1.Modify your greeting program so that if the user does not enter a name (i.e. they just press enter), the program responds "Hello, Stranger!". Otherwise, it should print a greeting with their name as before.

Code:

```
my_name=input("Hello, What is your name? ")
if my_name=="":
    print("Hello Stranger!")
else:
    print(f"Hello, {my_name}. Good to meet you! ")
```

Output:

```
Hello, What is your name?  Apeal
Hello, Apeal. Good to meet you!
```

Figure 1: When user enter name

```
Hello, What is your name?
Hello Stranger!
```

Figure 2: When user don't enter name

2. Write a program that simulates the way in which a user might choose a password. The program should prompt for a new password and then prompt again. If the two passwords entered are the same the program should say "Password Set" or similar, otherwise it should report an error.

Code:

```
password_1 = input("Enter your new password: ")
password_2 = input("Re_enter your new password: ")

if password_1==password_2:
    print("Password Set")
else:
    print("Error: Password do not match. Please try again..")
```

Output:

```
Enter your new password:  apeal
Re_enter your new password:  apeal
Password Set
```

3. Modify your previous program so that the password must be between 8 and 12 characters (inclusive) long.

Code:

```
password_1 = input("Enter your new password: ")

if len(password_1) < 8 or len(password_1) > 12:
    print("Error: Password must be between 8 and 12 characters long. Please try again.")
else:
    password_2 = input("Re_enter your new password: ")

    if password_1==password_2:
        print("Password Set")
    else:
        print("Error: Password do not match. Please try again..")
```

Output:

```
Enter your new password:  apeal123456
Re_enter your new password:  apeal123456
Password Set
```

4. Modify your program again so that the chosen password cannot be one of a list of common passwords, defined thus: BAD_PASSWORDS = ['password', 'letmein', 'sesame', 'hello', 'justinbieber']

Code:

```
BAD_PASSWORDS = ['password', 'letmein', 'sesame', 'hello', 'justinbieber']
password_1 = input("Enter your new password: ")

if password_1 in BAD_PASSWORDS:
    print("Error: Password cannot be one of the following: ['password', 'letmein', 'sesame',
'hello', 'justinbieber']!")
elif len(password_1) < 8 or len(password_1) > 12:
    print("Error: Password must be between 8 and 12 characters long. Please try again.")
else:
    password_2 = input("Re-enter your new password: ")

    if password_1 == password_2:
        print("Password Set")
    else:
        print("Error: Passwords do not match. Please try again.")
```

Output:

```
Enter your new password:  password
Error: Password cannot be one of the following: ['password', 'letmein', 'sesame', 'hello', 'justinbieber']!
```

Figure 3: When user try to Set password which found inside BAD_PASSWORDS

5. Modify your program a final time so that it executes until the user successfully chooses a password. That is, if the password chosen fails any of the checks, the program should return to asking for the password the first time.

Code:

```
while True:
   BAD_PASSWORDS = ['password', 'letmein', 'sesame', 'hello', 'justinbieber']
   password_1 = input("Enter your new password: ")

   if password_1 in BAD_PASSWORDS:
     print("Password connot be ['password', 'letmein', 'sesame', 'hello', 'justinbieber']!")
     continue
   if len(password_1) < 8 or len(password_1) > 12:
       print("Error: Password must be between 8 and 12 characters long. Please try again.")
       continue
   password_2 = input("Re_enter your new password: ")

   if password_1 == password_2:
    print("Password Set")
    break
   else:
    print("Error: Password do not match. Please try again..")
```

Output:

```
Enter your new password:  password
Password connot be ['password', 'letmein', 'sesame', 'hello', 'justinbieber']!
Enter your new password:  ok
Error: Password must be between 8 and 12 characters long. Please try again.
Enter your new password:  apeal123456
Re_enter your new password:  apeal123456
Password Set
```

6. Write a program that displays the "Seven Times Table". That is, the result of multiplying 7 by every number from 0 to 12 inclusive. The output might start:

0 x 7 = 0

1 x 7 = 7

2 x 7 = 14

and so on.

Code:

```
i = 7
for count in range(0,13):
    print(f"{count} * {i} = {count*i}")
```

Output:

```
0 * 7 = 0
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
10 * 7 = 70
11 * 7 = 77
12 * 7 = 84
```

7. Modify your "Times Table" program so that the user enters the number of the table they require. This number should be between 0 and 12 inclusive.

Code:

```
i=int(input("Enter your number for the Times Table: "))
for count in range(0,13):
    print(f"{count} * {i} = {count*i}")
```

Output:

```
Enter your number for the Times Table:  2
0 * 2 = 0
1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
4 * 2 = 8
5 * 2 = 10
6 * 2 = 12
7 * 2 = 14
8 * 2 = 16
9 * 2 = 18
10 * 2 = 20
11 * 2 = 22
12 * 2 = 24
```

8. Modify the "Times Table" again so that the user still enters the number of the table, but if this number is negative the table is printed backwards. So, entering "-7" would produce the Seven Times Table starting at "12 times" down to "0 times".

Code:

```
i = int(input("Enter a number for the Times Table: "))
if i < 0:
    for count in range(12, -1, -1):
        print(f"{count} * {i} = {count * i}")
else:
    for count in range(13):
        print(f"{count} * {i} = {count * i}")
```

Output:

```
Enter a number for the Times Table:  -5
12 * -5 = -60
11 * -5 = -55
10 * -5 = -50
9 * -5 = -45
8 * -5 = -40
7 * -5 = -35
6 * -5 = -30
5 * -5 = -25
4 * -5 = -20
3 * -5 = -15
2 * -5 = -10
1 * -5 = -5
0 * -5 = 0
```

Figure 4: When input number is negative

```
Enter a number for the Times Table:  5
0 * 5 = 0
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50
11 * 5 = 55
12 * 5 = 60
```

Figure 5: When input number is positive