```
#Генерация Биткоин-адреса
UPPER_LIMIT =
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

# 1 Выбрать случайное 32-байтовое число в качестве закрытого ключа
from random import randint
privkey = bytes.fromhex(hex(randint(1, UPPER_LIMIT))[2:].zfill(64))
print(privkey.hex())

# 2 Найти соответствующий открытый ключ с использованием алгоритма ECDSA-
SECP256k1
from ecdsa import SECP256k1, SigningKey
s = SigningKey.from_string(privkey, curve=SECP256k1)
v = bytes.fromhex("04") + s.verifying_key.to_string()
print(v.hex())

# 3 Вычислить хеш-значение открытого ключа SHA-256
import hashlib

a = hashlib.sha256(v).digest()
print(a.hex())

# 4 Вычислить хеш-значение RIPEMD-160 хеш-значения предыдущего шага
from ripemd.ripemd160 import ripemd160

b = ripemd160(a)
print(b.hex())

# 5 Дописать слева к результату предыдущей операции номер версии
# адреса Биткоин (для основной сети Биткойн - значение «0x00»)
c = bytes.fromhex("00") + b
print(c.hex())

# 6 Вычислить хеш-значение SHA-256 результата предыдущего шага
d = hashlib.sha256(c).digest()
print(d.hex())

# 7 Ещё раз вычислить хеш-значение SHA-256 результата предыдущего шага
e = hashlib.sha256(d).digest()
print(e.hex())

# 8 Дописать к результату шага 5 справа первые 4 байта результата шага 7
f = c+e[:4]
print(f.hex())

# 9 Записать получившееся значение в кодировке base58
from base58 import b58encode

address = b58encode(f)
print(address.decode())
```