Introduction to Python for Cybersecurity

NATIONAL
STUDENT
LEADERSHIP
CONFERENCE

Quick Overview

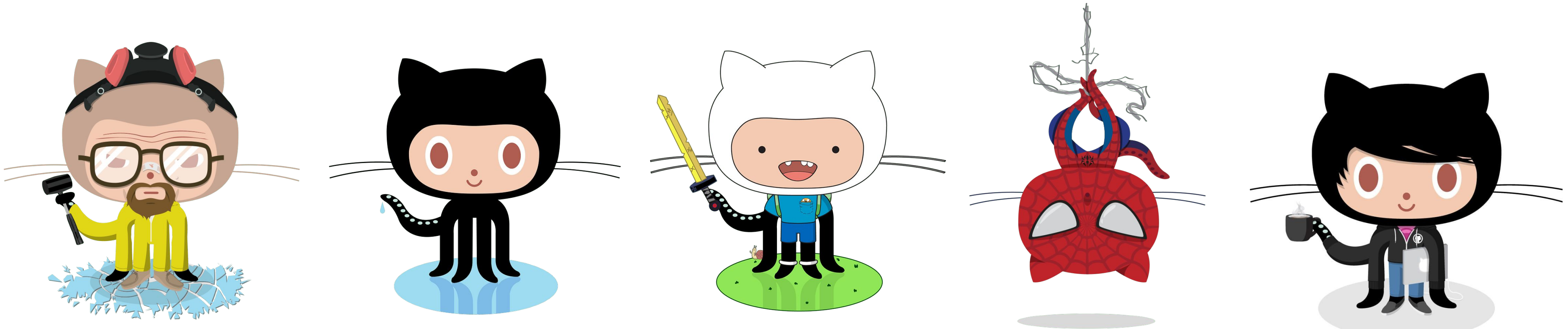
- ➔ Code in Kali Linux using nano!
- ➔ Learn how to encrypt messages using the Fernet Library!
- ➔ Find all IP’s in our subnet that are active and list them!
- ➔ Create a Caesar Cipher!

Let's Start!!!

https://octodex.github.com/

First, lets use github to access our codes!

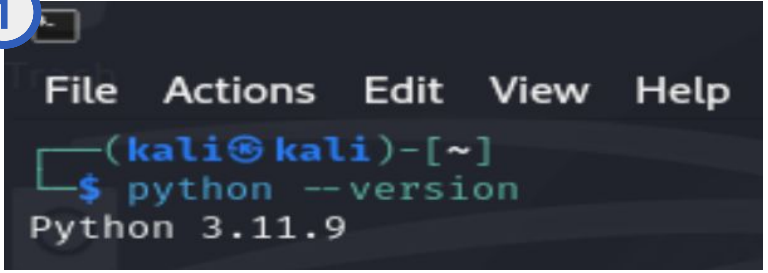
For our coding exercises and projects, we need to access code that has already been written. Instead of everyone writing the same code from scratch, we can use code that's stored in a GitHub repository. This makes it easier to start working on our projects quickly and ensures that everyone has the same starting point.



How can I code in Kali?

First go to your terminal and make sure you have all the libraries necessary for this workshop

1

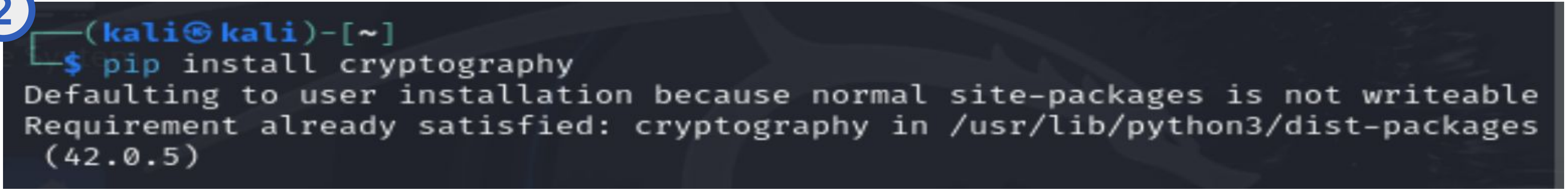


```
┌──(kali㉿kali)-[~]
└─$ python --version
Python 3.11.9
```

