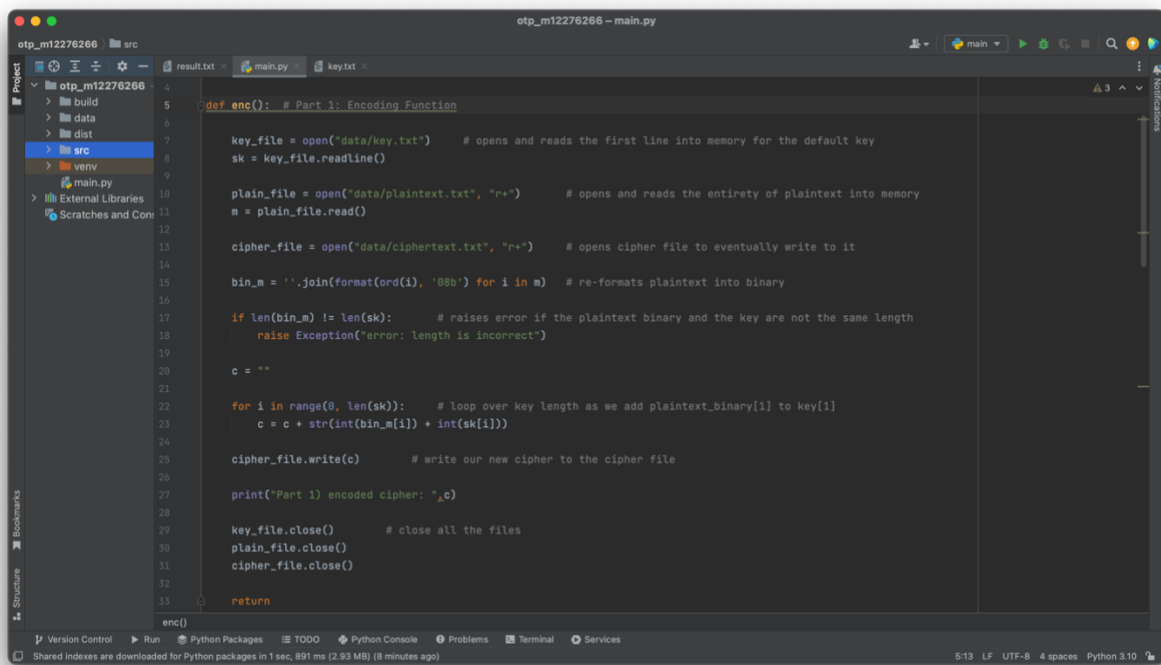


Name: Tamara Myers
M12276266

I used my M1 Mac for most of this project. I used PyCharm as my IDE since I programmed this in python. It was my first time using PyCharm, and it was very easy and powerful. I don't think I will be going back to VS code for python in the future. I have run this code on both VS Code and on PyCharm. All you need to do is open the otp_12276266 folder within VS code or PyCharm, then hit run on the main.py file.

Encoding Function:



```
def enc(): # Part 1: Encoding Function
    key_file = open("data/key.txt") # opens and reads the first line into memory for the default key
    sk = key_file.readline()

    plain_file = open("data/plaintext.txt", "r") # opens and reads the entirety of plaintext into memory
    m = plain_file.read()

    cipher_file = open("data/ciphertext.txt", "r+") # opens cipher file to eventually write to it

    bin_m = ''.join(format(ord(i), '08b') for i in m) # re-formats plaintext into binary

    if len(bin_m) != len(sk): # raises error if the plaintext binary and the key are not the same length
        raise Exception("error: length is incorrect")

    c = ""

    for i in range(0, len(sk)): # loop over key length as we add plaintext_binary[i] to key[i]
        c = c + str(int(bin_m[i]) + int(sk[i]))

    cipher_file.write(c) # write our new cipher to the cipher file

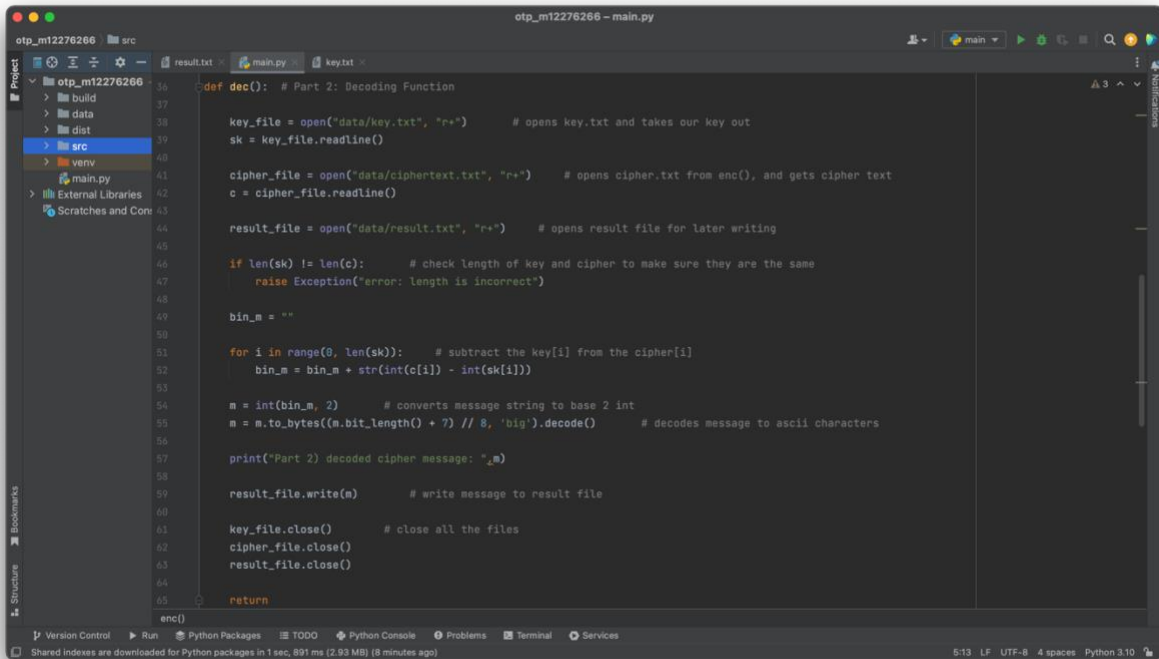
    print("Part 1) encoded cipher: " + c)

    key_file.close() # close all the files
    plain_file.close()
    cipher_file.close()

    return

enc()
```

