M.KANE

TD PYTHON 2nde A et B

Exercice 1

*Le maximum sous toutes ses formes

1. Écrire une fonction **maximum** qui prend en arguments deux nombres et qui renvoie le plus grand des deux.
2. Écrire une fonction **maximum3** qui prend en arguments trois nombres et qui renvoie le plus grand des trois. Vous ferez appel à votre fonction maximum.
3. Écrire une fonction **maximum3_input** qui ne prend pas d'arguments, qui demande trois nombres à l'utilisateur puis qui renvoie le maximum des 3 nombres. Vous ferez appel à votre fonction maximum3.

Exercice 2

Les fonctions demandées ne sont a priori pas dépendantes les unes des autres. Vous pouvez les écrire dans l'ordre que vous voulez.

1. Ecrire une fonction **f** qui prend un nombre x en argument et qui renvoie la valeur de $4x^2-x+5$.
2. Considérons un triangle ABC rectangle en A. Ecrire une fonction **coteAB**(coteAC, hypotenuse) qui prend en argument la longueur du côté AC et la longueur de l'hypoténuse et qui renvoie la longueur du côté AB.
3. Ecrire une fonction **conversion_feet_m** qui prend comme argument une longueur en pieds et qui renvoie la longueur en mètres correspondante. On utilisera le taux de conversion suivant: 1 pied=0.3048m.
4. Etant donnés deux points A et B de coordonnées (x_A,y_A) et (x_B,y_B), la distance Manhattan entre A et B est donnée par la formule suivante:

   $$d=|x_A-x_B|+|y_A-y_B|$$

   (les barres verticales désignent la valeur absolue, dont le nom en Python est *abs*) . Elle porte ce nom car c'est la distance qu'il faut parcourir pour aller de A à B en utilisant seulement des directions parallèles à l'axe des abscisses ou à l'axe des ordonnées (comme le quadrillage des rues de Manhattan). Ecrire une fonction **distance_manh**(xA, yA, xB, yB) qui renvoie la distance Manhattan entre A et B.

Exercice 3

Des petites fonctions numériques, la suite

1. Écrire une fonction **valeur_abs** qui prend en argument un nombre x et qui renvoie sa valeur absolue.
2. Écrire une fonction **signe_different** qui prend en arguments deux nombres x et y et qui renvoie True si x et y ont des signes différents, False sinon (si x ou y est égal à 0, la fonction doit renvoyer False).
3. Soit f la fonction mathématique définie sur R par $f(x) = 3x^2 + 2x + 3$. Écrire une fonction Python **f**

qui prend en argument un réel x et qui renvoie la valeur de f(x).

4. Écrire une fonction **nb_racines** qui prend en argument trois réels a, b, c et qui renvoie le nombre de racines réelles du polynôme $ax^2 + bx + c$. La fonction doit donc renvoyer un entier qui vaut 0, 1 ou 2 selon les cas. On rappelle la formule du discriminant : $\Delta = b^2 - 4ac$.

Exercice 4

Moyenne pondérée

Écrire une fonction **moyenne_ponderee** qui prend en arguments quatre nombres que l'on appellera note1, note2, coeff1 et coeff2 et qui calcule la moyenne pondérée de note1 et note2, ayant pour coefficients respectifs coeff1 et coeff2.

Exercice 5

Fonctions avec des chaines de caractères

Vous pouvez écrire les fonctions demandées dans l'ordre que vous voulez (et même en créer d'autres, si cela vous paraît utile). Votre programme principal ne sera pas évalué.

1. Ecrire une fonction **ajoute_prefixe** qui prend en argument un mot et un préfixe et qui renvoie la chaîne de caractère obtenue en concaténant le préfixe suivi du mot. Par exemple, *ajoute_prefixe("mentir", de")* vaut *"dementir"*.
2. Ecrire une fonction **repete** qui prend en argument un mot et un entier n et qui renvoie la chaîne de caractères obtenue en répétant le mot n fois à la suite (sans séparation). Par exemple, *repete("bla", 3)* vaut *"blablabla"*.
3. Ecrire une fonction **ajoute_longueur** qui prend comme argument un mot et qui ajouter sa longueur au début et à la fin. Par exemple, *ajoute_longueur("toto")* vaut *"4toto4"*.
4. *Note:* On rappelle que la longueur d'un mot est obtenue avec la fonction ***len***, par exemple *len("toto")* vaut *4*. De plus, pour convertir un entier ou flottant en chaînes de caractères, il faut utiliser le mot clé **str(...)**. Par exemple, *str(4)* vaut *"4"*.
5. Ecrire une fonction **que_des_nombres** qui prend en argument un mot et qui renvoie la chaîne de caractères obtenue à partir du mot en remplaçant chaque lettre par sa position dans le mot. Par exemple, *que_des_nombres("toto")* vaut *"1234"*, et *que_des_nombres("bonjour")* vaut *"1234567"*.

