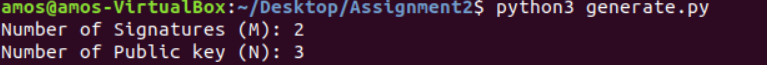
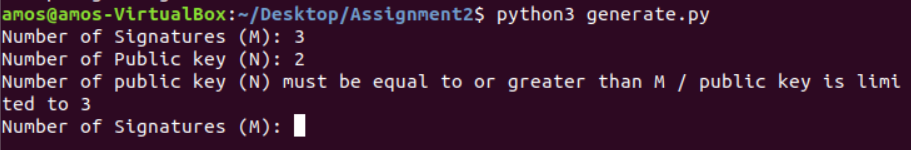
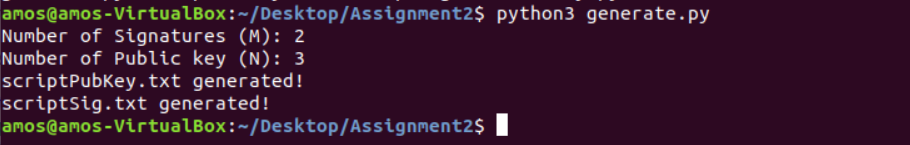
**generate.py**

1. Running generate.py will prompt user for number of Signature (M) and Public keys (N). M must be less than or equals to N and N cannot be more than 3 as P2MS - The locking script of a P2MS can get pretty sizeable with all the public keys.



If the condition above is not fulfilled, it will re-prompt user for number of N and M.

 It will generate random public keys (base on how many the user wants) using the while fixing p q g and write to scriptPubkey.txt. OP\_3 will also be write onto the file. It will also use the key generated previously to sign the message ‘CSCI301 Contemporary topics in security’ and write it to scriptSig.txt. OP\_1 is hard coded to scriptSig.txt Two files scriptPubKey.txt and scriptSig.txt will be created. Contents in scriptPubkey.txt and scriptSig.txt are also changed to hexadecimal as required.

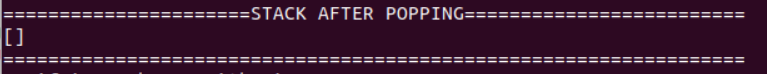


**verify.py**

verify.py will first push OP\_1 and signatures and also OP\_2 into the stack (list is used in my program) thereafter, pubkeys together with OP\_3 will be pushed into the stack. The program will then print out the stack as shown:



CHECKMULTISIG will start by popping OP\_3 and pubkeys followed by OP\_2, Signatures and OP1 from stack following the First In/Last Out rules.



Thereafter will be the verifying of all pub key(s) with all signature(s) while keeping the parameter p,q,g constant. If verifying is successful, it will print out True onto the console. If it fails, it will print out ‘X’. In this case, there are **3** pubkey and **2** sig required.

