

# Exercices spécification et tests

## Exercice 1 : Prototypage

Prototyper correctement les fonctions suivantes. Il est possible de les tester pour mieux les reconnaître.

<pre># Mystère 1 def a(b, c, d, e):     f = (b + c + d + e) / 4     return f</pre>	<pre># Mystère 2 def a(b):     if b%2 == 0:         return True     else:         return False</pre>	<pre># Mystère 3 def a(b, c):     if b &gt;= c:         return b     else:         return c</pre>
<pre># Mystère 4 def a(b, c):     d = 0     for _ in range(c):         d = d + b     return d</pre>	<pre># Mystère 5 def a(b, c):     d = b // c     e = b % c     return (d, e)</pre>	<pre># Mystère 6 def a(b, c, d):     b = b**2     c = c**2     d = d**2      e = b == c + d     e = e or c == b + d     e = e or d == b + c     return e</pre>
<pre># Mystère 7 def a(b, c, d):     e = b[c]     b[c] = b[d]     b[d] = e</pre>	<pre># Mystère 8 def a(b):     c = []     while b &gt; 0:         if b%2 == 0:             c.insert(0, 0)             b = b // 2         else:             c.insert(0, 1)             b = (b - 1) // 2     return c</pre>	

## Exercice 2 : Typage.

Typier les fonctions suivantes. Ils existe plusieurs réponses possibles.

<pre># Typage 1 def gagnant(score1, score2):     if score1 &gt; score2:         return "Joueur 1"     elif score1 &lt; score2:         return "Joueur 2"     else:         return "ex æquo"</pre>	<pre># Typage 2 from math import sqrt def distance(A, B):     xA = A[0]     yA = A[1]     xB = B[0]     yB = B[1]     return sqrt((xB - xA)**2 + (yB - yA)**2)</pre>
<pre># Typage 3 def puissance(valeur, exposant):     l = []     for i in range(1, exposant + 1):         l.append(valeur**i)     return l</pre>	

**Exercice 3 : Chaîne de documentation.**

Créer les chaînes de documentation des fonctions de l'exercice 2.  
Elle doivent être cohérentes avec vos typages.

**Exercice 4 : Tests.**

Écrire 3 jeux de tests **pertinents** pour chacune des fonctions de l'exercice 2.  
Ils doivent être cohérent avec vos typages.

**Exercice 5 : Assertions.**

- 1) Transformer vos tests de l'exercice 4 en assertions.
- 2) Pour chacune des fonctions de l'exercice 2, écrire 2 tests de préconditions sur les arguments, cohérents avec vos typages.

**Exercice 6 : Doctest.**

Intégrer vos tests de l'exercice 5 à la chaîne de documentation des fonctions et utiliser le module doctest.

**Exercice 7 : Type checker**

Écrire pour chacune des fonctions avec typage de l'exercice 2, un appel où le typage est respecté et un autre où il ne l'est pas puis utiliser mypy sur le site suivant pour analyser votre code. <https://mypy-play.net/>

**Bonus : Plusieurs types possibles**

Il existe plusieurs types possibles pour les paramètres des fonctions de l'exercice 2.  
Chercher en ligne comment laisser la possibilité de passer différents types aux fonctions et modifier vos assertions en conséquence.