Exercice 6

Le journal de M. Bizarre

M. Bizarre est rédacteur-en-chef d'un journal, et a souvent des idées farfelues. Suivant les jours de la semaine, il souhaite que les articles de son journal suivent des règles qu'il a inventées.

1. Chaque lundi, il souhaite que les mots soient répétés deux fois, séparés par un espace. Ecrivez une fonction **lundi** qui prend en argument un mot et qui transforme ce mot selon la règle du lundi. Par exemple, lundi("bonjour") doit valoir "bonjour bonjour".
2. Chaque mardi, il faut que les mots de longueur paire soit répétés 6 fois, en séparant par un tiret, alors que les mots de longueur impaire doivent être répétés 3 fois, en séparant par des virgules. Ecrire la fonction **mardi** qui prend en argument un mot et qui renvoie la chaîne de caractères suivant la règle du mardi. Rappel : la longueur d'une chaîne de caractères est donnée par len(chaine).
3. Chaque mercredi, il faut que les mots de longueur impaire soit remplacés par le mot "impair". Les

mots pairs doivent rester tels qu'ils sont. Ecrire la fonction **mercredi** (qui prend en argument un mot et qui renvoie la bonne chaîne de caractères).

4. Chaque jeudi, il faut que les mots soient répétés autant de fois que leur longueur modulo 3 (à la suite, sans espace). Ecrire la fonction **jeudi**. Par exemple, jeudi("merci") vaut "mercimerci", jeudi("bonbon") vaut "", jeudi("comment") vaut "comment".
5. Le vendredi, il faut que les mots soient écrits normalement. Et heureusement, il n'y a pas de journal le week-end
6. Ecrire une fonction **transforme(mot, num_jour)** qui prend en argument un mot et le numéro du jour (1 pour lundi, 2 pour mardi, etc ...) et qui renvoie le mot transformé selon la règle du jour correspondant.

Exercice 7

Factorielle qui depasse

Écrire une fonction **depasse** qui prend en argument un entier A et qui renvoie le plus petit entier n tel que n! soit supérieur ou égal à A.

Par exemple, *depasse(120)* renvoie 5 car 5!=120. De même, *depasse(20)* renvoie 4 car 3!=6 mais 4!=24.

Exercice 8

*Moustiques

Depuis le début de l'année 2017, deux scientifiques Marc et Alice étudient l'évolution d'une population de moustiques sur l'île Chépaou. Ils ont réussi à obtenir l'estimation suivante sur l'évolution de la population : si la population contient x moustiques au cours d'une année, alors il y a aura $1.09x - 200$ moustiques l'année suivante. Par contre, ils ne sont pour l'instant pas d'accord sur l'estimation de la population en 2017 : ils s'accordent seulement sur le fait que ce nombre est compris entre 8 000 et 12 000. Il faudra donc considérer cette donnée comme une variable.

1. Écrire une fonction f qui prend en argument le nombre x de moustiques à une certaine année et qui renvoie le nombre de moustiques l'année suivante. La valeur de retour de f doit arrondir la réponse à l'entier inférieur. Comme les valeurs obtenues seront toujours positives, vous pouvez utiliser au choix int(..) ou la fonction floor(.) du module math. Exemple: int(4.5) vaut 4 et math.floor(6.75) vaut 6.
2. Écrire une fonction nb_moustiques qui prend en arguments nb_debut, le nombre estimé de moustiques en 2017, et un entier annee_voulue. La fonction doit renvoyer le nombre de moustiques qu'il y aura en annee_voulue.
3. Écrire une fonction annee_atteindra qui prend en argument un entier seuil et un entier nb_debut (qui correspondra au nombre de moustiques en 2017) et qui renvoie l'année à partir de laquelle le nombre de moustiques sera supérieur ou égal à seuil.
4. Question non-évaluée par Caseine: Écrire un programme principal qui demande à Marc son estimation du nombre de moustiques, puis à Alice la sienne. Votre programme demandera ensuite une année et affichera le nombre de moustiques qu'il y aura cette année-là, selon l'estimation de Marc puis selon celle d'Alice. Enfin, votre programme demandera un seuil et affichera en quelle année on atteint ce seuil, en fonction de chacune des deux estimations.

