même longueur
```

```python
def padding(t, n, t1, lmax):
    for i in range(n):
        t1[i] = etoile(t[i], lmax)

# Fonction qui affiche un tableau avec des tirets entre les éléments
def affichertab(t, n):
    for i in range(0, n):
        print(t[i], end="-")

# Fonction qui crée un mot crypté à partir de la colonne j des mots complétés
def cryptermot(t1, n, j):
    mot = ""
    # Prend le j-ème caractère de chaque mot complété
    for i in range(n):
        mot = mot + t1[i][j]
    return mot

# Fonction qui effectue le cryptage par transposition colonne par colonne
def cryptage(t1, n, lmax, t2):
    # Pour chaque colonne, crée un mot crypté
    for i in range(lmax):
        t2[i] = cryptermot(t1, n, i)

# Fonction qui reconstruit le message crypté à partir des mots cryptés
def construiremessage(t2, lmax):
    msgcrypt = t2[0]
    # Joint tous les mots cryptés avec des espaces
    for i in range(1, lmax):
        msgcrypt = msgcrypt + " " + t2[i]
    return msgcrypt

# Programme principal
msg = saisir_message()                    # Demande le message à l'utilisateur
n = nbmot(msg)                            # Compte le nombre de mots
t = array([str]*n, dtype="U20")           # Tableau pour les mots originaux
t1 = array([str]*n, dtype="U20")          # Tableau pour les mots complétés avec étoiles
repartir(msg, t, n)                       # Sépare le message en mots
affichertab(t, n)                         # Affiche les mots originaux
lmax = longmot(t, n)                      # Trouve la longueur maximale des mots
t2 = array([str]*lmax, dtype="U20")       # Tableau pour les mots cryptés
padding(t, n, t1, lmax)                   # Complète tous les mots avec des étoiles
print()
affichertab(t1, n)                        # Affiche les mots complétés
print()
cryptage(t1, n, lmax, t2)                 # Effectue le cryptage par colonnes
affichertab(t2, lmax)                     # Affiche les mots cryptés
print()
print(construiremessage(t2, lmax))        # Affiche le message crypté final
```