Exercices fonction Correction

Christophe Viroulaud

Première - NSI

Lang 07

Exercice 1

```python
def est_pair(x: int) -> bool:
    """
    vérifie si x est pair
    Paramètres:
        x (int): entier

    Renvoie:
        bool: True si x est pair
    """
    if x % 2 == 0:
        return True
    else:
        return False
```

Code 1 – Création de la fonction

```python
print( est_pair(5) )
```

Code 2 – Programme principal

```python
def est_pair(x: int) -> bool:
    """
    vérifie si x est pair
    Paramètres:
        x (int): entier

    Renvoie:
        bool: True si x est pair
    """
    return x % 2 == 0
```

Code 3 – Autre version

À retenir

L'expression `x%2 == 0` est un booléen (`True` ou `False`) qui peut être renvoyé directement.

Exercice 2

```python
# Fonction
def valeur_absolue(x: int) -> int:
    """
    renvoie la valeur absolue de x
    """
    if x < 0:
        return -x
    else:
        return x

# Programme principal
print( valeur_absolue(-4) )
```

Remarque

La `docstring` est moins détaillée mais toujours présente.

Exercice 3

```python
def surface(r: int) -> float:
    """
    renvoie la surface du disque de rayon r
    """
    return 3.14*r**2
```

```python
# import
from math import pi

# fonction
def surface(r: int) -> float:
    """
    renvoie la surface du disque de rayon r
    utilise la bibliothèque math
    """
    return pi*r**2
```

Code 4 – 2 versions

Exercice 4


```
1 def volume(longueur: int, largeur: int, hauteur: int) ->int:
2     return longueur*largeur*hauteur
```

```
1 print( volume(3, 4, 5) )
```

Exercice 5

```python
def est_majeur(age: int) -> bool:
    """
    vérifie si la personne est majeur

    Paramètres:
        age (int): âge de la personne

    Renvoie:
        bool: True si majeur
    """
    if age >= 18:
        return True
    else:
        return False
```

```python
print( est_majeur(20) )
print( est_majeur(10) )
print( est_majeur(18) )
```

```python
def est_majeur(age: int) -> bool:
    """
    vérifie si la personne est majeur

    Parameters
    ----------
    age : int
        âge de la personne.
    Renvoie
    -------
    boolean

    """
    return age >= 18
```

Code 5 – Seconde version :`age >= 18` est une expression booléenne

Exercice 6

```python
def puissance(x: int, n: int) -> int:
    """
    élève x à la puissance n

    Parameters
    ----------
    x : int
        entier.
    n : int
        exposant.

    Renvoie
    -------
    res : int
    """
    res = 1
    for i in range(n):
        res = res* x
    return res
```

```
1 print( "2**5", puissance(2, 5) )
2 print( "5**2", puissance(5, 2) )
```

Code 6

Exercice 7

```python
# import
from random import randint

# fonction
def lancer_des() -> int:
    """
    renvoie la somme de deux dés
    """
    de1 = randint(1, 6)
    de2 = randint(1, 6)
    return de1 + de2

# programme principal
somme_des = lancer_des()
print(somme_des)
```

Exercice 8

```python
def pythagore(a: int, b: int, c: int) -> bool:
    """
    vérifie si le triangle a, b, c est rectangle
    """
    cotes = a**2+b**2
    hyp = c**2
    if cotes == hyp:
        return True
    else:
        return False
```

```python
print( pythagore(3, 4, 5) )
print( pythagore(3, 4, 10) )
```

```python
def pythagore(a: int, b: int, c: int) -> bool:
    """
    vérifie si le triangle a, b, c est rectangle

    Parameters
    ----------
    a, b, c: int
        mesures des côtés.

    Renvoie
    -------
    Boolean.

    """
    return a**2 + b**2 == c**2
```

Code 7 – L'expression `a**2 + b**2 == c**2` renvoie un booléen

Exercice 9

```python
def somme(n: int) -> int:
    """
    renvoie la somme des entiers de 1 à n
    """
    resultat = 0
    for i in range(n+1):
        resultat = resultat + i
    return resultat
```

```python
print( "somme 5", somme(5) )
print( "somme 0", somme(0) )
```

Exercice 10

```python
def est_parfait(n: int) -> bool:
    """
    vérifie si n est parfait
    """
    somme = 0
    for i in range(1, n):
        if n % i == 0:  # i est multiple de n
            somme = somme + i
    # la somme des multiples est égal à n
    return somme == n
```

Exercice 11

```python
def est_premier(x: int) -> bool:
    """
    renvoie True si x est un nombre premier
    """
    diviseur = 2
    # si le reste est nul, c'est que nous avons un diviseur
    while diviseur < x and not(x%diviseur == 0):
        diviseur = diviseur + 1
    # On a divisé par tous les nombres < x
    if diviseur == x:
        return True
    else:  # on s'est arrêté avant
        return False
```

Remarque

Cet algorithme compte tous les diviseurs. Il est possible de faire plus rapide.

Remarque

Optimisation possible : Mathématiquement, les diviseurs de `x` (autres que `x`) sont inférieurs ou égaux à `x/2`

```
1  def est_premier(x: int) -> bool:
2      """
3      renvoie True si x est un nombre premier
4      """
5      diviseur = 2
6      # si le reste est nul, c'est que nous avons un diviseur
7      while diviseur <= x//2 and not(x%diviseur == 0):
8          diviseur = diviseur + 1
9      # On a divisé par tous les nombres < x
10     if diviseur == x:
11         return True
12     else:  # on s'est arrêté avant
13         return False
```

```python
def est_premier(x: int) -> bool:
    """
    renvoie True si x est un nombre premier
    """
    for i in range(2, x):
        # si le reste est nul, c'est que nous avons un diviseur
        if x%i == 0:
            return False
    return True
```

Code 8 – Seconde version

Remarque

Cet algorithme s'arrête dès qu'on trouve un diviseur.

Exercice 12

```python
def collatz(n: int) -> bool:
    """
    renvoie True si la procédure termine
    Si la boucle ne se termine jamais, la conjecture
    est fausse.
    """
    while n != 1:
        if n % 2 == 0:  # pair
            n = n//2
        else:
            n = 3*n+1
    return True
```

Observation

La conjecture affirme que quelque soit la valeur de **n**, cette fonction renverra **True**.

Exercice 13

```python
# import
import turtle as t

# fonction
def triangle(c: int) -> None:
    """
    Trace un triangle noir
    """
    t.begin_fill()
    for _ in range(3):
        t.forward(c)
        t.left(120)
    t.end_fill()
```

Observation

Dans la boucle, la variable est nommée `_`. Ce n'est pas obligatoire, on peut la nommer `i`, `var`…

```
1 # programme principal
2 t.up()
3 for _ in range(3):
4     t.left(90)
5     t.forward(50)
6     t.right(90)
7     triangle(100)
8 t.done()
```

Observation

On lève le crayon (`t.up()`) pour ne pas tracer de trait entre les triangles.

Exercice 14

Observation

Le fichier `tests_unitaires.py` rassemble tous les tests. Si les fonctions sont correctement construites, l'exécution des tests ne doit rien afficher dans la console.