Introducción a la Programación con PYTHON

José Rodriguez

Unidad 10 - Criptografía

UNIDAD 10 – Criptografía

El módulo hashlib define una interfaz de programación para acceder a diferentes algoritmos criptográficos de hash. Para trabajar con un algoritmo específico de hash, usa la función constructora apropiada o new() para crear un objeto hash. A partir de ahí, los objetos utilizan la misma interfaz de programación, sin importar qué algoritmo se esté utilizando.

El método

hashlib.algorithms_guaranteed

devuelve una lista con los métodos de encriptación garantizados

```
import hashlib
print(hashlib.algorithms_guaranteed)
```

dando como resultado

'sha256', 'sha1', 'blake2b', 'blake2s', 'sha3_256', 'md5', 'sha512', 'shake_256', 'sha3_384', 'shake_128', 'sha384', 'sha3_512'}

Mientras que el método

hashlib.algorithms_available

devuelve todos los algoritmos disponibles en Python entre los que se encuentran, como es lógico, los citados anteriormente.

El algoritmo más usado es MD5. Se trata de un algoritmo que recibe como entrada una cadena de caracteres y devuelve una cadena de 128 bits que es la huella digital de la secuencia de entrada. Se utiliza normalmente para asegurar la integridad de un envío a través de la red y también para codificar las contraseñas en muchas aplicaciones.

Dispone de tres funciones básicas:

- encode() : convierte la cadena de entrada en una secuencia de bytes previa a la codificación.
- digest() : Devuelve la cadena codificada en formato byte.

- hexdigest() : Retorna la cadena codificada en formato hexadecimal

```
import hashlib
cadena="pepe"
result = hashlib.md5(cadena.encode())
print(result.hexdigest())
```

El resultado es

926e27eecdbc7a18858b3798ba99bddd

Con el algoritmo sha1 tendremos

```
import hashlib
cadena="pepe"
result = hashlib.sha1(cadena.encode())
print(result.hexdigest())
```

265392dc2782778664cc9d56c8e3cd9956661bb0