Decoding:



The screenshot shows a code editor window titled "otp_m12276266 - main.py". The left sidebar displays a project structure with folders like "build", "data", "dist", "src", "venv", and "main.py". The main editor area shows the following Python code:

```
def dec(): # Part 2: Decoding Function
    key_file = open("data/key.txt", "r+") # opens key.txt and takes our key out
    sk = key_file.readline()

    cipher_file = open("data/ciphertext.txt", "r+") # opens cipher.txt from enc(), and gets cipher text
    c = cipher_file.readline()

    result_file = open("data/result.txt", "r+") # opens result file for later writing

    if len(sk) != len(c): # check length of key and cipher to make sure they are the same
        raise Exception("error: length is incorrect")

    bin_m = ""

    for i in range(0, len(sk)): # subtract the key[i] from the cipher[i]
        bin_m = bin_m + str(int(c[i]) - int(sk[i]))

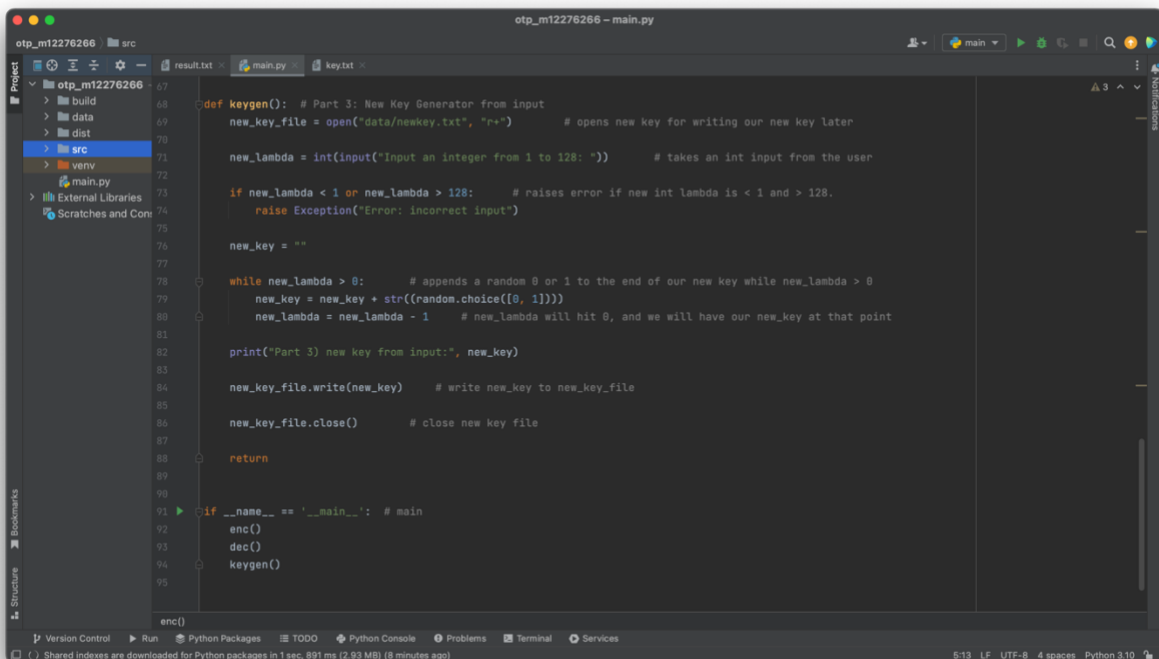
    m = int(bin_m, 2) # converts message string to base 2 int
    m = m.to_bytes((m.bit_length() + 7) // 8, 'big').decode() # decodes message to ascii characters
    print("Part 2) decoded cipher message: ", m)

    result_file.write(m) # write message to result file

    key_file.close() # close all the files
    cipher_file.close()
    result_file.close()

    return enc()
```

