# Key Derivation

Python has pretty limited support for key derivation built into the standard library. In fact, the only method that hashlib provides is the **pbkdf2_hmac** method, which is the PKCS#5 password-based key derivation function 2. It uses HMAC as its psuedorandom function. You might use something like this for hashing your password as it supports a salt and iterations. For example, if you were to use SHA-256 you would need a salt of at least 16 bytes and a minimum of 100,000 iterations.

As a quick aside, a salt is just random data that you use as additional input into your hash to make it harder to "unhash" your password. Basically it protects your password from dictionary attacks and pre-computed rainbow tables.

Let's look at a simple example:

```
import hashlib
import binascii
dk = hashlib.pbkdf2_hmac(hash_name='sha256',
        password=b'bad_password34',
        salt=b'bad_salt',
        iterations=100000)
print (binascii.hexlify(dk))
#b'6e97bad21f6200f9087036a71e7ca9fa01a59e1d697f7e0284cd7f9b897d7c02'
```

Here we create a SHA256 hash on a password using a lousy salt but with 100,000 iterations. Of course, SHA is not actually recommended for creating keys of passwords. Instead you should use something like **scrypt** instead. Another good option would be the 3rd party package, bcrypt. It is designed specifically with password hashing in mind.