**Vishal Jha**

**Python Assessment – 29/01/24**

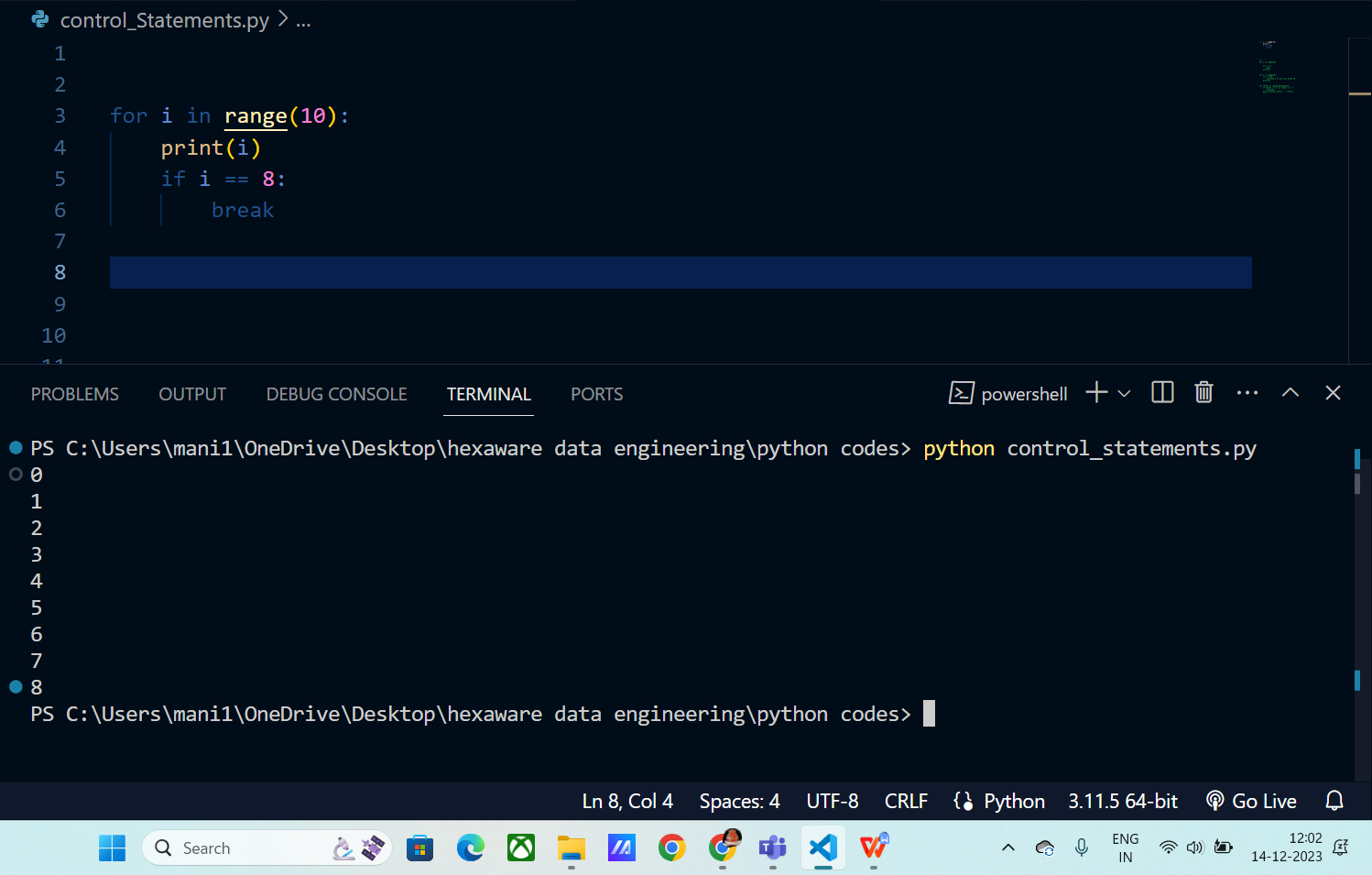
Topics covered:

* loop control statements like break , continue, pass
* While
* operators
* list slicing

Today we discussed about loop control statements like break, continue and pass.

