1. Generating ECDSA signatures

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. 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 library to generate signatures on arbitrary messages. You will need to reference the web3 eth_account library docs, which can be found here

Assignment

Modify the skeleton file "sign.py" to create a function, "sign," that takes in a single message *m*, creates an eth account, and uses the account's key-pair to sign the message *m*. Your function should return the Ethereum account address that the signature is valid under, as well as the signature.

> If you look at the ECDSA signing 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 not need to store the private key for the account that you generate – the autograder will not check that you