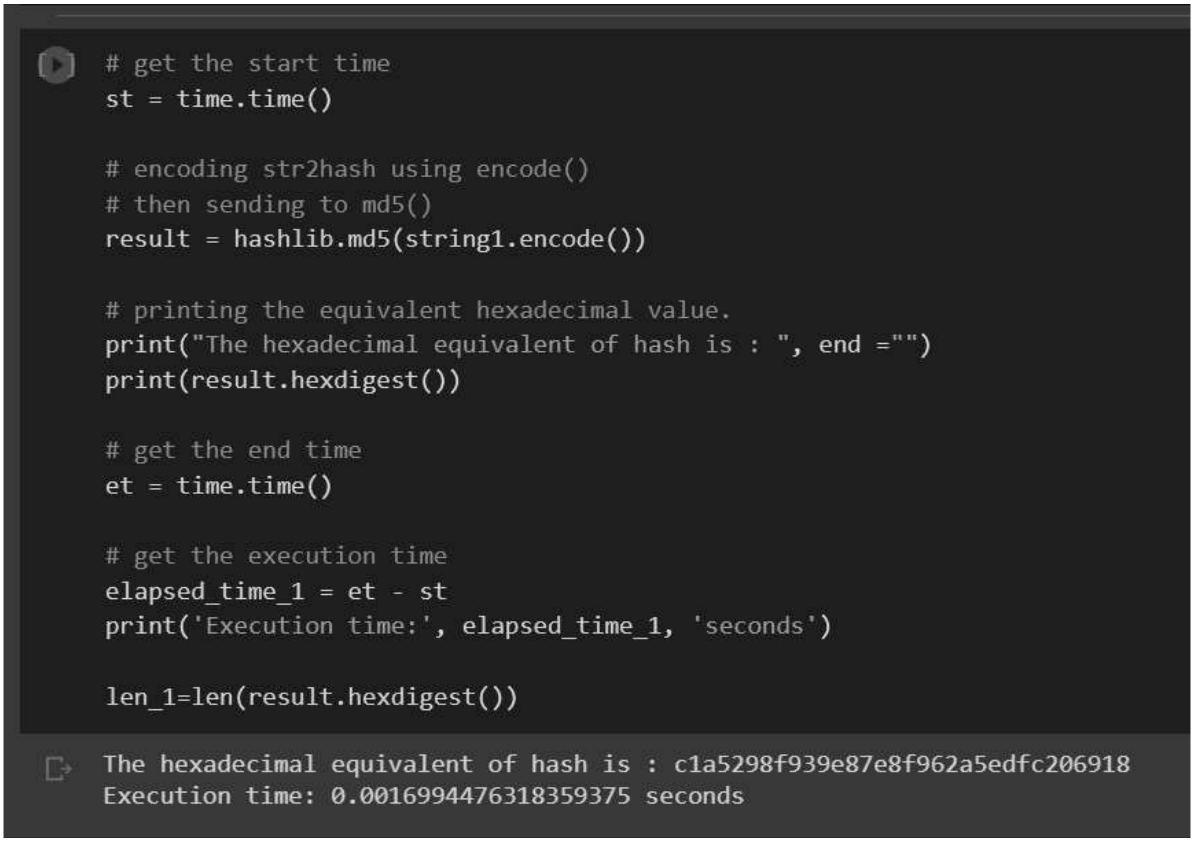
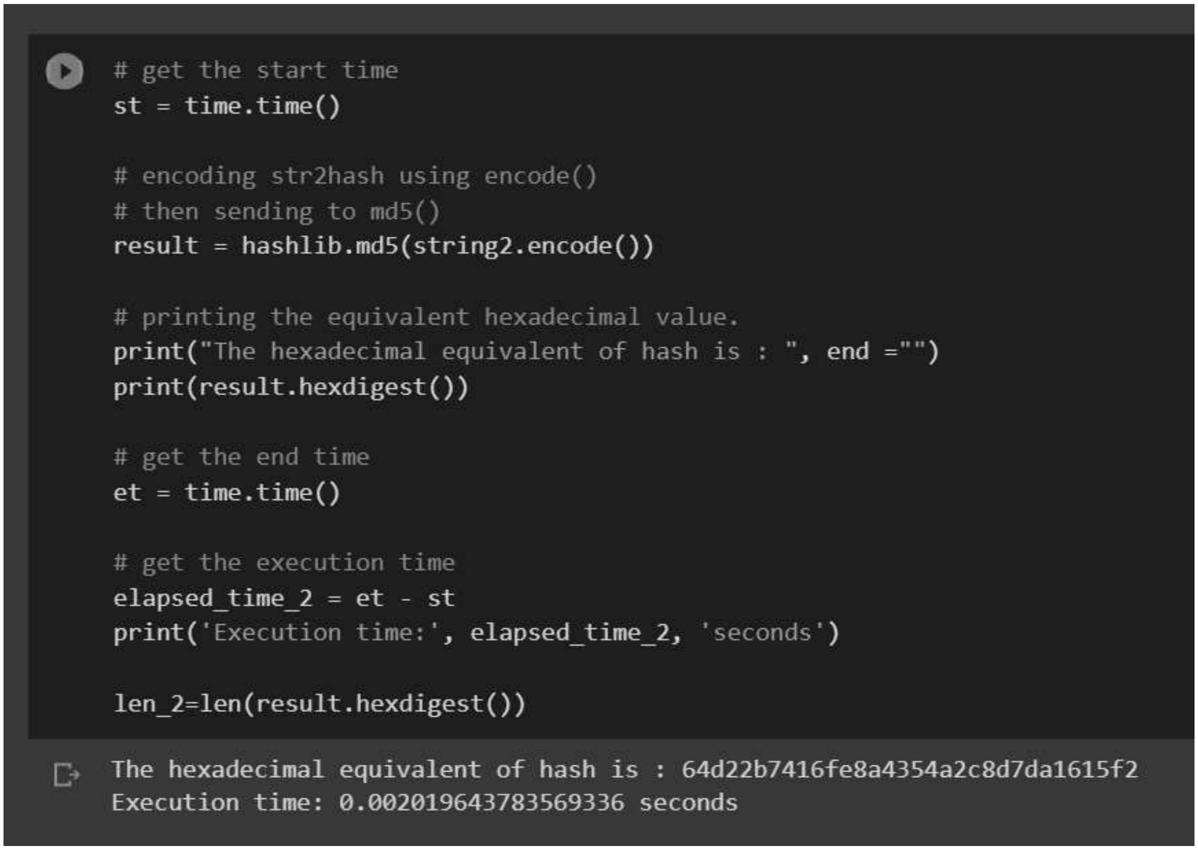


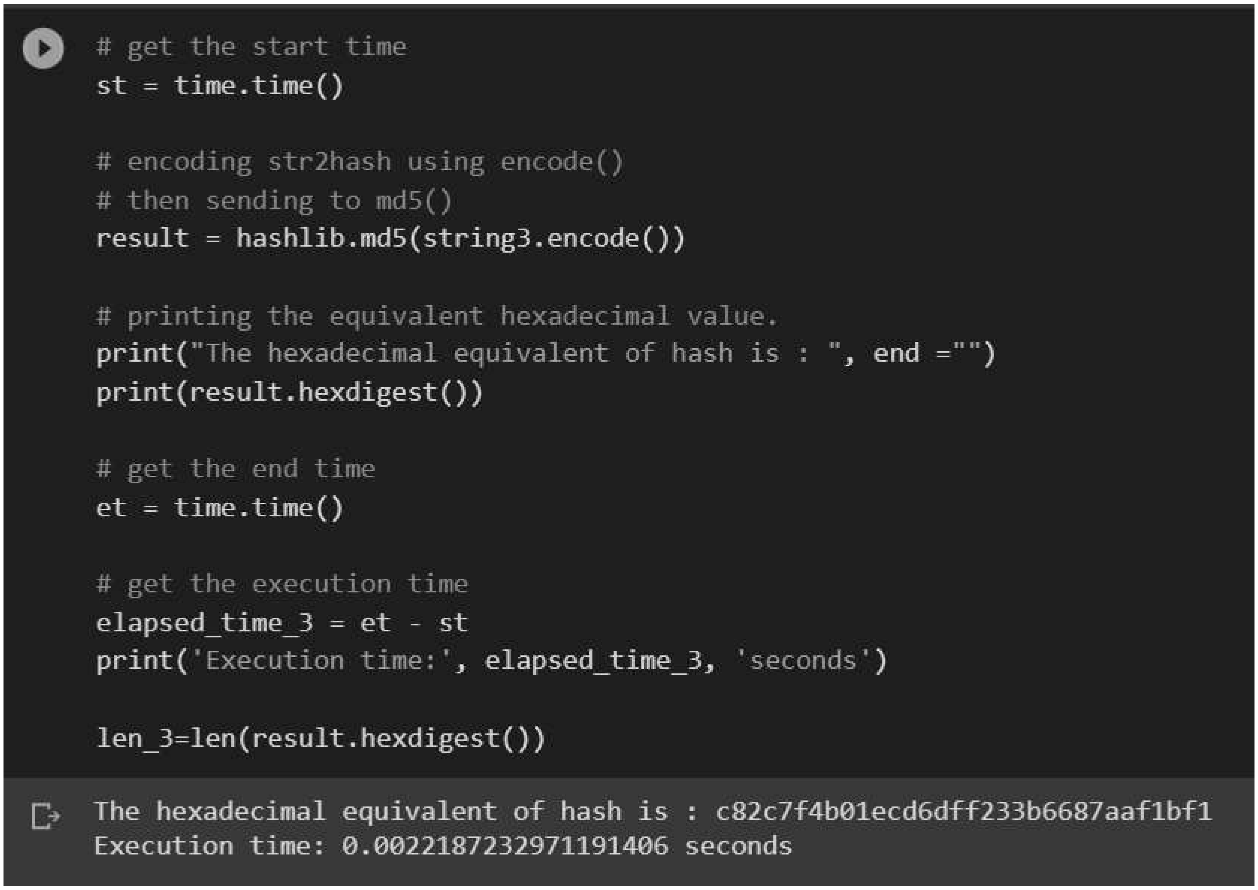
```
# get the start time
st = time.time()

# encoding str2hash using encode()
# then sending to md5()
result = hashlib.md5(string1.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of hash is : ", end ="")
print(result.hexdigest())

# get the end time
et = time.time()

# get the execution time
elapsed_time_1 = et - st
print('Execution time:', elapsed_time_1, 'seconds')

len_1=len(result.hexdigest())
```

```
The hexadecimal equivalent of hash is : c1a5298f939e87e8f962a5edfc206918
Execution time: 0.0016994476318359375 seconds
```



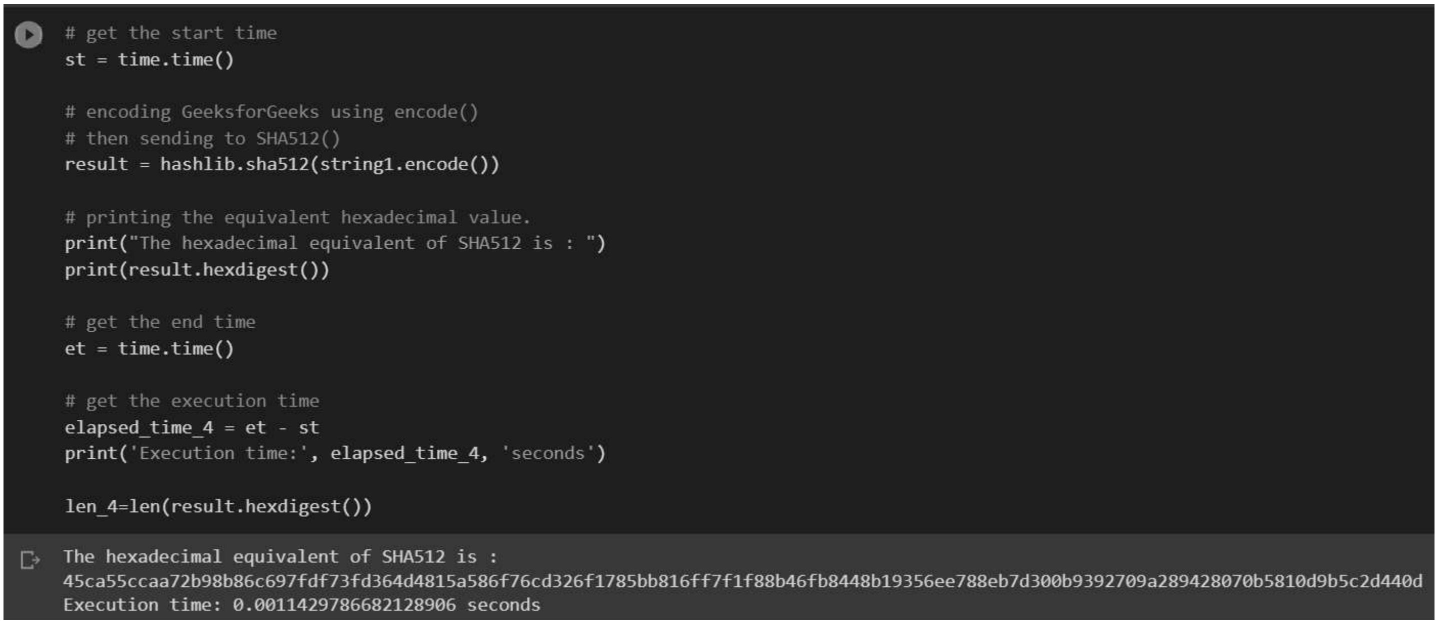
```
# get the start time
st = time.time()

