**Actividad relacionada con la lección 3:**

**Como hemos estudiado en esta lección existen diversas herramientas que nos facilitan la tarea de documentar nuestro código. Para la actividad de esta lección se proponen dos ejercicios:**

1. **Busque una herramienta alternativa a Sphinx para generar la documentación de código en Python.**

Sphinx

Sphinx es una herramienta que facilita la creación de una documentación inteligente y hermosa, escrita por Georg Brandl y licenciada bajo la licencia BSD.

Originalmente fue creada para la documentación de Python y tiene excelentes instalaciones. para la documentación de proyectos de Python, pero C / C ++ también es compatible, y se planea agregar soporte especial para otros idiomas también.

Doxygen

Doxygen es una herramienta para generar documentación a partir de fuentes C++ anotadas, así como otros lenguajes de programación populares como C, Objective-C, C #, PHP, Java, Python, IDL (CORBA, Microsoft y UNO / OpenOffice), Fortran, VHDL, Tcl, y hasta cierto punto D.

Ejemplo básico:

Utilizando un código y la interfaz de nuestro generador de documentación vamos a obtener una documentación completa con muy pocos pasos.

* Lo primero que vamos a hacer es configurar nuestro proyecto.
* Luego configuramos el lenguaje que hemos utilizado, en este caso Python.
* Después el formato de nuestra documentación.
* Luego seleccionamos si queremos que nos realice unos diagramas.
* Y finalmente tenemos la documentación

1. **Realice la documentación de alguno de los códigos que ha desarrollado a lo largo de este curso. La documentación puede generarla con Sphinx o con la herramienta que ha buscado en el primer ejercicio.**
2. class OperacionesMatematicas:
3. """
4. Operaciones matemáticas. Es una clase que permite realizar una serie de operaciones matemáticas con números enteros.
5. Atributos:
6. op1: es el primer operando de la operación matemática y debe ser de tipo entero. Si un método solo recibe un parámetro,
7. éste será op1.
8. op2: es el segundo operando de la operación matemática y debe ser de tipo entero.
9. Métodos:
10. suma:
11. suma los enteros op1 y op2
12. resta:
13. resta los enteros op1 y op2
14. producto:
15. multiplica los enteros op1 y op2
16. division:
17. divide el entero op1 entre op2. Si op2 vale 0 , el resultado será 0.
18. factorial:
19. calcula el factorial de op1
20. es\_primo:
21. determina si op1 es o no primo
22. >>> import OperacionesMatematicas
23. >>> oM = OperacionesMatematicas(4, 5)
24. >>> suma = OperacionesMatematicas.suma()
25. >>> fact = OperacionesMatematicas.factorial()
26. """
28. def \_\_init\_\_(self, op1, op2):
29. self.op1 = op1
30. self.op2 = op2
31. def suma(self):
32. """
33. Metodo suma. Aplica el algoritmo de la suma sobre los operandos op1 y op2.
34. Input:
35. La entrada son los operandos op1 y op2 que son atributos de la clase.
36. Output:
37. Resultado: variable de tipo entero que almacena el resultado de sumar op1 y op2
38. """
39. resultado = self.op1 + self.op2
40. return resultado
41. def resta(self):
42. """
43. Metodo resta. Aplica el algoritmo de la resta sobre los operandos op1 y op2.
44. Input:
45. La entrada son los operandos op1 y op2 que son atributos de la clase.
46. Output:
47. Resultado: variable de tipo entero que almacena el resultado de restar op1 y op2
48. """
49. resultado = self.op1 - self.op2
50. return resultado
51. def producto(self):
52. """
53. Metodo producto. Aplica el algoritmo de la multiplicacion sobre los operandos op1 y op2.
54. Input:
55. La entrada son los operandos op1 y op2 que son atributos de la clase.
56. Output:
57. Resultado: variable de tipo entero que almacena el resultado de multiplicar op1 y op2
58. """
59. resultado = self.op1 \* self.op2
61. return resultado
62. def division(self):
63. """
64. Metodo division. Aplica el algoritmo de la division sobre los operandos op1 y op2.
65. Input:
66. La entrada son los operandos op1 y op2 que son atributos de la clase.
67. Output:
68. Resultado: variable de tipo entero que almacena el resultado de dividir op1 y op2
69. """
70. resultado = 0
71. if(self.op2 != 0):
72. resultado = self.op1 / self.op2
73. return resultado
74. def primo(self):
75. """
76. Metodo primo. Determina si el operando op1 es un numero primo.
77. Inputs:
78. -------
79. self.op1
80. Outputs:
81. -------
82. True su self.op1 es primo, False en caso contrario
83. """
84. es\_primo = True
85. for i in (2, self.op1 - 1):
86. if(self.op1%i == 0):
87. es\_primo = False
88. return es\_primo
89. def factorial(self):
90. """
91. Metodo para calcular el factorial de un numero.
92. """
93. assert(n >= 0)
94. fct = 1
95. for i in range(1, self.op1 + 1):
96. fct \*= i
98. return fct