Make sure to have python installed

Install cryptography library for fernet

2



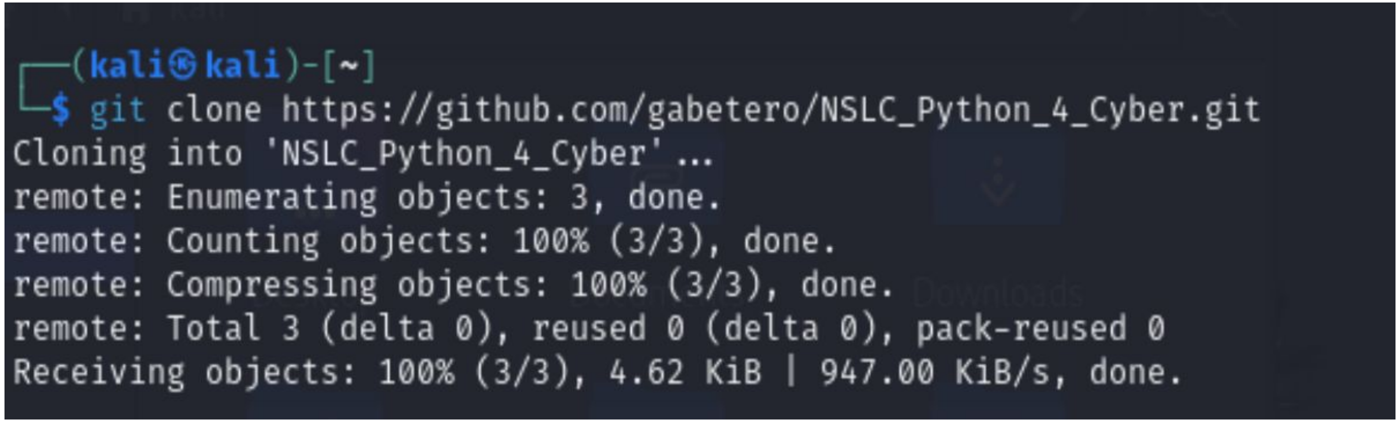
```
┌──(kali㉿kali)-[~]
└─$ pip install cryptography
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages
 (42.0.5)
```

How to use git command in terminal

Open your terminal.

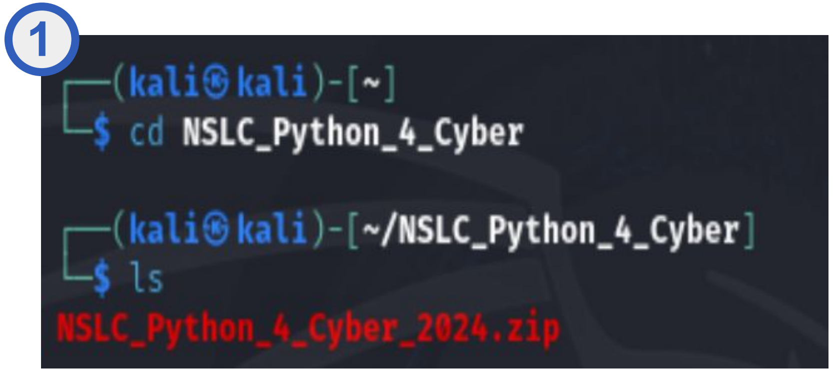
Type the following command and press Enter:

```
git clone https://github.com/gabetero/NSLC.git
```



