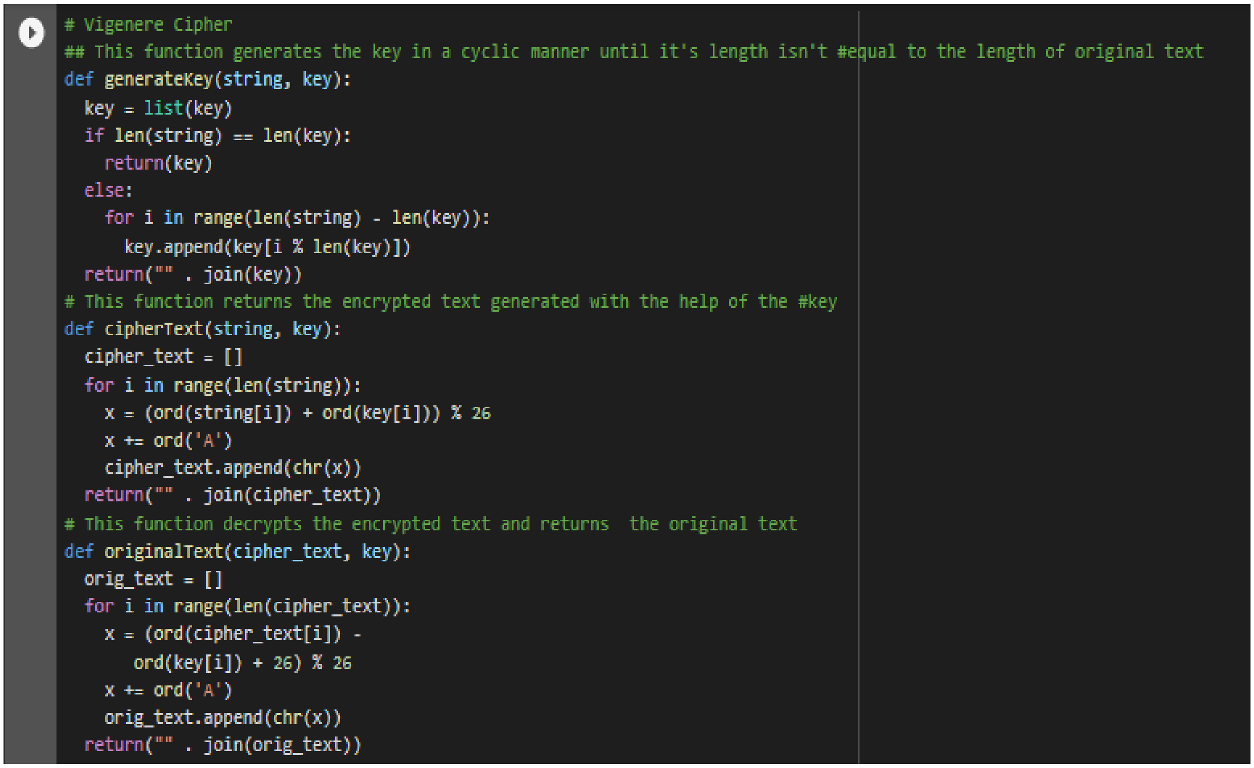
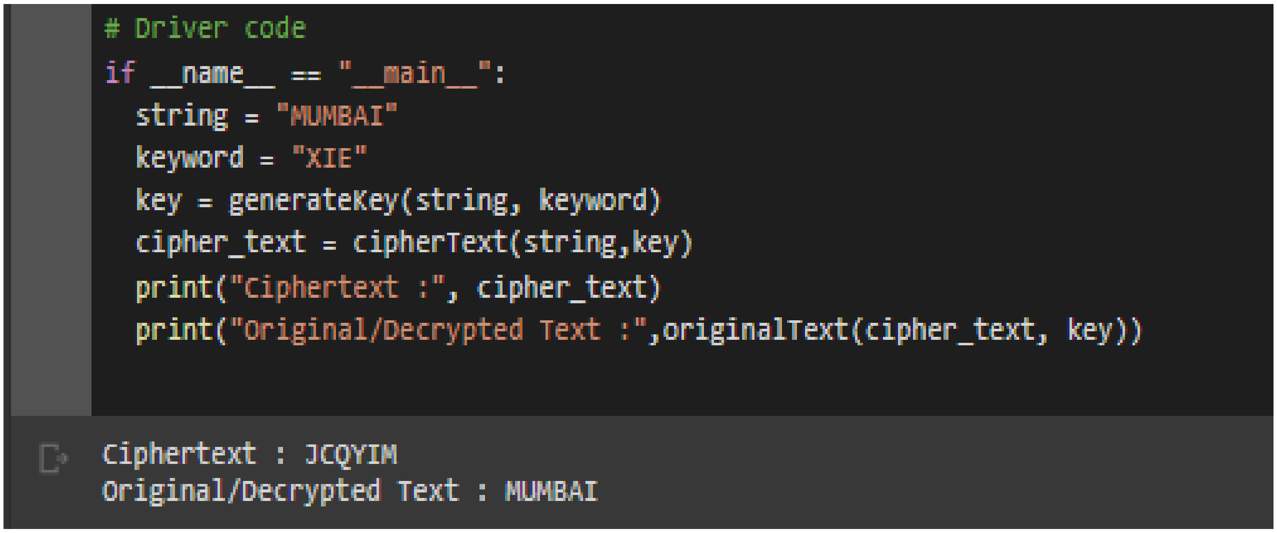
1. Vigenere Cipher



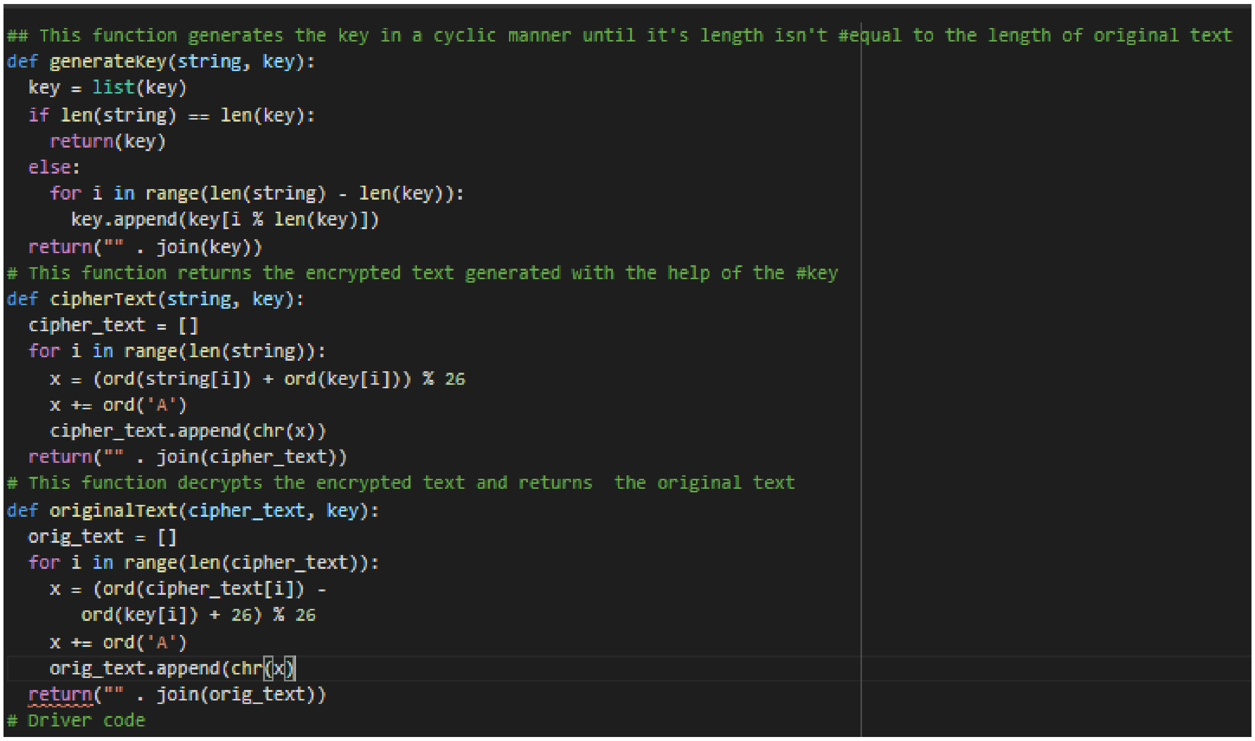
```python
# Vigenere Cipher
## This function generates the key in a cyclic manner until it's length isn't #equal to the length of original text
def generateKey(string, key):
  key = list(key)
  if len(string) == len(key):
    return(key)
  else:
    for i in range(len(string) - len(key)):
      key.append(key[i % len(key)])
  return("" . join(key))
# This function returns the encrypted text generated with the help of the #key
def cipherText(string, key):
  cipher_text = []
  for i in range(len(string)):
    x = (ord(string[i]) + ord(key[i])) % 26
    x += ord('A')
    cipher_text.append(chr(x))
  return("" . join(cipher_text))
# This function decrypts the encrypted text and returns  the original text
def originalText(cipher_text, key):
