**Generating signed messages**

On Ethereum (and other EVM chains), every transaction needs to be signed using the private key corresponding to the signer’s address. The [signer’s address is a truncation of the hash of their ECDSA public key](https://info.etherscan.com/what-is-an-ethereum-address/). So there’s no reason you can’t use your Ethereum key pair to sign and verify arbitrary messages (not just Ethereum transactions).

In this assignment, you’ll use the web3 eth\_account and web3.eth.account libraries to generate and verify signatures on arbitrary messages. You will need to reference the web3 eth\_account library docs, which can be found [here](https://web3py.readthedocs.io/en/latest/web3.eth.account.html)

**Assignment**

Modify the skeleton file “signature.py” to create the functions:

"sign", that takes in a single message *m*, creates an eth account, and uses the account’s key-pair to sign the message *m*. This function should return the Ethereum account address that the signature is valid under, as well as the signature.

"verify", that takes in a single message *m*, *public\_key* the Ethereum account address of the signer, and *signed\_message*, the signature. This function should return a boolean (True or False) for whether or not the message verifies.

If you look at the [ECDSA signing algorithm](https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm), you’ll see that an ECDSA signature actually consists of two integers, *r,s*. Fortunately, the eth.account library abstracts this away, and instead returns a “SignedMessage” object that includes both *r,s* (as well as other information).  
So your function, “sign,” should return two elements, an address and an object of type SignedMessage that holds the two components of the signature.

For the purposes of this assignment, you will generate new accounts every time “sign” is called. However, in future assignments you will need to be able to sign multiple messages with the same key, and pay for gasfees from an account where you have funds. So, you should consider saving at least one of the keypairs generated/printed by your code for use in these upcoming assignments.