

ENCRIPTAR Y DESENCRIPTAR DATOS EN PYHON, CON «cryptography».


👤 programacionpython80889555 📁 algoritmos, criptografía, datos, funciones en python, hacking, matemáticas con python, matplotlib, programación en python, programacion, python, seguridad, software, tech
🕒 marzo 3, 2020noviembre 27, 2022 ⌵ 4 minutos

Saludos y bienvenidos una semana más a vuestro blog sobre programación en Python. Hoy vamos a hablar de una tema que no tocamos desde hace un tiempo. Y es el relativo al cifrado y encriptado de información. Concretamente, hablaremos de «**cryptography**», una librería que nos permitirá realizar el encriptado de información (por ej: un mensaje de texto o un archivo) .

En el proceso de encriptado, utilizamos un algoritmo (en este caso del tipo **AES**) que lo que hará será generar una clave (en una archivo de extensión «**.key**») a través de la cual podremos encriptar nuestra información. A su vez, al tratarse de un método de cifrado simétrico, podremos usar esa misma clave, tanto para encriptar como para desencriptar nuestra información. A su vez, sobre el mencionado algoritmo (**AES**), en el siguiente enlaces tenéis información acerca del método de cifrado que emplea:

https://es.wikipedia.org/wiki/Advanced_Encryption_Standard
(https://es.wikipedia.org/wiki/Advanced_Encryption_Standard).

Hecha esta introducción, empezaremos a realizar nuestro primer ejercicio, encriptando un sencillo mensaje de texto con «**cryptography**», librería que deberemos instalar previamente en el equipo con «**pip install cryptography**».

 Símbolo del sistema

```
Microsoft Windows [Versión 10.0.18362.657]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Antonio>cd AppData\Local\Programs\Python\Python37\Scripts

C:\Users\Antonio\AppData\Local\Programs\Python\Python37\Scripts>pip install cryptography_
```

Para ello, empezaremos, creando una función (a la que hemos llamado «**genera_clave()**») que como se deduce por su nombre, se encargará de crear el archivo «**.key**» (al que llamaremos «**clave.key**»), usando «**Fernet.generate_key()**» (para lo cual hemos importado, previamente el módulo «**Fernet**»), que luego utilizaremos tanto para encriptar la información, como para desencriptarla:

```

>>> #IMPORTAMOS RECURSOS.
>>> from cryptography.fernet import Fernet
>>>
>>> #FUNCION PARA ESCRIBIR Y GUARDAR CLAVE.
>>>
>>> def genera_clave():
        clave = Fernet.generate_key()
        with open("clave.key", "wb") as archivo_clave:
            archivo_clave.write(clave)

```

Ya tenemos la función que creará nuestra clave. No obstante, para poder realizar el cifrado de nuestro mensaje, necesitamos contar con dicha clave, por lo que necesitaremos cargarla previamente. Para ello definiremos una segunda función (de nombre «**cargar_clave()**») que, simplemente, abrirá (función «**open()**») y leerá («**.read()**») el archivo «**clave.key**», creado anteriormente:

```

>>> #IMPORTAMOS RECURSOS.
>>> from cryptography.fernet import Fernet
>>>
>>> #FUNCION PARA ESCRIBIR Y GUARDAR CLAVE.
>>>
>>> def genera_clave():
        clave = Fernet.generate_key()
        with open("clave.key", "wb") as archivo_clave:
            archivo_clave.write(clave)

>>> #FUNCIÓN PARA CARGAR LA CLAVE.
>>>
>>> def cargar_clave():
        return open("clave.key", "rb").read()

```








Pasemos ahora a aplicar ambas funciones para encriptar un mensaje de texto. Así lo primero será ejecutar la función «**genera_clave()**», mediante la correspondiente llamada a la misma:

```

>>> #PROCESO DE ENCRIPTADO.
>>>
>>> #CREAMOS Y GUARDAMOS CLAVE.
>>> genera_clave()

```

La ejecución de esta función, generará en nuestro directorio actual, el archivo «**.key**» al que dimos el nombre de «**clave.key**». Es importante guardar este archivo por que es el que vamos a usar tanto para cifrar nuestro mensaje, como para luego descifrarlo de nuevo:

 CilindroB	26/04/2019 23:11	Python File	1 KB
 CilindroC	28/04/2019 12:03	Python File	1 KB
 circles	17/09/2019 0:18	Python File	2 KB
 clave.key	28/02/2020 10:12	Archivo KEY	1 KB
 clipboard	31/01/2020 18:32	Documento de te...	1 KB
 code	01/02/2020 14:08	Archivo PNG	1 KB
 Cold_Rise	23/02/2020 13:54	Archivo WAV	26.330 KB

A continuación, aplicaremos esta clave para cifrar nuestro mensaje, para lo cual, cargaremos dicha clave, usando la función «**cargar_clave()**» previamente definida. Tras lo cual, estableceremos el mensaje a cifrar, aplicandole a su vez, el método de cifrado «**encode()**»:

```
>>> #PROCESO DE ENCRIPTADO.
>>>
>>> #CREAMOS Y GUARDAMOS CLAVE.
>>> genera_clave()
>>>
>>> #CARGAMOS CLAVE.
>>> clave = cargar_clave()
>>>
>>> #ENCRIPTAMOS UN MENSAJE.
>>> mensaje = "Mi mensaje a encriptar".encode()
```

Tras ello, pasaremos a crear la versión cifrada de «**mensaje**». Para lo cual, empezaremos iniciando el método «**Fernet()**» (el cual, iniciado previamente, se encargará de asegurarse que nuestro mensaje no pueda leerse sin la clave creada) pasandole como argumento la clave creada y cargada «**clave.key**». Una vez hecho ello, pasaremos a encriptar nuestro mensaje haciendo uso del método «**encrypt()**» introduciendo como argumento la variable «**mensaje**»:

```

>>> #PROCESO DE ENCRİPTADO.
>>>
>>> #CREAMOS Y GUARDAMOS CLAVE.
>>> genera_clave()
>>>
>>> #CARGAMOS CLAVE.
>>> clave = cargar_clave()
>>>
>>> #ENCRIPTAMOS UN MENSAJE.
>>> mensaje = "Mi mensaje a encriptar".encode()
>>>
>>> #INICIAMOS "Fernet".
>>> f = Fernet(clave)
>>>
>>> #ENCRIPTAMOS MENSAJE.
>>> encriptado = f.encrypt(mensaje)

```

Con esto ya tendríamos nuestro mensaje cifrado en la variable a la que hemos dado el nombre de «**encriptado**». Para verlo podemos escribir «**print(encriptado)**»:

```

>>> #ENCRIPTAMOS MENSAJE.
>>> encriptado = f.encrypt(mensaje)
>>>
>>> #IMPRIMIMOS MENSAJE ENCRİPTADO.
>>> print(encriptado)
b'gAAAAABeV6jzGdEv6UktVrqliwX0JAGO52fCxMKuQwNGnXhau22mkAibWhrNx4MRzUGoVc3h44hJ56
RVhVGUyPJU9F42T-STu_jHlxe2QaptSACs7Q8SBOA='

```

Tal y como hemos apuntado más arriba, podemos usar, también, nuestra clave, para descifrar mensajes cifrados previamente con ella. Para ello, usaremos el método «**decrypt()**»:

```

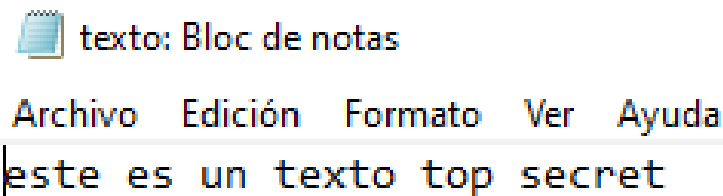
>>> #ENCRIPTAMOS MENSAJE.
>>> encriptado = f.encrypt(mensaje)
>>>
>>> #IMPRIMIMOS MENSAJE ENCRİPTADO.
>>> print(encriptado)
b'gAAAAABeV6jzGdEv6UktVrqliwX0JAGO52fCxMKuQwNGnXhau22mkAibWhrNx4MRzUGoVc3h44hJ56
RVhVGUyPJU9F42T-STu_jHlxe2QaptSACs7Q8SBOA='
>>>
>>> #PROCESO DE DESENCRIPTADO.
>>>
>>> #DESENCRIPTAR MENSAJE.
>>> desencriptado = f.decrypt(encriptado)
>>>
>>> #IMPRIMIMOS MENSAJE ORIGINAL.
>>> print(desencriptado)
b'Mi mensaje a encriptar'

```

El resultado de tal operación lo almacenamos en la variable a la que hemos llamado «**desencriptado**», que, a continuación (como en el anterior caso) hemos mostrado con «**print**».

