



Bases de l'informatique 1 – SPUF100

Année 2024-2025 – Examen terminal

Nom :

Prénom :

Numéro d'étudiant :

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

Durée : 2 heures.

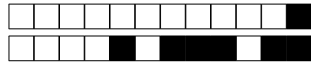
Aucun document n'est autorisé. L'usage de la calculatrice ou de tout autre appareil électronique est interdit.

Les exercices sont indépendants. Au sein d'un même exercice, vous pouvez utiliser les variables et fonctions des questions précédentes, même si vous n'avez pas su les faire; chaque question est donc indépendante.

À part les méthodes et fonctions de base, vous n'avez pas le droit d'utiliser les fonctions et les méthodes « avancées », sauf si l'énoncé vous conseille l'utilisation de certaines d'entre elles.

```
1 # Fonctions autorisées
2 range(...) len(...) print(...)
3
4 # Boucles autorisées
5 Boucles while
6 Boucles for avec un range
7 Boucles for sans range (for x in L)
8
9 # Vous pouvez utiliser les compréhensions ou les slices
10 [ x for x in range(L) ]
11 chaine[début:fin:pas]
12
13 # Les méthodes suivantes sont autorisés :
14 L.append(...)
```

```
1 # Par exemple les méthodes et fonctions suivantes sont entre autres interdites
2 max(...) min(...) sum(...) abs(...)
3 s.split(...) s.index(...) L.extend(...)
```



Exercice 1 Identifiants de code-barres 5,5 points

0 0,5 1 1,5 2 2,5 3 3,5 4 4,5 5 5,5

1. En utilisant les fonctions `chr` et `ord`, écrire la fonction `entier(chaine)` qui transforme une chaîne de longueur 1 représentant un chiffre en l'entier correspondant. Écrire de même la fonction `chiffre(n)` transformant un nombre entier entre 0 et 9 (compris) en chaîne.

```
1 >>> entier("3")
2 3
3 >>> chiffre(7)
4 '7'
```

.....

.....

.....

.....

.....

.....

.....

2. Écrire une fonction `tester_identifiant(ch)` qui teste si la chaîne donnée en paramètre est bien une chaîne de 7 chiffres. On ne pourra évidemment pas utiliser la méthode `ch.isdigit()`.

```
1 >>> tester_identifiant("1234567")
2 True
3 >>> tester_identifiant("12345678")
4 False
```

```
1 >>> tester_identifiant("4442222")
2 True
3 >>> tester_identifiant("444X222")
4 False
```

.....

.....

.....

.....

.....

.....

.....



3. Un identifiant pour code barre (pour la norme EAN-8) est composé de 7 chiffres quelconques et d'un 8^e chiffre (appelé la clé) calculé à partir des 7 premiers. Pour calculer la clé, on applique la méthode suivante : A) On ajoute les différents chiffres en les pondérant alternativement par 3 ou 1. B) Si le chiffre des unités du nombre obtenu est non nul, on soustrait ce chiffre à 10, sinon, la clé est zéro. Par exemple pour calculer la clé associée à l'identifiant "1564781". On commence par calculer la somme pondérée :
 $3 \times 1 + 1 \times 5 + 3 \times 6 + 1 \times 4 + 3 \times 7 + 1 \times 8 + 3 \times 1 = 62$ (le premier chiffre est multiplié par 3, le suivant par 1, celui d'après par 3 à nouveau, etc). Le chiffre des unités de 62 étant 2, la clé est donc $10 - 2 = 8$.

Écrire une fonction `ajouter_clé(ch)` qui à partir d'un identifiant de 7 chiffres renvoie une nouvelle chaîne de 8 chiffres obtenue en ajoutant la clé à la fin.

```
1 >>> ajouter_clé("1564781") # la clé calculée avec la méthode précédente est 8
2 '15647818'
```

.....