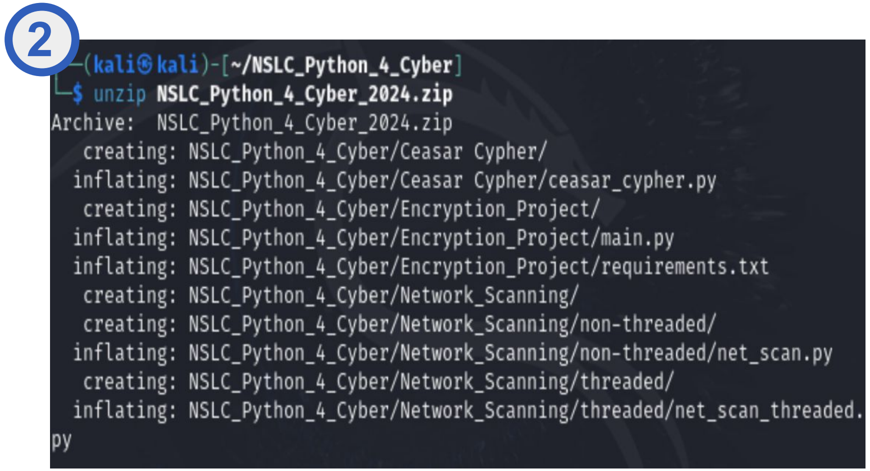
```
┌──(kali㉿kali)-[~]
└─$ git clone https://github.com/gabetero/NSLC_Python_4_Cyber.git
Cloning into 'NSLC_Python_4_Cyber'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), 4.62 KiB | 947.00 KiB/s, done.
```

Accessing and navigating the NSLC_*Python_4_Cyber* file



```
┌──(kali㉿kali)-[~]
└─$ cd NSLC_Python_4_Cyber

┌──(kali㉿kali)-[~/NSLC_Python_4_Cyber]
└─$ ls
NSLC_Python_4_Cyber_2024.zip
```

Change directory into your new file!

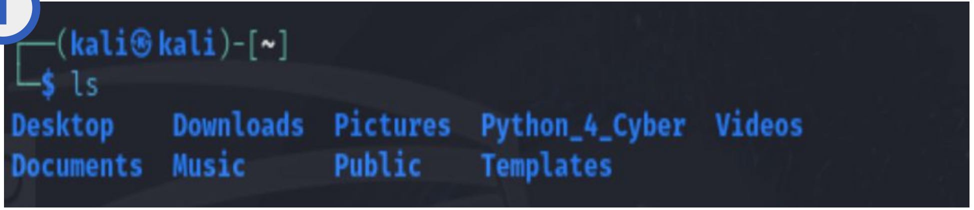


```
┌──(kali㉿kali)-[~/NSLC_Python_4_Cyber]
└─$ unzip NSLC_Python_4_Cyber_2024.zip
Archive:  NSLC_Python_4_Cyber_2024.zip
   creating: NSLC_Python_4_Cyber/Ceasar Cypher/
  inflating: NSLC_Python_4_Cyber/Ceasar Cypher/ceasar_cypher.py
   creating: NSLC_Python_4_Cyber/Encryption_Project/
  inflating: NSLC_Python_4_Cyber/Encryption_Project/main.py
  inflating: NSLC_Python_4_Cyber/Encryption_Project/requirements.txt
   creating: NSLC_Python_4_Cyber/Network_Scanning/
   creating: NSLC_Python_4_Cyber/Network_Scanning/non-threaded/
  inflating: NSLC_Python_4_Cyber/Network_Scanning/non-threaded/net_scan.py
   creating: NSLC_Python_4_Cyber/Network_Scanning/threaded/
  inflating: NSLC_Python_4_Cyber/Network_Scanning/threaded/net_scan_threaded.
py
```

Since its a zip file, we have to unzip before accessing.

Accessing and navigating the NSLC_*Python_4_Cyber* file

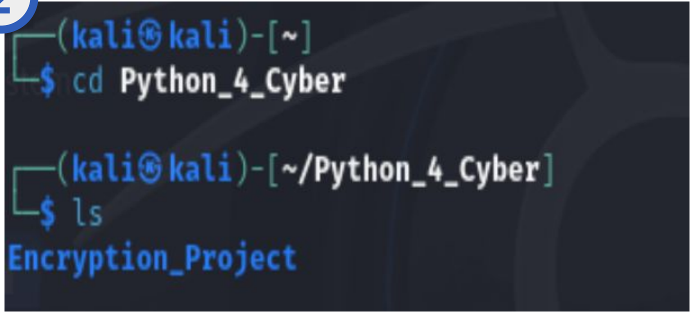
1



```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads  Pictures  Python_4_Cyber  Videos
Documents  Music      Public    Templates
```

