CRYPTOGRAPHY AND NETWORK SECURITY

LAB 11 : SHA – 512 ALGORITHM

NAME : OM SUBRATO DEY

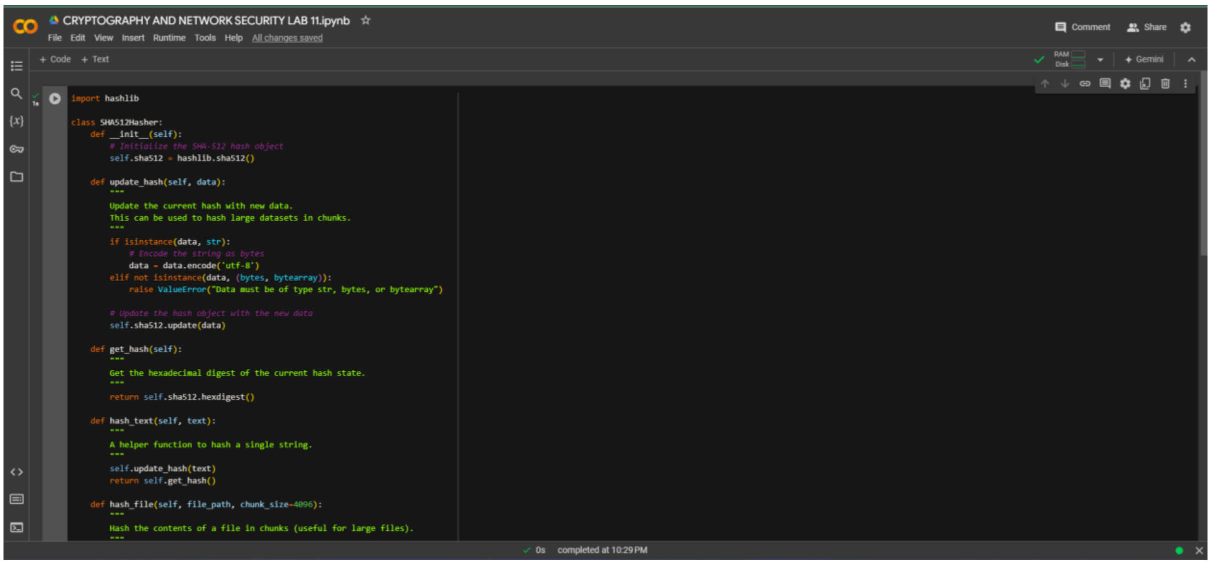
REGISTER NO. : 21BAI1876

CODE:

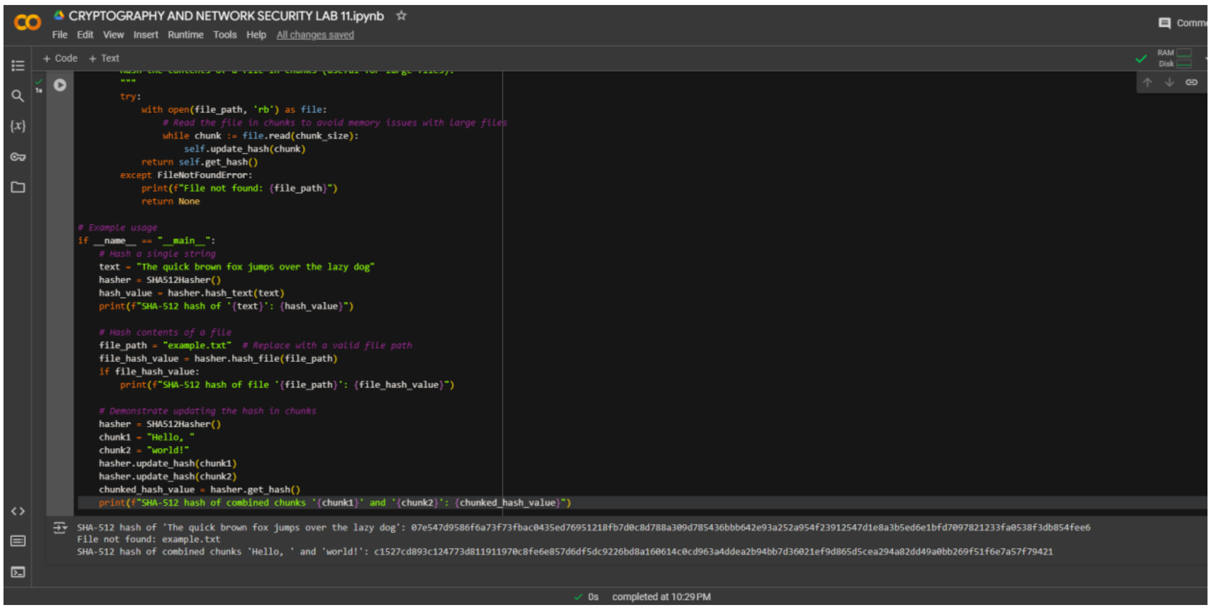
```
import hashlib
class SHA512Hasher:
    def __init__(self):
        # Initialize the SHA-512 hash object
        self.sha512 = hashlib.sha512()

    def update_hash(self, data):
        """
        Update the current hash with new data.
        This can be used to hash large datasets in
chunks.
        """
        if isinstance(data, str):
            # Encode the string as bytes
            data = data.encode('utf-8')
        elif not isinstance(data, (bytes, bytearray)):
            raise ValueError("Data must be of type str,
bytes, or bytearray")

        # Update the hash object with the new data
        self.sha512.update(data)

    def get_hash(self):
        """
        Get the hexadecimal digest of the current hash
state.
```

```
        """
        return self.sha512.hexdigest()

    def hash_text(self, text):
        """
        A helper function to hash a single string.
        """
        self.update_hash(text)
        return self.get_hash()

    def hash_file(self, file_path, chunk_size=4096):
        """
        Hash the contents of a file in chunks (useful for large files).
        """
        try:
            with open(file_path, 'rb') as file:
                # Read the file in chunks to avoid memory issues with large files
                while chunk := file.read(chunk_size):
                    self.update_hash(chunk)
            return self.get_hash()
        except FileNotFoundError:
            print(f"File not found: {file_path}")
            return None

# Example usage
if __name__ == "__main__":
    # Hash a single string
    text = "The quick brown fox jumps over the lazy dog"
    hasher = SHA512Hasher()
    hash_value = hasher.hash_text(text)
    print(f"SHA-512 hash of '{text}': {hash_value}")

    # Hash contents of a file
    file_path = "example.txt"  # Replace with a valid file path
    file_hash_value = hasher.hash_file(file_path)
    if file_hash_value:
        print(f"SHA-512 hash of file '{file_path}': {file_hash_value}")

    # Demonstrate updating the hash in chunks
    hasher = SHA512Hasher()
    chunk1 = "Hello, "
    chunk2 = "world!"
    hasher.update_hash(chunk1)
    hasher.update_hash(chunk2)
    chunked_hash_value = hasher.get_hash()
    print(f"SHA-512 hash of combined chunks '{chunk1}' and '{chunk2}': {chunked_hash_value}")
```