# encoding str2hash using encode()
# then sending to md5()
result = hashlib.md5(string2.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of hash is : ", end ="")
print(result.hexdigest())

# get the end time
et = time.time()

# get the execution time
elapsed_time_2 = et - st
print('Execution time:', elapsed_time_2, 'seconds')

len_2=len(result.hexdigest())
```

```
The hexadecimal equivalent of hash is : 64d22b7416fe8a4354a2c8d7da1615f2
Execution time: 0.002019643783569336 seconds
```



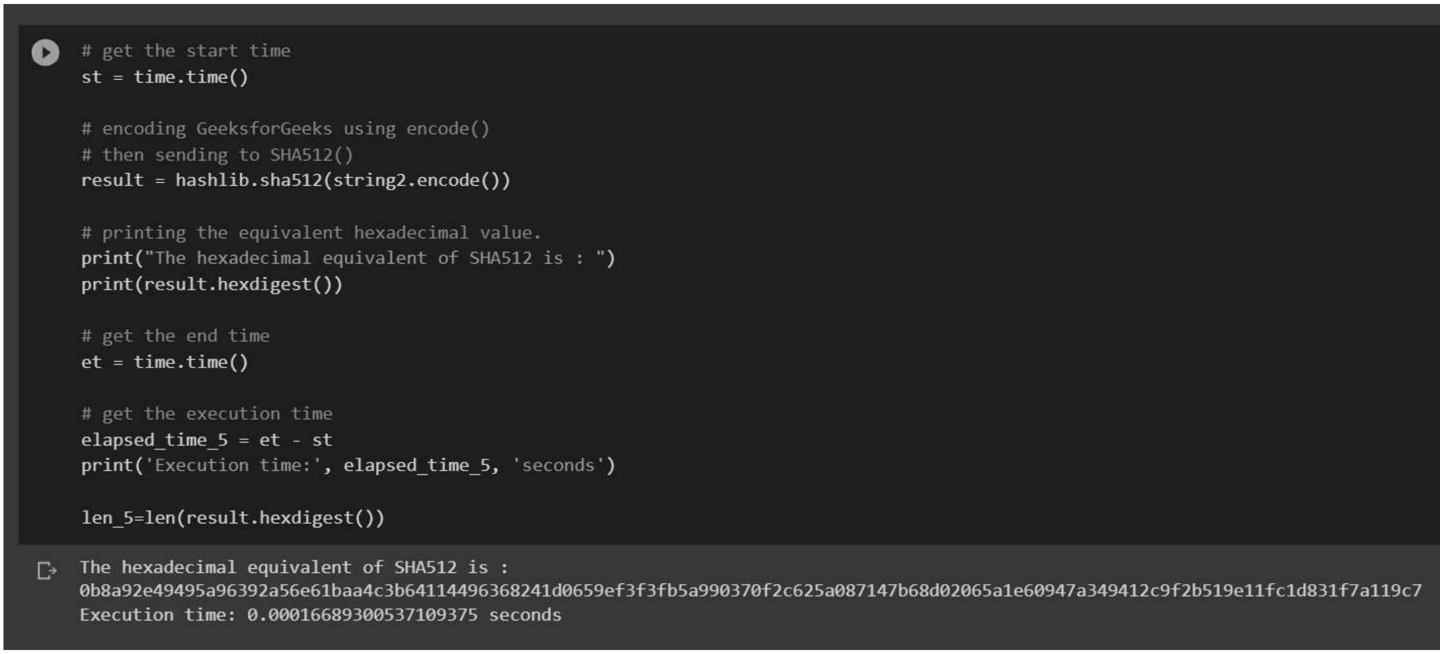
```python
# get the start time
st = time.time()

# encoding str2hash using encode()
# then sending to md5()
result = hashlib.md5(string3.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of hash is : ", end ="")
print(result.hexdigest())

# get the end time
et = time.time()

# get the execution time
elapsed_time_3 = et - st
print('Execution time:', elapsed_time_3, 'seconds')