List all directories to identify the folder

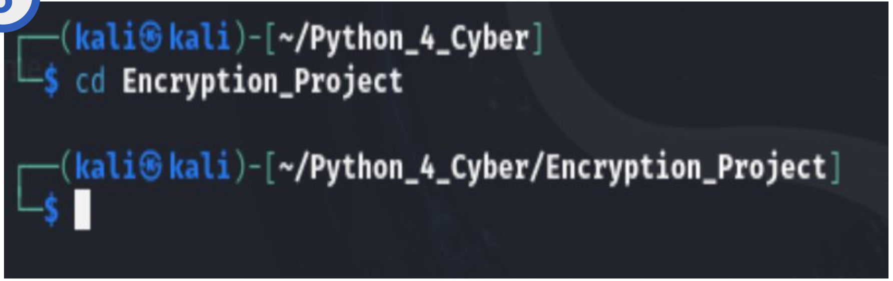
2



```
┌──(kali㉿kali)-[~]
└─$ cd Python_4_Cyber

┌──(kali㉿kali)-[~/Python_4_Cyber]
└─$ ls
Encryption_Project
```

List all directories to identify the folder

3

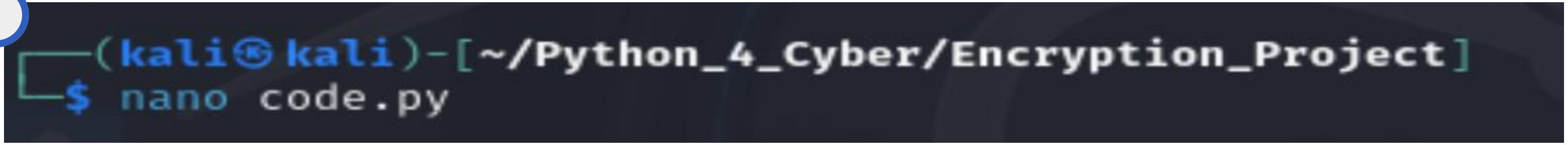


```
┌──(kali㉿kali)-[~/Python_4_Cyber]
└─$ cd Encryption_Project

┌──(kali㉿kali)-[~/Python_4_Cyber/Encryption_Project]
└─$ 
```

Go inside the project using cd (change directory)

