7 – Fonctions - Exercices fonctions (correction)

ACTIVITÉ 1

Exercice 1

Définir une fonction `test_Pythagore` qui prend trois entiers a, b et c en arguments et renvoie un booléen indiquant si $a^2 + b^2 = c^2$, ou $b^2 + c^2 = a^2$ ou $c^2 + a^2 = b^2$.

Tests et exemples d'usage :

```
>>> print(test_pythagore(3, 4, 5))
True
>>> print(test_pythagore(7, 2, 12))
False
>>> print(test_pythagore(3, 5, 4))
True
>>> print(test_pythagore(5, 4, 3))
True
```

[1]:
```
def test_pythagore(a, b, c):
    est_rectangle = a*a + b*b == c*c \
                    or b*b + c*c == a*a \
                    or c*c + a*a == b*b
    return est_rectangle


print(test_pythagore(3, 4, 5))
print(test_pythagore(7, 2, 12))
print(test_pythagore(3, 5, 4))
print(test_pythagore(5, 4, 3))
```

```
True
False
True
True
```

ACTIVITÉ 2

Exercice 2

Définir une fonction `valeur_absolue` qui prend un entier en argument et renvoie sa valeur absolue.

Exemples et tests :

```
>>> print(valeur_absolue(5))
5
>>> print(valeur_absolue(-5))
5
>>> print(valeur_absolue(0))
0
>>> print(valeur_absolue(-42))
42
```

[2]:
```
def valeur_absolue(n):
    if n >= 0:
        return n

    return -n


print(valeur_absolue(5))
print(valeur_absolue(-5))
print(valeur_absolue(0))
print(valeur_absolue(-42))
```

```
5
5
0
42
```

ACTIVITÉ 3

Exercice 3

Créer une fonction `multiples` pour qu'elle prenne la limite en argument plutôt que d'utiliser la valeur 999. En déduire une fonction `multiples_3_ou_5(borne_sup)` qui renvoie la somme des multiples de 3 ou 5 inférieurs ou égaux à `borne_sup`.

Exemples et tests :

```
>>> print(multiples_3_ou_5(2))
0
>>> print(multiples_3_ou_5(6))
14
>>> print(multiples_3_ou_5(10))
33
>>> print(multiples_3_ou_5(100))
2418
```

[4]:
```
def multiples_3_ou_5(borne_sup):
    somme = 0
    for i in range(borne_sup+1):
        if i % 3 == 0 or i % 5 == 0:
            somme += i
    return somme


print(multiples_3_ou_5(2))
print(multiples_3_ou_5(6))
print(multiples_3_ou_5(10))
print(multiples_3_ou_5(100))
```

```
0
14
33
2418
```

ACTIVITÉ 4

Exercice 4

Écrire une fonction `max2(a, b)` qui renvoie le plus grand des deux entiers `a` et `b`.

Exemples et tests :

```
>>> print(max(0, 0))
0
>>> print(max(1, 2))
2
>>> print(max(-2, -1))
-1
```

[5]:
```
def max2(a, b):
    if a >= b:
        return a

    return b


print(max(0,0))
print(max(1,2))
print(max(-2,-1))
```

```
0
2
-1
```

ACTIVITÉ 5

Exercice 5

Écrire une fonction `max3(a, b, c)` qui utilise la fonction `max2` de l'exercice précédent et qui renvoie le plus grand des trois entiers `a`, `b`, `c`.

La fonction `max2` est disponible et vous pouvez l'utiliser sans l'implémenter.

Exemples et tests :

```
>>> max3(0, 0, 0)
0
>>> print(max3(0,5,2))
5
>>> print(max3(-10,5,21))
21
```

[6]:
```
def max3(a, b, c):
    return max2(a, max2(b, c))


print(max3(0, 0, 0))
print(max3(0,5,2))
print(max3(-10,5,21))
```

```
0
5
21
```

ACTIVITÉ 6

Exercice 6

Écrire une fonction `puissance(x, k)` qui renvoie `x` à la puissance `k` (utilisation de l'opérateur `**` interdit! évidemment...). On utilisera une boucle `for` pour faire le calcul.

On suppose que $k \geq 0$ et on rappelle que $x^0 = 1$.

Exemples et tests :

```
>>> print(puissance(5, 2))
25
>>> print(puissance(100, 0))
1
>>> print(puissance(2, 10))
1024
```

[9]:
```
def puissance(x, k):
    resultat = 1
    for i in range(k):
        resultat = resultat * x

    return resultat


print(puissance(5, 2))
print(puissance(100, 0))
print(puissance(2, 10))
```

```
25
1
1024
```

ACTIVITÉ 7

Exercice 7

Écrire une fonction `est_bissextile(annee)` qui renvoie un booléen indiquant si l'année `annee` est une année bissextile.

