Correction

NSI - 2021 Étranger Jour 1 (21-NSIJ1G11)

Exercice 1 - Cryptage selon le « Code de César »

1. Ce code affiche dans la console :
   D
   A

2.

```
    def cryptage(self, texte):
        """ Retourne le texte chiffré """
        code = ""
        for i in texte:
            code += self.decale(i)
        return code
```

Remarque : On parle plutôt de « chiffrement » que de « cryptage ». Seule la notion de décryptage a un sens puisque qu'elle consiste à décrypter un code sans en connaître la clé. On parle de « chiffrement » puisque la clé est forcément connue au moment du chiffrement, on ne peut pas crypter sans connaître la clé de chiffrement.

3.

Solution 1

```
# Demande de saisir la clé de chiffrement
cle = int(input("clé de chiffrement :"))
# Crée un objet de classe CodeCesar
cesar = CodeCesar(cle)
# Demande le texte à chiffrer
texte = input("texte à chiffrer :")
# Affiche le texte chiffré
print(cesar.cryptage(texte))
```

Solution 2 (Plus robuste)

```
try:
    # Demande de saisir la clé de chiffrement
    cle = int(input("clé de chiffrement :"))
    if -26 < cle < 26:
        # Crée un objet de classe CodeCesar
        cesar = CodeCesar(cle)
        # Demande le texte à chiffrer
        texte = input("texte à chiffrer :")
        # Affiche le texte chiffré
        print(cesar.cryptage(texte))
except ValueError:
    print("La clé est incorrecte.")
except KeyboardInterrupt:
    pass
```

4. Il s'affiche dans le console :
`FIN`

Cette ligne de code crée une instance de la classe `CodeCesar` avec comme attribut une clé de chiffrement de `10` puis appelle sa méthode `transforme` avec comme argument la chaîne de caractère `"PSX"`.
Cette méthode transforme la clé de chiffrement en son opposé et appelle la méthode `cryptage`. Ainsi, au lieu de décaler de 10 places vers la droite, le chiffrement du texte, décale de 10 places vers la gauche :
P devient F
S devient I
X devient N

Correction

NSI - 2021 Étranger Jour 1 (21-NSIJ1G11)

Exercice 2 - Gestion d'un parc de vélo

1.

L'instruction `flotte[26]` renvoie le dictionnaire `{'type': 'classique', 'etat': 1, 'station': 'Coliseum'}`.
L'instruction `print(flotte[80]["etat"])` renvoie l'entier `0`.
L'instruction `print(flotte[99]["etat"])` génère une erreur : **KeyError** car la clé `99` n'existe pas dans le dictionnaire `flotte`.

2. a. Les valeurs possibles de la variable `choix` sont les chaînes de caractères `"electrique"` et `"classique"`.

2. b. Cette fonction renvoie la station du premier vélo disponible et correspondant au choix du type de l'utilisateur. Les dictionnaires en python étant non-ordonnés, on ne peut pas savoir quel sera ce premier vélo.

3. a.

```
# Affiche les identifiants des vélos disponibles à la station Citadelle
for idv in flotte:    # équivalent à for idv in flotte.keys() :
    if flotte[idv]["station"] == station and flotte[idv]["etat"] == 1:
        print(idv)
```

3. b.

```
# Renvoie les identifiants et les stations des vélos électrique qui ne sont pas en panne.
for idv in flotte:    # équivalent à for idv in flotte.keys()
    if flotte[idv]["type"] == 'electrique' and flotte[idv]["etat"] != -1:
        print(idv, flotte[idv]["station"])
```

4.

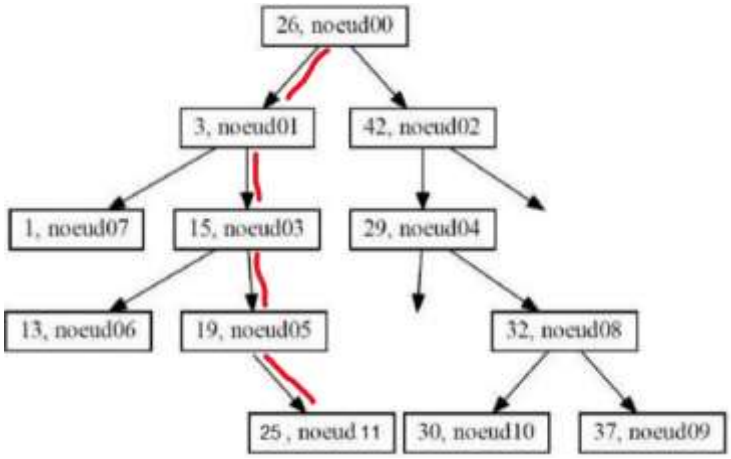
```
def dispo_proche(coord_u):
    """ Renvoie le nom, la distance et les identifiants des vélos disponibles
        pour chaque station située à moins de 800 mètres
        et ayant au moins un vélo disponible.
    """
    rep = []
    for nom_s, coord_s in stations.items():
        velos = dispo(nom_s)
        d = distance(coord_u, coord_s)
        if d < 800 and velos:
            rep.append((nom_s, d, velos))
    return rep
```

Correction

NSI - 2021 Étranger Jour 1 (21-NSIJ1G11)

Exercice 3 – Parcours d'arbre binaire de recherche

1.