OUTPUT:


CRYPTOGRAPHY AND NETWORK SECURITY LAB 11.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share
Gemini
0s completed at 10:29 PM

```python
import hashlib

class SHA512Hasher:
    def __init__(self):
        # Initialize the SHA-512 hash object
        self.sha512 = hashlib.sha512()

    def update_hash(self, data):
        """
        Update the current hash with new data.
        This can be used to hash large datasets in chunks.
        """
        if isinstance(data, str):
            # Encode the string as bytes
            data = data.encode('utf-8')
        elif not isinstance(data, (bytes, bytearray)):
            raise ValueError("Data must be of type str, bytes, or bytearray")

        # Update the hash object with the new data
        self.sha512.update(data)

    def get_hash(self):
        """
        Get the hexadecimal digest of the current hash state.
        """
        return self.sha512.hexdigest()

    def hash_text(self, text):
        """
        A helper function to hash a single string.
        """
        self.update_hash(text)
        return self.get_hash()

    def hash_file(self, file_path, chunk_size=4096):
        """
        Hash the contents of a file in chunks (useful for large files).
        """
```


CRYPTOGRAPHY AND NETWORK SECURITY LAB 11.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
0s completed at 10:29 PM

```python
        """
        try:
            with open(file_path, 'rb') as file:
                # Read the file in chunks to avoid memory issues with large files
                while chunk := file.read(chunk_size):
                    self.update_hash(chunk)
            return self.get_hash()
        except FileNotFoundError:
            print(f"File not found: {file_path}")
            return None

# Example usage
if __name__ == "__main__":
    # Hash a single string
    text = "The quick brown fox jumps over the lazy dog"
    hasher = SHA512Hasher()
    hash_value = hasher.hash_text(text)
    print(f"SHA-512 hash of '{text}': {hash_value}")

    # Hash contents of a file
    file_path = "example.txt"  # Replace with a valid file path
    file_hash_value = hasher.hash_file(file_path)
    if file_hash_value:
        print(f"SHA-512 hash of file '{file_path}': {file_hash_value}")

    # Demonstrate updating the hash in chunks
    hasher = SHA512Hasher()
    chunk1 = "Hello, "
    chunk2 = "world!"
    hasher.update_hash(chunk1)
    hasher.update_hash(chunk2)
    chunked_hash_value = hasher.get_hash()
    print(f"SHA-512 hash of combined chunks '{chunk1}' and '{chunk2}': {chunked_hash_value}")
```

```
SHA-512 hash of 'The quick brown fox jumps over the lazy dog': 07e547d9586f6a73f73fbac0435ed76951218fb7d0c8d788a309d785436bbb642e93a252a954f23912547d1e8a3b5ed6e1bfd7097821233fa0538f3db854fee6
File not found: example.txt
SHA-512 hash of combined chunks 'Hello, ' and 'world!': c1527cd893c124773d811911970c8fe6e857d6df5dc9226bd8a160614c0cd963a4ddea2b94bb7d36021ef9d865d5cea294a82dd49a0bb269f51f6e7a57f79421
```