```
E94121232@raspberrypi:~ $ python server.py
Waiting for connection...
Add connection from ('192.168.137.234', 43568)
Add connection from ('192.168.137.1', 57203)
```

```
E94121232@raspberrypi:~ $ python server.py
Waiting for connection...
Add connection from ('192.168.137.234', 43568)
Add connection from ('192.168.137.1', 57203)
Received from ('192.168.137.1', 57203): 8
Send to ('192.168.137.1', 57203): 21
conection closed
Received from ('192.168.137.234', 43568): 7
Send to ('192.168.137.234', 43568): 13
conection closed
```

```
 GNU nano 7.2                                                    server.py
# _*_ coding: utf-8 _*_
import socket
import threading

HOST = '192.168.137.234'                                #設定伺服器
PORT = 8000

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)   #建立socket
s.bind((HOST, PORT))
s.listen()

def handle_client(conn, addr):
        print(f'Add connection from {str(addr)}')
        a = 0                                           #設定參數
        b = 1
        c = 0
        n = conn.recv(1024).decode()                    #設定n為從客戶端得到並解碼的結果
        if not n:                                       #檢查字串是不是空的
                print(f'Received from {str(addr)}: {n}')
        if n == 'exit':
                print(f'Received from {str(addr)}: {n}')        #客戶端輸入exit時跳出迴圈中斷連線
        n = int(n)                                      #接收到的n為str 需要轉成int
        print(f'Received from {str(addr)}: {n}')
        if n == 0 or n == 1 :                           #根據n的值產生結果
                print(f'Send to {str(addr)}: {n}')
                conn.send(str(n).encode())              #將結果回傳給客戶端
        else:
                for i in range(n-1):
                        c = b;
                        b = b + a;
                        a = c;
                print(f'Send to {str(addr)}: {b}')
                conn.send(str(b).encode())
        print('conection closed')
        conn.close()                                    #迴傳後中斷連線

print('Waiting for connection...')
while True:                                             #每收到一個Client連線時,開一個新的Thread給那個Client

^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute   ^C Location  M-U Undo     M-A Set Mark  M-] To Bracket
```

```
  GNU nano 7.2                                                server.py
PORT = 8000

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)           #建立socket
s.bind((HOST, PORT))
s.listen()

def handle_client(conn, addr):
        print(f'Add connection from {str(addr)}')
        a = 0                                                   #設定參數
        b = 1
        c = 0
        n = conn.recv(1024).decode()                            #設定n為從客戶端得到並解碼的結果
        if not n:                                               #檢查字串是不是空的
                print(f'Received from {str(addr)}: {n}')
        if n == 'exit':
                print(f'Received from {str(addr)}: {n}')        #客戶端輸入exit時跳出迴圈中斷連線
        n = int(n)                                              #接收到的n為str 需要轉成int
        print(f'Received from {str(addr)}: {n}')
        if n == 0 or n == 1 :                                   #根據n的值產生結果
                print(f'Send to {str(addr)}: {n}')
                conn.send(str(n).encode())                      #將結果回傳給客戶端
        else:
                for i in range(n-1):
                        c = b;
                        b = b + a;
                        a = c;
                print(f'Send to {str(addr)}: {b}')
                conn.send(str(b).encode())
        print('conection closed')
        conn.close()                                            #迴傳後中斷連線

print('Waiting for connection...')
while True:                                                     #每收到一個Client連線時,開一個新的Thread給那個Client
        conn, addr = s.accept()
        client_handler = threading.Thread(target = handle_client, args=(conn, addr))
        client_handler.start()
s.close()

^G Help        ^O Write Out     ^W Where Is      ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket
```