Devoir surveillé d'informatique

Consignes

- La calculatrice n'est **pas autorisée**.
- On pourra toujours librement utiliser une fonction demandée à une question précédente même si cette question n'a pas été traitée.
- Veillez à présenter vos idées et vos réponses partielles même si vous ne trouvez pas la solution complète à une question.
- La clarté et la lisibilité de la rédaction et des programmes sont des éléments de notation.

❑ Exercice 1 : *Questions de cours*

1. Recopier et compléter le tableau suivant en donnant le type et la valeur de l'expression. Les lignes sur fond gris sont des exemples déjà complétées afin de vous aider.

Expression	Type	Valeur
`5 == 3`	`bool`	`False`
`3*8 + 1`	`int`	25
`2**5`	`int`	32
`72%9 == 0`	`bool`	True
`"ah"*3`	`str`	"ahahah"
`10/4`	`float`	2.5
`True or False`	`bool`	True
`len("math")!=3`	`bool`	True
`7//2 == 3.5`	`bool`	False
`"20"+"24"`	`str`	"2024"
`(2+7, 17%3)`	`tuple`	(9,2)
`"ab" >= "ac"`	`bool`	False

2. On suppose définie une variable `s` de type `str` contenant `"Extraordinaire"`. On a numéroté ci-dessous à partir de 0 les caractères de cette chaine :

E	x	t	r	a	o	r	d	i	n	a	i	r	e
0	1	2	3	4	5	6	7	8	9	10	11	12	13

a) Quel est le contenu des expression suivantes ?

- `s[7]`
- `s[len(s)-1]`
- `s[0:3]`

- `s[7] = "d"`
- `s[len(s)-1]="e"`
- `s[0:3]="Ext"`

b) Ecrire sous la forme d'une tranche de `s` une expression contenant `"ordi"`.

```
s[5:9]
```

c) Quel est l'effet de l'instruction `s[0]="e"` ? Expliquer

Cette instruction provoque une erreur (`TypeError: 'str' object does not support item assignment`] car les chaines de caractères ne sont pas mutables, on ne peut pas modifier les caractères qui les composent on doit construire une nouvelle chaine.

3. On suppose définie une variable `l` de type `list` contenant `[2, 3, 5, 7, 11, 13, 17]`
   a) Donner la valeur de `n` ainsi que le contenu de `l` après exécution de l'instruction `n = l.pop()`

   > `n = 17` et `l = [2, 3, 5, 7, 11, 13]` car `pop` supprime le dernier élément d'une liste et renvoie cet élément.

   b) Ecrire l'instruction permettant d'ajouter la valeur `19` à la fin de cette liste.

   > `l.append(19)`

   c) Quel est l'effet de l'instruction `l[0] = l[0] + l[3]` ?

   > Le premier élément de la liste `l` est modifié, il devient la somme de son ancienne valeur et de `l[3]` c'est à dire `2+7 = 9`.

4. Ecrire un programme (qui peut se limiter à une seule instruction) permettant de créer les listes suivantes :
   - `lst1` qui contient 14 fois l'entier 42.
   - `lst2` qui contient les entiers de 1 à 100.
   - `lst3` qui contient les 20 premières puissances positives de 2 (c'est à dire $2^0, 2^1, \dots, 2^{19}$)

   > - `lst1` qui contient 14 fois l'entier 42.
   >   `lst1 = [14]*42`
   > - `lst2` qui contient les entiers de 1 à 100.
   >   `lst2 = [i for i in range(1,101)]`
   > - `lst3` qui contient les 20 premières puissances positives de 2 (c'est à dire $2^0, 2^1, \dots, 2^{19}$)
   >   `lst3 = [2**i for i in range(0,20)]`

❑ **Exercice 2** : *Fonction* `mystere`
On considère la fonction `mystere` suivante :

```
1  def mystere(n:int) -> list[int]:
2      assert n>=0, "L'entier n doit être positif"
3      if n==0:
4          return [0]
5      res = []
6      c = 0
7      while n>0:
8          c = n%10
9          res.append(c)
10         n = n//10
11     return res
```

1. Donner le type attendu pour le paramètre `n` et le type de la valeur renvoyée par cette fonction.

   > Cette fonction prend en argument un entier n et renvoie une liste d'entiers.

2. Donner le résultat renvoyé par `mystere` lors des appels suivants :
   — `mystere(-10)`
   — `mystere(0)`
   — `mystere(7)`

— `mystere(-10)` : provoque une erreur `AssertionError` et affiche `"L'entier n doit être positif"`

— `mystere(0)` : renvoie `[0]`

— `mystere(7)` : renvoie `[7]`

3. On effectue à présent l'appel `mystere(2024)`, recopier et compléter le tableau suivant qui indique le le contenu des variables, `n, c` et `res` durant l'exécution.

	`n`	`c`	`res`
valeurs initiales	2024	0	`[]`
après un tour de la boucle `while`	202	4	`[4]`
après deux tour de la boucle `while`	20	2	`[4, 2]`
après trois tour de la boucle `while`	2	0	`[4, 2, 0]`
après quatre tour de la boucle `while`	0	2	`[4, 2, 0, 2]`

4. Proposer une spécification pour la fonction `mystere`, en specifiant le type des arguments et du résultat et les éventuelles préconditions.

La fonction mystère prend en entrée un entier `n` positif et renvoie la liste des chiffres de ce nombre dans l'ordre inverse. Par exemple `mystere(173)` renvoie `[3, 7, 1]`.