  orig_text = []
  for i in range(len(cipher_text)):
    x = (ord(cipher_text[i]) -
       ord(key[i]) + 26) % 26
    x += ord('A')
    orig_text.append(chr(x))
  return("" . join(orig_text))
```



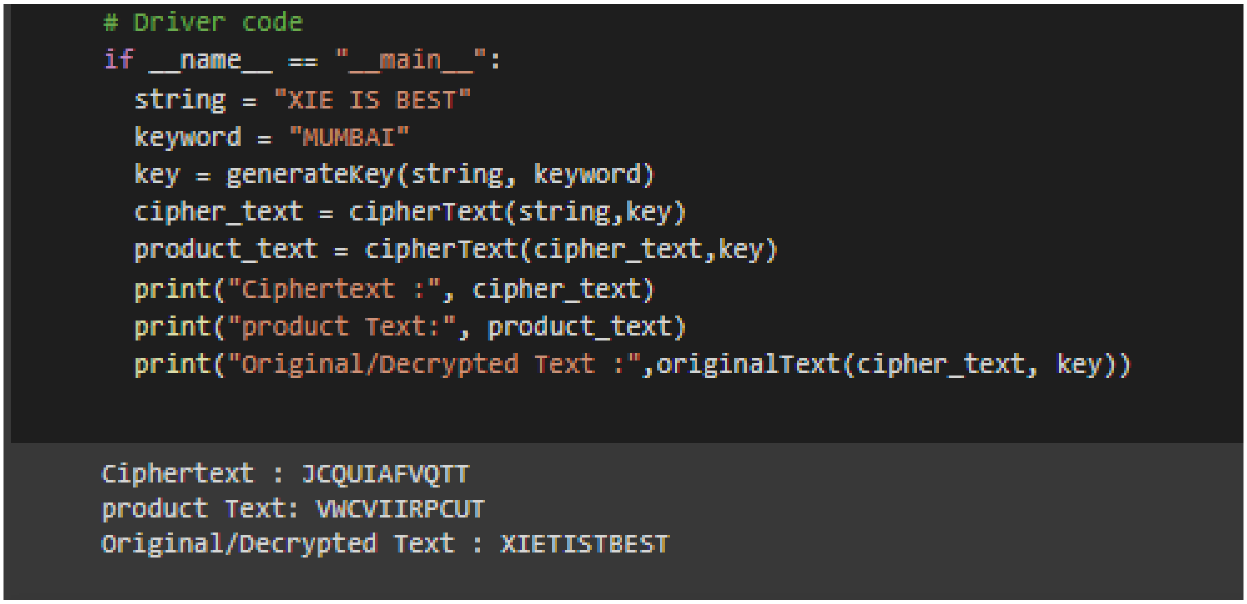
```python
# Driver code
if __name__ == "__main__":
  string = "MUMBAI"
  keyword = "XIE"
  key = generateKey(string, keyword)
  cipher_text = cipherText(string,key)
  print("Ciphertext :", cipher_text)
  print("Original/Decrypted Text :",originalText(cipher_text, key))
```

```
Ciphertext : JCQYIM
Original/Decrypted Text : MUMBAI
```

2. Product Cipher Encryption & Decryption using Vigenere



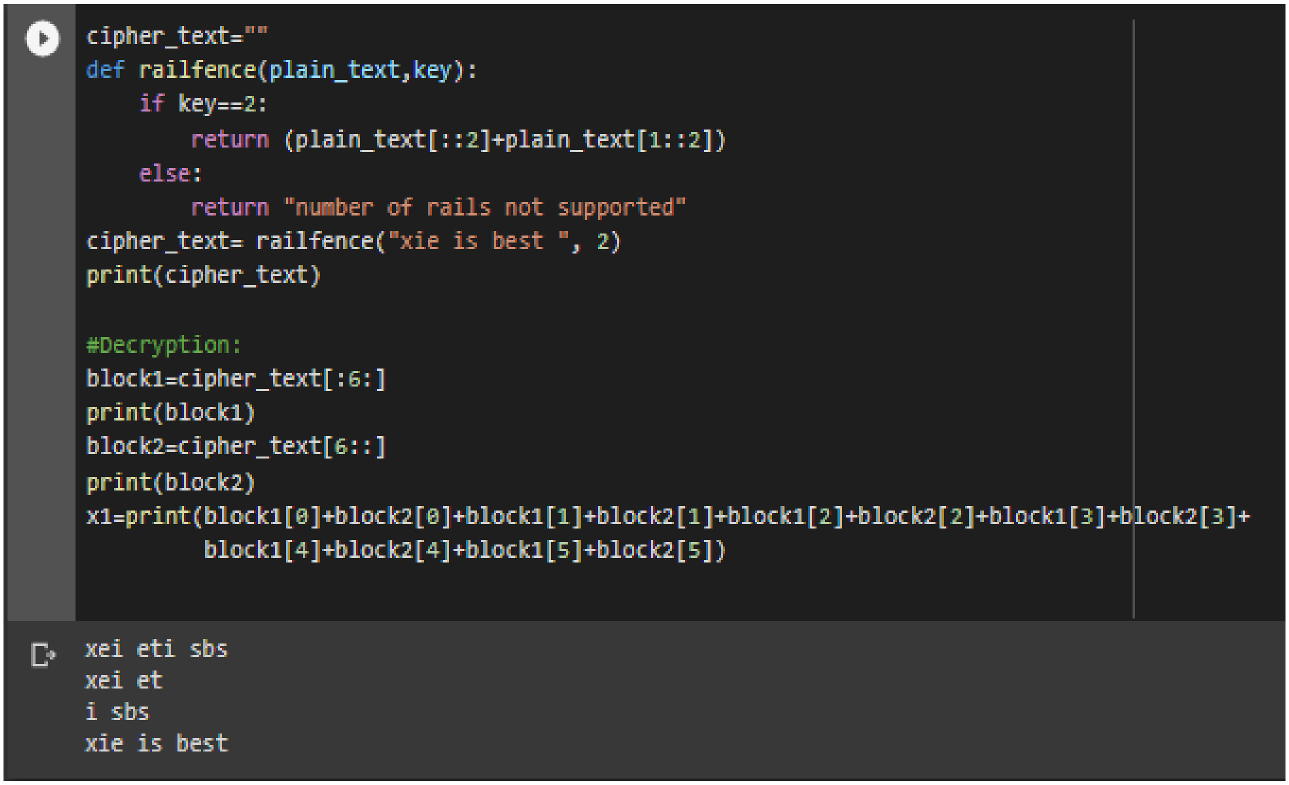
```
## This function generates the key in a cyclic manner until it's length isn't #equal to the length of original text
def generateKey(string, key):