ENCRYPTAR/DESENCRIPTAR ARCHIVO.

Hemos visto como encriptar un sencillo mensaje de texto. Sin embargo también podemos encriptar (y desencriptar) la información contenida en un archivo. Así, en este segundo ejemplo vamos a encriptar el contenido de un documento de texto, de nombre «**texto.txt**» y ubicado en nuestro directorio de ejecución:



Para encriptar su contenido, usaremos la misma clave que creamos en el caso anterior, con lo que no será necesario llamar a la función encargada de generarla. Lo que si haremos será definir las funciones «**carga_clave()**» y «**encrypt_archivo()**», para cargar la clave y encriptar el contenido de nuestro documento de texto:

```
>>> from cryptography.fernet import Fernet
>>>
>>> #FUNCION PARA CARGAR CLAVE.
>>> def carga_clave():
        return open("clave.key", "rb").read()

>>> #FUNCION PARA ENCRYPTAR ARCHIVO.
>>> def encrypt(nom_archivo,clave):
        f = Fernet(clave)
        with open(nom_archivo, "rb") as file:
            archivo_info = file.read()
        encrypted_data = f.encrypt(archivo_info)
        with open(nom_archivo, "wb") as file:
            file.write(encrypted_data)
```

Como se ve, hemos definido la función «**carga_clave()**» del mismo modo en que lo hicimos en el primer ejemplo. No obstante la cosa cambia con respecto a la función de encriptado. Lo cual, es debido a que, previo proceso de encriptado, debemos empezar abriendo el documento (para lo cual escribimos «**with open(nom_archivo,»rb») as file:**») y leerlo («**file.read()**»). La información leída de ese modo (variable «**archivo_info**») es la que en el paso siguiente encriptaremos mediante el método «**encrypt()**» que vimos en el ejemplo anterior. El contenido encriptado es el que almacenaremos en la variable «**encrypted_data**». Una vez obtenido el contenido encriptado, pasaremos a escribirlo en nuestro archivo original, «**nom_archivo**», sustituyendo al texto original («**file.write(encrypted_data)**»).

Para el caso de la función de desencriptado, usaremos exactamente el mismo procedimiento, solo que usando el método «**decrypt()**» (para desencriptar) en lugar de «**encrypt()**»:

```

>>> from cryptography.fernet import Fernet
>>>
>>> #FUNCION PARA CARGAR CLAVE.
>>> def carga_clave():
        return open("clave.key", "rb").read()

>>> #FUNCION PARA ENCRIPTAR ARCHIVO.
>>> def encript(nom_archivo,clave):
        f = Fernet(clave)
        with open(nom_archivo, "rb") as file:
            archivo_info = file.read()
        encrypted_data = f.encrypt(archivo_info)
        with open(nom_archivo, "wb") as file:
            file.write(encrypted_data)

>>> #FUNCION PARA DESENCRIPTAR ARCHIVO.
>>> def desencrypt(nom_archivo,clave):
        f = Fernet(clave)
        with open(nom_archivo, "rb") as file:
            encrypted_data = file.read()
        decrypted_data = f.decrypt(encrypted_data)
        with open(nom_archivo, "wb") as file:
            file.write(decrypted_data)

```

A continuación aplicaremos nuestras funciones sobre el referido documento «texto.txt»:


```

>>> #CARAGAMOS CLAVE.
>>> clave = carga_clave()
>>>
>>> #ARCHIVO A ENCRIPTAR.
>>> nom_archivo = "texto.txt"
>>>
>>> #ENCRIPTAR ARCHIVO.
>>> encript(nom_archivo,clave)

