Exercices Pythons

1

Table de multiplication par 9

Affichez la table de multiplication par 9 en une seule commande avec les instructions `range()` et `list()`.

Correction

```
print(list(range(0,90,9)))
```

2

Nombres pairs

Répondez à la question suivante en une seule commande. Combien y a-t-il de nombres pairs dans l'intervalle `[2, 10000]` inclus ?

Correction

```
print(len(list(range(2,10000,2))))
```

3

Parcours de liste

Créer une liste d'une taille de 100 dont les valeurs des éléments sont les entiers naturels croissants de 0 à 99.

1. Définir une fonction permettant le parcours de ce tableau élément par élément et l'afficher
2. Définir une fonction permettant le parcours de ce tableau de façon à ce que alternativement nous prenions l'élément n+3 puis l'élément n-1. Soit, le troisième élément suivant, puis l'élément précedent. Afficher à chaque fois la valeur de l'élement.
3. Définir une fonction parcourants la liste de la même manière que dans 1. Sur chaque élément vérifier s'il est paire ou impaire, et l'ajouter à une liste correspondante (liste_paire et liste_impaire). *Il peut être utile d'utiliser le reste de la division euclidienne.*

Correction

```
liste_naturelle = list(range(100))

def funAffichage(liste):
        for i in liste:
                print(i)

funAffichage(liste_naturelle)

def funParcoursChelou(liste):
```

```
    boole=True
    while i<len(liste)-1:
        if boole:
            i=i+3
            print(f"{liste[i]} & est impair")
            boole= not boole
        else:
            i=i-1
            print(f"{liste[i]} & est pair")
            boole= not boole

funParcoursChelou(liste_naturelle)

def funTrierPaire(liste):
    liste_paire= []
    liste_impaire= []
    for i in liste:
        if i %2==0:
            liste_paire.append(i)
        else:
            liste_impaire.append(i)
            return liste_paire, liste_impaire
liste_paire1, liste_impaire1 = funTrierPaire(liste_naturelle)
```

4

Plus fort que la pomme

Créer une liste d'une taille de 1000 dont les valeurs des éléments sont les entiers naturels croissants de 0 à 999.

1. Définir une fonction permettant de calculer la somme de l'intégralité des nombres stocké dans la liste précedente.
2. Était-ce la seule façon ?

Correction

```
list1=list(range(1000))
n=len(list1)-1

def funcSum(list1):
    x=0
    for i in list1:
        x=i+x
    return x

def funcGauss(n):
    return n*(n+1)/2
```

5

Les tables de multiplications

Créer un tableau où chacune des cases est un tableau de manière à reconstituer les tables de multiplications comme au collège. Pour celà nous pourrons utiliser les tableaux de tableaux, en effet, Python

nous permet de créer des listes, où les éléments de cette liste sont eux aussi des listes.

Correction

```
def multi(x):
    res = []
    for i in range(x):
        line = []
        for j in range(x):
            line.append(i*j)
        res.append(line)
    return res

multiX=multi(10)
```

6

Mini Maxi et sans tricher

Créer deux listes de tailles 100 dont la valeur de chacun des éléments est défini de façon aléatoire.

1. Créer une fonction permettant de trouver le maximum de cette liste et son indice
2. Créer une fonction permettant de trouver le minimum de cette liste et son indice

IL EST INTERDIT D'UTILISER MIN et MAX

Correction

```
import random
liste1=[]
for i in range(100): # parcourir toute la liste
    a=random.randint(0,100)
    liste1.append(a)

# trouver le maxi
def maxi(liste):
    x = liste[0]
    for i in liste:
        if x<i:
            x=i
    print(x)

maxi(liste1)

# trouver le mini
def mini(liste):
    y = liste[0]
    j=0
    for i in liste:
        if y>i:
            y=i
            pos_mini=j
        j+=1
    print(y,pos_mini)

mini(liste1)
```

7

Upside down

Créer une liste de taille 10 d'entier numérique croissant.

- Inverser cette liste sans utiliser *reverse()*.

Correction

```
import random
listeExo=list(range(10))
listeRandom=[]

for i in range(10):
    a=random.randint(0,100)
    listeRandom.append(a)

def funReverse(liste):
    liste1 = liste[:]
    liste2 = []
    for i in range(10):
        a=liste1[-1]
        liste1.pop()
        liste2.append(a)
    return liste2
```

8

Le bibliothécaire

Créer la liste de lettres représentant l'alphabet, les minuscules suffiront.
À partir de la liste précédente, créer une liste de lettre aléatoire de taille 100.

1. Créer une fonction permettant de compter le nombre de fois qu'une lettre apparait dans le tableau

```
def comptageLettreSimple(lettre):
        ## Compter le nombre de fois que la lettre en paramètre est présente
```

2. Créer une fonction permettant de compter chacune des lettres et stocker les résultat dans un tableau

```
def comptageLettre():
        ## Compter le nombre de fois que chacune des lettres de l'alphabet est présent dans
votre tableau
```

Correction

```
import random

liste1=list("abcdefghijklmnopqrstuvwxyz")
liste2= []

for i in range(100):
    a=random.randint(0,25)
    liste2.append(liste1[a])
```

```
def comptageSimple(lettre):
    a=0
    for i in liste2:
        if i==lettre:
            a=a+1
    return a

L = comptageSimple("a")

liste3= []

def comptageMultiple():
    for i in liste1:
        print(i)
        liste3.append(comptageSimple(i))
comptageMultiple()
```

9

Le loto

Créer une liste dans laquelle figure les numéros du loto.
Simulons le comportement du loto :

1. Créer une fonction permettant de choisir un élément de la liste précedente, puis de le supprimer.
2. Stocker cet élément dans une nouvelle liste
3. Effectuer un tirage du loto (5 chiffre)

Correction

```
import random
liste1=list(range(1,50))
liste2= []

def tirage(liste):
    a=liste1[random.randint(1,len(liste))]
    liste1.remove(a)
    liste2.append(a)
    return liste2
# print(tirage(liste1))

def loto(liste):
    for i in range(5):
        tirage(liste)
    return liste2
print(loto(liste1))
```

10

Le palindrome

Un palindrome est un mot ou une phrase qui se lit à l'envers. Nous allons créer une fonction qui permet de vérifier que notre mot, puis notre phrase est bien un palindrome.

1. Créer la liste de lettres représentant l'alphabet, les minuscules suffiront.

2. Créer une fonction qui vérifie que notre mot est un palindrome.

```
def palindromeMot(mot):
        is_palindrome = False
        ##écrire les instructions pour vérifier que notre mot est un palindrome

        return is_palindrome
```

3. Créer une fonction qui transforme une phrase en une chaine de charactères sans espaces, ni accent, ni ponctuation.

```
def transformationPhrase(phrase):
        ##écrire les instructions pour transformer notre phrase
        return phrase_transformee
```

4. Créer la fonction qui vérifie qu'une phrase est un palindrome.
5. Tester la fonction.

Correction

```
def palindromeMot(mot):
    for i in range(int(len(mot)/2)):
        a=mot[-i-1]
        b=mot[i]
        print(f"{a} & {b}")
        if a!=b:
            return False
    return True

palindromeMot(input("mot? : "))


def transformationPhrase(phrase):
        phrase = phrase.lower()
    listePhrase=[]
    for i in phrase:
        if i == " ":
            a = 0
        if i == "," or i == "." or i == "?" or i == "!":
            a = 0
        if i == "é" or i =="è" or i =="ê":
            listePhrase.append("e")
        if i == "à":
            listePhrase.append("a")
        if i == "ù":
            listePhrase.append("u")
        if i in liste1:
            listePhrase.append(i)
    print(listePhrase)

def palindrome(phrase):
    if palindrome(transformationPhrase(phrase)):
        print("je suis un palindrome")
    else:
        print("je suis pas un palindrome")
palindrome("Élu par cette crapule !")
```

11

Le chiffrement de EGUCT

Dans l'antiquité pour coder les messages, Cesar employait une méthode de chiffrement de ces messages. Le chiffrement par décalage ou chiffrement de Cesar est une méthode qui consiste à remplacer une lettre de l'alphabet par une lettre avec un décalage d'ordre de l'alphabet. Ce décalage était la clef secrète à connaitre pour pouvoir déchiffrer le message.

```
alphabet = ["
","a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","
w","x","y","z"]
clef = 6

def chiffrementCesar(phrase,clef):
        phraseChiffre = []
        ###Ici utiliser la clef et l'alphabet pour décaler les indices des lettres de la
phrase. Stocker dans la liste phraseChiffre les nouvelles lettres pour écrire la nouvelle
phrase chiffrée
```

De la même manière déchiffrer un message chiffré par la méthode de Cesar en connaissant la clef. Comment coder un programme pour deviner la clef ? Quel était la clef que j'ai utilisé pour le titre de l'exercice ?

Correction

```
alphabet = ["
","a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","
w","x","y","z"]
clef = 7

def trouverIndiceLettre(lettre):
    cpt=0
    for i in alphabet:
        if i==lettre:
            return cpt
        cpt+=1
    print("error, this letter is not in the alphabet")


print(trouverIndiceLettre("e"))

def chiffrementCesar(phrase,clef):
    phraseChiffre = []
    for i in phrase:
        indiceActuel = trouverIndiceLettre(i)
        indiceNouveau_temp = indiceActuel+clef
        indiceNouveau = indiceNouveau_temp % len(alphabet)
        lettreNouveau = alphabet[indiceNouveau]
        phraseChiffre.append(lettreNouveau)
    return phraseChiffre

def dechiffrementCesar(phraseChiffre,clef):
    phraseClaire = []
    for i in phraseChiffre:
        indiceActuel = trouverIndiceLettre(i) #indice récupéré pour calcul
        indiceNouveau_temp = indiceActuel-clef #nouvel indice, hors tableau
        indiceNouveau = indiceNouveau_temp % len(alphabet) #modulo sur tableau tourant
```

```
        lettreNouveau = alphabet[indiceNouveau]
        phraseClaire.append(lettreNouveau)
    return phraseClaire

phraseChiffre = chiffrementCesar("le chiffrement de cesar",6)
print(phraseChiffre)
print(dechiffrementCesar(phraseChiffre,6))
```

12

Le chiffrement de Vigenere

Le chiffrement de Cesar est très connu, et surtout facilement cassable en observant les récurrences d'une même lettre dans un mot. Alors Vigenere à tenter de créer une autre méthode plus difficile à cracker. Sa stratégie réside en prenant une clef à plusieurs chiffre, caché dans un mot. Cette série de chiffre est à répéter jusqu'à la fin. Aisni, il y a peu de chance pour qu'une lettre ai deux fois le même décalage.
https://fr.wikipedia.org/wiki/Chiffre_de_Vigenère
En se basant sur la description de Wikipédia, mettre en place le chiffrement de Vigenère. On pourra réaliser deux boucles imbriquée, une dédiée à la clef, une autre dédiée à la phrase à crypter.

13

Une histoire de nain

C'est l'histoire de sept nains qui récoltent de l'or pour le maître des nains. Les nains fabriquent avec cet or des lingots d'exactement 1 kg. Le problème est qu'un certain nombre de nains sont des voleurs (on ne sait pas combien) et à chaque fois qu'ils fabriquent un lingot ils volent 1g d'or et fabriquent ainsi des lingots de 999g.
Le maître des nains peut demander à chacun des nains de fabriquer autant de lingots qu'il le souhaite, les nains voleurs voleront systématiquement 1g. Les lingots sont numérotés donc il est possible de savoir qui a fabriqué le lingot.
Pour démasquer les voleurs le maître des nains dispose d'une balance très précise. Elle peut peser tout ce qu'on veut aussi lourd soit-il.
La question est : quelle est la stratégie du maître pour démasquer avec certitude tous les voleurs avec une seule pesée ?