Exercice 9

M.KANE

*La banque

Aline envisage d'ouvrir un compte à la banque Argento, mais elle veut d'abord savoir si cela sera rentable. Sur un tel compte, les intérêts sont de 5% par an, et la banque prélève un coût fixe annuel de 11 euros. Le capital de l'année n + 1 est donc obtenu par la formule $u_{n+1} = u_n \times 1.05 - 11$, où u_n désigne le capital à l'année n.

1. Écrire une fonction capital(nb_annees, capital_debut) qui renvoie le capital en euros qu'Aline aurait sur un tel compte au bout de nb_annees en plaçant initialement un capital égal à capital_debut (en euros).
2. Écrire une fonction gagne_argent(nb_annees, capital_debut) qui renvoie True si le capital au bout de nb_annees sur un tel compte est supérieur ou égal au capital de début.
3. Écrire une fonction capital_debut_min(nb_annees) qui renvoie le capital minimal qu'il faut mettre initialement sur le compte pour qu'après nb_annees, il soit supérieur ou égal au capital de début. On supposera ici que le capital de début est toujours un entier. Par exemple, capital_debut_min(7) renvoit 220.

Exercice 10

**Un peu d'arithmétique

Dans cet exercice, on ne considérera que des entiers strictement positifs.

1. Écrire une fonction plus_grand_diviseur_premier(n) qui renvoie le plus grand diviseur premier de l'entier n. Il vous est conseillé de commencer par redéfinir la fonction est_premier(n) que l'on a déjà vue.
2. Écrire une fonction pgcd(a,b) qui renvoie le plus grand commun diviseur des entiers a et b.
3. Écrire une fonction ppcm(a,b) qui renvoie le plus petit commun multiple de a et b.
4. Écrire une fonction irreductible(numerateur, dénominateur) qui renvoie True si la fraction correspondante numerateur/denominateur est irréductible, False sinon.

Exercice 11

**Codes secrets de la pyramide

Vous partez à la recherche d'un trésor caché dans une pyramide. Vous avez réussi à obtenir des informations sur les épreuves qui vous attendent à l'intérieur de celle-ci:

- dans le hall d'entrée, une statue de sphinx va vous demander un code secret sous la forme suivante : *"Je n'aime que les nombres p qui sont des nombres premiers, tels que p+2 soit aussi un nombre premier... Et mon nombre préféré de la journée est *un nombre qui change tous les jours*. Le code secret est le plus petit nombre supérieur ou égal à mon nombre préféré parmi les nombres que j'aime."*

- une fois entré, vous parcourez un labyrinthe dont vous vous êtes déjà procuré le plan, puis vous arrivez face à une statue d'Osiris qui garde la salle du trésor. Il va vous demander un code secret sous la forme suivante: *"Je n'aime que les nombres n tels que n+1 soit une puissance de 2..... et il faut également que le reste de la division de n par 5 soit égal à 3 ... Mon nombre préféré de la journée est *un nombre qui change tous les jours*. Le code secret est le plus petit nombre supérieur ou égal à mon nombre préféré, parmi les nombres que j'aime."*

Pour chacune des deux énigmes, vous n'aurez que 5 secondes pour répondre, sinon la sortie sera verrouillée et vous resterez enfermés dans la pyramide à tout jamais. Pour éviter cela à tout prix, vous décidez d'écrire un programme que vous emporterez sur votre smartphone et qui vous aidera à résoudre les énigmes. Pour

cela:

1. Ecrire une fonction est_premier qui prend en argument un entier supérieur ou égal à 2 et qui renvoie True si l'entier en premier, False sinon.
2. Ecrire une fonction sphinx_aime qui prend en argument un entier supérieur ou égal à 2 et qui renvoie True si l'entier est un nombre que le sphinx aime, False sinon.
3. Ecrire une fonction code_hall qui prend en argument le nombre préféré du sphinx ce jour-là (un entier strictement positif) et qui renvoie le code secret du hall.
4. Ecrire une fonction est_puissance2 qui prend en argument un entier positif et qui renvoie True s'il s'agit d'une puissance de 2, False sinon.
5. Ecrire une fonction osiris_aime qui prend en argument un entier positif et qui renvoie True si Osiris aime cet entier, False sinon.
6. Ecrire une fonction code_tresor qui prend en argument le nombre préféré d'Osiris ce jour-là (un entier strictement positif) et qui renvoie le code secret permettant d'accéder au trésor.

