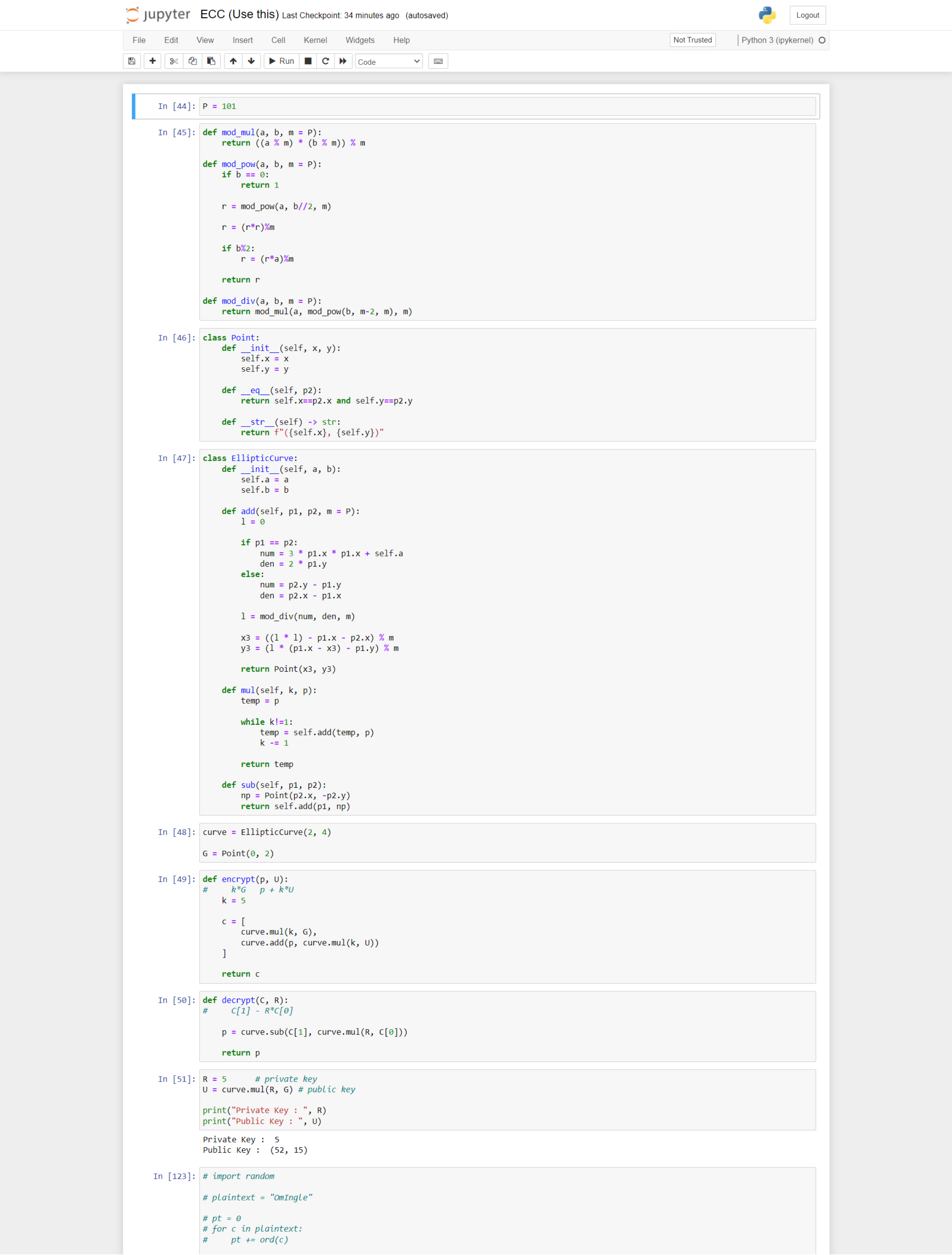

ECC (Use this) Last Checkpoint: 34 minutes ago (autosaved)

In [44]:
```
P = 101
```

In [45]:
```
def mod_mul(a, b, m = P):
    return ((a % m) * (b % m)) % m

def mod_pow(a, b, m = P):
    if b == 0:
        return 1

    r = mod_pow(a, b//2, m)

    r = (r*r)%m

    if b%2:
        r = (r*a)%m

    return r

def mod_div(a, b, m = P):
    return mod_mul(a, mod_pow(b, m-2, m), m)
```

In [46]:
```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __eq__(self, p2):
        return self.x==p2.x and self.y==p2.y

    def __str__(self) -> str:
        return f"({self.x}, {self.y})"
```

In [47]:
```
class EllipticCurve:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def add(self, p1, p2, m = P):
        l = 0

        if p1 == p2:
            num = 3 * p1.x * p1.x + self.a
            den = 2 * p1.y
        else:
            num = p2.y - p1.y
            den = p2.x - p1.x

        l = mod_div(num, den, m)

        x3 = ((l * l) - p1.x - p2.x) % m
        y3 = (l * (p1.x - x3) - p1.y) % m

        return Point(x3, y3)

    def mul(self, k, p):
        temp = p

        while k!=1:
            temp = self.add(temp, p)
            k -= 1

        return temp

    def sub(self, p1, p2):
        np = Point(p2.x, -p2.y)
        return self.add(p1, np)
```

In [48]:
```
curve = EllipticCurve(2, 4)

G = Point(0, 2)
```

In [49]:
```
def encrypt(p, U):
#     k*G   p + k*U
    k = 5

    c = [
        curve.mul(k, G),
        curve.add(p, curve.mul(k, U))
    ]

    return c
```

In [50]:
```
def decrypt(C, R):
#     C[1] - R*C[0]

    p = curve.sub(C[1], curve.mul(R, C[0]))

    return p
```

In [51]:
```
R = 5      # private key
U = curve.mul(R, G) # public key

print("Private Key : ", R)
print("Public Key : ", U)
```

```
Private Key :  5
Public Key :  (52, 15)
```

In [123]:
```
# import random

# plaintext = "OmIngle"

# pt = 0
# for c in plaintext:
#     pt += ord(c)
# x = pt%P
# y = len(plaintext)%P

# plaintext = Point(x, y)


plaintext = Point(3, 4)

print("Plaintext : ", plaintext)
```

```
Plaintext :  (3, 4)
```

In [124]:

```
ciphertext = encrypt(plaintext, U)

print("Ciphertext : ", ciphertext[0], ciphertext[1])
```

```
Ciphertext :  (52, 15) (9, 34)
```

In [125]:

```
decryptedtext = decrypt(ciphertext, R)

# if((plaintext.x%P == decryptedtext.x) and (plaintext.y%P == decryptedtext.y)):
#     print(plaintext, decryptedtext)
#     decryptedtext = plaintext


print("Decrypted Text: ", decryptedtext)
```

```
Decrypted Text:  (3, 4)
```

In [126]:

```
assert(decryptedtext == plaintext)
```

In [127]:

```
if decryptedtext == plaintext:
    print("Plaintext and Decrypted text is same")
```

```
Plaintext and Decrypted text is same
```

In [ ]: