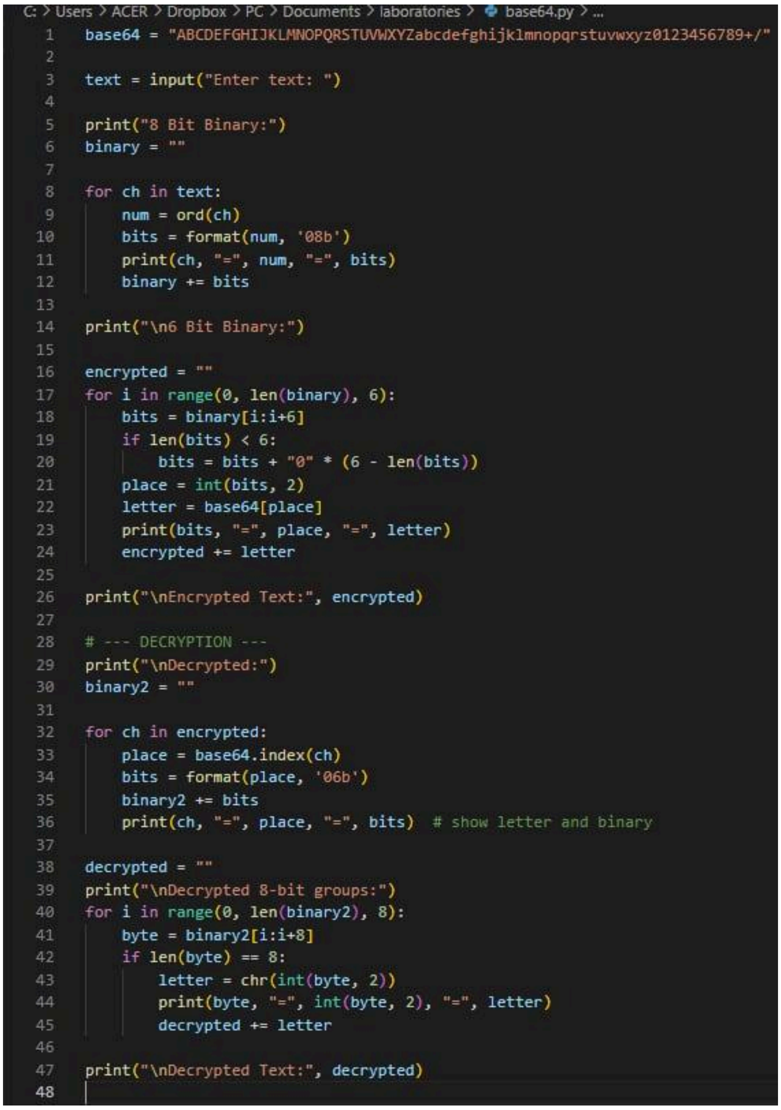
Renoel Jersean O. Ocampo
BSIT4-3

BASE64


C: > Users > ACER > Dropbox > PC > Documents > laboratories > base64.py > ...

```python
base64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

text = input("Enter text: ")

print("8 Bit Binary:")
binary = ""

for ch in text:
    num = ord(ch)
    bits = format(num, '08b')
    print(ch, "=", num, "=", bits)
    binary += bits

print("\n6 Bit Binary:")

encrypted = ""
for i in range(0, len(binary), 6):
    bits = binary[i:i+6]
    if len(bits) < 6:
        bits = bits + "0" * (6 - len(bits))
    place = int(bits, 2)
    letter = base64[place]
    print(bits, "=", place, "=", letter)
    encrypted += letter

print("\nEncrypted Text:", encrypted)

# --- DECRYPTION ---
print("\nDecrypted:")
binary2 = ""

for ch in encrypted:
    place = base64.index(ch)
    bits = format(place, '06b')
    binary2 += bits
    print(ch, "=", place, "=", bits)  # show letter and binary

decrypted = ""
print("\nDecrypted 8-bit groups:")
for i in range(0, len(binary2), 8):
    byte = binary2[i:i+8]
    if len(byte) == 8:
        letter = chr(int(byte, 2))
        print(byte, "=", int(byte, 2), "=", letter)
        decrypted += letter

print("\nDecrypted Text:", decrypted)

```

First, it defines the Base64 alphabet (a-z, 0-9, +, /). The user enters plaintext, then format(ord(ch), '08b') converts each character to 8-bit binary. These 8-bit segments are merged into a single lengthy binary string.

ENCRYPTION
Next, the program divides the binary string into 6-bit groups since Base64 represents data in 6-bit chunks. If the last group has fewer than six bits, it inserts more zeros to complete it. Each 6-bit value is transformed to a decimal integer, which is then used as an index to locate the appropriate character in the Base64 alphabet. These characters make up the encrypted (encoded) Base64 text.

DECRYPTION
To decrypt, the program reverses the procedure.
It translates each Base64 letter back to its 6-bit binary counterpart (depending on its index in the Base64 alphabet), then connects all of the 6-bit groups together to form 8-bit bytes. Each 8-bit chunk is then transformed back into an ASCII character using chr(int(byte, 2)), resulting in the reconstruction of the original plaintext.