在这段代码中，使用了Python的`Crypto`库来进行AES加解密。以下是这段代码中AES加解密的实现：

1. 密钥生成

首先，将密钥字符串`key_string`编码成字节串，然后使用SHA-256哈希算法对其进行哈希运算，生成32字节的哈希值。这个哈希值就是AES加解密所使用的密钥，即`AES_KEY`。

```
makefileCopy codekey_string = "This is a secret key string."
AES_KEY = hashlib.sha256(key_string.encode()).digest()
```

1. 加密

在加密时，需要先创建一个AES加密对象。这个对象需要传入密钥和加密模式，这里使用了`CBC`模式，还需要指定一个初始向量`iv`。然后使用`pad`函数对明文数据进行填充，将其长度调整为AES块大小的整数倍。最后使用AES加密对象的`encrypt`方法对填充后的数据进行加密。

```
makefileCopy codecipher = AES.new(AES_KEY, AES.MODE_CBC, iv)
plaintext = "This is a message to be encrypted."
padded_plaintext = pad(plaintext.encode(), AES.block_size)
ciphertext = cipher.encrypt(padded_plaintext)
```

1. 解密

在解密时，需要先创建一个AES解密对象，传入密钥、解密模式和初始向量`iv`。然后使用AES解密对象的`decrypt`方法对密文数据进行解密，得到填充后的明文数据。最后使用`unpad`函数对填充后的数据进行去除填充操作，得到原始的明文数据。

```
scssCopy codedecipher = AES.new(AES_KEY, AES.MODE_CBC, iv)
padded_plaintext = decipher.decrypt(ciphertext)
plaintext = unpad(padded_plaintext, AES.block_size).decode()
```

注意，以上代码中的`iv`（初始向量）并没有给出，它应该是一个随机生成的、与密钥长度相同的字节串。在加密和解密时需要使用相同的`iv`。