```
# === Substitution Cipher Encryptor (Google Colab Ready) ===
# Paste this entire cell into Colab and run.

import string, random

# Fixed key (A→cipher)
FIXED_KEY = "QWERTYUIOPASDFGHJKLZXCVBNM"
ALPHABET = string.ascii_uppercase

def encrypt_text(plaintext, key):
    mapping = str.maketrans(ALPHABET, key)
    ciphertext = plaintext.upper().translate(mapping)
    return ciphertext

def generate_random_key():
    letters = list(ALPHABET)
    random.shuffle(letters)
    return ''.join(letters)

print("=== SUBSTITUTION CIPHER ENCRYPTOR ===")
plain = input("Enter your plain text: ")

choice = input("Use fixed key (F) or random key (R)? ").strip().upper()
if choice == "R":
    key = generate_random_key()
    print("\n🔑 Random Key Used:", key)
else:
    key = FIXED_KEY
    print("\n🔑 Fixed Key Used:", key)

cipher = encrypt_text(plain, key)
print("\n✅ Cipher Text:\n", cipher)
print("\n📋 Save this key if you want to decrypt later!")
```

```
=== SUBSTITUTION CIPHER ENCRYPTOR ===
Enter your plain text: hello guys this is my new project checkout in github
Use fixed key (F) or random key (R)? r

🔑 Random Key Used: SYCPTVQZAFBJKHLGWIXNUEMORD

✅ Cipher Text:
 ZTJJL QURX NZAX AX KR HTM GILFTCN CZTCBLUN AH QANZUY

📋 Save this key if you want to decrypt later!
```

```
# ============================================================
# 🔐 Substitution Cipher - Perfect Encryption & Decryption
# Works 100% accurately (no guessing, key-based)
# ============================================================

import string
import random

ALPHABET = string.ascii_uppercase

# ---------------------- GENERATE RANDOM KEY ----------------------
def generate_key():
    letters = list(ALPHABET)
    random.shuffle(letters)
    key = ''.join(letters)
    return key

# ---------------------- ENCRYPT FUNCTION -------------------------
def encrypt(plaintext, key):
    table = str.maketrans(ALPHABET, key)
    ciphertext = plaintext.upper().translate(table)
    return ciphertext

# ---------------------- DECRYPT FUNCTION -------------------------
def decrypt(ciphertext, key):
    table = str.maketrans(key, ALPHABET)
    plaintext = ciphertext.upper().translate(table)
    return plaintext

# ---------------------- MAIN DRIVER ------------------------------
if __name__ == "__main__":
    print("=== Substitution Cipher (Accurate Version) ===")
    mode = input("Choose mode: [E]ncrypt / [D]ecrypt: ").strip().upper()

    if mode == "E":
        text = input("Enter your plaintext: ")
```

```
        text = input("Enter your plaintext: ")
        key = generate_key()
        cipher = encrypt(text, key)
        print("\n✅ Encrypted Ciphertext:\n", cipher)
        print("\n🔑 Save this key to decrypt later:\n", key)

    elif mode == "D":
        cipher = input("Enter your ciphertext: ")
        key = input("Enter the key used during encryption: ")
        plain = decrypt(cipher, key)
        print("\n✅ Decrypted Plaintext:\n", plain)
    else:
        print("Invalid option. Please enter E or D.")
```

```
=== Substitution Cipher (Accurate Version) ===
Choose mode: [E]ncrypt / [D]ecrypt: d
Enter your ciphertext: ZTJJL QURX NZAX AX KR HTM GILFTCN CZTCBLUN AH QANZUY
Enter the key used during encryption: SYCPTVQZAFBJKHLGWIXNUEMORD

✅ Decrypted Plaintext:
 HELLO GUYS THIS IS MY NEW PROJECT CHECKOUT IN GITHUB
```