Examen Final P17

Sujet adapté pour Python

Éléments de solutions

1ère Partie : Gestion d'un annuaire *(5 points)*

Vous devez proposer un programme Python de gestion d'un annuaire. Pour chaque personne stockée dans l'annuaire, les informations suivantes sont renseignées : nom, prénom, numéro (dans la rue), nom de la rue, numéro de téléphone, code postal, ville. Votre programme doit permettre la saisie des informations, des recherches et des extractions suivant des critères choisis par l'utilisateur.

1. Définissez la structure correspondant à la personne à l'aide des informations citées précédemment. Définissez l'annuaire en tant que tableau contenant ce type de données.
2. Votre programme de gestion de l'annuaire doit contenir certaines fonctionnalités structurées sous forme de fonctions et procédures appelées dans le programme principal. Vous devez définir :
    a. Une fonction *saisie_tab* permettant la saisie de toutes les données pour toutes les entrées de l'annuaire et retournant le tableau correspondant à cet annuaire.
    b. Une fonction *critere_recherche*, sans arguments, qui permet à l'utilisateur de choisir le critère de recherche (nom, prénom, nom de la rue, numéro de téléphone, code postal, ville). Cette fonction doit retourner le choix de l'utilisateur.
    c. Une fonction *recherche* à deux arguments : le tableau annuaire et le critère de recherche (Cf. 2.b). L'utilisateur doit pouvoir donner la valeur de recherche (par exemple Compiègne si le critère de recherche est 'ville'). Cette fonction doit retourner un tableau booléen de la taille du tableau annuaire contenant la valeur `True` pour les entrées de l'annuaire qui correspondent à la recherche demandée.
    d. Une procédure *affiche_tab* à deux arguments : le tableau annuaire et un tableau du type de celui retourné par la fonction *recherche* précédente. Toutes les informations relatives aux personnes correspondant au critère de recherche doivent être affichées à l'écran.

```
def saisie_tab():
    annuaire = []
    continuer = True
    while continuer:
        personne = {}
        personne['nom'] = input('Nom ? ')
        personne['prenom'] = input('Prénom ?')
        personne['rue'] = input('Rue ? ')
        personne['numero'] = input('Numéro dans la rue ? ')
        personne['tel'] = input('Numéro de téléphone ? ')
        personne['code'] = input('Code postal ? ')
        personne['ville'] = input('Ville ? ')
        annuaire.append(personne)
        reponse = input('Autre personne (O/N) ?')
```

```
        continuer = reponse.upper() == 'O'
    return annuaire

def critere_recherche():
    print('Citère de recherche : ')
    reponse = input('(N)om, (P)rénom, (R)ue, (T)éléphone, (C)ode postal,
(V)ille :')
    table = {'N':'nom','P':'prenom','R':'rue','T':'tel', 'C':'code',
'V':'ville'}
    return table[reponse]

def recherche(annuaire, critere):
    resultat = [False]*len(annuaire)
    valeur = input('Valeur')
    for i in range(len(annuaire)):
        if annuaire[i][critere] == valeur:
            resultat[i] = True
    return resultat

def affiche_tab(annuaire, resultat):
    for i in range(len(annuaire)):
        if resultat[i] :
            print('*******')
            for cle, valeur in annuaire[i].items():
                print (cle, valeur)
```

2ème Partie : Récursivité *(5 points)*

2. 1 : La fonction itérative suivante, `divin(a,b)`, retourne le résultat de la division entière de *a* par *b* :

```
def divin(a,b):
    d = 0
    while a >= b :
        d = d + 1
        a = a - b
    return d
```

Il s'agit en fait du calcul `a // b` en Python.

Vous devez écrire la version récursive de cette fonction. Pour cela :

a) Déterminez l'expression récursive et le critère d'arrêt de la séquence des appels récursifs.
b) Ecrivez la fonction récursive `divinRec(a,b)` qui fournit le résultat de `a div b`, en utilisant uniquement les opérateurs "+" et/ou "-".

```
def divinRec(a,b):
    if a < b:
        return 0
    else:
        return 1 + divinRec(a - b, b)
```

2.2 : On souhaite écrire une fonction récursive prenant en paramètres deux valeurs, *a* un réel et *b* un entier, et retournant la valeur réelle $\boxed{a^b}$

a) Quel(s) critère(s) d'arrêt proposez-vous ?
b) Ecrivez la fonction
c) Combien y a-t-il d'appels récursifs pour *a=2* et *b=3* ? Décrivez graphiquement les appels récursifs dans ce cas.

Pour b positif. On peut étendre facilement à b < 0

```
def puissance(a, b):
    if b == 0:
        return 1
    else:
        return a * puissance(a, b - 1)
```

3ème Partie : Fichiers *(5 points)*

Dans cette partie, on suppose que l'on dispose d'un fichier texte composé exactement d'un mot par ligne, en minuscules et sans espacement. Chaque procédure ou fonction fait les opérations d'ouverture et de fermeture des fichiers dont le nom est passé en paramètre sous forme de chaine de caractères.