❑ Exercice 3 : *Calculs de moyennes*

1. Moyenne simple

a) Ecrire une fonction `somme` qui prend en argument une liste de nombres et renvoie leur somme. Par exemple `somme([12, 7, 11, 18])` renvoie 48.

```
def somme(lst):
    s = 0
    for x in lst:
        s = s + x
    return s
```

b) Ecrire une fonction `moyenne` qui prend en argument une liste de nombres et renvoie leur moyenne (on pourra utiliser la fonction `somme` de la question précédente.)

```
def moyenne(lst):
    return somme(lst)/len(lst)
```

c) Quel sera le résultat de l'appel à `moyenne` sur une liste vide ? Quelle instruction permettrait de vérifier en amont que la liste n'est pas vide et de déclencher une erreur si ce n'est pas le cas ?

L'appel renverra une erreur (division par zéro), on peut ajouter une instruction `assert len(lst)!=0`.

2. Moyenne olympique

La *moyenne olympique* est utilisée pour noter les athlètes lors de certaines compétitions sportives. Pour la calculer, on enlève d'abord de la liste de notes *une* occurrence du maximum et *une* occurrence du minimum. Par exemple si les notes sont `[12; 7; 6; 15; 9; 6]` alors on fera la moyenne en supprimant une occurence du maximum (`15`) et une du minimum (`6`) et donc on calculera la moyenne de `[12; 7; 9; 6]`. On supposera dans toute la suite qu'on dispose d'une listes de notes contenant au moins 3 notes et on veut écrire une fonction renvoyant la moyenne olympique de ces notes.

a) Ecrire une fonction `maximum` qui renvoie le maximum des éléments d'une liste supposée non vide.

```python
def maximum(lst):
    maxi = lst[0]
    for x in lst:
        if x>maxi:
            maxi=x
    return maxi
```

b) Ecrire une fonction `minimum` qui renvoie le minimum des éléments d'une liste supposée non vide.

```python
def minimum(lst):
    mini = lst[0]
    for x in lst:
        if x<mini:
            mini=x
    return mini
```

c) En déduire une fonction `moyenne_olympique` qui renvoie la moyenne olympique de la liste de notes données en argument (on suppose que la liste contient au moins 3 notes).

```python
def moyenne_olympique(lst):
    somme = 0
    for x in lst:
        somme =  somme + x
    somme = somme - minimum(lst) - maximum(lst)
    return somme/(len(lst)-2)
```

3. Moyenne pondérée
Ecrire une fonction `moyenne_ponderee` qui prend en argument une liste de tuples de la forme `(note, coefficient)` et renvoie la moyenne pondérée des notes affectés des coefficient correspondants. Par exemple `moyenne_ponderee([(12,3),(17,1),(11,2)])` renvoie `12.5` en effet : $(12 \times 3 + 17 \times 1 + 11 \times 2)/6 = 12,5$. On supposera que la liste est non vide et que les coefficients sont strictement positifs.

```python
def moyenne_ponderee(lst):
    somme = 0
    somme_coeff = 0
    for x in lst:
        # x est un tuple de la forme (note, coeff) qu'on décompacte :
        note, coeff = x
        somme_coeff = somme_coeff + coeff
        somme = somme + note*coeff
    return somme/somme_coeff
```

❑ Exercice 4 : *Chiffrement de César*

En cryptographie, le chiffrement par décalage, aussi connu comme le chiffre de César ou le code de César (...), est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes (ce qui explique le nom « chiffre de César »).

(Wikipedia)

Pour coder un texte avec la code de César, on se donne une clé de codage c (un entier entre 1 et 25) puis on décale toutes les lettres de c emplacement dans l'alphabet *en recommençant au début lorsqu'on dépasse le* Z. Par exemple, si $c = 7$, voici la correspondance entre les lettres et leur chiffrement :

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
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

On remarquera qu'on a réecrit l'alphabet à partir du H et en revenant au début une fois le Z atteint.
Donc si on décide de chiffrer "PCSI" avec une clé de 7, on obtient "WJZP".
Le but de l'exercice est d'écrire une fonction `cesar` qui prend en entrée une chaine de caractères et une clé et renvoie la chaine chiffrée avec cette clé. Si les caractères de la chaine ne sont pas des lettres majuscules on les laisse intactes. Par exemple `chiffre("MP2I",1)` renvoie `"NQ2J"` (le 2 est inchangé).

1. Ecrire une fonction `numero` qui prend en entrée un caractère, si ce caractère est une lettre majuscule on renvoie son numéro dans l'alphabet (en commençant la numérotation à zéro) sinon on renvoie `-1`. Par exemple `numero('A')` renvoie `0`, `numero('B')` renvoie `1`, ... et `numero('2')` renvoie `-1`.

```
def numero(car):
    lettres = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    for i in range(len(lettres)):
        if car==lettres[i]:
            return i
    return -1
```

2. Ecrire une fonction `decalage` qui prend en entrée un entier `cle` et renvoie une chaine contenant l'alphabet décalé de `cle` emplacements. Par exemple `decalage(7)` renvoie `"HIJKLMNOPQRSTUVWXYZABCDEFG"`
Indication : on pourra penser à utiliser les tranches de la chaine contenant les lettres de l'alphabet.

```
def decalage(cle):
    lettres = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    return lettres[cle:]+lettres[:cle]
```

3. En utilisant les deux fonction précédentes, écrire une fonction `cesar` qui prend en entrée une chaine de caractère `chaine` et un entier `c` et renvoie cette chaine chiffré avec la clé `c`.

```
def cesar(chaine, cle):
    chiffre = ""
    dec = decalage(cle)
    for c in chaine:
        if numero(c)==-1:
            chiffre +=c
        else:
            chiffre += dec[numero(c)]
    return chiffre
```