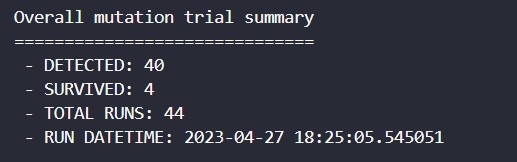
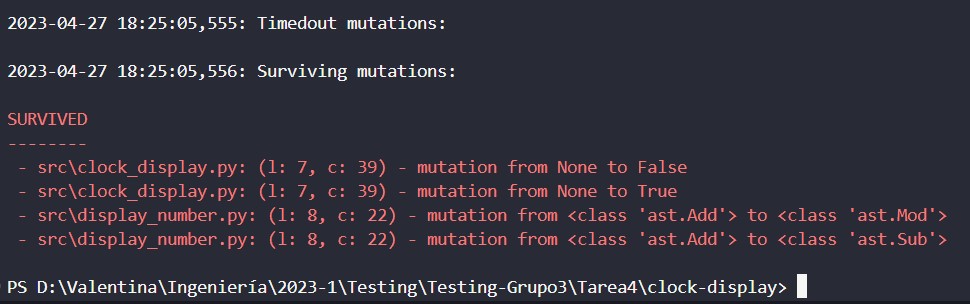
# Resporte Mutaciones





- src\clock\_factory.py: (l: 8, c: 26) - mutation from None to True/False y src\display\_number.py: (l: 3, c: 41) - mutation from None to True/False

Estas mutaciones se hacen en la línea de la definición del método init (def \_\_init\_\_(self) -> None). Se está indicando que el método retorna None, por lo que la mutación a True/False indicaría que el método retorna el booleano correspondiente. Por la forma en que está construido Python, esta mutación no es necesario eliminarla ya que cuando un método \_\_init\_\_() retorna algo, se levanta un error de TypeError lo que no permitiría la ejecución del código. <https://docs.python.org/3/reference/datamodel.html?highlight=__init__#object.__init__>

- src\display\_number.py: (l: 8, c: 22) - mutation from <class 'ast.Add'> to <class 'ast.Sub'>

Esta mutación ocurre en el método increase de la clase NumberDisplay indicando que el primer “+” es cambiado por un “-” en “(self.value + self.limit + 1) % self.limit”. Tests no fallan dada la naturaleza de la operación módulo (división con resto), tal que A%B = (A+B)%B = (A-B)%B, con el resto restringido a 0<= r < B. Así, considerando A = value + 1, se puede entender entonces el por qué la mutación no provoca que la prueba falle.

- src\display\_number.py: (l: 8, c: 22) - mutation from <class 'ast.Add'> to <class 'ast.Mod'>

Mutación indica que el primer “+” es cambiado por un “%”. Esto no provoca que la prueba falle en tanto sacar el módulo a un número, sumarle uno y a ese resultado volver a sacarle el módulo respecto del mismo número, es equivalente a tomar la suma de un número y uno, y a ese resultado sacarle el módulo. EJ: (3%7+1)%7 = 4 = (3+7+1)%7 = (3+1)%7 donde se usó la idea de la mutación anterior. Notar funciona para todo valor limit>0 respetando las propiedades de la división con resto.

**REPORTE TESTS SMELLS ACTIVDAD 3 USO PYNGUIN**

MÓDULO ‘display\_number.py’

MÓDULO ‘clock\_display.py’

* EJEMPLOS DONDE SE OBSERVAN LOS TESTS SMELLS