1ère Spécialité NSI *Mercredi 20 novembre 2019*

Devoir surveillé n°2

Héron d'Alexandrie

Héron d'Alexandrie est un ingénieur, un mécanicien et un mathématicien grec du premier siècle après J.-C.

On ne sait pas grand chose de la vie d'Héron, si ce n'est qu'il était originaire d'Alexandrie ; les historiens se sont même longtemps divisés sur l'époque où il a vécu. Leurs estimations allaient du 1er siècle avant J.-C. au 3ème siècle de notre ère. Aujourd'hui, la querelle est éteinte : il est clairement établi que Héron est postérieur à Vitruve mort en -20, et contemporain de Pline l'Ancien (23 – 79), en étant actif autour de l'an 62. Il a donc bien vécu au 1er siècle après J.-C. et sans doute au début du 2ème siècle, donc sous l'Empire romain, mais dans l'Alexandrie grecque.

On attribue à Héron d'Alexandrie plusieurs formules mathématiques et une méthode efficace d'extraction de racine carrée, c'est-à-dire de résolution de l'équation $x^2 = a$, avec a positif :

1. Choisir une première valeur raisonnable que l'on note G.
2. Améliorer cette valeur en calculant la moyenne des valeurs G et x/G.
3. Vérifier si cette nouvelle valeur convient.
4. Reprendre l'étape 2 tant que la valeur calculée n'est pas satisfaisante.

L'objectif de ce problème est de mettre en œuvre cette méthode en Python.

Votre fichier réponse devra se terminer par les instructions :

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

et chaque fonction devra incorporer dans sa documentation les informations qui vous permettront de tester sa validité (reprendre donc systématiquement les spécifications que je vous donne).

1. Définir la fonction `moyenne` dont la spécification est

```
def moyenne(a: float, b: float) -> float:
    """ Calcule et retourne la moyenne des deux nombres a et b passés en argument.
    >>> moyenne(12, 16)
    14.0
    >>> moyenne(0, 8)
    4.0
    """
```

2. Définir la fonction `valeur_absolue` dont la spécification est :

```
def valeur_absolue(x: float) -> float:
    """
    Calcule et retourne la valeur absolue du nombre x passé en argument.
```

```
    >>> valeur_absolue(3)
    3
    >>> valeur_absolue(0)
    0
    >>> valeur_absolue(-3)
    3
    """
```

Remarque. Il est interdit d'utiliser la fonction `abs` intégrée à Python.

3. Définir la fonction `puissance` dont la spécification est :

```
def puissance(x: float, n: int) -> float:
    """ calcule et retourne le résultat de x à la puissance entière n :
    x^n = x . x . x . ... . x (n fois)

    >>> puissance(2, 8)
    256
    >>> puissance(0, 2)
    0
    >>> puissance(3, 0)
    1
    """
```

Remarque. Il est interdit d'utiliser l'opérateur `**` intégré à Python ou la fonction `pow` du module `math`.

4. Définir la fonction `amelioration_essai` dont la spécification est :

```
def amelioration_essai(essai: float, x: float) -> float:
    """ Calcule et retourne la moyenne des nombres essai (strictement positif)
    et x/essai.

    >>> amelioration_essai(1, 2)
    1.5
    >>> amelioration_essai(2, 1)
    1.25
    """
```

Remarque. Cette fonction doit utiliser la fonction `moyenne` définie plus haut.

5. Définir le prédicat `est_suffisamment_bon` dont la spécification est :

```
def est_suffisamment_bon(essai: float, x: float) -> float:
    """
    Retourne True si la valeur absolue de la différence du carré du nombre essai
    et du nombre x est inférieure à une tolérance donnée (prendre 0.001).
    Retourne False sinon.
```

```
    >>> est_suffisamment_bon(1.9, 4)
    False
    >>> est_suffisamment_bon(1.999, 4)
    False
    >>> est_suffisamment_bon(1.9999, 4)
    True
    """
```

Remarque. Cette fonction doit utiliser les fonctions `valeur_absolue` et `puissance` définies ci-dessus.

6. Définir la fonction `test` qui implémente l'algorithme de Héron d'Alexandrie. La spécification de la fonction est :

```
def test(essai: float, x: float) -> float:
    """
    Retourne la racine carrée du nombre x. Le calcul est effectué grâce à un
    raisonnement itératif depuis une première valeur strictement positive notée
essai.

    >>> test(2, 4)
    2
    >>> test(1, 4)
    2.0000000929222947
    >>> test(7, 4)
    2.0000000271231317
    """
```

Remarque. Cette fonction doit utiliser les fonctions `est_suffisamment_bon` et `amelioration_essai`.

7. Définir la fonction `racine_carree` dont la spécification est :

```
def racine_carree(x: float) -> float:
    """
    Retourne le résultat de l'appel de la fonction test avec la valeur 1
    pour l'argument essai.

    >>> racine_carree(4)
    2.0000000929222947
    >>> racine_carree(9)
    3.00009155413138
    >>> racine_carree(16)
    4.000000636692939
    """
```

8. Définir la fonction `main` qui affiche à l'écran la liste des racines carrées de tous les nombres pairs compris dans l'intervalle $[1; 100]$.