

# Cryptography Homework 2a—Create your Python Environment

## Required Reading

Cryptology2 slides

Cracking Codes with Python by Sweigart

Introduction, 'Crypto regarded as a weapon' and xxv – xxviii, Installing Python

Chapter 1, pp. 1 - 11 (or <https://inventwithpython.com/cracking/chapter1.html>) about crypto paper tools

Chapter 2, pp. 12 - 19 (or <https://inventwithpython.com/cracking/chapter2.html>) about the Python interactive shell

Chapter 3, pp. 30 - 33 about the Idle file editor.

Chapter 5, pp. 54 - 67 (or <https://inventwithpython.com/cracking/chapter5.html>) about the Caesar cipher

## Python

We will use Python for our computations in Cryptology for two reasons:

- 1) Cryptology involves huge integers, and Python will handle them automatically without having to use special classes, like Java's BigInteger.
- 2) Python is a very popular language among IT security specialists

You can download basic instructions on installing Python from the book, *Cracking Codes with Python*, by Al Sweigart. You can buy the book from No Starch Press <https://nostarch.com/crackingcodes>, but Mr. Sweigart has kindly made the book available online at <https://inventwithpython.com/cracking/>. The link for the Introduction has some Python instructions.  
<https://inventwithpython.com/cracking/chapter0.html>.

Python is installed by default on most Linux variants (the version is 3.6.x in Ubuntu, 2.7.x in other distros.) In Windows, Python needs to be installed (instructions later.)

## Choosing an Environment

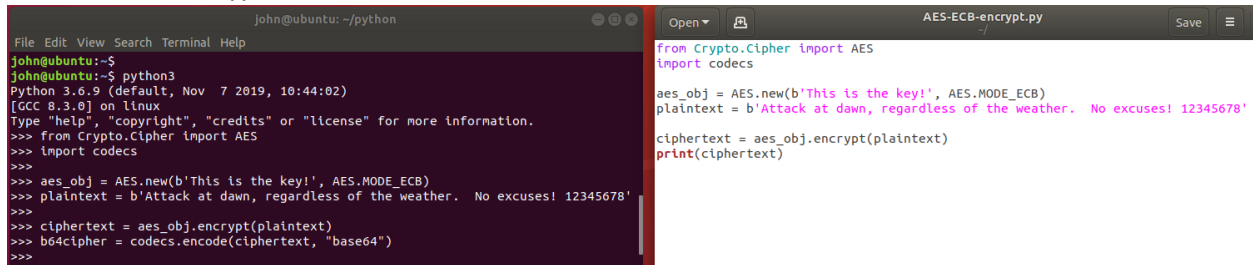
There are many ways to write and use Python.

- Use a terminal Command Line Interface (CLI) and a text editor
- Use Python's built-in development environment, called Idle
- Use a full-feature development environment

## Terminal and Text Editor

This is the simplest method, and the easiest to use when you move between environments. In this method, I have the terminal with the Python interactive prompt and the text editor side by side. I will often test a line or two of code in the terminal and then paste it into the text editor for safe keeping.

Other times I will type a few lines of code into the text editor and then test them in the terminal.



