Report
Mongo Database
Using python

Pubudu Tharuka Costa

HDSE18.2f-037

What I am using

NetBeans Idle

Mongo Database Windows Version

MongoDB plugin for NetBeans

Python plugin for NetBeans

Hashing Strings with Python (hashing library)

NetBeans IDLE

NetBeans is a very friendly IDLE for me then I was referred how to add python to NetBeans. NetBeans version 8.2 was supported the python and new version of NetBeans does not support the python

And using mongo DB plugin NetBeans can control and edit the Database using graphical interface

What are the Errors and How to Fix

How to import Library to the NetBeans

>>PIP INSTALL command not affected for the NetBeans

After the mongo DB installation, we should need to install the pip of pymongo this task can do by using python IDLE but It was not affected for the NetBeans python platform source path then we need to select the part tool > python platform > source path > and set to C:\Users\Pubba\AppData\Roaming\NetBeans\8.2\jython-2.7.0\Lib\site-packages

Hashing

A hash function is a function that takes input of a variable length sequence of bytes and converts it to a fixed length sequence. It is a one way function. This means if *f* is the hashing function, calculating *f(x)* is pretty fast and simple, but trying to obtain *x* again will take years. The value returned by a hash function is often called a hash, message digest, hash value, or checksum. Most of the time a hash function will produce unique output for a given input. However depending on the algorithm, there is a possibility to find a collision due to the mathematical theory behind these functions.

- **MD5**: Message digest algorithm producing a 128-bit hash value. This is widely used to check data integrity. It is not suitable for use in other fields due to the security vulnerabilities of MD5.
- **SHA:** Group of algorithms designed by the U. S's NSA that are part of the U.S Federal Information processing standard. These algorithms are used widely in several cryptographic applications. The message length ranges from 160 bits to 512 bits.

The `hashlib` module, included in The Python Standard library is a module containing an interface to the most popular hashing algorithms. `hashlib` implements some of the algorithms, however if you have OpenSSL installed, `hashlib` is able to use this algorithm as well.

First, import the `hashlib` module:

```
1 import hashlib
```

Now we use `algorithms_available` or `algorithms_guaranteed` to list the algorithms available.

```
1 print(hashlib.algorithms_available)
2 print(hashlib.algorithms_guaranteed)
```

The `algorithms_available` method lists all the algorithms available in the system, including the ones available trough OpenSSl. In this case you may see duplicate names in the list. `algorithms_guaranteed` only lists the algorithms present in the module. `md5, sha1, sha224, sha256, sha384, sha512` are always present.

MD5

```
import hashlib
hash_object = hashlib.md5(b'Hello World')
print(hash_object.hexdigest())
```

The code above takes the "Hello World" string and prints the HEX digest of that string. `hexdigest` returns a HEX string representing the hash, in case you need the sequence of bytes you should use `digest` instead.

It is important to note the "b" preceding the string literal, this converts the string to bytes, because the hashing function only takes a sequence of bytes as a parameter. In previous versions of the library, it used to take a string literal. So, if you need to take some input from the console, and hash this input, do not forget to encode the string in a sequence of bytes:

```
import hashlib
mystring = input('Enter String to hash: ')
# Assumes the default UTF-8
hash_object = hashlib.md5(mystring.encode())
print(hash_object.hexdigest())
```

SHA1

```
import hashlib
hash_object = hashlib.sha1(b'Hello World')
hex_dig = hash_object.hexdigest()
print(hex_dig)
```

SHA224

```
import hashlib
hash_object = hashlib.sha224(b'Hello World')
hex_dig = hash_object.hexdigest()
print(hex_dig)
```

SHA256

```
1 import hashlib
2 hash_object = hashlib.sha256(b'Hello World')
3 hex_dig = hash_object.hexdigest()
4 print(hex_dig)
```

SHA384

```
1 import hashlib
2 hash_object = hashlib.sha384(b'Hello World')
3 hex_dig = hash_object.hexdigest()
4 print(hex_dig)
```

SHA512

```
1 import hashlib
2 hash_object = hashlib.sha512(b'Hello World')
3 hex_dig = hash_object.hexdigest()
4 print(hex_dig)
```

Using OpenSSL Algorithms

Now suppose you need an algorithm provided by OpenSSL. Using `algorithms_available`, we can find the name of the algorithm you want to use. In this case, "DSA" is available on my computer. You can then use the `new` and `update` methods:

```
1 import hashlib
2 hash_object = hashlib.new('DSA')
3 hash_object.update(b'Hello World')
4 print(hash_object.hexdigest())
```

Practical example: hashing passwords

In the following example we are hashing a password in order to store it in a database. In this example we are using a salt. A salt is a random sequence added to the password string before using the hash function. The salt is used in order to prevent dictionary attacks and rainbow tables attacks. However, if you are making real world applications and working with users' passwords, make sure to be updated about the latest vulnerabilities in this field. I you want to find out more about secure passwords please refer to this article

EX :

```
1  import uuid
2  import hashlib
3
4  def hash_password(password):
5      # uuid is used to generate a random number
6      salt = uuid.uuid4().hex
7      return hashlib.sha256(salt.encode() + password.encode()).hexdigest() + ':' + salt
8
9  def check_password(hashed_password, user_password):
10     password, salt = hashed_password.split(':')
11     return password == hashlib.sha256(salt.encode() + user_password.encode()).hexdigest()
12
13 new_pass = input('Please enter a password: ')
14 hashed_password = hash_password(new_pass)
15 print('The string to store in the db is: ' + hashed_password)
16 old_pass = input('Now please enter the password again to check: ')
17 if check_password(hashed_password, old_pass):
18     print('You entered the right password')
19 else:
20     print('I am sorry but the password does not match')
```

ASSIGMENT

```
import hashlib
from pymongo import MongoClient
username = input("Enter Your Username")
password = input("Enter Your password")
sign = input("press 2 for create account")
if sign !=2:
#hash_object = hashlib.md5(password)
hash_object = hashlib.md5(password.encode())
password2 = hash_object.hexdigest()
print('The string to store in the db is: ' + password2)
Client = MongoClient()
db = Client["Login"]
collection = db["user"]
user ={}
user['userName'] = unam
user['userPassword'] =upas
if user['userName'] == username and user['userPassword'] == password2:
   print("welcome")
  elif sign == 2:
      user ["userName"] = username
user ["userPassword"] = password2
collection.insert(user)
else:
    print("try agin")
    else
    print("???????")
```