len_3=len(result.hexdigest())
```

```
The hexadecimal equivalent of hash is : c82c7f4b01ecd6dff233b6687aaf1bf1
Execution time: 0.0022187232971191406 seconds
```



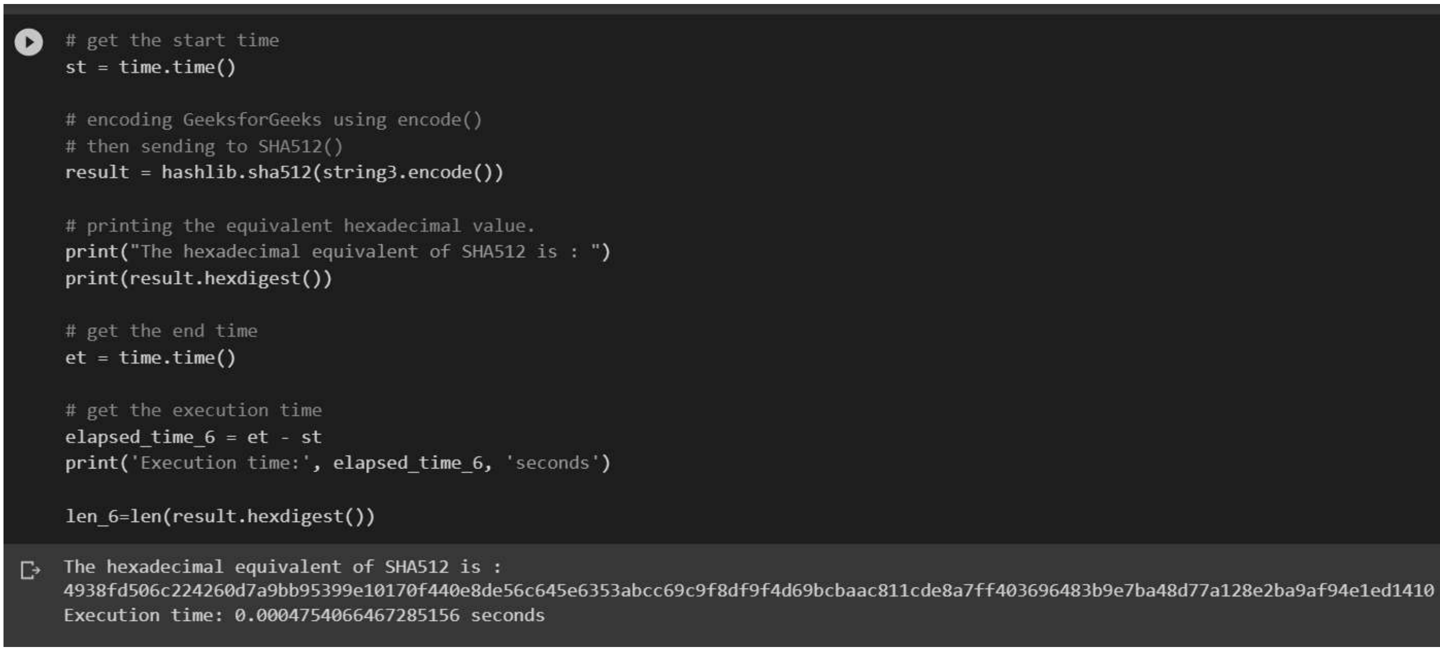
```python
# get the start time
st = time.time()

# encoding GeeksforGeeks using encode()
# then sending to SHA512()
result = hashlib.sha512(string1.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())

# get the end time
et = time.time()

# get the execution time
elapsed_time_4 = et - st
print('Execution time:', elapsed_time_4, 'seconds')

len_4=len(result.hexdigest())
```

```
The hexadecimal equivalent of SHA512 is : 
45ca55ccaa72b98b86c697fdf73fd364d4815a586f76cd326f1785bb816ff7f1f88b46fb8448b19356ee788eb7d300b9392709a289428070b5810d9b5c2d440d
Execution time: 0.0011429786682128906 seconds
```