The image shows a text editor window titled 'AES-ECB-encrypt.py' with the following code:

```
from Crypto.Cipher import AES
import codecs

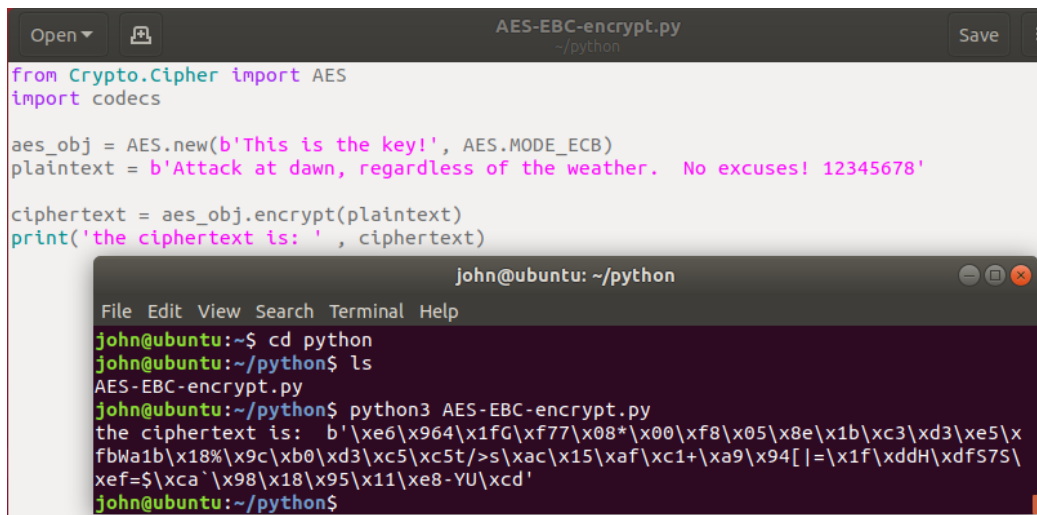
aes_obj = AES.new(b'This is the key!', AES.MODE_ECB)
plaintext = b'Attack at dawn, regardless of the weather. No excuses! 12345678'

ciphertext = aes_obj.encrypt(plaintext)
print(ciphertext)
```

Below the editor is a terminal window with the following output:

```
john@ubuntu: ~/python
File Edit View Search Terminal Help
john@ubuntu:~$ python3
Python 3.6.9 (default, Nov 7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> from Crypto.Cipher import AES
>>> import codecs
>>>
>>> aes_obj = AES.new(b'This is the key!', AES.MODE_ECB)
>>> plaintext = b'Attack at dawn, regardless of the weather. No excuses! 12345678'
>>>
>>> ciphertext = aes_obj.encrypt(plaintext)
>>> b64cipher = codecs.encode(ciphertext, "base64")
>>>
```

Once I have enough of the code written to have a testable script, I run the entire script in the terminal and begin debugging the errors.



The image shows a text editor window titled 'AES-ECB-encrypt.py' with the following code:

```
from Crypto.Cipher import AES
import codecs

aes_obj = AES.new(b'This is the key!', AES.MODE_ECB)
plaintext = b'Attack at dawn, regardless of the weather. No excuses! 12345678'

ciphertext = aes_obj.encrypt(plaintext)
print('the ciphertext is: ', ciphertext)
```

Below the editor is a terminal window with the following output:

```
john@ubuntu: ~/python
File Edit View Search Terminal Help
john@ubuntu:~$ cd python
john@ubuntu:~/python$ ls
AES-ECB-encrypt.py
john@ubuntu:~/python$ python3 AES-ECB-encrypt.py
the ciphertext is:  b'\xe6\x964\x1fG\x77\x08*\x00\xf8\x05\x8e\x1b\xc3\xd3\xe5\xfbWa1b\x18%\x9c\xb0\xd3\xc5\xc5t/>s\xac\x15\xaf\xc1+\xa9\x94[|= \x1f\xddH\xdf57S\xef=$\xca`\x98\x18\x95\x11\xe8-YU\xcd'
```

It is important that the working directory of the terminal is the same as the directory where the file is stored; that way you don't have to specify a path.

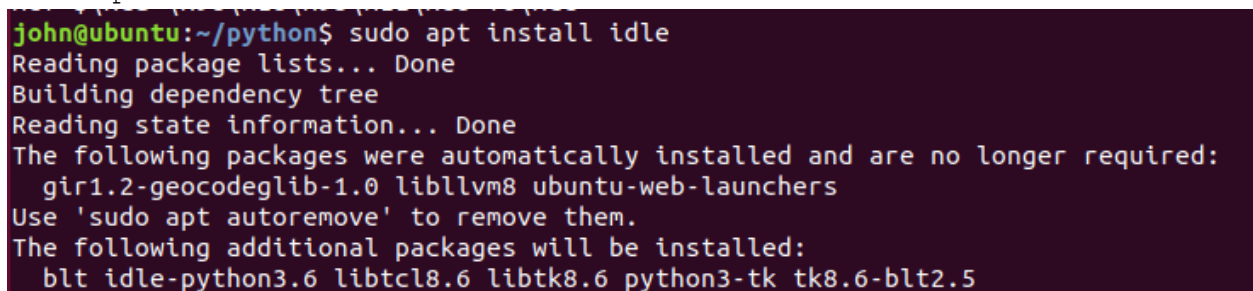
In Linux, the gedit text editor understands Python and will color code your scripts. This handy for spotting errors like forgetting to close quotes. The default editor in Windows, notepad, does not do this.

