Affine Cipher -

The Affine cipher is a type of monoalphabetic substitution cipher, wherein each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. The formula used means that each letter encrypts to one other letter, and back again, meaning the cipher is essentially a standard substitution cipher with a rule governing which letter goes to which.
The whole process relies on working modulo m (the length of the alphabet used). In the affine cipher, the letters of an alphabet of size m are first mapped to the integers in the range 0 … m-1.

The 'key' for the Affine cipher consists of 2 numbers, we'll call them a and b. The following discussion assumes the use of a 26 character alphabet (m = 26). a should be chosen to be relatively prime to m (i.e. a should have no factors in common with m).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Encryption
E (x) = (a x + b) mod m
modulus m: size of the alphabet
a and b: key of the cipher.
a must be chosen such that a and m are coprime.

Decryption
D (x) = a^-1 (x - b) mod m
a^-1 : modular multiplicative inverse of a modulo m. i.e., it satisfies the equation
1 = a a^-1 mod m .

To find a multiplicative inverse

```
[g,x,d] = gcd(a,m);    % we can ignore g and d, we dont need them
x = mod(x,m);
```

Implementation of Affine Cipher -

```python
def egcd(a, b):
    x,y, u,v = 0,1, 1,0
    while a != 0:
        q, r = b//a, b%a
        m, n = x-u*q, y-v*q
        b,a, x,y, u,v = a,r, u,v, m,n
    gcd = b
    return gcd, x, y

def modinv(a, m):
    gcd, x, y = egcd(a, m)
    if gcd != 1:
        return None  # modular inverse does not exist
    else:
        return x % m


# affine cipher encryption function
# returns the cipher text
def affine_encrypt(text, key):
    '''
    C = (a*P + b) % 26
    '''
    return ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26)
                + ord('A')) for t in text.upper().replace(' ', '') ])


# affine cipher decryption function
# returns original text
def affine_decrypt(cipher, key):
    '''
    P = (a^-1 * (C - b)) % 26
    '''
    return ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') - key[1]))
                % 26) + ord('A')) for c in cipher ])


# Driver Code to test the above functions
def main():
    # declaring text and key
    text = 'AFFINE CIPHER'
    key = [17, 20]

    # calling encryption function
    affine_encrypted_text = affine_encrypt(text, key)

    print('Encrypted Text: {}'.format( affine_encrypted_text ))

    # calling decryption function
    print('Decrypted Text: {}'.format
```

Output -

```
┌──(kali㉿kali)-[~]
└─$ python affine.py
Encrypted Text: UBBAHKCAPJKX
Decrypted Text: AFFINECIPHER
```