SPYDER

Rafael Fernández Ortiz

Se denomina ***Spyder*** o ***Web Spyder*** al bot/script que se utiliza para analizar o rasrear el código html de una dirección url. De este modo, puedes identificar las palabras y por tanto hacer un análisis de frecuencia de ciertas palabras, o identificar los distintos hipervínculos que existen en dicha url.

Este es un primer proyecto de clase, así repaso el concepto y la utilidad de las clases, de un objeto *spyder*.

Las distintas **funciones** o **métodos** del *spyder* permiten, o bien extraer los links de direcciones *url* y *pdfs*, y mostrarlo en pantalla mediante una lista; o bien, escribirlo en un archivo *.txt*

Comentarios: Este es una tercera versión del código. Aún hay que afinarlo muchísimo.

```
import urllib.request as url

class Spyder(object):

    def __init__(self, url):
        self.url = str(url)
        self.html = self.__getHtml().lower()
        self.dominio = self.__getDominio()

    def __getDominio(self):
        url = self.url
        protocolo = url[:url.find('://') + 3]
        url = url[url.find('://') + 3:]
        dominio = url[:url.find('/')]
        return protocolo + dominio

    def __getHtml(self):
        html = ""
        try:
            dir = url.urlopen(self.url)
            html = str(dir.read())
        except:
            print("Error al intentar conectarme con: " + self.url)
            pass
        return html

    def __getImages(self):
        img = []
        html = self.html
        while True:
            pos = html.find('src="')
            if pos == -1:
                break
            html = html[pos + 5:]
            imagen = html[:html.find('"')]
```

```
            if imagen.startswith("/"):
                imagen = self.dominio + imagen
            img.append(imagen)
        return img

    def __getUrls(self):
        urls = []
        html = self.html
        while True:
            pos = html.find('href="')
            if pos == -1:
                break
            html = html[pos + 6:]
            url = html[:html.find('"')]
            if url.startswith("/"):
                url = self.dominio + url
            urls.append(url)
        return urls

    def getUrls(self):
        return filter(lambda x: x.startswith("http"), self.__getUrls())

    def getPdfs(self):
        return filter(lambda x: x.endswith(".pdf"), self.__getUrls())

    def getImages(self):
        img = []
        imagenes = self.__getImages()
        img += filter(lambda x: x.endswith(".jpg"), imagenes)
        img += filter(lambda x: x.endswith(".jpeg"), imagenes)
        img += filter(lambda x: x.endswith(".gif"), imagenes)
        img += filter(lambda x: x.endswith(".png"), imagenes)
        return img

    def __linksToTxt(self, archivo, links):
        fichero = open(archivo, 'w')
        for i in links:
            fichero.write(i + "\n")
        fichero.close()

    def getUrlsToTxt(self):
        self.__linksToTxt("links.txt", self.getUrls())

    def getImagesToTxt(self):
        self.__linksToTxt("img.txt", self.getImages())

    def getPdfsToTxt(self):
        self.__linksToTxt("pdfs.txt", self.getPdfs())

    # Este metodo pretende buscar la frecuencia de palabras dentro del contenido de la
pagina web
    # Sin embargo, está regular, habria que darle una buena pensada
    #
```

```
    # def frecuencias(self):
    #     texto = self.html
    #     diccionario = dict()
    #     separadores = ",.;:!?¿¡+*(){}|"
    #     # etiquetas = """<>=\/"""
    #     for i in separadores:
    #         texto = texto.replace(i, ' ')
    #     texto = texto.split()
    #     for p in texto:
    #         if len(p) <= 8:     #Afinar esto
    #             diccionario[p] = diccionario.get(p, 0) + 1
    #     newfich = open("frecuencias.txt", 'w')
    #     for k in diccionario.keys():
    #         newfich.write("{0} := {1}".format(k, diccionario.get(k)) + '\n')
    #     newfich.close()

# Ejemplo
spyder = Spyder("https://www.boe.es/boe/dias/2019/01/02/")
# spyder.frecuencias()
spyder.getUrlsToTxt()
spyder.getPdfsToTxt()
spyder.getImagesToTxt()
```