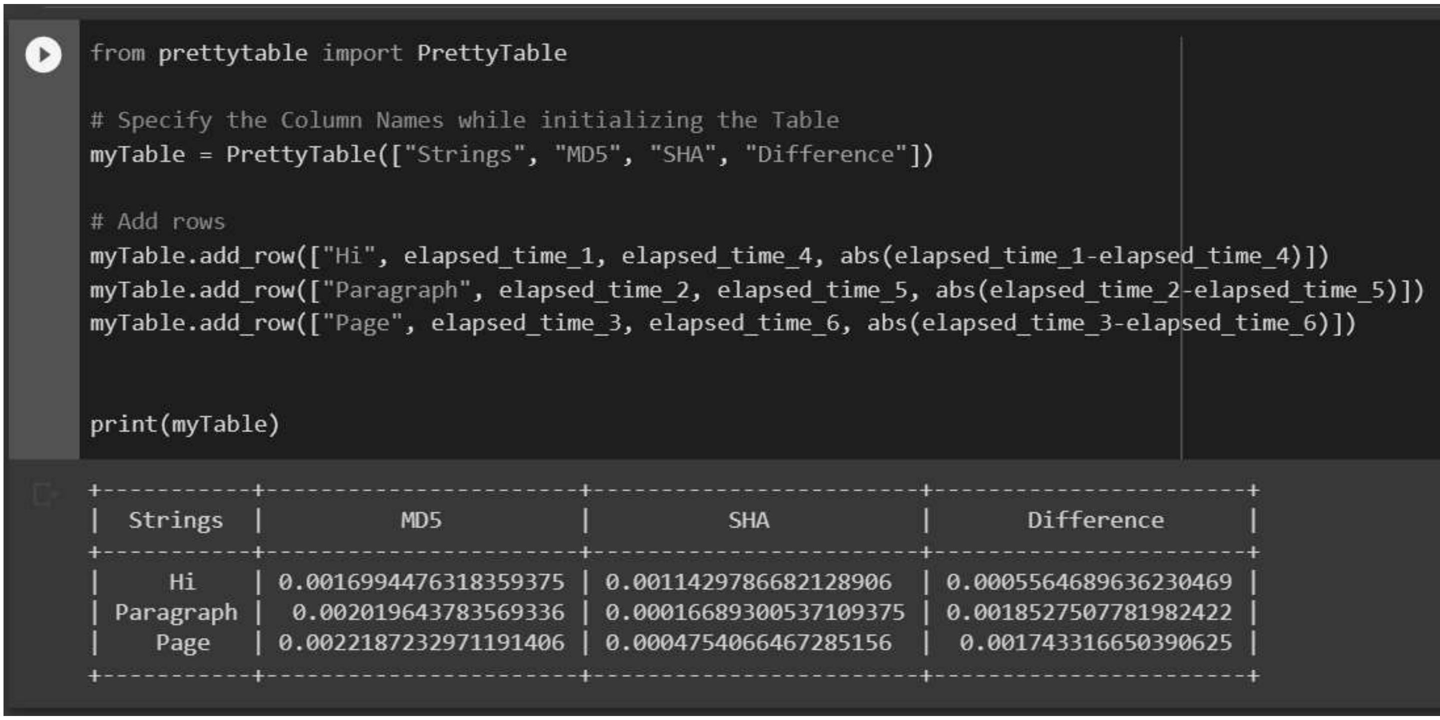
```
# get the start time
st = time.time()

# encoding GeeksforGeeks using encode()
# then sending to SHA512()
result = hashlib.sha512(string2.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())

# get the end time
et = time.time()

# get the execution time
elapsed_time_5 = et - st
print('Execution time:', elapsed_time_5, 'seconds')

len_5=len(result.hexdigest())
```

```
The hexadecimal equivalent of SHA512 is :
0b8a92e49495a96392a56e61baa4c3b64114496368241d0659ef3f3fb5a990370f2c625a087147b68d02065a1e60947a349412c9f2b519e11fc1d831f7a119c7
Execution time: 0.00016689300537109375 seconds
```



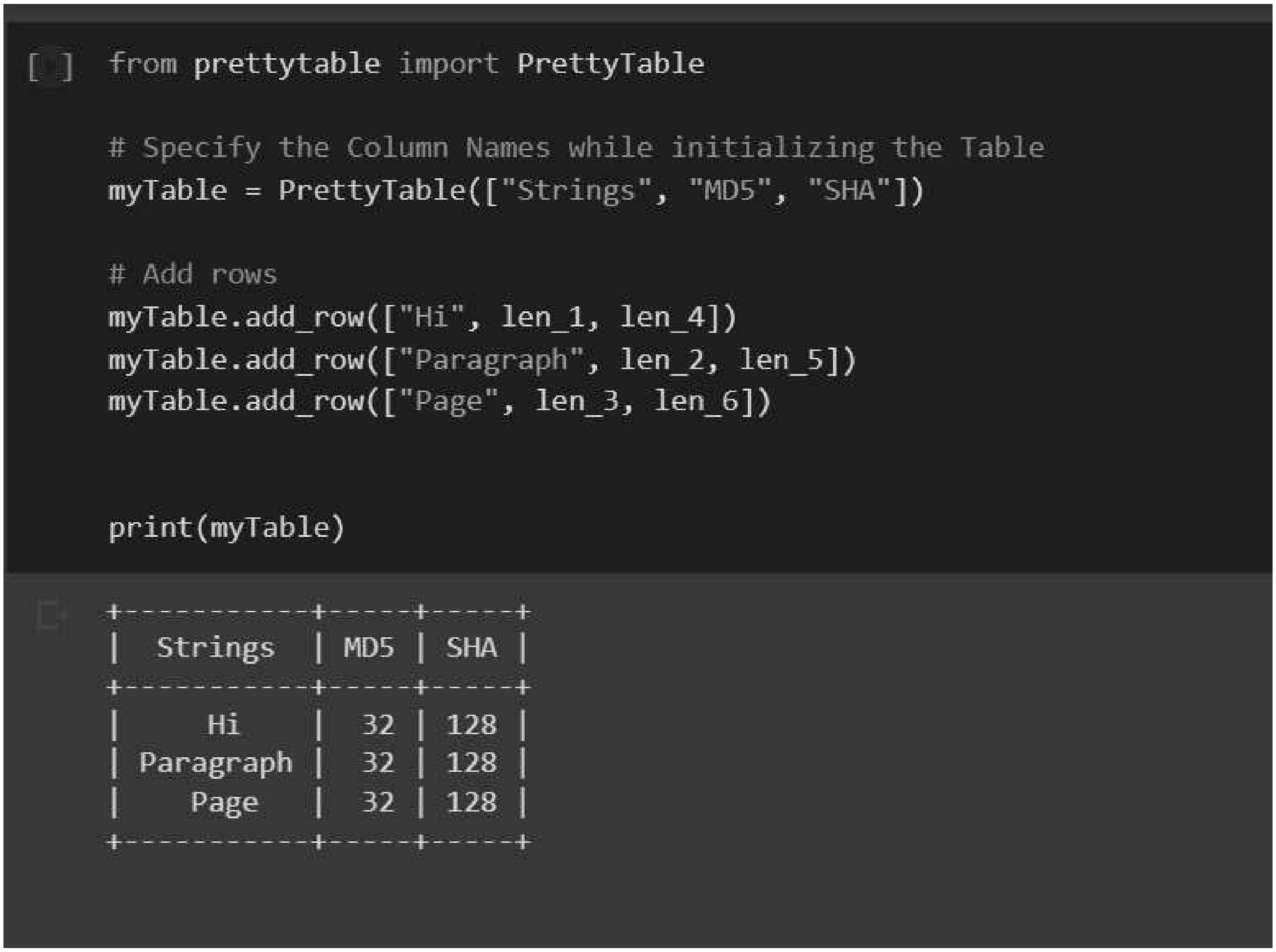
```
# get the start time
st = time.time()

# encoding GeeksforGeeks using encode()
# then sending to SHA512()
result = hashlib.sha512(string3.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())

