Note: My program takes input from user using input() method. So, after running the aestest file it will ask for input string to be entered.

aestest:

Takes input string from user and generates a new key using secrets library.

Then it converts the hexadecimal key into array (matrix) of hexadecimal values of the key and passes the plaintext and the hexadecimal key array to encryption function in aesencrypt file.

The aesencrypt file prints the ciphertext and returns the ciphertext to aestest file.

Then aestest file passes the returned ciphertext to aesdecrypt along with the generated hexadecimal key array.

aesencrypt:

Converts the plaintext string into an array of hexadecimal values and calls key expansion function and generates an array of long (11 words) key from the input key that will be used in every round for encryption.

Also, while converting the input string to hex, it will check for the length of the input whether it is 128 bits or not and will apply padding accordingly. And it will generate list of matrices of 128 bits of hexadecimal plaintext blocks.

Then first word of expanded key (128 bits) which is the original key is XORed with the plaintext which generates input for first round of the cipher.

Then the function iterates through a loop for 9 times and calls 4 different functions in the loop – subBytes, shiftRows, mixColumns, addRoundKey.

- subBytes: Takes the output of the previous round and substitutes each pair of hexadecimal value with a new pair of hexadecimal value based on the sbox matrix given.
- shiftRows: Takes the output of subBytes and shifts the rows of the output matrix according to the subBytes specifications of AES.
- mixColumns: Takes the output of shiftRows and multiplies the matrix with a fixed matrix mentioned in AES encryption.
- addRoundKey: Takes the output of mixColumns and takes one word from expanded key according to the current round and XORes the two inputs to generate the final output of the round.

The final output of 9th round is the again given to subBytes function then to shiftRows and then to addRoundKey to generate the final cipher.

Then this cipher is converted to a single dimensional array of hexadecimal values and printed on the console.

Key Expansion: A 16 bytes (4 words) key is expanded to 176 bytes (44 words) key so that each round mentioned above can use different 16 bytes of the expanded key. Key expansion is done by applying functions – rotWord, subWord, and XOR with a unique round value.

- rotWord: Rotates each word of the key (4 bytes of a row) according to the AES specifications.
- subWord: Works in same way as subBytes.
- XOR with a hexadecimal number in each round.

These above functions are only applied to first word of the key.
Then the output is XORed with the output of previous round.

aesdecrypt:

It will get input from aestest, and the input will be cipher text which will be hexadecimal array, and a key generated by aestest which will be a hexadecimal array as well.

Calls key expansion function and generates an array of long (11 words) key from the input key that will be used in every round for decryption.

Then first word of expanded key (128 bits) which is the original key is XORed with the ciphertext which generates input for first round of the function.

Then the function iterates through a loop for 9 times and calls 4 different functions in the loop – invShiftRows, invSubBytes, addRoundKey, invMixColumns.