On rappelle qu'une année bissextile est une année multiple de 4 mais pas de 100, ou multiple de 400.

Exemples et tests :

```
>>> print(est_bissextile(1996))
True
```

```
>>> print(est_bissextile(2024))
True
>>> print(est_bissextile(2022))
False
```

[10]:
```
def est_bissextile(annee):
    est_div_4   = annee % 4   == 0
    est_div_100 = annee % 100 == 0
    est_div_400 = annee % 400 == 0
    if (est_div_4 and not est_div_100) \
        or est_div_400:
        return True

    return False


print(est_bissextile(1996))
print(est_bissextile(2024))
print(est_bissextile(2022))
```

```
True
True
False
```

ACTIVITÉ 8

Exercice 8

Écrire une fonctions `nb_jours_annee(annee)` qui renvoie le nombre de jour de l'année `annee`. La fonction `est_bissextile` de l'exercice précédent est disponible et vous pouvez l'utiliser (ce qui est bien pratique quand même).

Exemples et tests :

```
>>> print(nb_jours_annee(1996))
366
>>> print(nb_jours_annee(2024))
```

```
366
>>> print(nb_jours_annee(2022))
365
```

[11]:
```
def nb_jours_annee(annee):
    if est_bissextile(annee):
        return 366
    return 365


print(nb_jours_annee(1996))
print(nb_jours_annee(2024))
print(nb_jours_annee(2022))
```

```
366
366
365
```

ACTIVITÉ 9

Exercice 9

Écrire une fonction `nb_jours_mois(annee, mois)` qui renvoie le nombre de jours dans le mois `mois` de l'année `annee`. La fonction `est_bissextile` de l'exercice précédent est disponible (ce qui est bien pratique).

On suppose que le mois `mois` est un entier compris entre `1` (pour janvier) et `12` (décembre).

Exemples et tests :

```
>>> print(nb_jours_mois(1900, 1))
31
>>> print(nb_jours_mois(1900, 2))
28
```

```
>>> print(nb_jours_mois(1904, 2))
29
>>> print(nb_jours_mois(2021, 9))
30
```

[13]:
```
def nb_jours_mois(annee, mois):
    if mois == 2:
        if est_bissextile(annee):
            return 29
        else:
            return 28

    if mois == 4 or mois == 6 or mois == 9 or mois == 11:
        return 30

    return 31


print(nb_jours_mois(1900, 1))
print(nb_jours_mois(1900, 2))
print(nb_jours_mois(1904, 2))
print(nb_jours_mois(2021, 9))
```

```
31
28
29
30
```

Exercice 10

En utilisant les fonctions `nb_jours_annnee` et `nb_jours_mois` (qui sont disponibles), écrire une fonction `nb_jours(j_debut, m_debut, a_debut, j_fin, m_fin, a_fin)` qui renvoie le nombre de jours compris entre deux dates données (incluses).

Exemples et tests :

```
>>> print(nb_jours(1,1,2021,15,1,2021))
15
>>> print(nb_jours(1,1,2021,31,12,2021))
365
>>> print(nb_jours(1,1,2024,31,12,2024))
366
>>> print(nb_jours(1,1,1970,31,12,2021))
18993
```

[18]:
```python
def nb_jours(j_debut, m_debut, a_debut, j_fin, m_fin, a_fin):
    # nombre de jours avant la date de
    # départ pour compléter la première année
    nbjours_avant = 0
    nbjours_avant = nbjours_avant + j_debut - 1
    for m in range (1, m_debut):
        nbjours_avant = nbjours_avant + nb_jours_mois(a_debut, m)

    # nombre de jours après la date de
    # fin nécessaires pour compléter la dernière année
    nbjours_apres = 0
    nbjours_apres = nbjours_apres + nb_jours_mois(a_fin, m_fin) - j_fin
    for m in range(m_fin + 1, 13):
        nbjours_apres = nbjours_apres + nb_jours_mois(a_fin, m)

    # nombres de jours entre l'année de début et l'année de fin
    total = 0
    for a in range(a_debut, a_fin+1):
        total = total + nb_jours_annee(a)

    # retirer les jours en trop des
    # premières et dernières années
    return total - nbjours_avant - nbjours_apres
```

[16]:
```python
print(nb_jours(1,1,2021,15,1,2021))
print(nb_jours(1,1,2021,31,12,2021))
print(nb_jours(1,1,2024,31,12,2024))
print(nb_jours(1,1,1970,31,12,2021))
```

```
15
365
366
18993
```