# get the end time
et = time.time()

# get the execution time
elapsed_time_6 = et - st
print('Execution time:', elapsed_time_6, 'seconds')

len_6=len(result.hexdigest())
```

```
The hexadecimal equivalent of SHA512 is :
4938fd506c224260d7a9bb95399e10170f440e8de56c645e6353abcc69c9f8df9f4d69bcbaac811cde8a7ff403696483b9e7ba48d77a128e2ba9af94e1ed1410
Execution time: 0.0004754066467285156 seconds
```



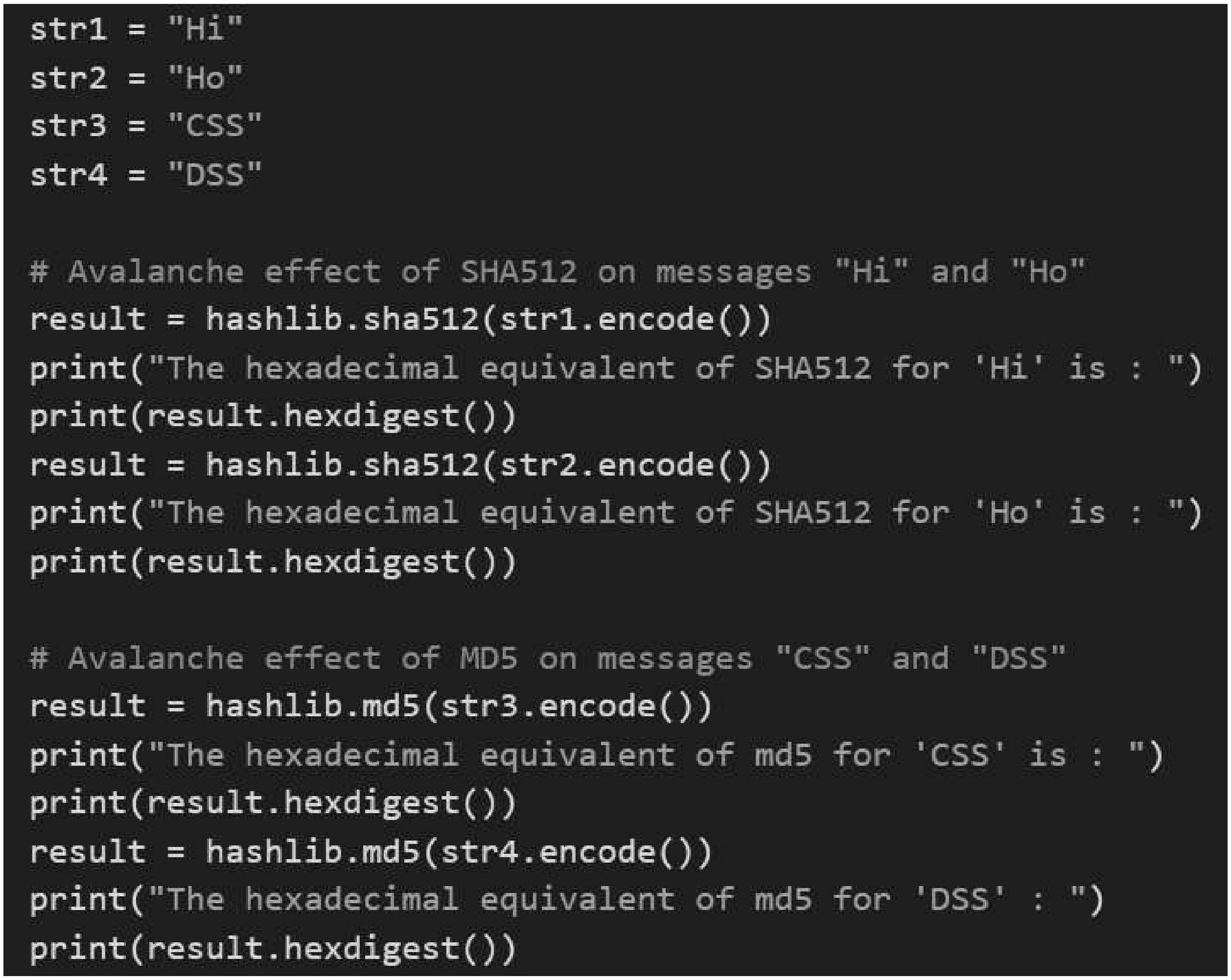
```
from prettytable import PrettyTable

# Specify the Column Names while initializing the Table
myTable = PrettyTable(["Strings", "MD5", "SHA", "Difference"])

# Add rows
myTable.add_row(["Hi", elapsed_time_1, elapsed_time_4, abs(elapsed_time_1-elapsed_time_4)])
myTable.add_row(["Paragraph", elapsed_time_2, elapsed_time_5, abs(elapsed_time_2-elapsed_time_5)])
