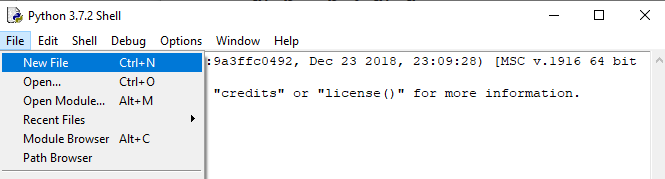
This note shows an example of working back and forth between the file mode and interactive mode of Python, using IDLE.

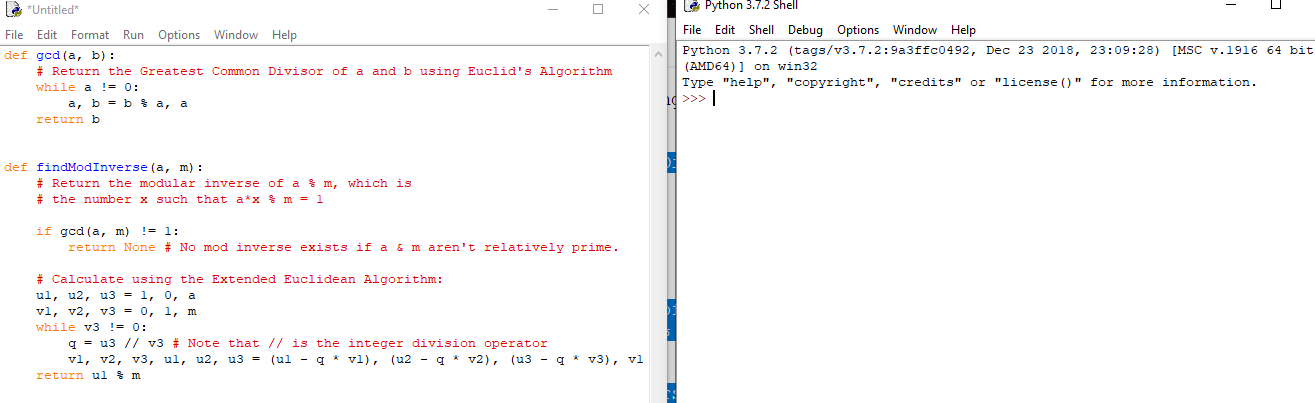
Problem: Generate simple RSA keys, then encrypt and decrypt a number.

Select p and q: I Googled for the 100 lowest prime numbers, and then selected 41 and 67.

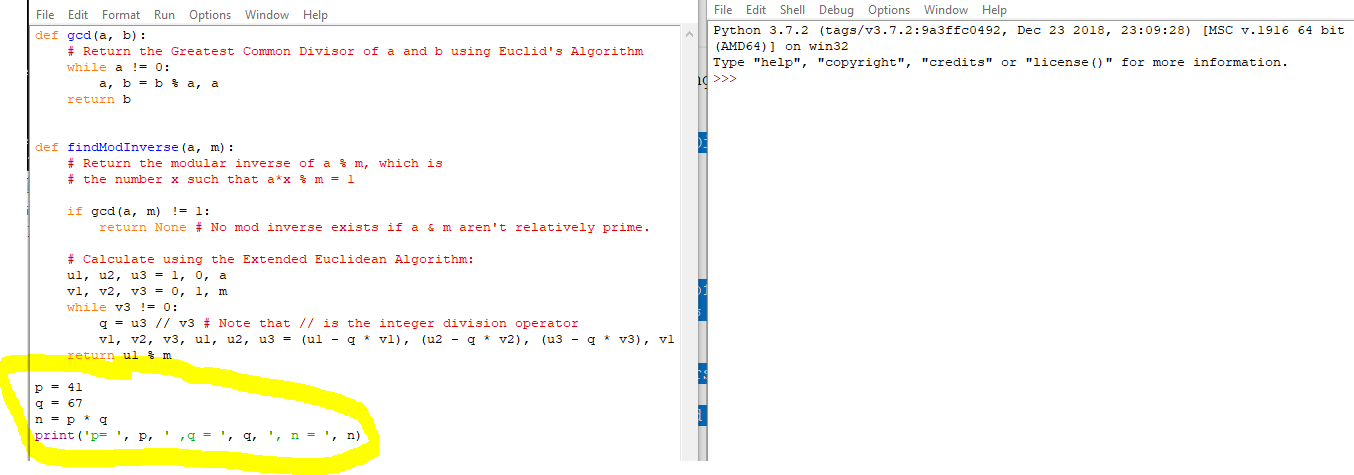
Open IDLE, then create a new file.  