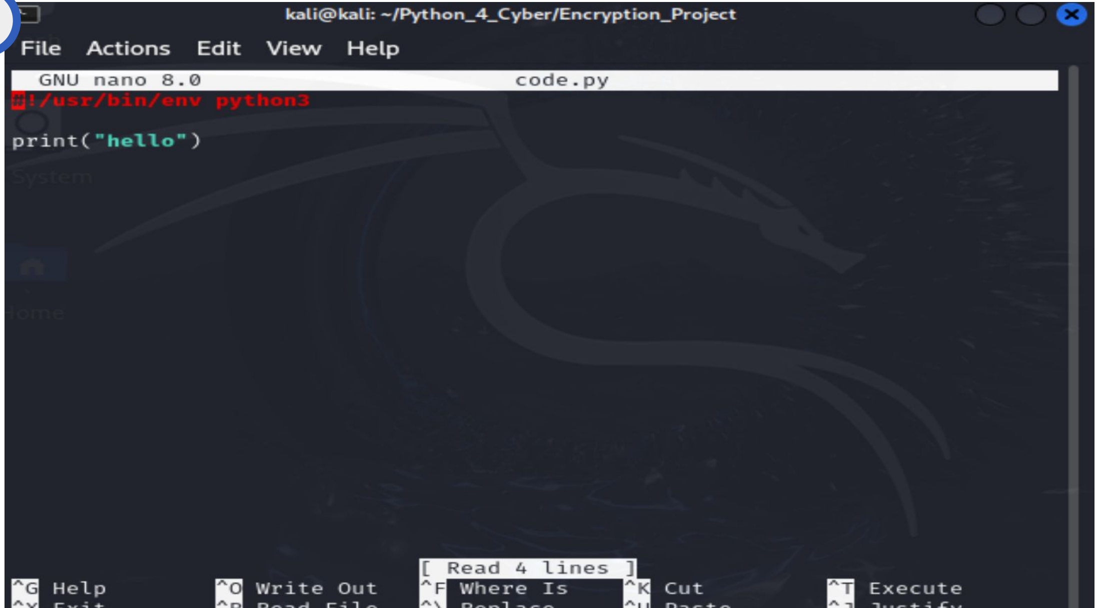
How to start coding in Kali

1



```
┌──(kali㉿kali)-[~/Python_4_Cyber/Encryption_Project]
└─$ nano code.py
```

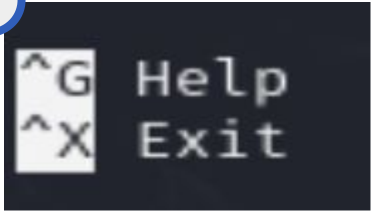
2



kali@kali: ~/Python_4_Cyber/Encryption_Project
File Actions Edit View Help

```
  GNU nano 8.0                          code.py
#!/usr/bin/env python3

print("hello")





                              [ Read 4 lines ]
^G Help          ^O Write Out     ^F Where Is      ^K Cut           ^T Execute
```

3

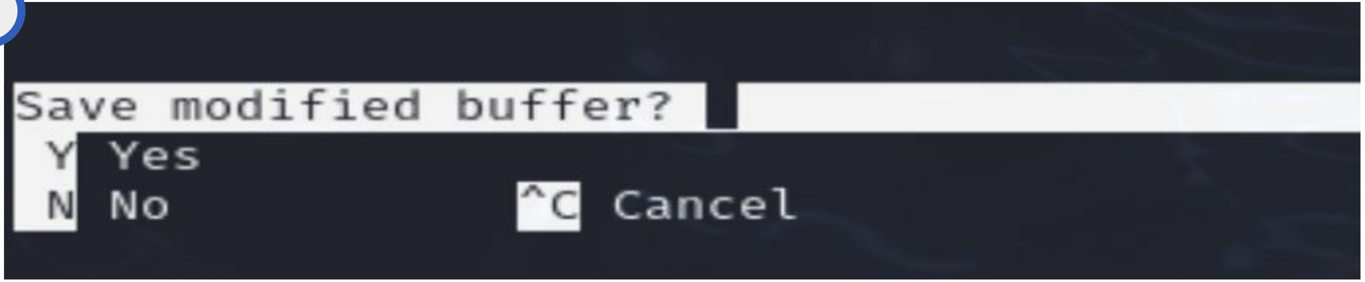


```
^G Help
^X Exit
```

To exit your work, press Ctrl+X

Then type “Y” to save

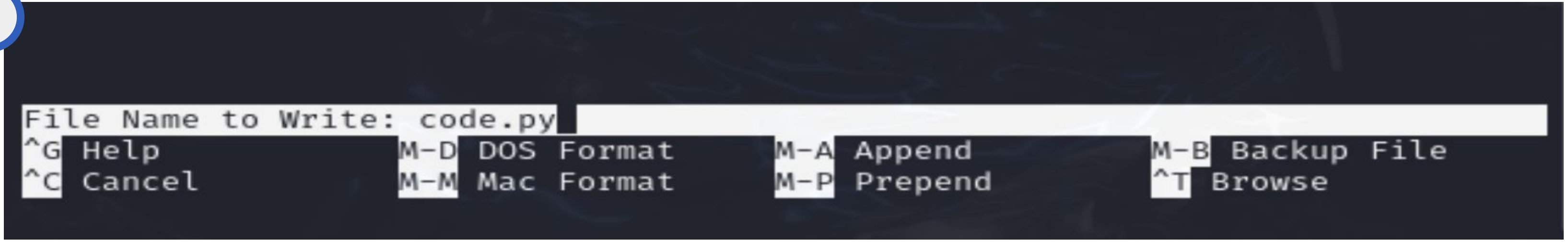
4



```
Save modified buffer?
 Y Yes
 N No                ^C Cancel
```

Modify your code’s name if desired, click enter

5



```
File Name to Write: code.py
^G Help          M-D DOS Format     M-A Append      M-B Backup File
^C Cancel        M-M Mac Format     M-P Prepend     ^T Browse
```

Cryptography

- Cryptography is like a secret code that only you and your friend can understand.

