Initiation à la programmation avec Python
Correction des travaux pratiques

Matthieu Falce

Novembre 2020

1 Algorithmes, pseudo-code et algorigrammes

1.1 Un bien grand nom

L'algorithme fait les actions suivantes :

- il déclare une variable (chaîne) contenant un nom d'utilisateur
- il calcule la taille de cette chaîne
- si la taille est supérieure à 10 caractères, il affiche "votre nom est long", sinon il affiche : "votre nom est court"

La version pseudo code ressemble à ceci :

```
DEBUT
    CHAINE  Nom_Utilisateur <-- "Matthieu Falce"
    ENTIER  Nb_Lettres <-- nombre de lettres de Nom_Utilisateur
    SI Nb_Lettres < 10 ALORS
        AFFICHE "Votre nom est court"
    SINON
        AFFICHE "Votre nom est long"
FIN
```

1.2 C'est lui

La version pseudo code ressemble à ceci :

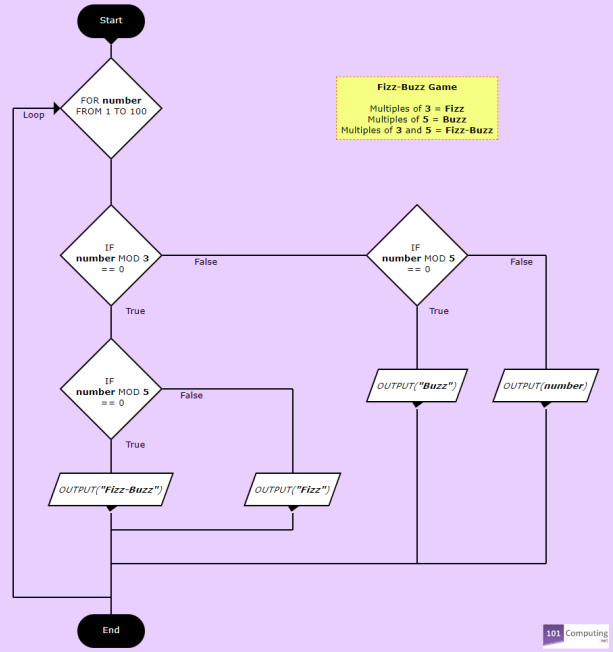
```
DEBUT
    LISTE  nombres <-- 1, 2, 3, 4, 5
    ENTIER  max <-- --1, min <-- 101, nb_nombres <-- taille de nombres

    POUR i <-- 0 A nb_nombres FAIRE
        ENTIER nombre <-- nombres[i]
        SI nombre > max ALORS
            max <-- nombre
        SI nombre < min ALORS
            min <-- nombre
FIN
```

2 Manipulation de la syntaxe python

2.1 *Fizz Buzz*

L'algorithme peut se représenter sous la forme de cet algorigramme :

```
POUR i <-- 1 A 100 FAIRE
    SI i est divisible par 3 ALORS
        SI i est divisible par 5
            AFFICHE "FizzBuzz"
        SINON
            AFFICHE "Fizz"
    SINON SI i est divisible par 5 ALORS
        AFFICHE "Buzz"
    SINON
        AFFICHE i
```



Source : `https://www.101computing.net/fizz-buzz-game-algorithm/`

Pour information, il existe plusieurs manières de résoudre ce problème. L'algorithme d'une autre méthode serait le suivant :

```
POUR i <-- 1 A 100 FAIRE
    SI i est divisible par 3 ET i est divisible par 5 ALORS
        AFFICHE "FizzBuzz"
    SINON SI i est divisible par 3 ALORS
        AFFICHE "Fizz"
    SINON SI i est divisible par 5 ALORS
        AFFICHE "Buzz"
    SINON
        AFFICHE i
```

Voir le code de correction situé : `corrections/syntaxe/fizzBuzz.py`

2.2 Plus ou moins

Réponses :

1.
```
DEBUT
    FONCTION ALEA
    ENTIER  cible <-- ALEA(0, 100), entree <-- -1
    TANT QUE cible != entree FAIRE
        AFFICHER "Entrer un entier: "
        LIRE entree
        SI entree > cible ALORS
            AFFICHER "Le nombre à trouver est plus petit"
        SINON SI entree < cible ALORS
            AFFICHER "Le nombre à trouver est plus grand"
        SINON
            AFFICHER "BRAVO, vous avez trouvé !"
FIN
```