## Idle

Idle is a simple development interface that comes with Python. In Windows it is installed by default when Python is installed. In Linux, you can install Idle with apt.

<https://www.techrepublic.com/article/how-to-install-the-idle-python-ide-on-ubuntu-desktop-19-10/>

```
sudo apt install idle
```



```
john@ubuntu:~/python$ sudo apt install idle
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-geocodeglib-1.0 libllvm8 ubuntu-web-launchers
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  blt idle-python3.6 libtcl8.6 libtk8.6 python3-tk tk8.6-blt2.5
```

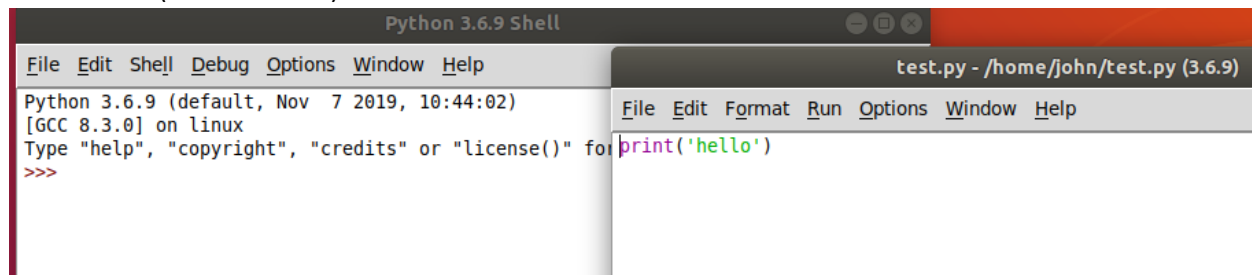
<snip>

In Linux, make an Idle desktop shortcut by clicking on Activities in the upper-left corner. Type idle, right-click on the IDLE (using Pyth..) icon, and select Add to Favorites.

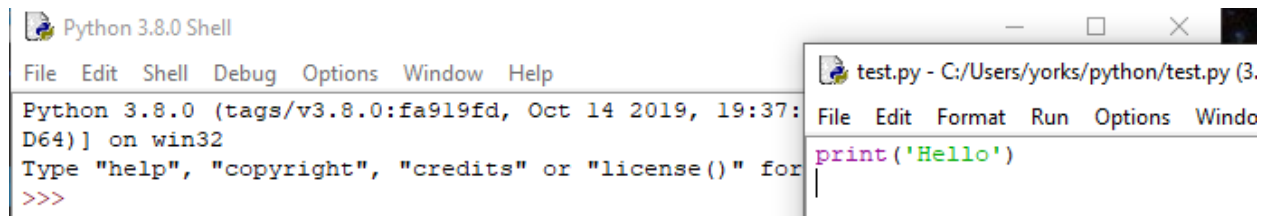


Idle works the same way in both Windows and Linux, so I will use it in all the examples and exercises.

Idle in Linux (Ubuntu 18.04).

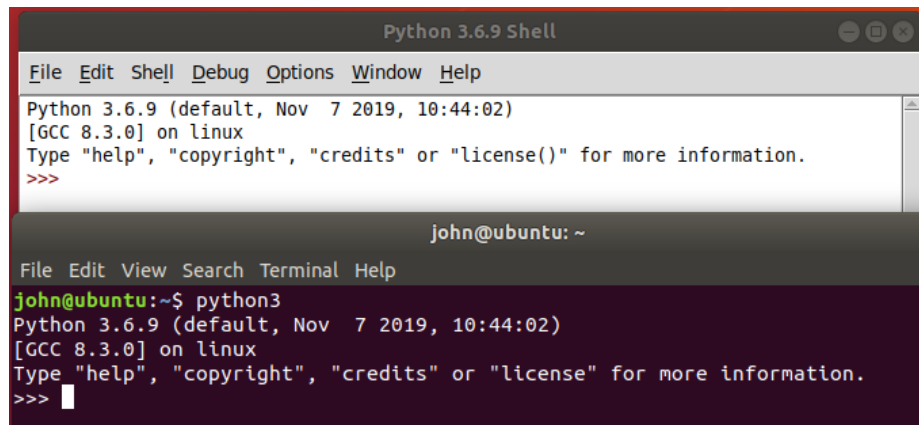


Idle in Windows 10.