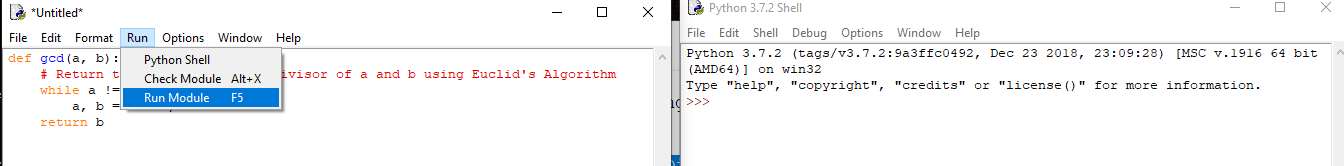
I’m going to pretend I was never able to import cryptomath, so I’ll just open the text file and paste the functions into my file window.

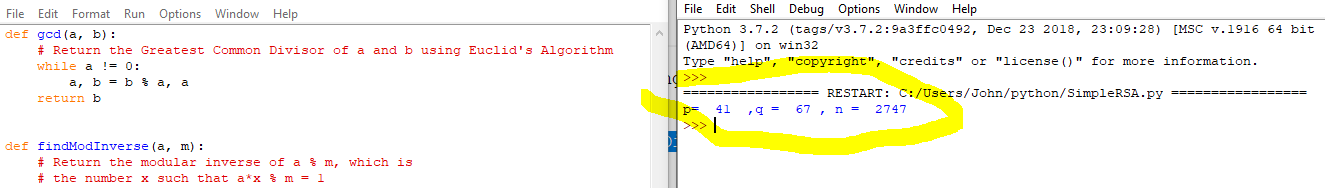


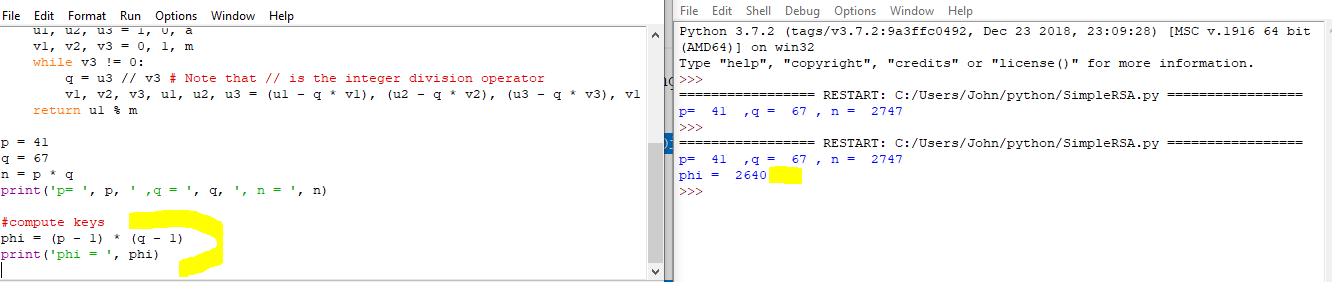
Now I can just use the functions in my file.  