25 < 26, on se déplace vers le fils gauche (nœud01).

25 > 3, on se déplace vers le fils droit (nœud 03).

25 > 15, on se déplace vers le fils droit (nœud05).

25 > 19, on place le nœud11 en fils droit.

2. En fils gauche du noeu04, on peut placer les valeurs 27 et 28. En effet en-dessous de 26, on passe par le fils gauche (noeud01) de la racine (noeud00) tandis qu'au-dessus de 29, on passe en fils droit du noeud04.

3. a. 26, 3, 1, 15, 13, 19, 25 (nœud ajouté à la question 1), 42, 29, 32, 30, 37

3. b. Il s'agit d'un parcours préfixe d'un arbre binaire.
En effet, on affiche le nœud lors du premier passage sur ce nœud.

4. Pour afficher les valeurs dans l'ordre croissant pour un arbre binaire de recherche, il faut réaliser un parcours infixe (deuxième passage) :

```
Parcours(A) # A est un arbre binaire de recherche
   Parcours(A.fils_gauche)
   Afficher(A.valeur)
   Parcours(A.fils_droit)
```

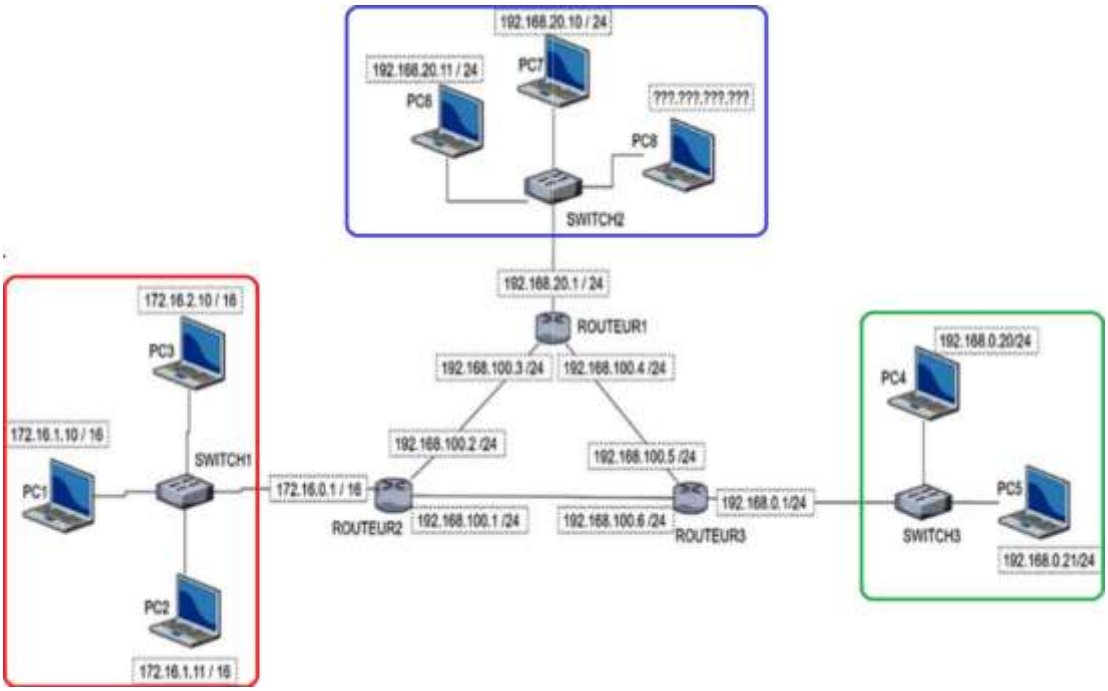
Correction

NSI - 2021 Étranger Jour 1 (21-NSIJ1G11)

Exercice 4 - Étude d'un réseau informatique

PARTIE A : Étude de l'adressage IP

1.



2. a. Quatre octets sont nécessaire pour composer une adresse IPv4.

2. b. $168_{(10)} = 10101000_{(2)}$ et $10_{(10)} = 00001010_{(2)}$

Adresse IP (V4) du PC7	Ligne 1	192	168	20	10
	Ligne 2	11000000	10101000	00010100	00001010

2. c. et d.

