Devoir maison N°1

Exercice 1: Cryptage selon le "code de César"

Q1 - Attribut cle = 3.

Le code renvoie (affiche):

D
A

Q2 -

```
def cryptage(self, texte):
    """ Crypte le texte avec la clé
    définie dans l'attribut clé
    Arg : texte : str
    """ retour : str
    crypte = ''
    for char in texte:
        crypte += self.decale(char)
    return crypte
```

Q3 -

```
def cryptage_texte():
    """ Crypte un texte demandé avec la
    clé demandée
    """
    cle = int(input('Quelle est la clé ?'))
```

```
Code = CodeCesar(cle)
Print (code.cryptage(texte))
```

Q4. Le programme applique le code de César avec une clé négative, cela a pour effet de décaler les lettres vers la gauche.

Affichage obtenu :

'FiN'

Exercice 2

Q5. flotte[26] -> {'type': 'classique', 'etat': 1, 'station': 'coliseum'}.

flotte[80]['etat'] -> 0

flotte[99]['etat'] -> 'erreur'

Q6. choix peut contenir les valeurs :

- "electrique"
- "classique"

La fonction renvoie la première station contenant un vélo disponible de type choix

Q7.

```
for id_ in flotte :
    if flotte [id_] ['station'] == 'Citadelle' :
        print (id_)
```

Q8.

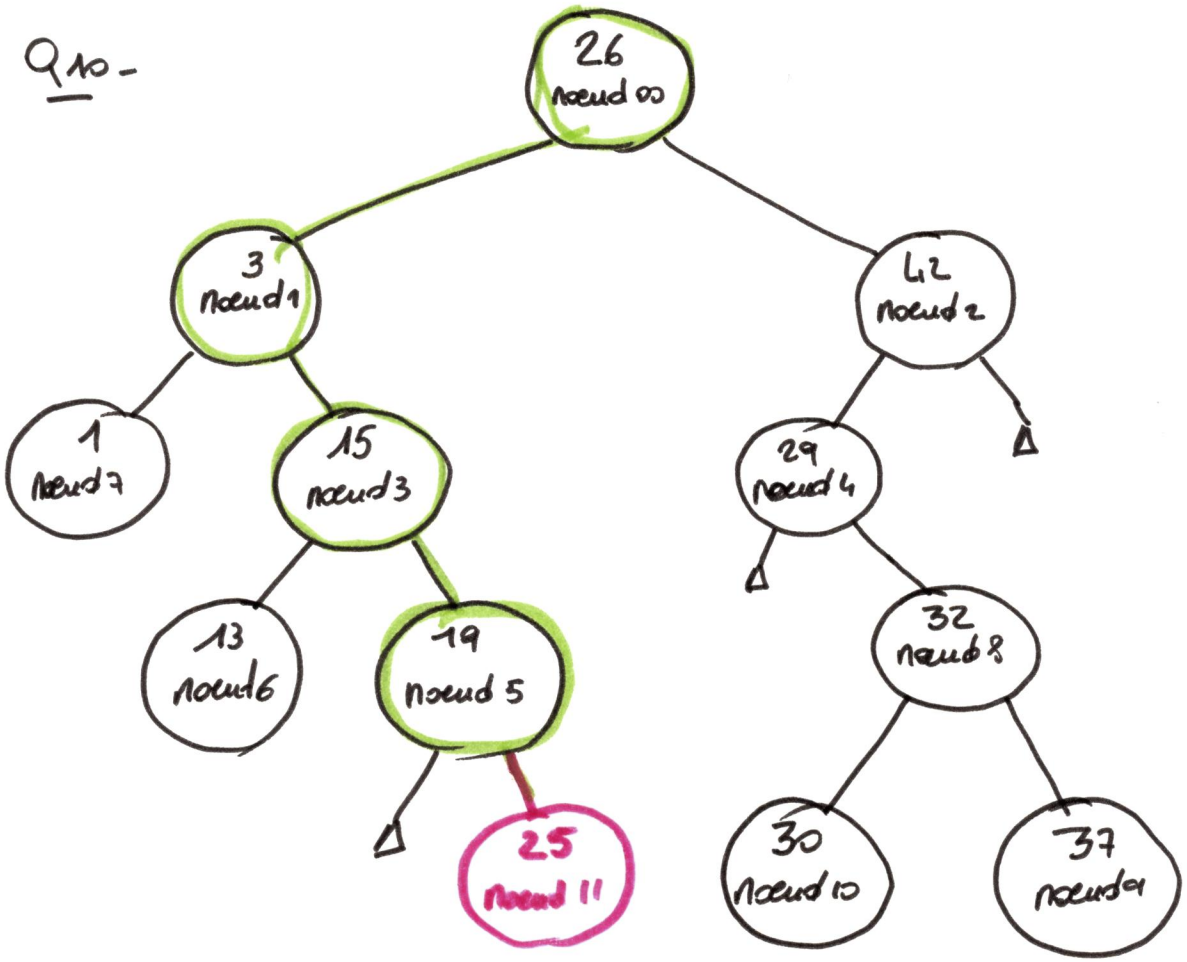
```
for id_ in flotte :
    if flotte [id_] ['type'] == 'electrique' and
       flotte [id_] ['etat'] != -1 :
        print (id, flotte [id_] ['station']
```

Q9.

```
def velo_proche (pos) :
    liste_stations = []
    for nom in stations :
        d = distance (stations[nom], pos)
        if d < 800 :
            velos_dispo = [id_ for id_ in flotte if
                           flotte [id_] ['etat'] == 1
                           and flotte [id_] ['station'] == nom]
            liste_stations . append ((nom, d, velos_dispo))
    return liste_stations
```

Exercice 3

Q10-



- La valeur 25 est placé en fils droit du noeud 05 de valeur 19.

Etape	Test effectué	chemin suivi
1	25 < 26	fils gauche de 26
2	25 > 3	fils droit de 3
3	25 > 15	fils droit de 15
4	25 > 19	fils droit de 19

Q11- Le fils gauche des noeuds peut contenir les valeurs comprises entre :

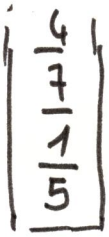
$[26, 29[$

Q12- Liste des valeurs affichées :

[26, 3, 1, 15, 13, 19, 42, 29, 32, 30, 37].

Exercice 4

Q13.



empiler(P, 8)

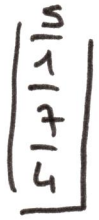
dépiler(P)

Valeur renvoyée
8

est_vide(P)

Valeur envoyée
False

Q14- transforme(P)



Q15-
```
def maximum(P):
    """Recherche la valeur maximale de
    la pile P """
    max=depile(P)
    while not(est_vide(P)):
```

Q16- Il faut créer une pile secondaire et une variable initialisé à 0 (cpt). Pour connaitre la taille il faut dépiler la pile dans la pile secondaire et pour chaque élément, ajouter 1 au compteur.
Dès que la pile est vide on la réinitialise avec le résultat de transforme(pile_secondaire).

```
def taille(p):
    """Détermine le nombre d'élément
    dans la pile P"""
    p_sec = creer_pile()
    cpt = 0
    while not(est_vide(p)):
        empile(p_sec, depile(p))
        cpt += 1
    p = transforme(p_sec)
    return cpt
```