  key = list(key)
  if len(string) == len(key):
    return(key)
  else:
    for i in range(len(string) - len(key)):
      key.append(key[i % len(key)])
  return("" . join(key))
# This function returns the encrypted text generated with the help of the #key
def cipherText(string, key):
  cipher_text = []
  for i in range(len(string)):
    x = (ord(string[i]) + ord(key[i])) % 26
    x += ord('A')
    cipher_text.append(chr(x))
  return("" . join(cipher_text))
# This function decrypts the encrypted text and returns  the original text
def originalText(cipher_text, key):
  orig_text = []
  for i in range(len(cipher_text)):
    x = (ord(cipher_text[i]) -
       ord(key[i]) + 26) % 26
    x += ord('A')
    orig_text.append(chr(x))
  return("" . join(orig_text))
# Driver code
```



```
# Driver code
if __name__ == "__main__":
  string = "XIE IS BEST"
  keyword = "MUMBAI"
  key = generateKey(string, keyword)
  cipher_text = cipherText(string,key)
  product_text = cipherText(cipher_text,key)
  print("Ciphertext :", cipher_text)
  print("product Text:", product_text)
  print("Original/Decrypted Text :",originalText(cipher_text, key))
```

```
Ciphertext : JCQUIAFVQTT
product Text: VWCVIIRPCUT
Original/Decrypted Text : XIETISTBEST
```

3. Railfence Technique

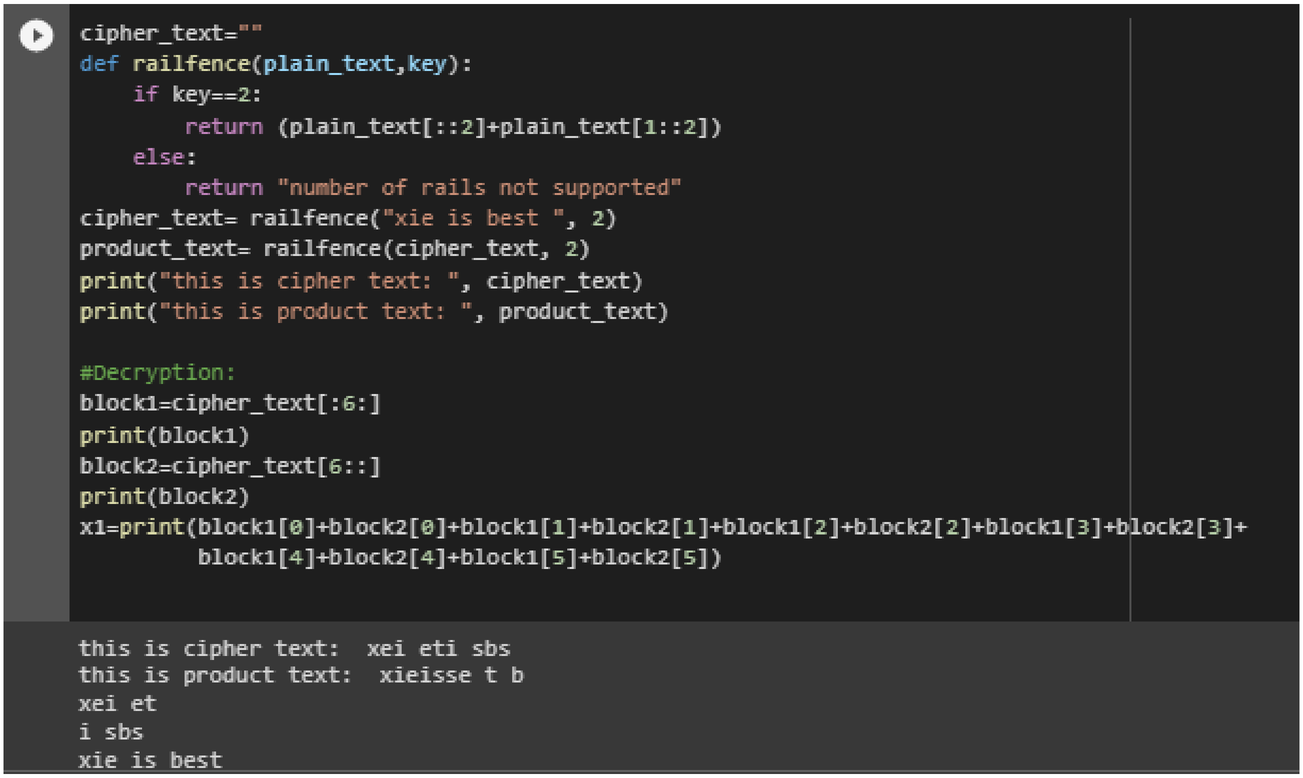


```
cipher_text=""
def railfence(plain_text,key):
    if key==2:
        return (plain_text[::2]+plain_text[1::2])
    else:
        return "number of rails not supported"
cipher_text= railfence("xie is best ", 2)
print(cipher_text)

#Decryption:
block1=cipher_text[:6:]
print(block1)
block2=cipher_text[6::]
print(block2)
x1=print(block1[0]+block2[0]+block1[1]+block2[1]+block1[2]+block2[2]+block1[3]+block2[3]+
         block1[4]+block2[4]+block1[5]+block2[5])
```

```
xei eti sbs
xei et
i sbs
xie is best
```

4. Product Cipher Encryption & Decryption using Rail fence



```
cipher_text=""
def railfence(plain_text,key):
    if key==2:
        return (plain_text[::2]+plain_text[1::2])
    else:
        return "number of rails not supported"
cipher_text= railfence("xie is best ", 2)
product_text= railfence(cipher_text, 2)
print("this is cipher text: ", cipher_text)
print("this is product text: ", product_text)

#Decryption:
block1=cipher_text[:6:]
print(block1)
block2=cipher_text[6::]
print(block2)
x1=print(block1[0]+block2[0]+block1[1]+block2[1]+block1[2]+block2[2]+block1[3]+block2[3]+
         block1[4]+block2[4]+block1[5]+block2[5])
```

```
this is cipher text:  xei eti sbs
this is product text:  xieisse t b
xei et
i sbs
xie is best
```