

Éléments de correction
sujet 03

Exercice 1

1. On doit arriver à 25 caractères, on a pour le moment 17 caractères, on doit donc ajouter 8 espaces.
2. `An(3 espaces)algorithm(3 espaces)must(2 espaces)be`
3. `assert nb_caracteres + (nb_mots-1) <= justification`
- 4.

```
def ajout_espace(liste_mots, justification):
    nb_caracteres = sum([len(mot) for mot in liste_mots])
    nb_mots = len(liste_mots)
    nb_espace_total = justification - nb_caracteres
    if nb_mots == 1:
        return liste_mots[0] + " " * nb_espace_total
    else :
        q = nb_espace_total // (nb_mots - 1)
        r = nb_espace_total % (nb_mots - 1)
        reponse = liste_mots[0]
        for i in range(1, r + 1):
            reponse = reponse + " " * (q + 1) + liste_mots[i]
        for i in range(r + 1, nb_mots):
            reponse = reponse + " " * q + liste_mots[i]
        #ligne ajoutée sinon par le dernier mot :
        reponse = reponse + liste_mots[-1]
    return reponse
```

5. On place le plus de mots possibles sur la première ligne puis de passer à la 2e ligne (on place là aussi le plus de mots possible) et ainsi de suite
 - 6.
- ```
def affiche_justifie(liste_mots, decoupage, justification):
 for i,j in decoupage:
 ligne_justifiee = ajout_espace(liste_mots[i:j],
justification)
 print(ligne_justifiee)
```
- 7.

| Coût total du découpage : 147 |             |           |            |         |      |
|-------------------------------|-------------|-----------|------------|---------|------|
| i mot deb                     | i mot fin+1 | nbre mots | nombre car | esp sup | coût |
| 0                             | 2           | 2         | 11         | 3       | 9    |
| 2                             | 4           | 2         | 6          | 8       | 64   |
| 4                             | 7           | 3         | 8          | 5       | 25   |
| 7                             | 8           | 1         | 8          | 7       | 49   |

8.

```
def cout(i, j, liste_mots, justification):
 n_mots = j - i
 n_caracteres = sum([len(liste_mots[k]) for k in range(i,j)])
 if n_caracteres + (n_mots - 1) > justification:
 return 1000000
 else:
 return (justification - n_caracteres - n_mots + 1) ** 2
```

9.

Non, ce n'est pas raisonnable, car la complexité algorithmique est trop grande ( $2^n$ )

10.

coût quadratique car boucles imbriquées

11.

```
cout_mini[j] = min(cout_mini[j], cout(i, j, liste_mots, justification))
pour j compris entre i+1 et n exclu (voir la boucle for)
```

12.

```
return [cout_mini[0], decoupage]
```