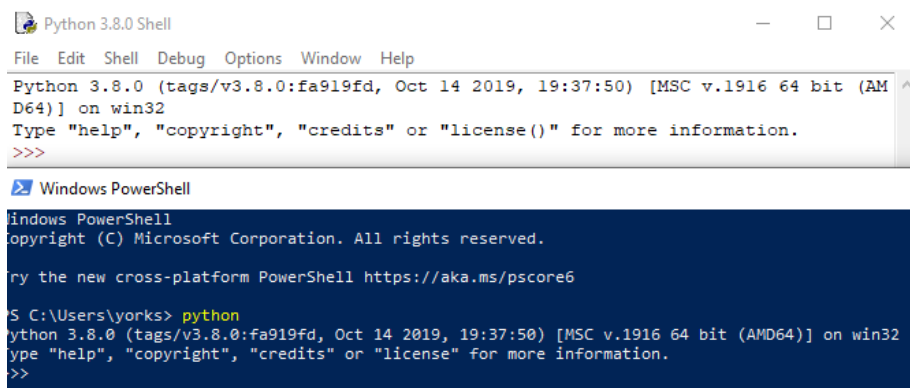


When you open Idle directly (not by opening a Python script in Idle first) you get the Python Shell window.

Linux

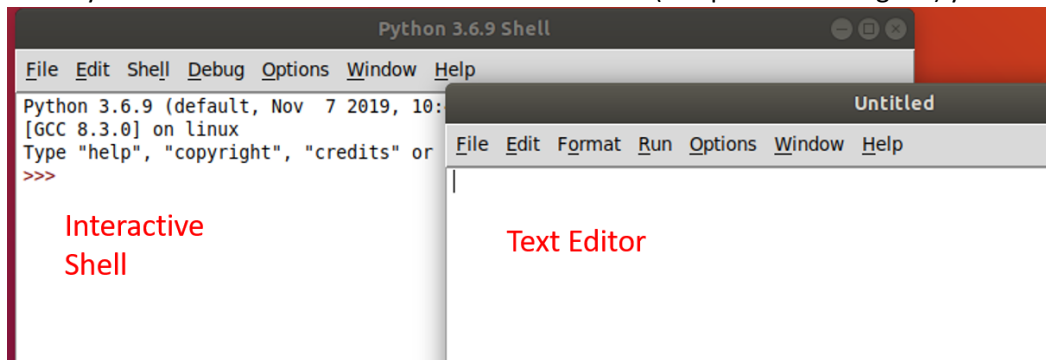


## Windows



The Idle Shell window is the same as the interactive Python prompt in a terminal, except that it has options along the top.

When you use File > New File in the Idle Shell window (or open an existing file) you see Idle's Text editor.



You can jump back and forth between the shell and editor just like you would between a terminal and text editor. When you run a script from the editor, it will run in the shell. For more information, see Chapter 3 of *Cracking Codes with Python*.

Idle is a minimal development environment, but it is useable. The most annoying thing for me is that you scroll through previous commands in the Shell window using Alt-P (previous) and Alt-N (next). I would much rather use the up and down arrows.