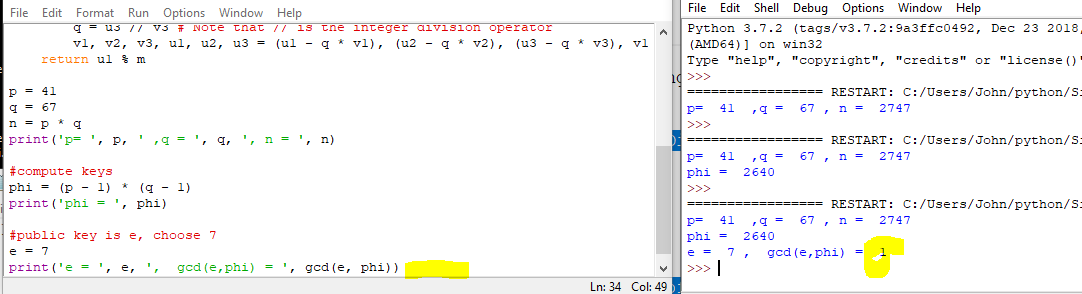
In the file window, put in p and q. Compute n. Print out the values so I know they are correct.



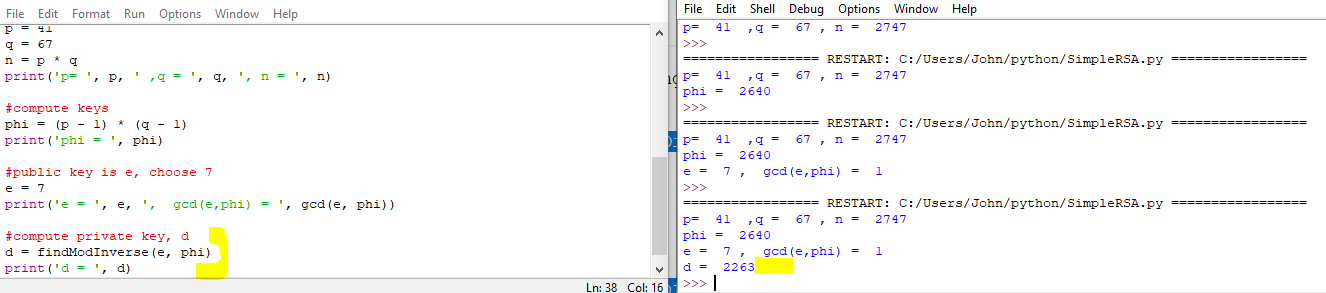
Run the file (module) so I see how I’m doing.  


Worked. Looks like I need to work on my spacing, though.  


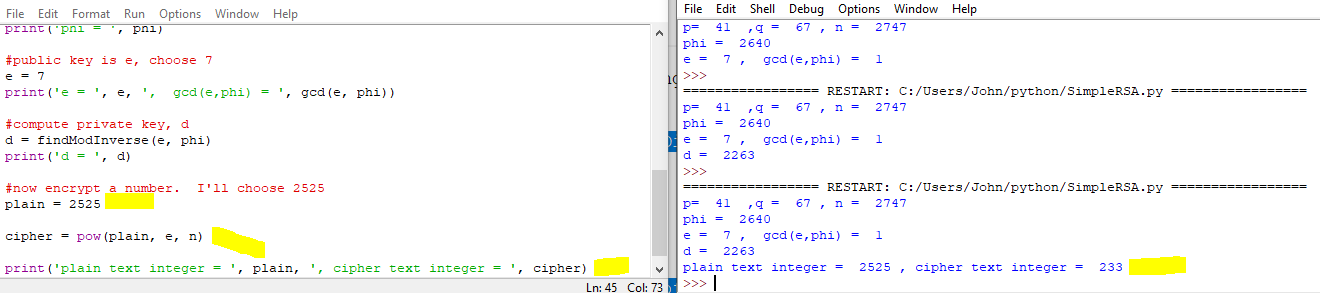
Now compute phi = (p-1)(q-1) so I can create keys. Then run it again.  


OK. Now pick a public key and make sure it is relatively prime with phi.  


Looks good, gcd is 1 so 7 is relatively prime with phi (2460).

Now compute the modular inverse of e (7) mod phi (2460). The answer will be d, the private key. Since I pasted the findModInverse function into the top of the file, I can just call it.  


Cool. I now have my public key, e, and my private key, d.

Now, let’s encrypt a number. The number must be < n (2747 in this case). I’ll choose 2525, and call it my plain text integer). To encrypt the number, take it to the power of the public key, mod n.  


Now to decrypt the number, take it to the power of the private key, mod n.  
