Premiers pas en programmation

Chap. 2,1

Réponse 1.

1. *Affichage de la spécification d'une fonction :* Utiliser la fonction `help`.

   D'après la documentation la fonction `sum` attend une séquence de nombres et non pas deux nombres.

   ```
   >>> help(sum) Help on built-in function sum in module builtins:
     sum(iterable, start=0, /) Return the sum of a 'start' value (default: 0) plus
     an iterable of numbers
     When the iterable is empty, return the start value. This function is intended
     specifically for use with numeric values and may reject non-numeric types.
   ```

2. La fonction `int` convertit n'importe quel nombre ou une chaîne de caractères en nombre entier (si possible).

   ```
   >>> help(int) Help on class int in module builtins:
     class int(object) | int([x]) -> integer | int(x, base=10) -> integer | | Convert
     a number or string to an integer, or return 0 if no arguments | are given. If
     x is a number, return x.__int__(). For floating point | numbers, this truncates
     towards zero.
   ```

   De même les fonctions `float`, `str` et `bool` convertissent, si possible, un nombre, une chaîne de caractères en approximation des nombres réels, chaîne de caractères, nombre booléen.

3. L'application de la fonction `pow` à deux nombres x et y donne le nombre x^y.

   ```
   >>> import math as m
   >>> help(m.pow) Help on built-in function pow in module math:
     pow(x, y, /) Return x**y (x to the power of y).
   ```

4. *Méthode de récherche d'une fonction mathématique :* 1) Importer le module `math` ; 2) Lister, avec la fonction `dir`, toutes les fonctions de ce module et rechercher celle qui semble réaliser l'opération souhaitée ; 3) Confirmer le choix effectué à l'aide de la fonction `help`.

   ```
   >>> import math
   >>> dir(math) ['__doc__', '__file__', '__loader__', '__name__', '__package__',
   '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
   'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs',
   'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf',
   'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10',
   'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin',
   'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
   >>> help(math.sqrt)
     Help on built-in function sqrt in module math:
     sqrt(x, /) Return the square root of x.
   ```

Réponse 2.

```
>>> import random
>>> help(random.randint) Help on method randint in module random:
  randint(a, b) method of random.Random instance Return random integer in range [a, b],
  including both end points.
```

1. La documentation nous apprend que la fonction attend deux arguments.
2. La fonction retourne un entier compris entre les deux bornes passées en argument.
3. `>>> random.randint(1, 10)`
4. La fonction retourne un entier compris entre 1 et 10 inclus.

Réponse 3.

1. Du point de vue syntaxique la fonction est correcte.
2. La fonction telle qu'elle est définie retourne la valeur de $1 - n$ et ne correspond donc pas à la spécification.
3.

```
def retire_un(n: float) -> float:
    """ Retourne la valeur de n - 1 """
    return n - 1
```

Réponse 4.

1.

```
from math import pow
def polynomiale(a:int, b:int, c:int, d:int, x: float) -> float:
    """ Retourne la valeur ax^3 + bx^2 + cx + d
    >>> polynomiale(1, 1, 1, 1, 2) 15.0
    >>> polynomiale(1, 1, 1, 1, 3) 40.0
    """
    return a * pow(x, 3) + b * pow(x, 2) + c * x + d
```

2.

```
import math
def polynomiale_carre(a:int, b:int, c:int, x: float) -> float:
    """ Retourne la valeur ax^4 + bx^2 + c
    >>> polynomiale(1, 1, 1, 1, 2) 15.0
    >>> polynomiale(1, 1, 1, 1, 3) 40.0
    """
    return a * pow(x, 4) + b * pow(x, 2) + c
```

Réponse 5.

```
def somme(x:float, y: float, z: float) -> float:
    """ Retourne la somme des trois nombres passés en argument. """
    return x + y + z

def moyenne(a: float, b: float, c: float) -> float:
    """ Retourne la moyenne des trois nombres passés comme arguments. """
    return somme(a, b, c) / 3

def moyenne_ponderee(x:float, y:float, z:float, a:int, b:int, c:int) -> float:
    """ Retourne la moyenne pondérée des nombres x, y et z par les coefficients
    a, b et c.

    >>> moyenne_ponderee(5, 10, 15, 1, 1, 1)
    10.0
    >>> moyenne_ponderee(5, 10, 15, 1, 1, 0)
    7.5
    """
    return somme(a * x, b * y, c * z) / somme(a, b, c)
```

Réponse 6.

```
from math import pow, pi

def surface_rectangle(a: float, b: float) -> float:
    """ Détermine la surface du rectangle de côtés a et b. """
    return a * b

def volume_parallelepipede(a: float, b: float, h: float) -> float:
    """ Retourne le volume du parallélépipède rectangle de côtés a, b
    et de hauteur h. """
    return surface_rectangle(a, b) * h

def surface_disque(r: float) -> float:
    """ Returne la surface du disque de rayon r. """
    return pi * pow(r, 2)

def surface_couronne(r1: float, r2: float) -> float:
    """ Retourne la surface de la couronne comprise entre les rayons
    r1 et r2. """
    return surface_disque(r2) - surface_disque(r1)

def volume_tube(r1, r2, l) -> float:
    """ Retourne le volume de la partie pleine d'un tube de longueur l,
    dont la section est une couronne de rayon intérieur r1 et de rayon
    extérieur r2. """
    return surface_couronne(r1, r2) * l
```

Réponse 7.

```
from math import pow, sqrt
def distance(x1: float, y1: float, x2: float, y2: float) -> float:
    """ Retourne la distance dans le plan entre les deux points de coordonnées
    (x1, y1) et (x2, y2).

    >>> distance(0, 0, 1, 1)
    1.4142135623730951
    """
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2))
```

Réponse 8.

```
from random import random
def tirage_entier(a: int) -> int:
    """ Retourne un nombre entier aléatoire compris entre 1 et a (inclus). """
    return int(random() * a + 1)
```