Keygen:



The screenshot shows the same code editor window, now displaying the key generation function. The code is as follows:

```
def keygen(): # Part 3: New Key Generator from input
    new_key_file = open("data/newkey.txt", "r+") # opens new key for writing our new key later

    new_lambda = int(input("Input an integer from 1 to 128: ")) # takes an int input from the user

    if new_lambda < 1 or new_lambda > 128: # raises error if new int lambda is < 1 and > 128.
        raise Exception("Error: incorrect input")

    new_key = ""

    while new_lambda > 0: # appends a random 0 or 1 to the end of our new key while new_lambda > 0
        new_key = new_key + str(random.choice([0, 1]))
        new_lambda = new_lambda - 1 # new_lambda will hit 0, and we will have our new_key at that point

    print("Part 3) new key from input:", new_key)

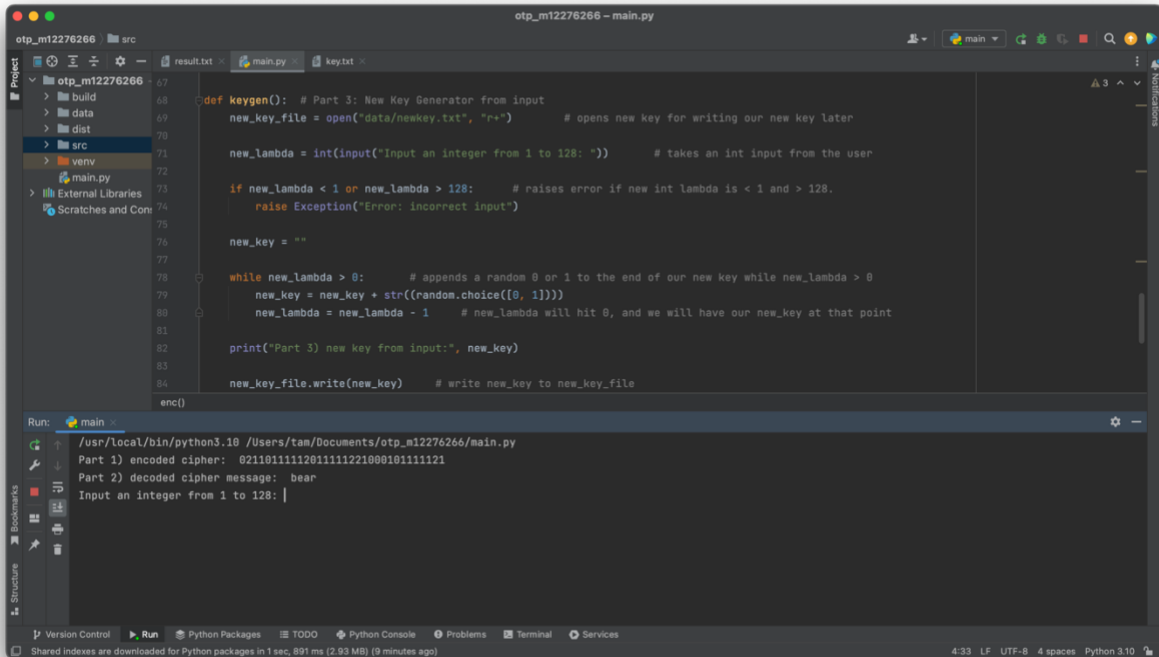
    new_key_file.write(new_key) # write new_key to new_key_file

    new_key_file.close() # close new key file

    return

if __name__ == '__main__': # main
    enc()
    dec()
    keygen()
```

Running the code:

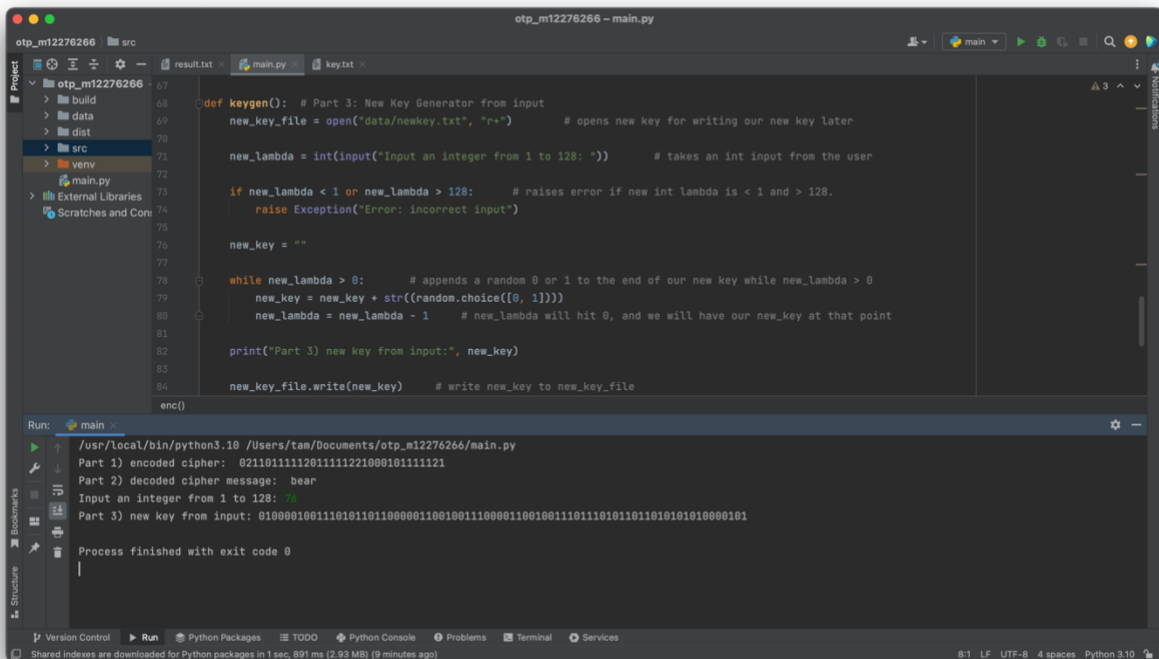


```
otp_m12276266 - main.py
Project: otp_m12276266
  build
  data
  dist
  src
  venv
  main.py
  External Libraries
  Scratches and Con...
result.txt
main.py
key.txt

def keygen(): # Part 3: New Key Generator from input
    new_key_file = open("data/newkey.txt", "r+") # opens new key for writing our new key later
    new_lambda = int(input("Input an integer from 1 to 128: ")) # takes an int input from the user
    if new_lambda < 1 or new_lambda > 128: # raises error if new int lambda is < 1 and > 128.
        raise Exception("Error: incorrect input")
    new_key = ""
    while new_lambda > 0: # appends a random 0 or 1 to the end of our new key while new_lambda > 0
        new_key = new_key + str(random.choice([0, 1]))
        new_lambda = new_lambda - 1 # new_lambda will hit 0, and we will have our new_key at that point
    print("Part 3) new key from input:", new_key)
    new_key_file.write(new_key) # write new_key to new_key_file
enc()

Run: main
/usr/local/bin/python3.10 /Users/tam/Documents/otp_m12276266/main.py
Part 1) encoded cipher: 02110111112011111221000101111121
Part 2) decoded cipher message: bear
Input an integer from 1 to 128: |
```

Inputting "76" for my new key length:



```
otp_m12276266 - main.py
Project: otp_m12276266
  build
  data
  dist
  src
  venv
  main.py
  External Libraries
  Scratches and Con...
result.txt
main.py
key.txt

def keygen(): # Part 3: New Key Generator from input
    new_key_file = open("data/newkey.txt", "r+") # opens new key for writing our new key later
    new_lambda = int(input("Input an integer from 1 to 128: ")) # takes an int input from the user
    if new_lambda < 1 or new_lambda > 128: # raises error if new int lambda is < 1 and > 128.
        raise Exception("Error: incorrect input")
    new_key = ""
    while new_lambda > 0: # appends a random 0 or 1 to the end of our new key while new_lambda > 0
        new_key = new_key + str(random.choice([0, 1]))
        new_lambda = new_lambda - 1 # new_lambda will hit 0, and we will have our new_key at that point
    print("Part 3) new key from input:", new_key)
    new_key_file.write(new_key) # write new_key to new_key_file
enc()

Run: main
/usr/local/bin/python3.10 /Users/tam/Documents/otp_m12276266/main.py
Part 1) encoded cipher: 02110111112011111221000101111121
Part 2) decoded cipher message: bear
Input an integer from 1 to 128: 76
Part 3) new key from input: 01000010011010101100000110010011100001100100110110101101010101010000101
Process finished with exit code 0
```