myTable.add_row(["Page", elapsed_time_3, elapsed_time_6, abs(elapsed_time_3-elapsed_time_6)])


print(myTable)
```

```
+-----------+-----------------------+------------------------+-----------------------+
|  Strings  |          MD5          |          SHA           |       Difference      |
+-----------+-----------------------+------------------------+-----------------------+
|     Hi    | 0.0016994476318359375 | 0.0011429786682128906  | 0.0005564689636230469 |
| Paragraph |  0.002019643783569336 | 0.00016689300537109375 | 0.0018527507781982422 |
|    Page   | 0.0022187232971191406 | 0.0004754066467285156  |  0.001743316650390625 |
+-----------+-----------------------+------------------------+-----------------------+
```

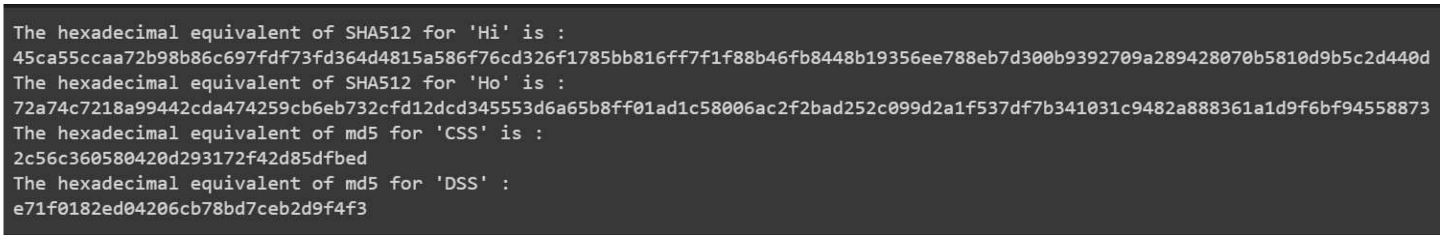


```
from prettytable import PrettyTable

