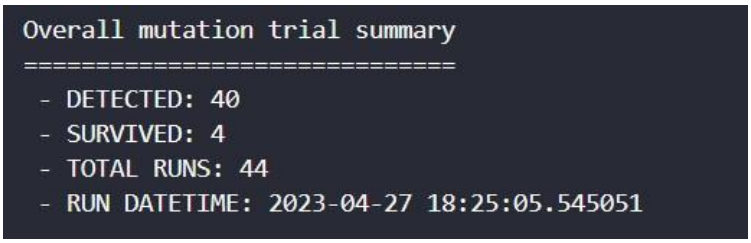
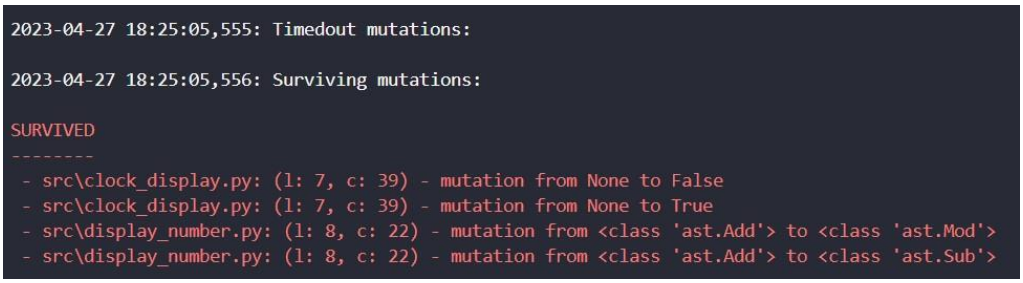
REPORTE MUTACIONES ACTIVIDAD 2





- src\clock_factory.py: (l: 8, c: 26)- mutation from None to True/False

- 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

Identificamos los siguentes test smells:

- Unknown Test, ya que hay multiples tests sin una declaración assert o equivalente. Por ejemplo, test cases 1, 2,3 y 4 de clock_display y tests 2, 3 y 5 de display_number.
- Eager Test, ya que ocurre en un par de tests que se invocan varios métodos de un objeto. Por ejemplo, en el test_case_4 de display_number y tests 0 y 5 de clock_display.
- Invocación de métodos del objeto luego de haber declarado un assert: esto no tiene sentido sentido en tanto no aporta nada a los tests ya que la correctitud de la performance de esas líneas no es evaluada. Ejemplo de ello, son los test 0 y 5 de clock_display y test 1 y 4 de de display_number.
- Variables declaradas no utilizadas: tal como se describe se instancian clases o definen variables que no son utilizadas dentro de la ejecución del test. Ejemplo de esto, es en los tests 0, 1, 2 y 4 de clock_display, donde var_0 no es ocupada.
- Error en los argumentos entregados al instanciar una clase: Al momento de crear una instancia de clase se entregan argumentos de distinto tipo al requerido por la clase. Por ejemplo, en los tests 0, 1 y 2 de clock_display se entrega un número en vez de un iterable. Que el número entregado sea en formato binario no es problema.
- Llamado de métodos de clase sin sentido: Por ejemplo, en el test_case_3 se aplica el método increment() sobre un booleano, que es el tipo de dato var_0 al tomar el valor de retorno de una aplicación de increment() sobre un ClockDisplay. También, en el test_case_4 de display_number se aplica el método increase() sobre booleanos lo cual arrojaría un error en la ejecución.