```

Así, lo primero que hacemos es definir la variable «clave», cuyo valor será el resultado de la ejecución de la función «carga_clave()» vista con anterioridad. Tras ello, introducimos el nombre del documento de texto cuya información queremos encriptar, (variable «nom_archivo»). Finalmente, para el proceso de encriptado del

contenido, llamaremos a la función «**encrypt()**», definida más arriba, introduciéndole el nombre del documento de texto («**nom_archivo**») y la clave a emplear («**clave**»). Una vez hecho esto, si miramos al contenido de nuestro documento, podremos ver como ha sido cifrado el texto original:

 texto: Bloc de notas


Archivo Edición Formato Ver Ayuda

gAAAAABeXRRtsx4qE0017Ukj60gZxFpZCh28Y7B4Zu6hurhYqUS587qWjH6YbAzWtG0YZk1CQo0NUknkFvFfPFhkn5keQqnLJHmZX2TN1QCY4aKP41JLPdE=

Para recuperar el contenido original, procederemos con a ejecutar la función para descryptar el contenido (aquella a la que llamamos «**desencrypt()**»), pasandole, nuevamente, el nombre de nuestro documento de texto y la clave (ya que, como recordamos, nos vale tanto para encriptar como para descryptar la información):

```
>>> #CARAGAMOS CLAVE.
>>> clave = carga_clave()
>>>
>>> #ARCHIVO A ENCRYPTAR.
>>> nom_archivo = "texto.txt"
>>>
>>> #ENCRYPTAR ARCHIVO.
>>> encrypt(nom_archivo,clave)
>>>
>>> #DESCRYPTAR ARCHIVO.
>>> desencrypt(nom_archivo,clave)
```

Hecha esta última operación, comprobaremos en nuestro documento, aparece, ahora, el texto original:

 texto: Bloc de notas

Archivo Edición Formato Ver Ayuda

este es un texto top secret

Como se ve, «**cryptography**» constituye una interesante y efectiva herramienta, que podemos usar en nuestros proyectos y a través de la cual, podremos proteger nuestros archivos e información más valiosa de eventuales intromisiones de terceros.

Podéis ver el código de la primera parte de este tutorial en el siguiente enlace:

<https://github.com/antonioam82/ejercicios-python/blob/master/encryption.py>
(<https://github.com/antonioam82/ejercicios-python/blob/master/encryption.py>).

Saludos.

Etiquetado:

algoritmos,
criptografía,
encriptado,
programación en python,
programacion informatica,
python,
python3.7,
software,
software libre



Publicado por programacionpython80889555

[Ver todas las entradas de programacionpython80889555](#)

4 comentarios sobre “ENCRIPTAR Y DESENCRIPTAR DATOS EN PYTHON, CON «cryptography».”

Facu dice:

octubre 12, 2020 a las 3:21 am

Copié el código y la parte de desencript no funciona.

Lo que pasó fue lo siguiente: Encripte un txt y efectivamente lo hizo.

Cuando quise utilizar la función de desencript para desencriptarlo, da error.

Si ejecutamos ambos codigos(encriptar y desencriptar) al mismo tiempo (cosa que no tiene mucho sentido, porque el contenido ya fue encriptado) funciona bien, devolviendo el primer encriptado. Se entiende?

↪ Responder

[programacionpython80889555](#) dice:

octubre 12, 2020 a las 6:56 am

Hola, acabo de repetir el ejercicio,(con el que pretendo mostrar como se puede encriptar un texto usando una clave y como realizar la acción inversa usando la misma clave) encriptando el archivo de texto, y desencriptándolo después y no me da ningún problema, ¿podría decirme que error te sale?

Saludos.

↪ Responder

Hellaheim dice:

noviembre 27, 2022 a las 3:25 pm

Hola a mi me sale un error en el apartado de f.encrypt el error me dice que no esta definido

[programacionpython80889555](#) dice:

noviembre 27, 2022 a las 4:07 pm

Asegúrese de haber hecho las importaciones necesarias y de haber declarado la variable «f» como se muestra aquí:

<https://github.com/antonioam82/ejercicios-python/blob/master/encryption.py>

Si le sigue dando error, dígame que es lo que le sale en el mensaje.