Par exemple, si le nombre préféré du sphinx est 15, alors le code secret du hall ce jour-là est 17 (car 17 et 19 sont premiers). Si le nombre préféré d'Osiris ce jour-là est 20, alors le code secret de la salle du trésor est 63 (car 63+1 est une puissance de 2, et 63= 12 x 5+3).

Exercice 12

Se familiariser avec les arguments optionnels

1) Ecrire une fonction **moyenne** qui prend en argument deux notes, ainsi qu'un argument optionnel que l'on appelle bonus et dont la valeur par défaut est 0. La fonction doit renvoyer la moyenne de l'étudiant calculée ainsi: d'abord, on calcule la moyenne "normale" entre les deux notes, puis on ajoute le bonus. Attention, la moyenne renvoyée ne devra pas dépasser 20, même si le calcul expliqué précédemment donne un nombre strictement supérieur à 20.

Exemples:
moyenne(12,14) vaut 13.0
moyenne(12,14,1) vaut 14.0 (car 13 + 1)
moyenne(18,19,2) vaut 20

2) Supposons que l'on joue à un jeu où chaque manche se déroule de la manière suivante: d'abord, on lance un dé, et son résultat donne le nombre de points en jeu. Ensuite, le joueur peut, s'il le souhaite, déclencher son bonus *"compte double"* ou son bonus *"compte triple"*. Ensuite, le joueur répond à un certain nombre de questions, et calcule son pourcentage de bonnes réponses. Les points attribués au joueur pour cette manche sont alors calculés en fonction du pourcentage de bonnes réponses et du nombre de points en jeu arrondi à l'entier le plus proche (par exemple, avec 50% de bonnes réponses et 6 points en jeu, le joueur obtient 3 points; avec 45% de bonnes réponses et 8 points en jeu, le joueur obtient 4 points). Enfin, ce nombre de points est éventuellement multiplié par deux ou trois si le bonus correspondant a été utilisé.

Ecrire une fonction **points_manche** qui prend en argument le pourcentage, le nombre de points en jeu, et comme argument optionnel le multiplicateur (qui vaut 2 lors d'un compte double et 3 lors d'un compte triple), et qui calcule le nombre de points en jeu. Vous utiliserez la fonction round pour arrondir, par exemple round(3.6) renvoie 4.

Exemples:
points_manche(40,6) vaut 2
points_manche(45,8) vaut 4
points_manche(50,9,2) vaut 8
points_manche(20,3,3) vaut 3

3) Ecrire une fonction **division_arrondi** qui prend en argument un numérateur, un dénominateur, et comme arguments optionnels un booléen arrondi (valant False par défaut) et un entier decimales (valant 0 par défaut) et qui calcule la valeur de la fraction numerateur/denominateur (sauf si le dénominateur est nul, auquel cas votre fonction affichera un message d'erreur et ne renverra rien). Si le booléen arrondi vaut False, la fonction renverra ce résultat sans l'arrondir (si decimales est précisé lors de l'appel, on l'ignorera). Si le booléen arrondi vaut True, le résultat sera arrondi au nombre de décimales demandé (par défaut, à l'entier le plus proche). Pour faire vos arrondis, vous utiliserez round(nombre_a_arrondir, nombre_de_decimales).

Exemples:
division_arrondi(1,3) vaut 0.3333333333333333
division_arrondi(1,3,arrondi=True, decimales=2) vaut 0.3
division_arrondi(15,8) vaut 1.875
division_arrondi(15,8, True) vaut 1
division_arrondi(15,8, arrondi=True, decimales=2) vaut 1.88

Note: il se peut que, durant vos tests, vous remarquiez quelques aberrations de la fonction round. Ceci est un phénomène connu, dû à la difficulté de stocker certains nombres décimaux sur un ordinateur. Les nombres utilisés dans les tests automatiques ont été choisis de manière à ne pas tomber sur un cas problématique.

Exercice 13

**Carte et billet SNCF

Le but de cet exercice est de programmer un automate destiné à vendre des cartes de réduction et billets de train pour la SCNF.

Pour simplifier, on ne considérera que deux types de carte de réduction:

- la carte Jeune, prix 50€
- la carte Senior, prix 60€.