Une fois la stratégie trouvée, modéliser le problème dans un programme Python et le résoudre de façon à ce que le programme trouve automatiquement le nain voleur.

Correction

```
import random
##crÃ©ation des nain
nain=[0]*7
##nombres de lingot par nain
for i in range(7):
    nain[i]=random.randint(64, 100)
##crÃ©ation du nain escrot
liste_escroc = []
for i in range(7):
    liste_escroc.append(random.randint(0, 1))

##On modÃ©lise les sac des nains
sacNain=[0]*7
```

```
##On attribue les lingots des nains
for i in range(len(sacNain)):
    x=1
    if liste_escroc[i]==1:
        x=0.99
    sacNain[i]=[x]*nain[i]


##Faire la pesée sachant que on a le droit qu'à  une seule pesée pour trouver le nain
escrot.
balance = 0

for j in range(7):
    nbLingot = 2**j
    balance += sacNain[j][0]*nbLingot
```

14

Le jeu du pendu

Le jeu du pendu consiste à faire deviner un mot à son adversaire en un nombre de coût limité. Afin de trouver le bon résultat, le joueur adverse doit proposer des lettres afin de compléter le mot. Lorsque la lettre est présente dans le mot, celle ci s'affiche dans le mot, sinon, il prend un malus. Lorsqu'il devine le mot le joueur à gagné, sinon, si ses malus dépassent 5 il a perdu.

Pour coder ce jeu il faut décomposer le programme en plusieurs fonctions :

1. Une fonction qui initialise le jeu en affichant des _ au lieu des lettres, sauf la première lettre.

```
def initialisationJeu(mot):
        ### Affiche uniquement des _ sauf pour la première lettre
```

2. Une fonction qui va vérifier que la lettre proposée est dans le mot à deviner.

```
def verification(lettre,mot):
        ### A coder
```

3. Une fonction qui va permettre d'afficher la lettre trouvée

```
def affichageMot(lettre,mot,mot_affiche_precedent):
        ### Afficher la lettre dans le mot, tout en gardant en mémoire le mot affiché
précedemment.
        return mot_affiche
```

4. Une fonction principale dans laquelle le joueur joue jusqu'à ce qu'il ai gagné ou qu'il ai perdu.

```
def jeu(mot):
        gagne = False
        while not gagne:
                ###Jouer au jeu
```

Correction

```
def initialisationJeu(mot):
    mot_initial = list(mot)
    for i in range(1,len(mot)):
        mot_initial[i]="_"
    return mot_initial

def verification(lettre,mot):
    if lettre in mot:
        return True
    else:
        return False


def affichageMot(lettre, mot, mot_affiche_precedent):
    if mot_affiche_precedent is not list:
        mot_affiche_precedent =list(mot_affiche_precedent)
    for i in range(len(mot)):
        if mot[i]==lettre:
            mot_affiche_precedent[i]=mot[i]
    return mot_affiche_precedent

def jeu(mot):
    gagne = False
    vie=5
    mot_affiche_precedent=initialisationJeu(mot)
    print("".join(mot_affiche_precedent))
    while not gagne:
        lettre_demande = input("Quelle lettre ?")
        if len(lettre_demande)!=1:
            print("Recommencez la saisie")
            continue
        else:
            if verification(lettre_demande,mot):
                print("Bien joué")

mot_affiche_precedent=affichageMot(lettre_demande,mot,mot_affiche_precedent)
                print("".join(mot_affiche_precedent))
            else:
                vie-=1
                if vie<0:
                    print("C'est perdu !")
                    return
                print(f"Dommage il ne te reste que {vie} coups")
        gagne=mot_affiche_precedent==list(mot)
    print("Bravo !")

jeu("test")
```

