**Usage**

**Time-based OTPs**

totp = pyotp.TOTP('base32secret3232')

totp.now() # => '492039'

# OTP verified for current time

totp.verify('492039') # => True

time.sleep(30)

totp.verify('492039') # => False

**Counter-based OTPs**

hotp = pyotp.HOTP('base32secret3232')

hotp.at(0) # => '260182'

hotp.at(1) # => '055283'

hotp.at(1401) # => '316439'

# OTP verified with a counter

hotp.verify('316439', 1401) # => True

hotp.verify('316439', 1402) # => False

**Generating a base32 Secret Key**

pyotp.random\_base32() # returns a 16 character base32 secret. Compatible with Google Authenticator and other OTP apps

**Google Authenticator Compatible**

PyOTP works with the Google Authenticator iPhone and Android app, as well as other OTP apps like Authy. PyOTP includes the ability to generate provisioning URIs for use with the QR Code scanner built into these MFA client apps:

pyotp.totp.TOTP('JBSWY3DPEHPK3PXP').provisioning\_uri("alice@google.com", issuer\_name="Secure App")

>>> 'otpauth://totp/Secure%20App:alice%40google.com?secret=JBSWY3DPEHPK3PXP&issuer=Secure%20App'

pyotp.hotp.HOTP('JBSWY3DPEHPK3PXP').provisioning\_uri("alice@google.com", initial\_count=0, issuer\_name="Secure App")

>>> 'otpauth://hotp/Secure%20App:alice%40google.com?secret=JBSWY3DPEHPK3PXP&issuer=Secure%20App&counter=0'

This URL can then be rendered as a QR Code (for example, using <https://github.com/neocotic/qrious>) which can then be scanned and added to the users list of OTP credentials.

**Working example**

Scan the following barcode with your phone's OTP app (e.g. Google Authenticator):

Now run the following and compare the output:

import pyotp

totp = pyotp.TOTP("JBSWY3DPEHPK3PXP")

print("Current OTP:", totp.now())