Épreuves écrites de NSI
Correction du sujet
Centres étrangers 2021
Jour 1

Exercice 1

1. Le code renverra le résultat suivant :

```
D
A
```

2. on peut écrire le code suivant :

```
def cryptage(self, texte) :
    res = ""
    for car in texte :
        res = res + self.decale(car)
    return res
```

3. On peut écrire le programme suivant :

```
cle = input("Choisir la clé de chiffrement : ")
cle = int(cle)
code = CodeCesar(cle)
mes = input("Taper le message à chiffrer : ")
mes_chif = code.cryptage(mes)
print(mes_chif)
```

4. La méthode `transforme` exécute la méthode cryptage après avoir remplacée la clé de chiffrement par sa valeur opposée, rétablit la valeur initiale de la clé de chiffrement puis renvoie le résultat obtenu par le chiffrement avec une clé opposée. Elle permet donc d'inverser le processus et d'obtenir le message initial à partir du message chiffré.
L'instruction va donc transformer le message "PSX" en décalant les lettres de 10 places *vers la gauche*. On obtiendra alors le mot :

```
FIN
```

Exercice 2

1. a. `flotte[26]` renvoie le dictionnaire suivant :
`{"type" : "classique", "etat" : 1, "station" : "Coliseum"}`

b. `flotte[80]["etat"]` renvoie `0`.

c. `flotte[99]["etat"]` génère un message d'erreur car il n'existe pas de dictionnaire associé à la clé `99` dans le dictionnaire `flotte`.

2. a. L'argument `choix` doit être une valeur associée à la clé `"type"` dans les dictionnaires de la forme `dict_velo`, donc soit `"electrique"` soit `"classique"`.

b. La méthode `proposition` renvoie le nom d'une station pour laquelle il existe un vélo disponible pour le type `choix` (donc soit `"electrique"` soit `"classique"`) placé en argument.

3. a. On peut écrire le script suivant :

```
res = []
```

```
for k in flotte :
    if flotte[k]["station"] == "Citadelle" \
    and flotte[k]["etat"] == 1 :
        res.append(k)
print(res)
```

b. On peut écrire le script suivant :

```
res = []
for k in flotte :
    if flotte[k]["etat"] != -1 \
    and flotte[k]["type"] == "electrique" :
        res.append((k, flotte[k]["station"]))
print(res)
```

4. Proposons une fonction `possibilites(coord)` qui renvoie un dictionnaire dont les clés sont les noms de station et les valeurs associées sont des tuples formés d'une distance et d'un tableau des identifiants de vélos disponibles. On peut alors écrire le code suivant :

```
def possibilites(coord) :
    res = {}
    for id in flotte :
        s = flotte[id]["station"]
        dist = distance(stations[s], coord)
        if dist < 800 and flotte[k]["etat"] == 1 :
            if not s in res :
                res[s] = (dist, [k])
            else :
                res[s][1].append(k)
    return res
```

Exercice 5

1. On a le résultat suivant :

Étape 0 Pile d'origine P	Étape 1 `empiler(P, 8)`	Étape 2 `depiler(P)`	Étape 3 `est_vide(P)`
4 7 1 5	8 4 7 1 5	4 7 1 5	4 7 1 5
/	None	8	False

2. L'exécution de `transforme(P)` renvoie un tuple à deux éléments dont :

* le premier est une pile vide
* le second est la pile placée en argument, soit :

4
7
1
5

3. On peut écrire le code suivant :

```
def maximum(P) :
    if est_vide(P) :
        return None
    maxi = depiler(P)
    while not est_vide(P) :
        elem = depiler(P)
        if elem > maxi
            maxi = elem
    return maxi
```

4. En mettant à profit la fonction transforme du 2., on peut écrire le code suivant :

```
def taille(P) :
    P, Q = transforme(P)
    cpt = 0
    while not est_vide(Q) :
        elem = depiler(Q)
        empiler(P, elem)
        cpt += 1
    return cpt
```