Here's how it works:

★ Imagine you have a special box with a lock. This lock has a unique key that only you and your friend have.

★ To send a secret message, you write it on a piece of paper and put it inside the box.

★ Before closing the box, you use the lock and key to secure it. This locks the message inside and makes it unreadable to anyone who doesn't have the key.

Cryptography

- ★ You then give the locked box to your friend, who has the key. They can unlock the box and read the secret message inside.
- ★ If anyone else tries to open the box without the key, they won't be able to read the message because it's locked and encrypted.

Let’s talk Fernet and Cryptography

Fernet is a part of the `cryptography` library in Python. It is a system for symmetric key encryption, which means the same key is used for both encryption and decryption. Here are some reasons why we use Fernet:

1. **Simplicity**: Fernet is designed to be easy to use. It handles the complexities of encryption and decryption, allowing us to focus on understanding the basic concepts.
2. **Security**: Fernet uses modern encryption techniques to ensure data is secure. It uses AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode with a 128-bit key for encryption, which is considered very secure.
3. **Authentication**: Fernet also ensures that the data has not been tampered with. It includes a Message Authentication Code (MAC) that helps verify the integrity and authenticity of the data.

CODE REVIEW TIME

Python to scan a network

We'll write a Python program that will:

1. **Ping Each Device:** We'll send a little message (a "ping") to each device on the network.
2. **Check Responses:** We'll see which devices respond back to our ping. This tells us that those devices are turned on and ready to communicate.
3. **Print Results:** Finally, we'll print out a list of all the devices that are active on our network.

How It Works:

- **IP Addresses:** Every device connected to a network has a unique address called an IP address, like a home address for the internet. We'll use Python to check each address in our network range.
- **Ping Command:** In our Python code, we use a command called `ping`. It's like knocking on a door to see if someone's home. If the device is there and responds, we'll know it's active.
- **Subnet:** We define the range of IP addresses we want to check using something called a "subnet". For example, `192.168.1.0/24` means we're looking at all addresses from `192.168.1.1` to `192.168.1.254`.

CODE REVIEW TIME

Open Questions

Is there anything that stuck out to you while executing this script?

It’s too slow!

When conducting real life network scans, it is important to consider how scalable our script is!

Scanning a subnet of a few possible active IP's pales in comparison to scanning a busy network with hundreds of active IP’s

This is where multithreading comes along

Imagine:

Every car in this lane is a task in your computer. This means that if there are many cars (tasks) waiting, it can be slow and inefficient because each car has to wait for the one in front of it to move.

With multithreading that is not an issue, because each car will have it's own lane. Thanks to multithreading, the computer can execute several processes at the same time

CODE REVIEW TIME

The Caesar Cipher

- One of the oldest and simplest encryption techniques, named after Julius Caesar, the famous Roman general and statesman.

- Around 58 BC, Julius Caesar used this cipher to protect his military messages.

- By encrypting his communications, Caesar ensured that even if his messages were intercepted by enemies, they wouldn't understand the true content without knowing the cipher technique.