3.1 – Ecrire la fonction `nbMotsAvecVoyelle(nomf)` :

qui renvoie le nombre de mots commençant par une voyelle et présents dans le fichier de nom `nomf`.

3.2 – On considère maintenant un fichier de nom `nomf1` trié par ordre lexicographique (*i.e.* alphabétique) croissant.

Ecrire la procédure `compterChaqueMot(nomf1,nomf2)` qui écrit chaque mot du fichier `nomf1` de façon unique dans le fichier `nomf2`, suivi sur la même ligne d'un espace et du nombre d'occurrences de ce mot.

N.B. : La procédure ne fera qu'un seul passage sur le fichier `nomf1`. Celui-ci étant trié, on utilisera le fait que les occurrences d'un même mot sont forcément consécutives.

```
def nbMotsAvecVoyelle(filename):
    voyelles = "aeiouy"
    nb = 0
    infile = open(filename, 'r')
    for ligne in infile:
        if ligne[0] in voyelles:
            nb = nb + 1
    infile.close()
    return nb

def compterChaqueMot(nomf1, nomf2):
    infile = open(nomf1, 'r')
```

```
    outfile = open(nomf2, 'w')
    mot = infile.readline()
    while mot != '':
        nbMots = 1
        mot_suivant = infile.readline()
        while mot_suivant != '' and mot_suivant == mot:
            nbMots = nbMots + 1
            mot_suivant = infile.readline()
        outfile.write(mot + ' ' + str(nbMots) + '\n')
        mot = mot_suivant

    infile.close()
    outfile.close()


print(nbMotsAvecVoyelle('mots.txt'))
compterChaqueMot('mots.txt', 'comptemots.txt')
```

4ème Partie : Retour vers le futur *(5 points)*

1) On considère le programme Python suivant :

   On souhaite représenter une personne à l'aide des champs suivants :
   - nom # nom de la personne (chaine de caractères)
   - annee # année actuelle de la personne (entier)
   - temps (entier) # temps (en secondes) nécessaire pour # revenir en 2017

   Ecrire en Python une fonction `Saisie()` permettant de saisir le nom des personnes et retournant ces noms dans un tableau `t`
   Le nombre de personnes n'est pas connu à l'avance. Ce tableau sera par la suite utilisé pour calculer et ajouter les champs `annee` et `temps` de chaque personne.

2) Chaque personne doit choisir une période d'au moins 10 ans au cours de laquelle elle souhaite faire un voyage dans le temps. Le programme choisit aléatoirement une année de départ au cours de cette période.

   Ecrire une procédure `calculAnnee` en Python qui pour chaque personne du tableau `t` :
   - demande dans quelle période, donnée par `annee_min` et `annee_max`, elle souhaite faire un voyage dans le temps. Cette période sera comprise entre -10 000 ans et 10 000 ans.
   - appelle la fonction `random.randint(annee_min, annee_max)` et stocke l'année renvoyée par cette fonction dans le champ `annee` de la personne.

3) Le retour en l'an 2017 se fait également de manière aléatoire par essais successifs. Ecrire une procédure `calculTemps` en Python qui, pour chaque personne du tableau `t`, calcule le temps nécessaire pour revenir en l'an 2017 et le stocke dans le champ `temps` de cette personne, sachant que :

   - la période de saut est forcément de 10 ans autour de 2017, i.e. entre 2012 et 2022,

- le temps pour chaque saut nécessite 10 secondes.

4) Prendriez-vous ce risque ?

```
def saisie():
    t = []
    continuer = True
    while continuer:
        personne = {}
        personne['nom'] = input('Nom ? ')
        t.append(personne)
        reponse = input('Autre personne (O/N) ?')
        continuer = reponse.upper() == 'O'
    return t

def annee(personne):
    " fonction qui retourne une année aléatoire correcte dans une période
choisie "
    print('Vers quelle période souhaitez-vous voyager ', personne['nom'],
'?' )
    print("Période d'au moins 10 ans et comprise entre -10000 et +10000 :")
    annee_min = int(input('Année de début ? '))
    annee_max = int(input('Année de fin ? '))

    while annee_min < -10000 or annee_max > 10000 or annee_max - annee_min
< 10 or annee_max < annee_min :
        print('Période incorrecte')
        annee_min = int(input('Année de début ? '))
        annee_max = int(input('Année de fin ? '))

    import random
    annee = random.randint(annee_min, annee_max)
    personne['annee'] = annee

def calculAnnee(t):
    " détermine l'année pour chaque personne"
    for i in range(len(t)):
        annee(t[i])

def calculTempsRetour(personne):
    " temps de retour pour une personne"
    AnneeCible = 2017
    mini = 2012
    maxi = 2022
    temps = 0
    annee = personne['annee']
    import random
    while annee != AnneeCible :
        annee = random.randint(mini, maxi)
        temps += 10
    personne['temps'] = temps

def calculTemps(t):
```

```
    for i in range(len(t)):
        calculTempsRetour(t[i])


t = saisie()
calculAnnee(t)
calculTemps(t)
print(t)
```

N.B. : la question 4 était optionnelle. Elle n’a pas été notée.