De même on ne considérera que les trajets suivants :

- Grenoble - Paris, prix 100€
- Grenoble - Lyon, prix 20€
- Lyon - Paris, prix 80€

Les prix correspondent à un aller simple plein tarif, et on suppose que le prix du retour est identique à celui de l'aller (par exemple, un Lyon - Grenoble vaut aussi 20€).

Les cartes de réduction offrent les avantages suivants :

- En période bleue (période creuse): 50% de réduction
- En période blanche (période de pointe): 30% de réduction avec une carte Jeune, et 25% de réduction avec une carte Senior.

De plus, un passager sans carte de réduction peut tout de même obtenir une réduction de 10% s'il choisit un billet non modifiable.

1) Ecrire une fonction tarif_carte qui prend en argument une chaîne de caractère correspondant au nom de la carte ("Jeune" ou "Senior") et qui renvoie le prix de la carte. Si un autre nom que "Jeune" ou "Senior" est utilisé, la fonction affichera "Carte inconnue" et ne renverra rien (ou renverra None, ce qui revient au même).

2) Ecrire une fonction plein_tarif qui prend en argument deux chaînes de caractères correspondant

respectivement à la ville de départ de la ville d'arrivée et qui renvoie le prix d'un billet plein tarif sur ce trajet-là. Si le trajet demandé n'est aucun des 6 trajets existants dans cet exercice, la fonction affichera "Trajet inconnu" et renverra None (ou rien).

3) Ecrire une fonction tarif_billet qui prend en argument la ville de départ, la ville d'arrivée et 3 arguments optionnels : un booléen modifiable indiquant si le billet est modifiable (True par défaut), un argument carte correspondant au nom de l'éventuelle carte de réduction (None par défaut; pourra valoir "Jeune" ou "Senior" dans les appels), et enfin un argument periode (None par défaut; pourra valoir "bleue" ou "blanche" dans les appels). Si le trajet ou la carte demandé(e) est inconnu(e), un message d'erreur s'affichera et la fonction renverra None.

4) *Ecrire une fonction prix_client qui ne prend aucun argument et qui renverra le prix total de la commande du client. Pour cela, votre fonction commencera par demander "Voulez-vous acheter une carte de réduction ? (oui/non) ". Si client tape *non*, on passe directement à l'achat du billet (décrit plus loin); si le client tape *oui*, on lui demande "Choisissez votre carte : (Jeune/Senior) " et enregistre le prix correspondant. Ensuite, on lui demande "Voulez-vous acheter un billet ? (oui/non) ". Si l'utilisateur tape *oui*, on lui demande "Depart : " puis "Destination : ". Si l'utilisateur vient d'acheter une carte de réduction, il doit voir s'afficher "Vous avez une carte de réduction" puis "Precisez la periode (bleue/blanche) : ". La fonction peut alors calculer le prix. Si l'utilisateur ne vient pas d'acheter une carte de réduction, on doit lui demander "Autres precisions a fournir ? (oui/non) ". Si l'utilisateur tape *non*, le prix calculé doit être celui d'un plein tarif modifiable. Sinon, on doit lui demander "Carte de reduction (Jeune, Senior, ou aucune)? ". Si l'utilisateur indique une carte, il faut lui demander "Precisez la periode (bleue/blanche): ", sinon il faut lui demander "Billet modifiable ? (oui/non) ". Avec toutes ces informations, la fonction calcule le bon prix. Dans tous les cas (si l'utilisateur n'achète qu'une carte, ou qu'un billet, ou aucun des deux), votre fonction doit terminer en affichant "Prix total :" suivi du prix total et doit renvoyer le prix total.

Note: Vous êtes libres de rajouter des fonctions intermédiaires si vous jugez cela pertinent.