Masque de sous réseau	Ligne 3	11111111	11111111	11111111	00000000
	Ligne 4	255	255	255	0

i. et ii.

Pour obtenir l'adresse réseau binaire, on réalise un ET(&) logique entre chaque bit de l'adresse IP (ligne 2) et du masque de sous réseau (ligne3)					
Adresse du réseau	Ligne 5	11000000	10101000	00010100	00000000
	Ligne 6	192	168	20	0

3. Pour connecter le PC8, les adresses IP possibles sont :

~~192.168.20.0~~ Réservé pour l'adresse du sous-réseau

~~192.256.20.11~~ Mauvaise adresse du sous-réseau. Quand bien même, il y aurait un conflit d'adresse IP avec le PC6.

192.168.20.30

192.168.20.230

~~192.168.20.260~~ Adresse impossible. La valeur maximale est 254. Le 255 est réservé pour le broadcast.

~~192.168.27.11~~ L'adresse du sous-réseau n'est pas correcte

Correction

NSI - 2021 Étranger Jour 1 (21-NSIJ1G11)

PARTIE B : Une fonction pour convertir une adresse IP en décimal pointé en notation binaire.

```
def IP_bin(add_IP):
    """ Renvoie la conversion en binaire d'une adresse IP décimale. """
    add_IP_bin = []
    for n in add_IP:
        add_IP_bin.append(dec_bin(n))
    return add_IP_bin
```

Hors-sujet :

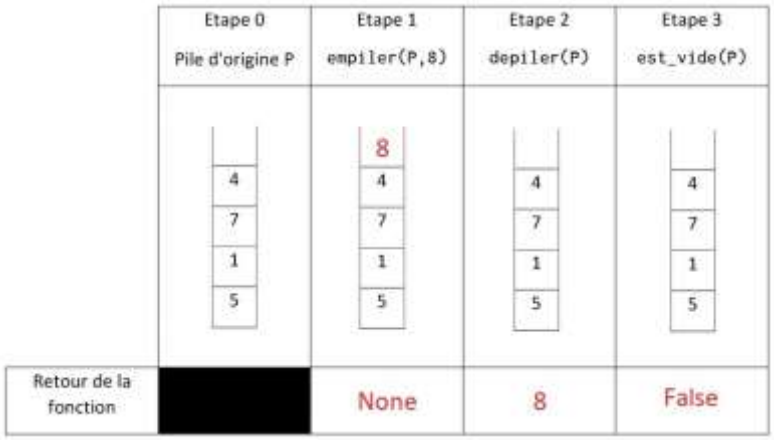
```
def dec_bin(n):
    """ Convertie l'entier n en binaire (liste). """
    n = list(str(bin(n))[2:])
    n = list(map(lambda x: int(x), n))
    while len(n) < 8:
        n = [0]+n
    return n
```

Correction

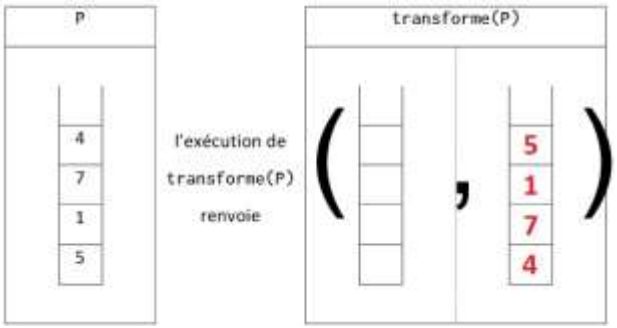
NSI - 2021 Étranger Jour 1 (21-NSIJ1G11)

Exercice 5 - Objet Pile

1.



2.



3.

```
def maximum(pile):
    """ Renvoie la valeur maximale de la pile. """
    if len(pile) == 0:
        return None
    m = depiler(pile)
    while not est_vide(pile):
        temp = depiler(pile)
        if temp > max:
            m = temp
    return m
```

4. a.

Le sujet ne le précise pas mais je fais l'hypothèse que la pile finale doit être identique à la pile initiale.

On initialise un compteur à 0.
Tant que la pile1 n'est pas vide, on dépile ses éléments en incrémentant un compteur et on les empile dans une pile2.
Enfin on dépile tous les éléments de pile2 pour les remettre dans la pile1 afin qu'elle retrouve sont état initial.
On retourne la valeur du compteur.

4. b.

```
def taille(pile1):
    """ Renvoie la taille de la pile. """
    n = 0  # compteur
    pile2 = creer_pile()
    while not est_vide(pile1):
        empiler(pile2, depiler(pile1))
        n += 1
    # restoration de pile1
    while not est_vide(pile2):
        empiler(pile1, depiler(pile2))
    return n
```