# Specify the Column Names while initializing the Table
myTable = PrettyTable(["Strings", "MD5", "SHA"])

# Add rows
myTable.add_row(["Hi", len_1, len_4])
myTable.add_row(["Paragraph", len_2, len_5])
myTable.add_row(["Page", len_3, len_6])


print(myTable)
```

```
+-----------+-----+-----+
|  Strings  | MD5 | SHA |
+-----------+-----+-----+
|     Hi    |  32 | 128 |
| Paragraph |  32 | 128 |
|    Page   |  32 | 128 |
+-----------+-----+-----+
```



```python
str1 = "Hi"
str2 = "Ho"
str3 = "CSS"
str4 = "DSS"

# Avalanche effect of SHA512 on messages "Hi" and "Ho"
result = hashlib.sha512(str1.encode())
print("The hexadecimal equivalent of SHA512 for 'Hi' is : ")
print(result.hexdigest())
result = hashlib.sha512(str2.encode())
print("The hexadecimal equivalent of SHA512 for 'Ho' is : ")
print(result.hexdigest())

# Avalanche effect of MD5 on messages "CSS" and "DSS"
result = hashlib.md5(str3.encode())
print("The hexadecimal equivalent of md5 for 'CSS' is : ")
print(result.hexdigest())
result = hashlib.md5(str4.encode())
print("The hexadecimal equivalent of md5 for 'DSS' : ")
print(result.hexdigest())
```



```
The hexadecimal equivalent of SHA512 for 'Hi' is :
45ca55ccaa72b98b86c697fdf73fd364d4815a586f76cd326f1785bb816ff7f1f88b46fb8448b19356ee788eb7d300b9392709a289428070b5810d9b5c2d440d
The hexadecimal equivalent of SHA512 for 'Ho' is :
72a74c7218a99442cda474259cb6eb732cfd12dcd345553d6a65b8ff01ad1c58006ac2f2bad252c099d2a1f537df7b341031c9482a888361a1d9f6bf94558873
The hexadecimal equivalent of md5 for 'CSS' is :
2c56c360580420d293172f42d85dfbed
The hexadecimal equivalent of md5 for 'DSS' :
e71f0182ed04206cb78bd7ceb2d9f4f3
```