2. on utilise la fonction `randint` qui provient du module `random`
3. on utilise la fonction `builtin` qui permet de demander une entrée à l'utilisateur : `input`
4. on ne sait pas à l'avance le nombre d'itérations nous devons faire, cela dépend du joueur et du nombre d'essais qu'il lui faut pour valider la condition d'arrêt. On doit donc utiliser une boucle `while`

Voir le code de correction situé : `corrections/syntaxe/plusOuMoins.py`

2.3 Magic 8 ball

Sous *Windows* il peut y avoir des soucis à l'ouverture du fichier à cause de l'encodage de caractère de cet *OS*. Il faut indiquer un encodage lors de l'ouverture du fichier dans ce cas.

Réponses :

1. on utilise la fonction `open(nomDuFichier)`
2. on utilise la méthode `.readlines()` du fichier ouvert précédemment
3. pour tirer un élément au hasard dans une liste, on peut générer un nombre aléatoire qui aura comme taille maximale la longueur de la liste -1. Sinon on utilise la fonction `choice` du module `random` .

Voir le code de correction situé : `corrections/syntaxe/magic8ball.py`

3 Projet

3.1 Contexte

Rien à faire dans cette section.

3.2 Avant propos

Les forces de python :

— dans le service, nous manipulons surtout du texte et c'est facile en python
— le langage est accessible au plus grand nombre, même peu sensibilisé au développement
— l'écosystème est très riche et permet de gérer la quasi-totalité des cas que nous allons rencontrer
— ...

3.3 Cryptanalyse

Réponses :

1. la chaine des lettres en majuscule est contenue dans `string.ascii_lowercase` et les majuscules dans `string.ascii_upperase` . Pour les convertir en liste, on utilise simplement la fonction `list`
2. la façon la plus élégante consiste à utiliser du slicing. Par exemple, pour un décalage de 2 on aurait `lettres[2:] + lettres[:2]` . Sinon, on peut le faire en utilisant une boucle et l'accès aux éléments de la liste en fonction d'un index (attention à bien mettre un modulo pour ne pas dépasser la taille d'index maximale).
3. si le caractère n'est pas dans `string.ascii_lowercase` ou `string.ascii_uppercase` ce n'est pas une lettre de l'alphabet ?
4. si le caractère n'est pas une lettre de l'alphabet nous ne le modifions pas, sinon nous le décalons ?
5. si l'on sait décaler une lettre, il suffit d'itérer sur la chaîne entière et de décaler tous les caractères un après l'autre ?
6. voir dans la correction du projet

3.3.1 Craquage

Réponses :

1. on utilise un dictionnaire avec comme clé la lettre et comme valeur le nombre de répétitions. Sinon la bibliothèque standard fourni `collections.Counter` qui fait le même travail.
2. il suffit de soustraire les valeurs ASCII des caractères. `ord("C") - ord("A") == 2`
3. pour craquer le code, nous devons
   — trouver quelle est la lettre la plus commune
   — regarder le nombre de lettres de décalage avec "e" qui est la lettre la plus commune en français.
   — décaler le texte chiffré avec la valeur trouvée à l'étape précédente

3.3.2 Déchiffrement

Voir le code de correction du sujet.

3.4 Extraction

Réponses :

1. on utilise la fonction `open`
2. on utilise la méthode `readlines()` pour lire toutes les lignes du fichier dans une liste.

3.5 Collecte

Réponses :

1. nous pouvons utiliser `requests` car elle est plus simple d'utilisation
2. on utilise la commande `python -m pip install requests --user` ?
3. on regarde l'attribut `rep.status_code` qui indique le code de statuts de la requête ?
4. C'est `random.150ml.com/cesi/boring_wozniak.txt`

3.6 La solution

Réponses :

1. oui, il est chiffré avec un code de César ayant une rotation de 13
2. il enquêtait sur Bernardo Provenzano qui est connu pour avoir utilisé ce chiffrement.

Voir le code de correction situé : `corrections/projet/dechiffrageCesar.py`