Exemple 1 pour un appel de prix_client() qui renvoie 50:
Voulez-vous acheter une carte de réduction ? (oui/non) *oui*
Choisissez votre carte : (Jeune/Senior) *Jeune*
Voulez-vous acheter un billet ? (oui/non) *non*
Prix total : 50 euros

Exemple 2 pour un appel de prix_client() qui renvoie 75.0:
Voulez-vous acheter une carte de réduction ? (oui/non) *oui*
Choisissez votre carte : (Jeune/Senior) *Senior*
Voulez-vous acheter un billet ? (oui/non) *oui*
Depart : *Lyon*
Destination : *Grenoble*
Vous avez une carte de reduction
Precisez la periode (bleue/blanche) : *blanche*
Prix total : 75.0 euros

Exemple 3 pour un appel de prix_client() qui renvoie 80:
Voulez-vous acheter une carte de réduction ? (oui/non) *non*
Voulez-vous acheter un billet ? (oui/non) *oui*
Depart : *Paris*
Destination : *Lyon*
Autres precisions a fournir ? (oui/non) *non*
Prix total : 80 euros

Exemple 4 pour un appel de prix_client() qui renvoie 90.0:
Voulez-vous acheter une carte de réduction ? (oui/non) *non*
Voulez-vous acheter un billet ? (oui/non) *oui*
Depart : *Paris*

Destination : *Grenoble*
Autres precisions a fournir ? (oui/non) *oui*
Carte de reduction (Jeune, Senior, ou aucune)? *aucune*
Billet modifiable ? (oui/non) *non*
Prix total : 90.0 euros

Exemple 5 pour un appel de prix_client() qui renvoie 10.0:
Voulez-vous acheter une carte de réduction ? (oui/non) *non*
Voulez-vous acheter un billet ? (oui/non) *oui*
Depart : *Grenoble*
Destination : *Lyon*
Autres precisions a fournir ? (oui/non) *oui*
Carte de reduction (Jeune, Senior, ou aucune)? *Senior*
Precisez la periode (bleue/blanche): *bleue*
Prix total : 10.0 euros

Exercice 14

Lire une liste d'entiers

1. Ecrire une fonction LireListeEntiers sans argument qui permet de lire des entiers positifs ou nuls saisis au clavier. La saisie s'arrête lorsque l'on entre un nombre négatif. La fonction retourne la liste des entiers saisis (ne contenant pas de valeur négative).
2. Ecrire le programme principal qui permet d'appeler la fonction et d'afficher ensuite la liste obtenue avec la fonction print.
3. Proposer une fonction LireListeReelsBornes avec deux arguments facultatifs bmin et bmax, qui permet de lire des réels saisis au clavier. La saisie s'arrête lorsque l'utilisateur saisit une valeur en dehors de l'intervalle [bmin ; bmax]. Par défaut, les valeurs de min et max sont fixées à 0 et 100 respectivement. La fonction doit retourner une liste (ne contenant pas de valeur hors borne).
4. Définir une fonction MMSListe qui prend en argument une liste et qui retourne le minimum, le maximum et la somme de la liste. Votre fonction doit utiliser une boucle while ou for.
5. Complétez votre programme principal en utilisant LireListeReelsBornes pour saisir une liste de réels au clavier et MMSListe pour obtenir le minimum, le maximum et la somme des valeurs saisies. Afficher les 3 valeurs obtenues à la console.

Exercice 15

Générer des Listes

1) Ecrire une fonction **liste_decroissante** prenant en argument un entier n et qui renvoie la liste de tous les entiers de n à 0.

Exemple: liste_decroissante(5) vaut [5, 4, 3, 2, 1, 0]

2) Ecrire une fonction **multiples** qui prend en argument un entier m et un entier longueur et qui renvoie une liste contenant les multiples de m, en commençant par 0, et dont la longueur est déterminée par l'argument longueur.

Exemple: multiples(5, 6) vaut [0, 5, 10, 15, 20, 25]

3) Ecrire une fonction **pairs** qui prend un argument longueur qui renvoie une liste contenant les entiers pairs dans l'ordre croissant, en commençant par 0, et dont la longueur est donnée par l'argument longueur. Vous pouvez bien sûr vous aider des fonctions précédentes si vous jugez cela pertinent.

Exemple: pairs(10) vaut [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

Exercice 16

Construire une liste à partir d'une autre liste

Dans toutes les fonctions demandées dans cet exercice, la liste passée en argument ne doit pas être modifiée par l'appel. Pensez à vous aider de Python Tutor pour visualiser le comportement de vos fonctions.

- Ecrire une fonction decale qui prend en argument une liste l et un nombre d et qui renvoie la liste obtenue à partir de l en ajoutant un décalage de d à chaque élément de l.

  Exemple : decale([4, 17, 12], 3) vaut [7, 20, 15].

- Ecrire une fonction intercale_zeros qui prend en argument une liste l et qui renvoie la liste obtenue à partir de l en intercalant un zéro après chaque élément de l.

  Exemple : intercale_zeros([4, 17, 12]) vaut [4, 0, 17, 0, 12, 0].

