Fonctions (2)

Exercices

Exercice 0 *QCM – vérification des prérequis* Pour chaque question, une seule réponse est correcte.

1. Quelle est la différence entre `print` et `return` dans une fonction ?

— A. `print` renvoie une valeur, `return` l'affiche

— B. `print` affiche une valeur à l'écran, `return` renvoie une valeur à l'appelant

— C. `print` et `return` font exactement la même chose

— D. `return` n'est utilisable que dans le programme principal

2. Que vaut `x` après l'exécution du code suivant ?

```
def double(n):
    return n * 2

x = double(5) + 1
```

— A. `10`

— B. `11`

— C. `None`

— D. Une erreur

3. Quelle annotation de type correspond à une fonction qui prend un entier et renvoie un booléen ?

— A. `def f(n: bool) -> int:`

— B. `def f(n: int) -> bool:`

— C. `def f(n: int) -> str:`

— D. `def f(n) -> int:`

4. Que se passe-t-il si une fonction ne contient pas de `return` ?

— A. Python lève une erreur `SyntaxError`

— B. La fonction renvoie `0`

— C. La fonction renvoie `None`

— D. La fonction ne peut pas être appelée

Exercice 1 *Paquets et poids*

1. Écrire une fonction `poids_total(n, poids)` qui prend en paramètres deux entiers : un nombre de paquets et le poids d'un paquet (tous les paquets ont le même poids). Si le poids total est strictement inférieur à 105 kg, la fonction doit renvoyer `True`, sinon elle doit renvoyer `False`.

2. Écrire le programme principal qui doit reproduire cette situation : on veut faire un premier envoi de 10 paquets de 13 kg chacun, puis un second envoi de 12 paquets de 5 kg chacun. Pour chaque envoi, le programme doit appeler la fonction et afficher `"envoi Ok"` ou `"envoi impossible"`.

Exercice 2 ***Température*** Écrire une fonction `stat_temp()` qui recueille au clavier les températures de 10 jours successifs, puis affiche les températures minimale, maximale et moyenne.

Exercice 3 ***Produit des entiers successifs*** Écrire une fonction `produit(n)` qui prend un nombre `n` en paramètre et qui renvoie le produit des entiers successifs de `1` jusqu'à `n`.

Exemples : `produit(3)` doit renvoyer `6` (résultat de $1 \times 2 \times 3$) ; `produit(5)` doit renvoyer `120` (résultat de $1 \times 2 \times 3 \times 4 \times 5$).

Exercice 4 ***Moyenne*** Écrire une fonction qui calcule la moyenne d'un certain nombre de notes entrées par l'utilisateur. Cette fonction prendra comme paramètre un nombre `n`, qui correspond au nombre de notes qui seront demandées à l'utilisateur. En sortie, elle renverra la moyenne des différentes notes.

Pour tester : `moyenne(5)` en entrant `4`, `5`, `6`, `7` et `8` devra renvoyer `6` ; `moyenne(3)` en entrant `4`, `5` et `6` devra renvoyer `5`.

Exercice 5 ***Années bissextiles*** Les années bissextiles sont les années divisibles par 4, à l'exception de celles qui sont divisibles par 100 sans l'être par 400.

Par exemple : 2000 est une année bissextile (divisible par 400) ; 2012 est bissextile (divisible par 4 mais pas par 100) ; 1900 n'est pas bissextile (divisible par 100 mais pas par 400).

Définissez une fonction dont le paramètre est un entier naturel non nul (l'année) puis qui renvoie `True` lorsque l'année est bissextile et `False` dans le cas contraire.

Exercice 6 ***Suite de Fibonacci*** On appelle nombres de Fibonacci les nombres obtenus par somme des deux précédents. Les deux premiers sont $F_0 = 1$ et $F_1 = 1$. On a ensuite $F_2 = 2$, $F_3 = 3$, $F_4 = 5$, $F_5 = 8$, $F_6 = 13$, etc.

Compléter le corps de la fonction `fibo(n)` qui calcule le n-ième terme de la suite de Fibonacci.

Exercice 7 ***Nombres de Armstrong*** Un nombre de Armstrong est un entier positif égal à la somme des cubes de ses chiffres. Exemple : $153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$.

1. Écrire une fonction `est_armstrong` qui prend un nombre `n` en entrée et renvoie `True` si ce nombre est un nombre de Armstrong et `False` sinon.
2. Écrire une fonction `armstrong` qui affiche les nombres de Armstrong inférieurs à un nombre `n` passé en paramètre.

Exercice 8 ***ADN*** Une molécule d'ADN est formée de nucléotides notés `A`, `C`, `T` ou `G`.

Écrire une fonction `presence_de_A(sequence)` qui teste la présence du nucléotide `A` dans une séquence d'ADN.

Tests : `presence_de_A("CTTGCT")` doit renvoyer `False` ; `presence_de_A("CTTACT")` renvoie `True`.

Exercice 9 ***ADN 2 (difficile)*** Écrire une fonction `position_de_AT(sequence)` qui teste si une séquence contient le nucléotide `A` suivi du nucléotide `T` et renvoie l'indice de la première occurrence trouvée.

Exemples : `position_de_AT("CTTATGCT")` renvoie `3` ; `position_de_AT("GATATAT")` renvoie `1` ; `position_de_AT("GACCGTA")` renvoie `None`.

Exercice 10 ***ADN 3 (encore plus difficile)*** Écrire une fonction `position(code, sequence)` qui teste si une séquence contient un code donné et renvoie l'indice de la première occurrence. Si la séquence n'est pas trouvée, la fonction doit renvoyer `-1`.

Exemples : `position("CCG", "CTCCGTT")` renvoie `2` ; `position("GT", "TACGTTA")` renvoie `3` ; `position("GAT", "TACGTTA")` renvoie `-1`.