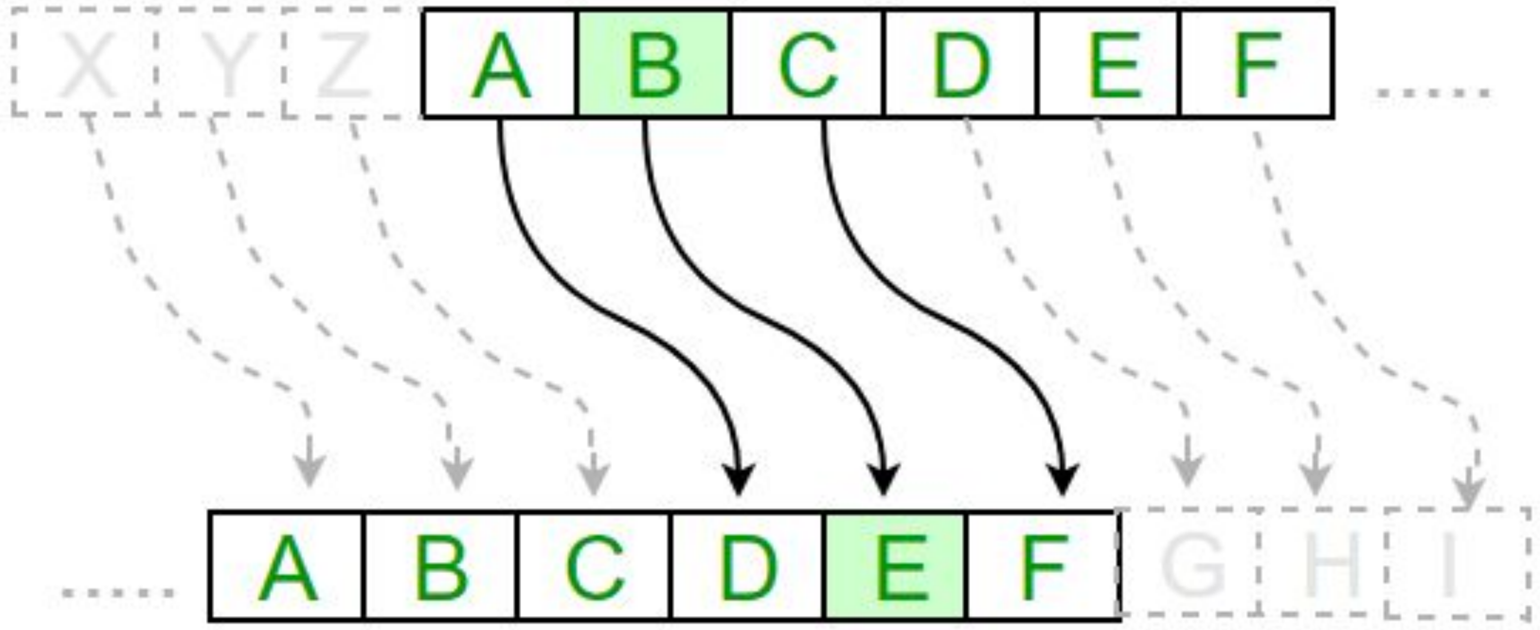
How does it work?

The Caesar cipher is a substitution cipher. This means that each letter in the plaintext is shifted a certain number of places down or up the alphabet. For example, with a shift of 3:

- **A** becomes **D**
- **B** becomes **E**
- **C** becomes **F**

and so on. When you reach the end of the alphabet, it wraps around to the beginning.

For instance, if you encrypt the word **"HELLO"** with a shift of 3, it becomes **"KHOOR"**.


X
Y
Z
A
B
C
D
E
F
A
B
C
D
E
F
G
H
I

How can we code this?

We can do this employing two useful components:

1. ASCII Table
2. Modulus Operator (**%**)

ASCII Table

- ASCII is a character encoding standard used in computers and electronic devices to represent text. Each character (like letters, numbers, and symbols) is assigned a unique numerical code.

For example:

- '**A**' is **65**
- '**B**' is **66**
- '**C**' is **67**
- And so on up to '**Z**' which is **90**

Similarly, lowercase letters '**a**' to '**z**' are mapped from **97** to **122**.

- When implementing a Caesar cipher in a programming language, understanding ASCII is crucial because it allows us to manipulate characters using their numerical codes.

ASCII Table

Dec = Decimal Value
Char = Character

'5' has the int value 53
if we write '5'-'0' it evaluates to 53-48, or the int 5
if we write char c = 'B'+32; then c stores 'b'

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Modulus Operator

- In the Caesar cipher, the modulus operator (`**%**`) helps us shift letters in a way that keeps everything inside the alphabet.
- For example, when we shift '**Z**' by **1**, it loops back to '**A**'.
- This makes sure our encoded messages stay readable and can be decoded using the same method.

CODE REVIEW TIME