## Full-featured Development Environment

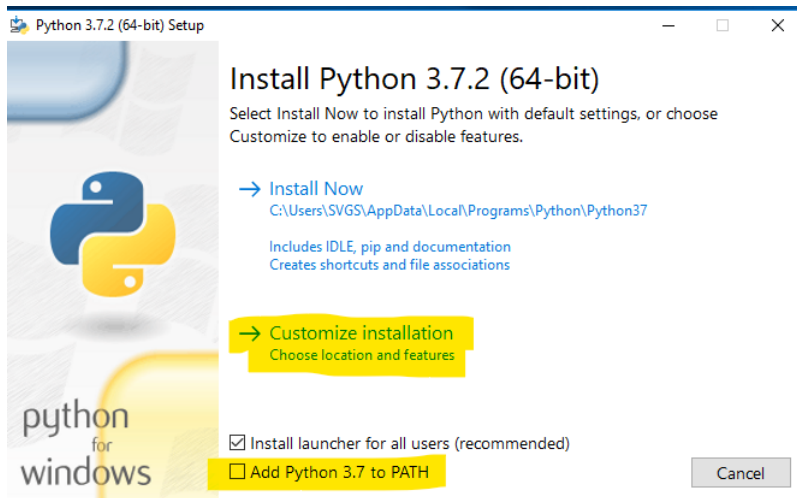
If you do a lot of work in Python, you will undoubtedly select a better Integrated Development Environment (IDE). You can find several by searching the Internet for “python best IDE”. It takes a while to learn to use a complicated IDE; we don't have much time to spare so we won't use a full IDE in this class.

Many of the security people I know use Visual Studio Code, which is a free download. They like it because it supports the other languages they use, so they only need one IDE.

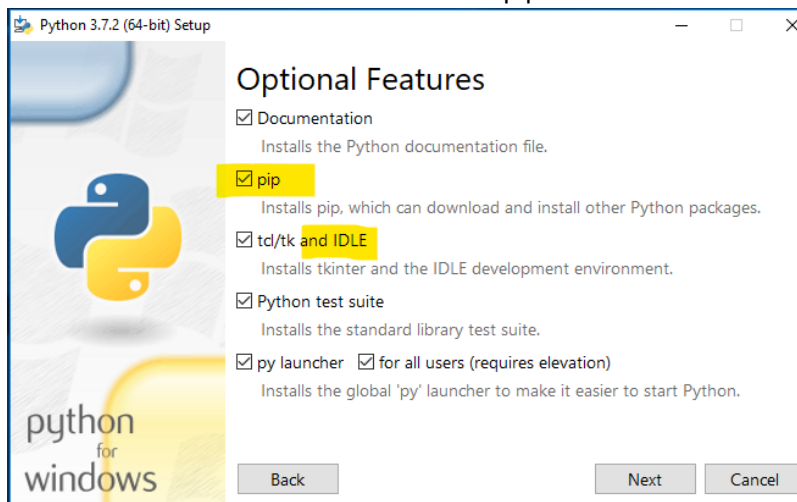
## Installing Python on Windows

Windows users can download Python from <https://www.python.org/downloads/windows/>. This will also install the Idle editor for Python. When you install Python on Windows be sure to check the box to

add Python to your path!! It will make life much easier. It is very handy to just type “python” in a terminal and have it work, rather than typing the entire path.



In the custom installation make sure that pip is selected.



Windows has a limit of 260 characters in a file path. Since Python installs itself on a long file path (like C:\Users\SVGS\AppData\Local\Programs\Python\Python37\), the limit can cause problems.



### Running someone's Python scripts in a terminal

Often you can find scripts that someone has already written that suit your purpose. You could paste their script into your text editor or IDE, but you can also run their script directly from a file. We will do that (question 5 below) using the caesarCipher.py script from CrackingCodesFiles.zip. I recommend you make a directory for your python projects and put the extracted files from CrackingCodesFiles.zip in it.

Download <https://inventwithpython.com/CrackingCodesFiles.zip>, unzip it, and put the contents into a convenient directory. In Ubuntu I used  
/home/{myUserName}/python/crackingcodesfiles.

In Windows, I used C:\Users\{myUserName}\python\CrackingCodesFiles

Then you can run those scripts from the command line like this.