- Ecrire une fonction supprime qui prend en argument une liste l et un élément elem et qui renvoie la liste obtenue à partir de l en supprimant toutes les occurrences de elem.

  Exemple : supprime([4, 7, 12, 4, 4, 0, 4, 5], 4) vaut [7, 12, 0, 5].

- Ecrire une fonction insere_milieu qui prend en argument une liste l et un élément elem et qui renvoie la liste obtenue à partir de l en ajoutant elem "au milieu" de l, c'est-à-dire:
  - Si la longueur de l est paire, l'élément est ajouté une fois, au milieu de la liste.
  - Si la longueur de l est impaire: l'élément est ajouté de part à d'autre de l'élément central.

Exemple : insere_milieu([4, 7, 12, 3], 0) vaut [4, 7, 0, 12, 3] et insere_milieu([9, 3, 5, 6, 2], 1) vaut [9, 3, 1, 5, 1, 6, 2].

- Ecrire une fonction decoupe qui prend en argument une liste de nombres l et un nombre seuil et qui renvoie deux listes: la première est obtenue à partir de l en ne gardant que les nombres inférieurs ou égaux à seuil; la deuxième est obtenue à partir de l en ne gardant que les nombres strictement supérieurs à seuil.

  Exemple : decoupe([14, 27, 12, 0, 40, 34, 20, 11], 20) a pour valeur de retour [14, 12, 0, 20, 11] et [27, 40, 34].

Exercice 17

*Jeux de mots

Remarques:

- vous avez bien sûr le droit de définir des fonctions intermédiaires qui ne sont pas demandées, lorsque vous jugez cela pertinent. Lisez l'énoncé en entier avant de commencer pour repérer les éventuelles opérations qui se répètent souvent.
- dans cet exercice, vous avez souvent besoin de convertir un mot en liste de lettres. Pour cela, il faut utiliser list(...) comme dans l'exemple suivant: list('oui') vaut ['o', 'u', 'i'].
- on suppose que tous les mots considérés dans cet exercice ne contiennent que des lettres, sans accents et en minuscule (pas de caractères spéciaux, tirets, espaces, etc...)

1) Ecrire une fonction commence_par prenant en argument une lettre et un mot et qui renvoie True si le mot commence par la lettre donnée en argument, False sinon.

Exemple: commence_par('h', 'hello') vaut True mais commence_par('e', 'roue') vaut False.

2) Ecrire une fonction contient_voyelle qui prend en argument un mot et qui renvoie True si le mot contient une voyelle, False sinon.

3) Ecrire une fonction derniere_consonne qui prend en argument un mot et qui renvoie deux valeurs de retour: l'indice de sa dernière consonne ainsi que la dernière consonne (comme pour les listes, on considérera que l'indice de la première lettre est zéro). On ne traitera pas le cas problématique où le mot ne contient pas de consonne.

Exemple: derniere_consonne('arrivee') renvoie 4, 'v'

4) Ecrire une fonction double_consonne qui prend en argument un mot et qui a deux valeurs de retour: un booléen valant True si le mot contient une double consonne (deux fois la même consonne à la suite), et dans ce cas la deuxième valeur de retour est la consonne qui est doublée ; s'il n'y a pas de consonne doublée, la fonction doit renvoyer False et None. *Pour information: pour simplifier l'exercice, on ne testera pas votre fonction sur un mot contenant plusieurs double consonnes (par exemple, 'successeur').*

Exemples:

- double_consonne('arrivee') vaut True, 'r'.
- double_consonne('bonbon') vaut False, None
- double_consonne('reussite') vaut True, 's'

5) Ecrire une fonction envers qui prend en argument une liste li (attention, pas un mot, contrairement aux autres fonctions de cet exercice) et qui renvoie une liste obtenue à partir de li en inversant l'ordre des éléments.

Exemple: envers(['a', 'b', 'c', 'd']) vaut ['d', 'c', 'b', 'a']

6) Ecrire une fonction palindrome qui prend en argument un mot et qui renvoie un booléen indiquant si le mot est un palindrome. Un palindrome est un mot qui reste identique lorsqu'il est lu de droite à gauche au lieu de l'ordre habituel de gauche à droite.

Exemple: palindrome('ici') vaut True mais palindrome('aller') vaut False.

7) Ecrire une fonction mot_autorise prenant en argument un mot et une liste de mots interdits, et qui renvoie True si le mot est autorisé, et False si le mot est interdit.

