Ejercicio 6. Unidad 3.

Tripletas pitagóricas. Existen triángulos rectángulos en los que las longitudes de sus lados son enteros. El conjunto de tres valores enteros cuyos valores verifican el Teorema de Pitágoras decimos que es una tripleta pitagórica. Escribir todas las tripletas pitagóricas en las que ningún lado sea mayor que 100. Utilizar tres iteraciones *for* anidadas que pruebe todas las posibilidades. Este es un ejemplo de computación "por fuerza bruta", semejante al del triángulo estudiado en este mismo capítulo. Existe un grán número de problemas interesantes para los cuales no existe ningún enfoque algorítmico conocido aparte del uso de la fuerza bruta.

```
# tripleta pitagórica mediante fuerza bruta
lista = [(a, b, c)
 for a in range(1, 20)
   for b in range(1,20)
      for c in range(1,20)
         if a**2 + b**2 == c**2]
print(lista)
```

[(3, 4, 5), (4, 3, 5), (5, 12, 13), (6, 8, 10), (8, 6, 10), (8, 15, 17), (9, 12, 15), (12, 5, 13), (12, 9, 15), (15, 8, 17)]

Como es lógico, se observa que dada cualquier combinación de a y b, también existe la combinación contraria, por lo que hay ternas no necesarias. Por ejemplo: 8, 6, 10 hace que también sea válida la combinación 6, 8, 10. Se pueden eliminar las combinaciones contrarias de a y b imponiendo por ejemplo que b>a. Bastará con sustituir `for b in range(1,20) por for b in range(a+1,20).`

Otra forma de hacerlo sería usando la función append. Por ejemplo, imponiendo que a<b<c:

```
lista = []
for a in range(1,20):
   for b in range(a+1, 20):
      for c in range(b+1,20):
         if a**2 + b**2 == c**2:
            lista.append((a, b, c))
print(lista)
```

[(3, 4, 5), (5, 12, 13), (6, 8, 10), (8, 15, 17), (9, 12, 15)]

El hecho de hacer `for c in range(b+1,20):` optimiza el rendimiento del programa ya que no se testean combinaciones que de entrada se sabe que no van a cumplir la condición requerida.