Linux

```
john@ubuntu:~$ cd python/crackingcodesfiles/
john@ubuntu:~/python/crackingcodesfiles$ python3 ./caesarCipher.py
QHIJKLMNOP
Traceback (most recent call last):
  File "./caesarCipher.py", line 45, in <module>
    pyperclip.copy(translated)
  File "/home/john/python/crackingcodesfiles/pyperclip.py", line 574, in lazy_load_stub_copy
    return copy(text)
  File "/home/john/python/crackingcodesfiles/pyperclip.py", line 284, in __call__
    raise PyperclipException(EXCEPT_MSG)
pyperclip.PyperclipException:
Pyperclip could not find a copy/paste mechanism for your system.
For more information, please visit https://pyperclip.readthedocs.io/en/latest/introduction.html#not-implemented-error
```

Note: Ignore the error about pyperclip--it's not installed on this VM. The script ran successfully and output the encrypted version of the hard-coded input, 0123456789 with a key of 21. To remove the error, I should have installed pyperclip or removed the lines in caesarCipher.py that refer to pyperclip.

This script is very simple; a more practical version would allow you to specify input and output files as command line arguments or pipe the input to the script.

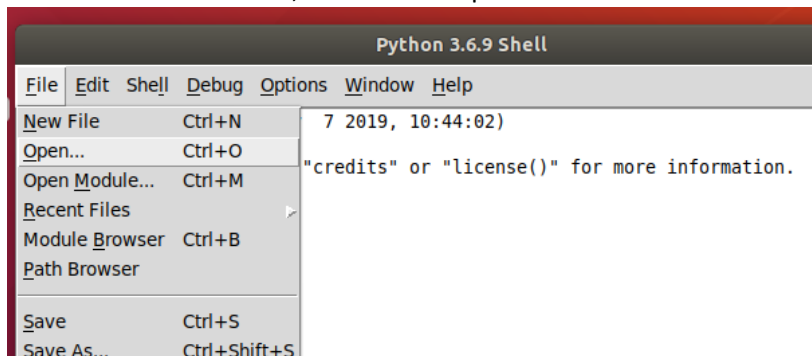
## Windows

```
PS C:\Users\John> cd .\python\CrackingCodesFiles\  
PS C:\Users\John\python\CrackingCodesFiles> python .\caesarCipher.py  
QHIJKLMNOP  
PS C:\Users\John\python\CrackingCodesFiles>
```

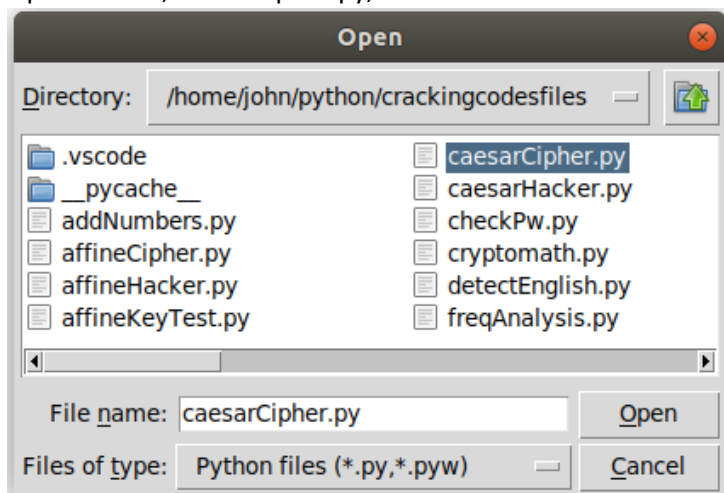
## Running a Script in Idle

Start Idle in either Linux or Windows. I'll show Linux, but Windows is the same.