Exercice 2 Encodage 5 points

0 0,5 1 1,5 2 2,5 3 3,5 4 4,5 5

1. Écrire une fonction `chaînes_vers_liste(ch)` qui prend en argument une chaîne composée de 10 groupes de quatre chiffres séparés par des tirets et la transforme en une liste de 10 chaînes (chaque chaîne correspondant à un groupe de quatre chiffres). On ne pourra pas utiliser la méthode `ch.split("-")`, mais on pourra utiliser les `slices` (`chaîne[début:fin:pas]`).

```
1 >>> valeurs = "3211-2221-2122-1411-1132-1231-1114-1312-1213-3112"
2 >>> chaînes_vers_liste(valeurs)
3 ['3211', '2221', '2122', '1411', '1132', '1231', '1114', '1312', '1213', '3112']
```

Dorénavant, et pour la suite de l'exercice, on définit la variable `encodage` comme ci-dessous.

```
1 >>> encodage = chaînes_vers_liste(valeurs)
```




Exercice 3 Dessinons le code-barre 3 points

0 0,5 1 1,5 2 2,5 3

La représentation d'un code-barre au format binaire n'est pas très lisible. Nous allons écrire avec Tkinter un petit programme qui affiche le motif associé à un code-barre comme dans l'image ci-contre.

Concrètement chaque 1 correspondra à une colonne noire et chaque 0 à une colonne blanche. Une bande noire épaisse est simplement une succession de plusieurs colonnes noires.

On commence à se donner le code ci-dessous qui crée et initialise la fenêtre Tk.



```
1 import tkinter as tk
2 # Variables globales
3 Hauteur = 500
4 Largeur = 700
5 Marge = 100
6 # On crée une fenêtre et un canevas (zone de dessin)
7 root = tk.Tk()
8 root.title("Générateur de code barre")
9 Dessin=tk.Canvas(root,height=Hauteur,width=Largeur,bg="white")
10 Dessin.pack()
```

1. On cherche à calculer la largeur d'une colonne à partir de la largeur, des marges de chaque côté et du nombre n de colonnes donné en paramètre. Écrire la fonction `calculer_delta(n)` renvoyant la largeur d'une telle colonne.

.....
.....
.....

2. Écrire alors une fonction `tracer_colonne(n, d)` qui trace en noir et en partant de la marge de gauche la n^{ème} colonne de largeur delta.

On utilisera la fonction vue en cours, `Dessin.create_rectangle(x0,y0,x1,y1,fill="black")` où `(x0,y0)` et `(x1,y1)` sont les coordonnées de deux sommets diagonaux du rectangle.

.....
.....
.....
.....
.....
.....



Exercice 5 Base de données et tickets de caisse 3,5 points

0 0,5 1 1,5 2 2,5 3 3,5

On se donne un fichier texte intitulé `bdd.txt` qui contient l'ensemble des références d'un magasin. Chaque ligne contient trois éléments séparés par des virgules : le code-barre, la description et le prix unitaire.

```
1 1234561, TABLETTE CHOCOLAT, 10.23
2 1785436, SAUCISSON, 5.26
3 7895647, DÉODORANT, 3.25
4 7535417, SLIP DE LUXE, 57.75
```

1. En déduire une fonction `lire_base_de_données(nom_fichier)` qui à partir du contenu du fichier donné en paramètre renvoie un dictionnaire dont les clés sont les codes-barres et les valeurs, des couples de la forme `(description, prix)`. On pourra utiliser la fonction `float("12.5")` pour convertir des chaînes en flottant ainsi que la fonction `découper` de l'exercice précédent.

```
1 >>> BDD = lire_base_de_données("bdd.txt")
2 >>> BDD['1785436']
3 ('SAUCISSON', 5.26)
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

2. Écrire une fonction `ticket_de_caisse(liste_codes, BDD, nom_fichier)` qui à partir d'une liste de codes-barres construit le ticket de caisse correspondant. Le résultat ne sera pas affiché mais stocké dans un fichier texte dont le nom est fourni par la chaîne `nom_fichier`.

On fera attention de gérer proprement les alignements et on calculera deux totaux : le total des prix des produits alimentaires (ceux dont le code-barre commence par 1) et le total complet de tous les prix.

```
1 >>> ticket_de_caisse(["1785436", "7535417", "1785436", "1234561"], BDD, "ticket.txt")
```

```
1 SAUCISSON      5.26€
2 SLIP DE LUXE  57.75€
3 SAUCISSON      5.26€
4 TABLETTE CHO 10.00€
5 -----
6 TOTAL         : 78.27€
7 DONT ALIM.    : 20.52€
```



A series of 20 horizontal dotted lines for writing.