Exemple:

- mot_autorise('fric', ['sous', 'fric', 'thune', 'ble']) vaut False
- mot_autorise('argent', ['sous', 'fric', 'thune', 'ble']) vaut True

Exercice 18

*Codage et décodage

1. Ecrire une fonction NextElem qui prend en argument un élément elm et une liste d'au moins deux éléments liste. La fonction renvoie l'élément suivant elm dans la liste liste, ou le premier élément de la liste si elm est le dernier de liste. Elle renvoie None si elm n'est pas dans liste.
   Exemple:
   - NextElem('a',['a','b','c'])) vaut 'b'
   - NextElem('c',['a','b','c'] vaut 'a'
   - NextElem('d',['a','b','c']) vaut None
2. Ecrire une fonction encode qui prend en argument un texte (en minuscule, sans accent) et une liste ordre contenant toutes les lettres minuscules et la ponctuation dans un certain ordre (comme ordre_alpha et ordre2 qui vous sont fournis, par exemple) et qui renvoie le texte chiffré, où chaque caractère du texte de départ a été remplacé par son élément suivant dans la liste ordre.

Note: On remarquera qu'une chaîne de caractères se transforme en liste, grâce à la fonction list.

Exercice 19

L'obsession des nombres pairs

Note: vous êtes libres d'écrire les fonctions demandées dans l'ordre que vous souhaitez. Lisez l'énoncé en entier avant de commencer. Vous pouvez bien sûr, à l'intérieur d'une fonction, faire des appels aux fonctions que vous avez écrites plus haut.

De plus, dans cet exercice, tous les nombres considérés seront des entiers positifs ou nuls. Les listes passées en argument ne doivent pas être modifiées.

- Ecrire une fonction somme_pairs qui prend en argument une liste et qui renvoie la somme des nombres pairs contenus dans la liste.

  Exemple : somme_pairs([4, 7, 12, 0, 21, 5]) vaut 16 (car 16=4+12+0).
- Ecrire une fonction nb_elem_pairs qui prend en argument une liste et qui renvoie le nombre d'entiers pairs contenus dans la liste.

  Exemple : nb_elem_pairs([4, 7, 12, 0, 21, 5]) vaut 3 (car 4, 12 et 0 sont pairs).
- Ecrire une fonction max_pair qui prend en argument une liste et qui renvoie le plus grand entier pair contenu dans la liste. On supposera pour simplifier (dans cette question uniquement) que la liste contient toujours au moins un nombre pair.

Exemple : max_pairs([4, 7, 12, 0, 21, 5]) vaut 12.

- Ecrire une fonction min_pair qui prend en argument une liste et qui renvoie le plus petit entier pair contenu dans la liste. Si la liste ne contient aucun entier pair, la fonction renverra None.

  Exemple : min_pairs([4, 7, 12, 0, 21, 5]) vaut 0 et min_pairs([9, 3, 1]) vaut None.

- Ecrire une fonction indice_de qui prend en argument un entier (supposé pair) et une liste, et qui renvoie l'indice auquel apparaît cet entier dans la liste. Si l'entier n'apparaît pas, la fonction renverra None. On suppose ici pour simplifier que l'entier cherché n'apparaît pas plusieurs fois dans la liste. *Indice: si la boucle for e in liste ne vous convient pas, n'oubliez pas que vous pouvez faire un for + range ou un while...*

  Exemple : indice_de(12, [4, 7, 12, 0, 21, 5]) vaut 2 (car 12 est placé à l'indice 2), et indice_de(6, [4, 7, 12, 0, 21, 5]) vaut None.

- Ecrire une fonction trouve_premier_pair qui prend en argument une liste et qui renvoie l'entier pair qui apparaît en premier dans la liste. Si la liste ne contient pas d'entier pair, la fonction renverra None.

  Exemple : trouve_premier_pair([1, 15, 4, 7, 12, 3]) vaut 4 et trouve_premier_pair([1, 17, 7]) vaut None.

- Ecrire une fonction extrait_pairs qui prend en argument une liste l1 et qui renvoie la liste obtenue à partir de l1 en ne gardant que les entiers pairs (et sans changer leur ordre). Attention, l1 ne doit pas être modifiée par la fonction.

  Exemple : extrait_pairs([4, 7, 12, 0, 3]) vaut [4, 12, 0] et extrait_pairs([21, 17, 3]) vaut [ ].