Contrôle 1 – NSI3 (104) – 25/09/20219

Première partie : sur ordinateur. **Durée : du début jusqu’à 10h10**. Les programmes devront être enregistrés sur le bureau, ou bien dans un dossier qui restera ouvert afin que je puisse les récupérer facilement sur clé, sans vous déranger, pendant la deuxième partie du contrôle (QCM de 12 minutes).

Pour cette partie du contrôle uniquement, vous avez droit au cours, aux exercices que vous avez déjà fait, et uniquement à la documentation Python en ligne <https://docs.python.org/3/index.html> (ceci dit elle ne devrait vous servir à rien sur ce contrôle). **Rien d’autre**, notamment sur le web. **Vérifiez que tout programme qui n’est pas en rapport avec ce qui précède est fermé** (attention aux ouvertures automatiques en début de session). Le non-respect de ces consignes est considéré comme une fraude, et promet des conséquences très néfastes sur la note finale (au minimum -5 points, en général 0)

*Les fonctions seront spécifiées.*

Exercice 1: de l’efficacité du sabre laser pour piller des coffres-forts (10 points)

Obiwan Kenobi a décidé de changer de boulot et d’aller braquer le coffre-fort de la Banque de France Naboo à l’aide de son sabre laser, qui découpe plutôt bien le métal. Le souci est que la porte est épaisse de 70 cm, et que plus le métal fond, plus il gêne pour aller plus profond.

En 1 seconde, le sabre perce le blindage sur une épaisseur *e* et fait fondre un volume *v* de métal. Les deux sont liés par la relation suivante :  et .

*Le problème est découpé ci-dessous en questions qui facilitent son étude. Vous pouvez aller plus vite et répondre directement à la question 4.*

1. Écrire un programme qui calcule le temps nécessaire pour percer la porte. L’enregistrer sous Nom\_Prénom\_ex1.py. Mettre en 1ère ligne de commentaire votre email, si vous souhaitez une correction personnalisée de vos erreurs.
2. Modifier le programme précédent pour qu’il précise en plus calcule le temps nécessaire pour percer la moitié de la porte.
3. Structurer le programme précédent en fonctions. Les calculs  et devront figurer une seule fois dans le programme. On rappelle que l’on peut faire un retour multiple sur les fonctions : truc, bidule = fonction(machin , chose, zaffair)
4. Modifier le programme précédent de manière à que seules les entrées-sorties figurent dans le programme principal. L’utilisateur rentre l’épaisseur de la porte, le programme affiche les temps des questions 1 et 2.
5. Approfondissement (cette question est notée sur 0,01 point). Modifier le programme précédent pour qu’il cherche si une porte peut être inviolable à partir d’une certaine épaisseur. Précisez ce que vous choisirez comme condition pour « inviolable ».

Exercice 2 (6 points).

On donne une chaîne de caractères, dans laquelle figurent également des chiffres. Écrire une fonction somme\_1\_2(*chaîne*) qui renvoie la somme des 1 et 2 figurant dans la chaîne. On rappelle qu’une chaîne est un type particulier de liste, que len(*chaîne*) donne la longueur de la chaîne et que chaîne[*i*] permet d’accéder à l’élément d’indice *i* de la chaîne. Si vous le souhaitez, plutôt que d’écrire une chaîne vous pouvez utiliser une liste (inconvénient : c’est plus long à taper). Enregistrer le programme sous Nom\_Prénom\_ex2.py

*Exemples :*

* somme\_1\_2("j’ai mangé 12 œufs") renvoie 3
* somme\_1\_2("j’ai mangé 3 œufs") renvoie 0
* somme\_1\_2("j’ai mangé 12 œufs, 1 bœuf, 5 gâteaux et un chapeau ") renvoie 4

Deuxième partie : 4 points, 12 minutes.

Cet exercice est un QCM où chaque question comporte 4 ou 5 réponses. Une seule est bonne. Toute bonne réponse rapporte 3 points, toute mauvaise réponse enlève 1 point, toute absence de réponse rapporte 0 point.

Répondre dans le cadre ci-dessous en mettant la lettre correspondant à la bonne réponse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NOM Prénom | Question 1 | Question 2 | Question 3 | Question4 |
|  |  |  |  |  |

1. A désignant un entier, lequel des codes suivants ne se termine pas ?

|  |  |  |  |
| --- | --- | --- | --- |
| Code A | Code B | Code C | Code D |
| i = A + 1  while i < A :  i = i - 1 | i = A +1  while i < A :  i = i + 1 | i = A - 1  while i < A :  i = i - 1 | i = A - 1  while i < A :  i = i + 1 |

1. On donne le programme suivant

def calcul(var):

var =var\*2

return(var)

var = 0

for i in range(5):

var = var + i

print(calcul(var))

Après exécution, la variable var contient :

1. 10 B. 15 C. 20 D. 30 E. autre réponse
2. Pour obtenir un nombre aléatoire *n* tel que , après avoir importé la bibliothèque random, il faut écrire :

A. random.randint(12) B. random.randint(0,13)

C. random.randint(0,12) D. random.randint(13)

E. Autre réponse

1. Que donne l’exécution du programme suivant ?

a = 5

b = 4

if a == 5 :

b = b + 1

else :

a = a - 1

if b == 5 :

a = a - 1

else :

b = b + 1

A. *a* = 5 et *b* = 6 B. *a* = 3 et *b* = 4 C. *a* = 3 et *b* = 6 D. *a* = 4 et *b* = 5