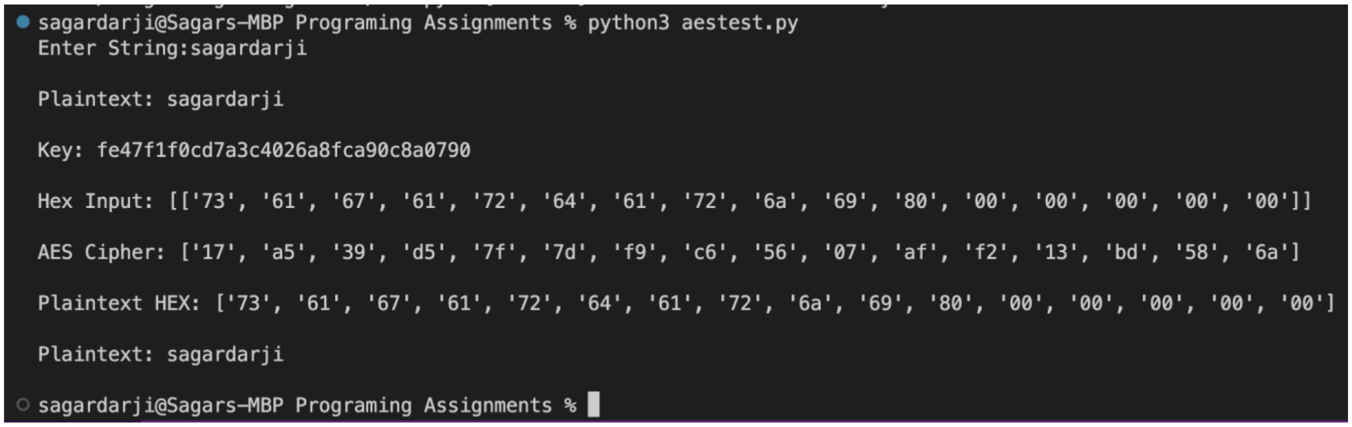
- invShiftRows: Takes the output of previous round and shifts the rows of the output matrix according to the subBytes specifications of AES for decryption (reverse from encryption).
- invSubBytes: Takes the output of the invShiftRows and substitutes each pair of hexadecimal value with a new pair of hexadecimal value based on the sbox matrix given which is inverse of the encryption.
- addRoundKey: Takes the output of invSubBytes and takes one word from expanded key according to the current round and XORes the two inputs.
- invMixColumns: Takes the output of addRoundKey and multiplies the matrix with a fixed matrix mentioned in AES decryption to generate the final output of the round.

The final output of 9th round is the again given to invShiftRows function then to invSubBytes and then to addRoundKey to generate the final decrypted plaintext.

Then this cipher is converted to a string from an array of hexadecimal values and printed on the console.

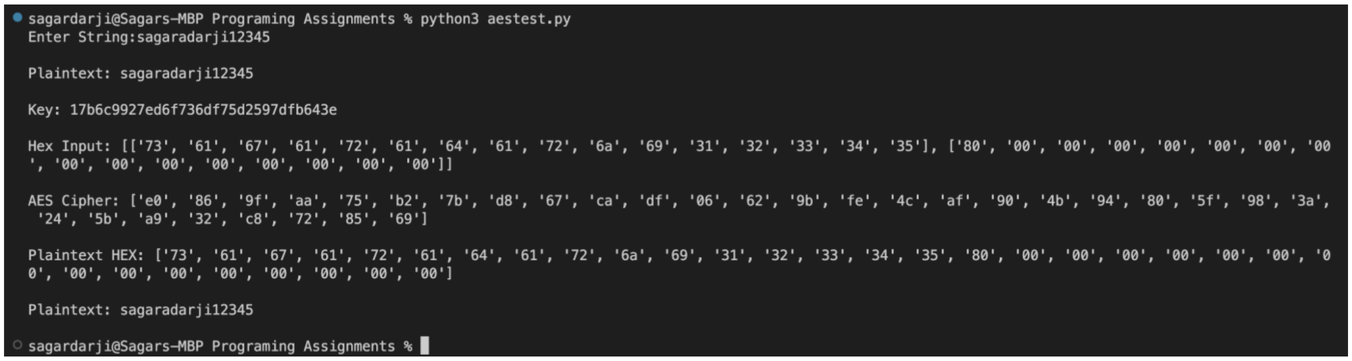
Screen Captures:

Less than 128 bits input:



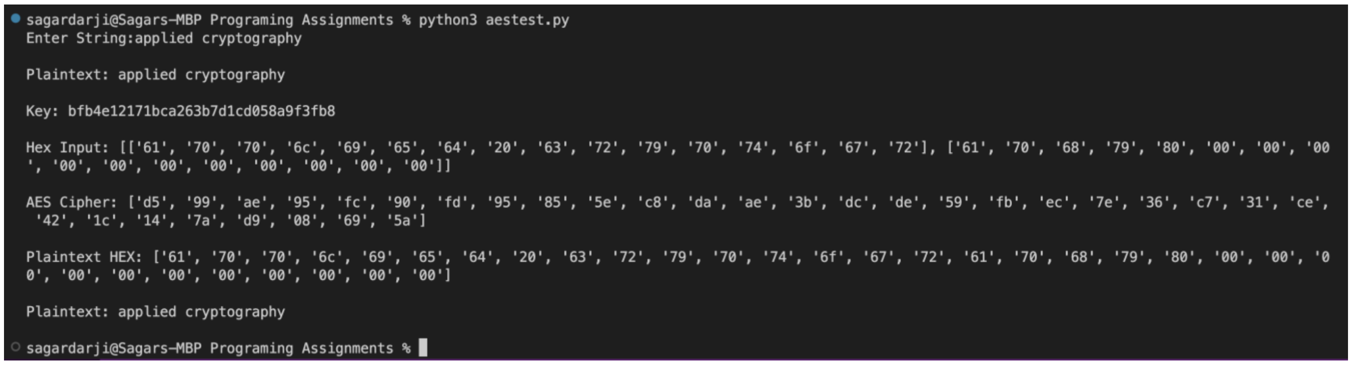
```
sagardarji@Sagars-MBP Programing Assignments % python3 aestest.py
Enter String:sagardarji

Plaintext: sagardarji

Key: fe47f1f0cd7a3c4026a8fca90c8a0790

Hex Input: [['73', '61', '67', '61', '72', '64', '61', '72', '6a', '69', '80', '00', '00', '00', '00', '00']]

AES Cipher: ['17', 'a5', '39', 'd5', '7f', '7d', 'f9', 'c6', '56', '07', 'af', 'f2', '13', 'bd', '58', '6a']

Plaintext HEX: ['73', '61', '67', '61', '72', '64', '61', '72', '6a', '69', '80', '00', '00', '00', '00', '00']

Plaintext: sagardarji

sagardarji@Sagars-MBP Programing Assignments %
```

128 bits input:



```
sagardarji@Sagars-MBP Programing Assignments % python3 aestest.py
Enter String:sagaradarji12345

Plaintext: sagaradarji12345

Key: 17b6c9927ed6f736df75d2597dfb643e

Hex Input: [['73', '61', '67', '61', '72', '61', '64', '61', '72', '6a', '69', '31', '32', '33', '34', '35'], ['80', '00', '00', '00', '00', '00', '00', '00
', '00', '00', '00', '00', '00', '00', '00', '00']]

AES Cipher: ['e0', '86', '9f', 'aa', '75', 'b2', '7b', 'd8', '67', 'ca', 'df', '06', '62', '9b', 'fe', '4c', 'af', '90', '4b', '94', '80', '5f', '98', '3a',
 '24', '5b', 'a9', '32', 'c8', '72', '85', '69']

Plaintext HEX: ['73', '61', '67', '61', '72', '61', '64', '61', '72', '6a', '69', '31', '32', '33', '34', '35', '80', '00', '00', '00', '00', '00', '00', '0
0', '00', '00', '00', '00', '00', '00', '00', '00']

Plaintext: sagaradarji12345

sagardarji@Sagars-MBP Programing Assignments %
```

More the 128 bits input:



```
sagardarji@Sagars-MBP Programing Assignments % python3 aestest.py
Enter String:applied cryptography

Plaintext: applied cryptography

Key: bfb4e12171bca263b7d1cd058a9f3fb8

Hex Input: [['61', '70', '70', '6c', '69', '65', '64', '20', '63', '72', '79', '70', '74', '6f', '67', '72'], ['61', '70', '68', '79', '80', '00', '00', '00
', '00', '00', '00', '00', '00', '00', '00', '00']]

AES Cipher: ['d5', '99', 'ae', '95', 'fc', '90', 'fd', '95', '85', '5e', 'c8', 'da', 'ae', '3b', 'dc', 'de', '59', 'fb', 'ec', '7e', '36', 'c7', '31', 'ce',
 '42', '1c', '14', '7a', 'd9', '08', '69', '5a']

Plaintext HEX: ['61', '70', '70', '6c', '69', '65', '64', '20', '63', '72', '79', '70', '74', '6f', '67', '72', '61', '70', '68', '79', '80', '00', '00', '0
0', '00', '00', '00', '00', '00', '00', '00', '00']

Plaintext: applied cryptography

sagardarji@Sagars-MBP Programing Assignments %
```