15

Echo echo echo echo

La récursivité permet de relancer une fonction pendant son execution. En suivant le cours sur la récursivité, essayons de créer une fonction qui nous permet d'afficher Echo dans la console un nombre infini de fois.

```
def infiniteEcho():
        print("Echo")
        ### S'appeler soi même pour faire un echo infini
```

Utiliser ctrl+c pour arrêter le traitement

Essayons de recoder cette fonction de façon à ce qu'elle s'effectue que 3 fois.

```
def tripleEcho(n):
        ###Avec une condition répeter trois
```

Correction

```
def infiniteEcho():
        print("Echo")
        infiniteEcho()
```

```
def tripleEcho(n):
    if n==0:
        return
    print("Echo")
    tripleEcho(n-1)

tripleEcho(3)
```

16

```
phrase = "¿Hola que tal?"
```

En utilisant la récursivité compter son nombre de caractères

Correction

```
def longueur(phrase):
        if len(phrase)<1:
                return 0
        return 1+longueur(phrase[1:])

longueur("kururukukurukusteuchsteuch")
```

17

Mon ami Fibo

Toute les suite arithmétiques peuvent s'écrire sous forme de fonction récursives.
Quel est la fonction récursive qui permet d'écrire la suite de Fibonacci ?

Correction

```
def fiboCursif(n):
    if n in [0,1]:
        return n
    return fiboCursif(n-1)+fiboCursif(n-2)

liste_fibo=[]

for i in range(10):
```

```
    liste_fibo.append(fiboCursif(i))
```

18

La paire y en a pas deux

Un nombre N est paire si (N−1) est impaire, et un nombre N est impair si (N−1) est pair.
Ecrire deux fonctions récursives mutuelles pair(N) et impair(N) permettant de savoir si un nombre N est pair et si un nombre N est impair.

Correction

```
def paire(n):
    if n==0:
        return True
    return impaire(n-1)

def impaire(n):
    if n==0:
        return False
    return paire(n-1)

print(paire(10))
print(paire(7))
print(impaire(10))
print(impaire(7))
```

19

Maximus

Soit un tableau X de N entiers, écrire une fonction récursive simple permettant de déterminer le maximum du tableau.

Correction

```
def maximum(list):
    if len(list) == 1:
        return list[0]
    else:
        m = maximum(list[1:])
        if m > list[0]:
            return m
        else:
            return list[0]
```

Il existe une multitude de façon de réaliser cet exercice. Le mécanisme principal est de faire réduire la liste jusqu'à trouver un seul élement. Dans cete façon je compare toujours le maximum que j'avais avant au premier élement de ma liste. J'aurais pu aussi faire une recherche par dichotomie pour aller plus vite.

20

Palindrôme récursif

Je dois vraiment l'expliquer ?

Correction

```
def palindromRecursif(phrase):
    if len(phrase)<2:
        return True
    if phrase[0]!=phrase[-1]:
        return False
    else:
        return palindromRecursif(phrase[1:-1])
```

Il est possible aussi d'ajouter des lignes afin de traiter la phrase pour supprimer les espaces et la ponctuation.

21

Arbre de noël

- Écrire une fonction triangle en langage python qui prend en paramètre un entier (n) et permet d'afficher un triangle isocèle formé d'étoile(*).
- La hauteur du triangle (c'est a dire le nombre des lignes (n))sera fournie en programme principale,comme dans l exemple ci dessous.

```
saisir la hauteur du triangle : 10
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
*******************
```

Correction

```
def arbreNoel(n):
    for i in range(1,n+1,2):
        print(int((n-i)/2)*" "+i*"*")

arbreNoel(10)
```