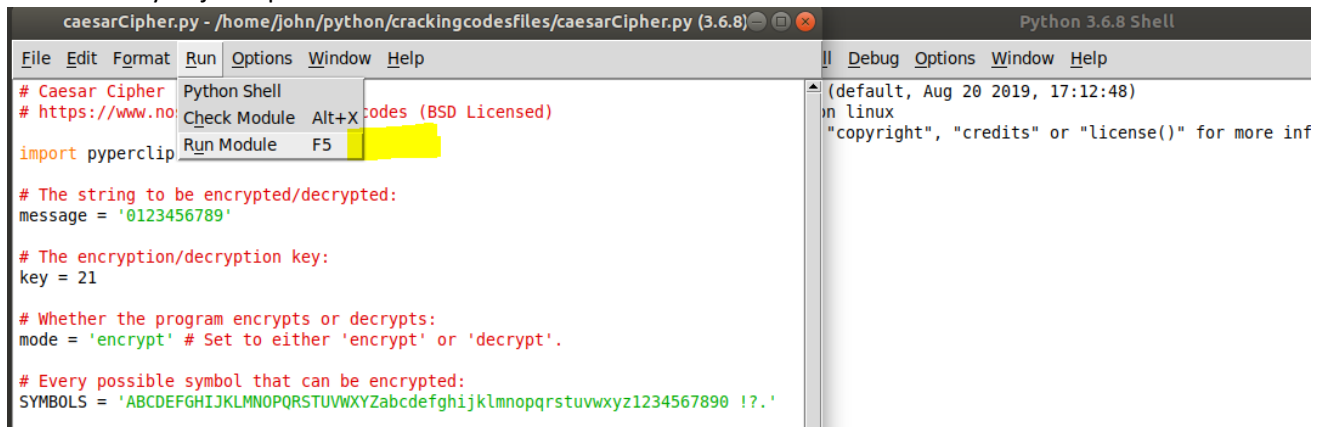
In the Idle Shell window, select File > Open.



Open the file, caesarCipher.py, in this case

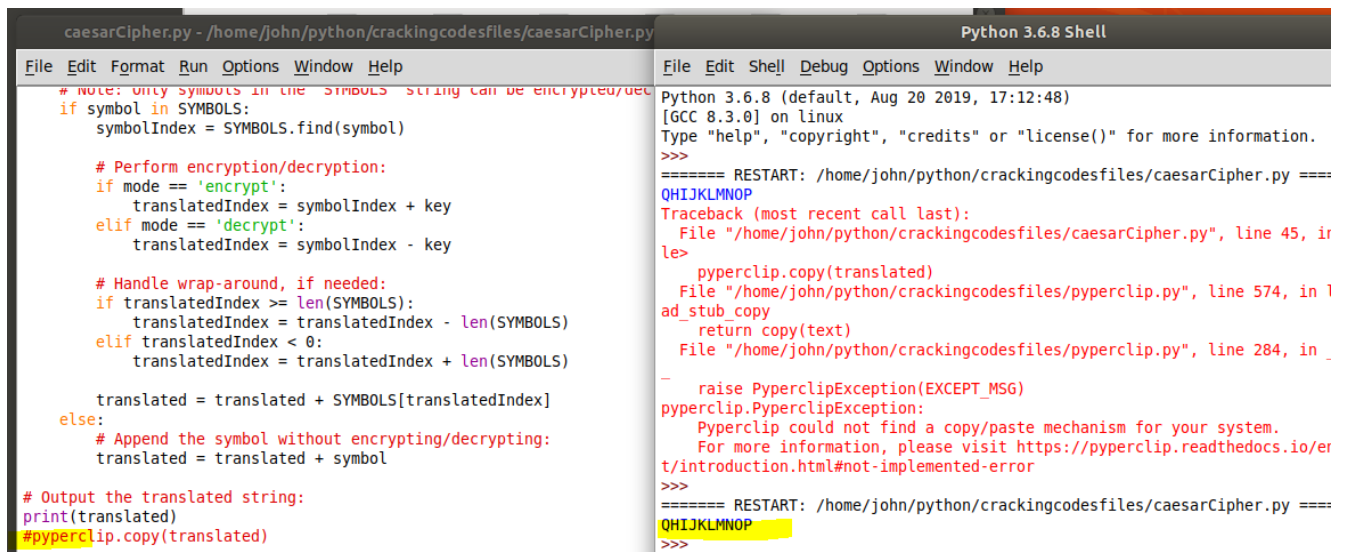


Run the file you just opened.



```
caesarCipher.py - /home/john/python/crackingcodesfiles/caesarCipher.py (3.6.8) Python 3.6.8 Shell
File Edit Format Run Options Window Help
# Caesar Cipher
# https://www.no
import pyperclip
# The string to be encrypted/decrypted:
message = '0123456789'
# The encryption/decryption key:
key = 21
# Whether the program encrypts or decrypts:
mode = 'encrypt' # Set to either 'encrypt' or 'decrypt'.
# Every possible symbol that can be encrypted:
SYMBOLS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789 !?.'
```

