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

2. **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.**

```
3. class OperacionesMatematicas:
4.     """
5.     Operaciones matemáticas. Es una clase que permite realizar una
   serie de operaciones matemáticas con números enteros.
6.     Atributos:
7.         op1: es el primer operando de la operación matemática y
   debe ser de tipo entero. Si un método solo recibe un parámetro,
8.         éste será op1.
9.         op2: es el segundo operando de la operación matemática y
   debe ser de tipo entero.
10.    Métodos:
11.        suma:
12.            suma los enteros op1 y op2
13.        resta:
14.            resta los enteros op1 y op2
15.        producto:
16.            multiplica los enteros op1 y op2
17.        division:
18.            divide el entero op1 entre op2. Si op2 vale 0 , el
   resultado será 0.
19.        factorial:
20.            calcula el factorial de op1
21.        es_primo:
22.            determina si op1 es o no primo
23.
24.    >>> import OperacionesMatematicas
25.    >>> oM = OperacionesMatematicas(4, 5)
26.    >>> suma = OperacionesMatematicas.suma()
27.    >>> fact = OperacionesMatematicas.factorial()
28.    """
29.
30.    def __init__(self, op1, op2):
31.        self.op1 = op1
32.        self.op2 = op2
33.
34.    def suma(self):
35.        """
36.        Metodo suma. Aplica el algoritmo de la suma sobre los
   operandos op1 y op2.
37.        Input:
38.            La entrada son los operandos op1 y op2 que son
   atributos de la clase.
39.        Output:
40.            Resultado: variable de tipo entero que almacena el
   resultado de sumar op1 y op2
41.        """
42.        resultado = self.op1 + self.op2
43.        return resultado
44.
```

```
45.     def resta(self):
46.         """
47.         Metodo resta. Aplica el algoritmo de la resta sobre los
    operandos op1 y op2.
48.         Input:
49.             La entrada son los operandos op1 y op2 que son
    atributos de la clase.
50.         Output:
51.             Resultado: variable de tipo entero que almacena el
    resultado de restar op1 y op2
52.         """
53.         resultado = self.op1 - self.op2
54.         return resultado
55.
56.     def producto(self):
57.         """
58.         Metodo producto. Aplica el algoritmo de la multiplicacion
    sobre los operandos op1 y op2.
59.         Input:
60.             La entrada son los operandos op1 y op2 que son
    atributos de la clase.
61.         Output:
62.             Resultado: variable de tipo entero que almacena el
    resultado de multiplicar op1 y op2
63.         """
64.         resultado = self.op1 * self.op2
65.
66.         return resultado
67.
68.     def division(self):
69.         """
70.         Metodo division. Aplica el algoritmo de la division sobre
    los operandos op1 y op2.
71.         Input:
72.             La entrada son los operandos op1 y op2 que son
    atributos de la clase.
73.         Output:
74.             Resultado: variable de tipo entero que almacena el
    resultado de dividir op1 y op2
75.         """
76.         resultado = 0
77.         if(self.op2 != 0):
78.             resultado = self.op1 / self.op2
79.
80.         return resultado
81.
82.     def primo(self):
83.         """
```

```
84.         Metodo primo. Determina si el operando op1 es un numero
    primo.
85.         Inputs:
86.         -------
87.             self.op1
88.         Outputs:
89.         -------
90.             True su self.op1 es primo, False en caso contrario
91.         """
92.         es_primo = True
93.         for i in (2, self.op1 - 1):
94.             if(self.op1%i == 0):
95.                 es_primo = False
96.
97.         return es_primo
98.
99.     def factorial(self):
100.             """
101.             Metodo para calcular el factorial de un numero.
102.             """
103.             assert(n >= 0)
104.
105.             fct = 1
106.             for i in range(1, self.op1 + 1):
107.                 fct *= i
108.
109.             return fct
```