The file will run in the Shell window. If you don't have a Shell window open, Idle will open one for you.



```
caesarCipher.py - /home/john/python/crackingcodesfiles/caesarCipher.py Python 3.6.8 Shell
File Edit Format Run Options Window Help
# Note: Only symbols in the SYMBOLS string can be encrypted/decrypted
if symbol in SYMBOLS:
    symbolIndex = SYMBOLS.find(symbol)
    # Perform encryption/decryption:
    if mode == 'encrypt':
        translatedIndex = symbolIndex + key
    elif mode == 'decrypt':
        translatedIndex = symbolIndex - key
    # Handle wrap-around, if needed:
    if translatedIndex >= len(SYMBOLS):
        translatedIndex = translatedIndex - len(SYMBOLS)
    elif translatedIndex < 0:
        translatedIndex = translatedIndex + len(SYMBOLS)
    translated = translated + SYMBOLS[translatedIndex]
else:
    # Append the symbol without encrypting/decrypting:
    translated = translated + symbol
# Output the translated string:
print(translated)
#pyperclip.copy(translated)
Python 3.6.8 (default, Aug 20 2019, 17:12:48)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/john/python/crackingcodesfiles/caesarCipher.py =====
QHIJKLMNOP
Traceback (most recent call last):
  File "/home/john/python/crackingcodesfiles/caesarCipher.py", line 45, in
    le>
    pyperclip.copy(translated)
  File "/home/john/python/crackingcodesfiles/pyperclip.py", line 574, in l
    ad_stub_copy
    return copy(text)
  File "/home/john/python/crackingcodesfiles/pyperclip.py", line 284, in _
    - raise PyperclipException(EXCEPT_MSG)
pyperclip.PyperclipException:
Pyperclip could not find a copy/paste mechanism for your system.
For more information, please visit https://pyperclip.readthedocs.io/en
t/introduction.html#not-implemented-error
>>>
===== RESTART: /home/john/python/crackingcodesfiles/caesarCipher.py =====
QHIJKLMNOP
>>>
```

Oops, nasty errors. I haven't installed pyperclip and the script tries to import pyperclip and then use it to copy its results to the clipboard. Let's get rid of the nasty errors by removing these lines from caesarCipher.py.

```
import pyperclip #near the top of the file
pyperclip.copy(translated) #at the bottom of the file.
```

Save the file and run it again.

```
>>>
===== RESTART: /home/john/python/CrackingCodesFiles/caesarCipher.py =====
QHIJKLMNOP
>>>
```

Ah, the error is gone.

If you want to play with the Caesar cipher script, edit the code to change message (plaintext), key, and mode (encrypt or decrypt).



We will talk about how the caesarCipher.py script works in class, or you can read about it in the Cracking Codes book in chapter 5, pp. 53 -68, or at <https://inventwithpython.com/cracking/chapter5.html>

#### For Turn In

- 1) Open the file from CrackingCodes.zip, caesarCipher.py and run it. If you haven't already, remove the pyperclip mentions from the file to prevent errors.
  - a. To show that you have a working version of Python, execute caesarCipher.py either from Idle or a terminal (either Linux or Windows, your choice) and hand in a screenshot.
  - b. Edit the message and the key in caesarCipher.py to encrypt a message of your choosing. Then change to decrypt mode and decrypt your message. Hand in a screenshot.
  - c. In caesarCipher.py, the symbol set is expanded to include upper and lower case, numbers, and some punctuation symbols.  
SYMBOLS =  
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890 !?.'  
The caesarCipher.py code has a problem, which you can see by encrypting a message with the key = 40. What happened? How can you fix it?
- 2) The affineCipher.py script also uses an expanded symbol set.  
SYMBOLS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890 !?.'
  - a. This symbol set is not a field, because not all elements have a multiplicative inverse. What caused that to happen? (Hint: Look at the length of SYMBOLS.)
  - b. Add or remove characters from SYMBOLS so that all elements of SYMBOLS have multiplicative inverses. Again, len(SYMBOLS) is the key.
    - i. What SYMBOLS set did you choose, and what is its length?
    - ii. Encrypt a message. Hand in the encrypted message and the key you used.