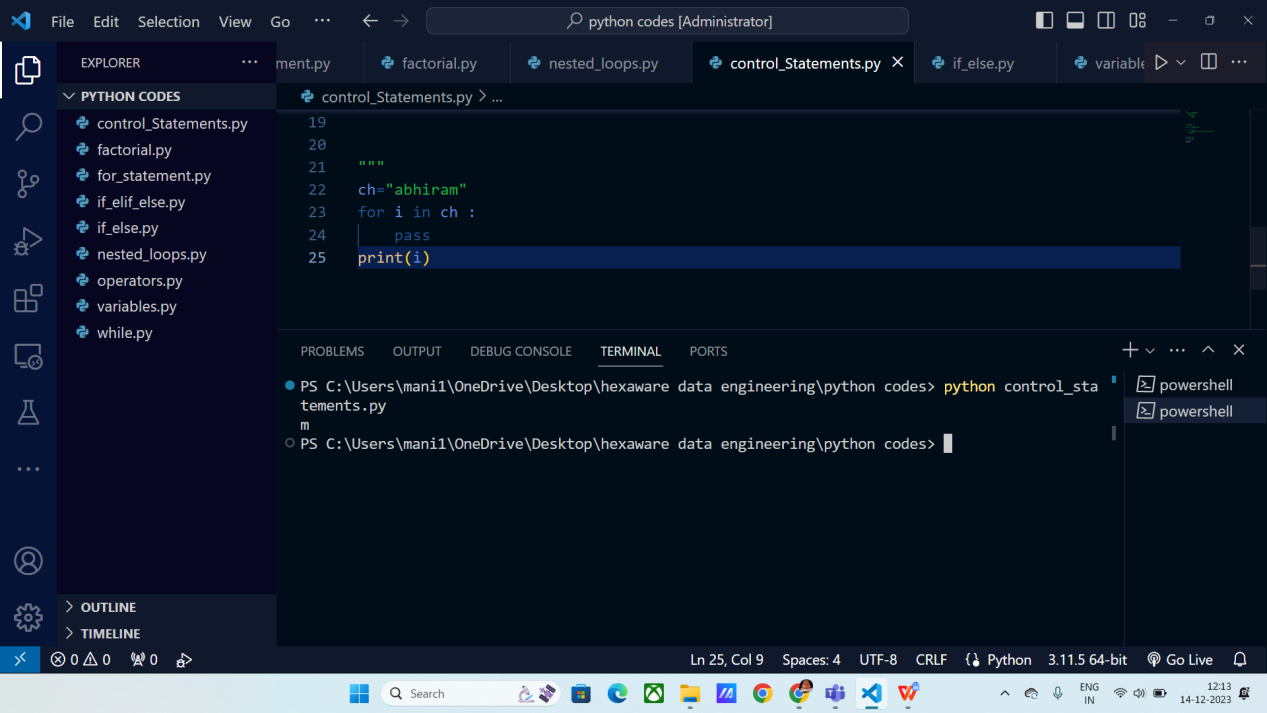
**Break statement** :

It will terminate the loop.



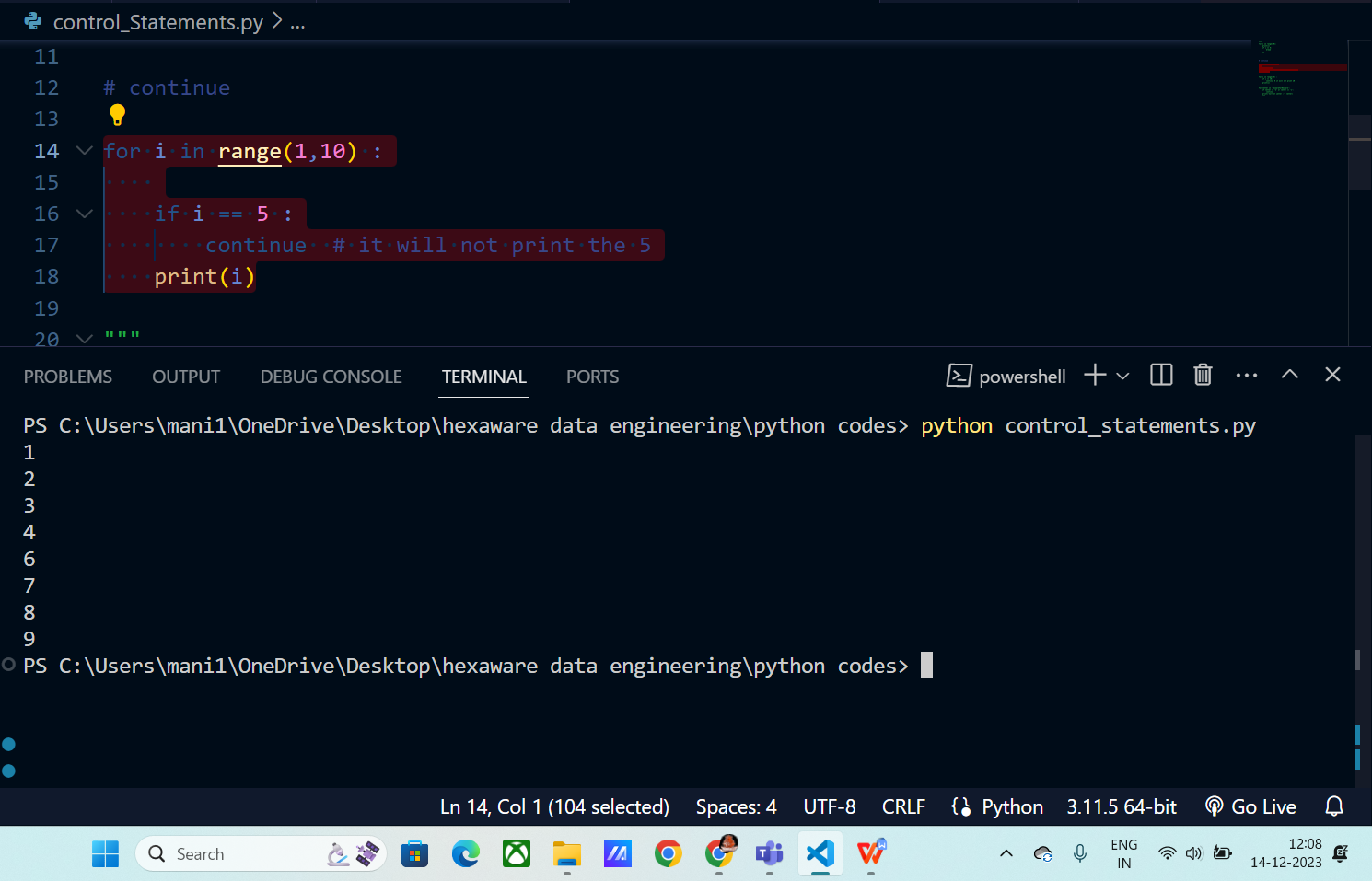
**Pass statement** :

It is used to make an empty loop.

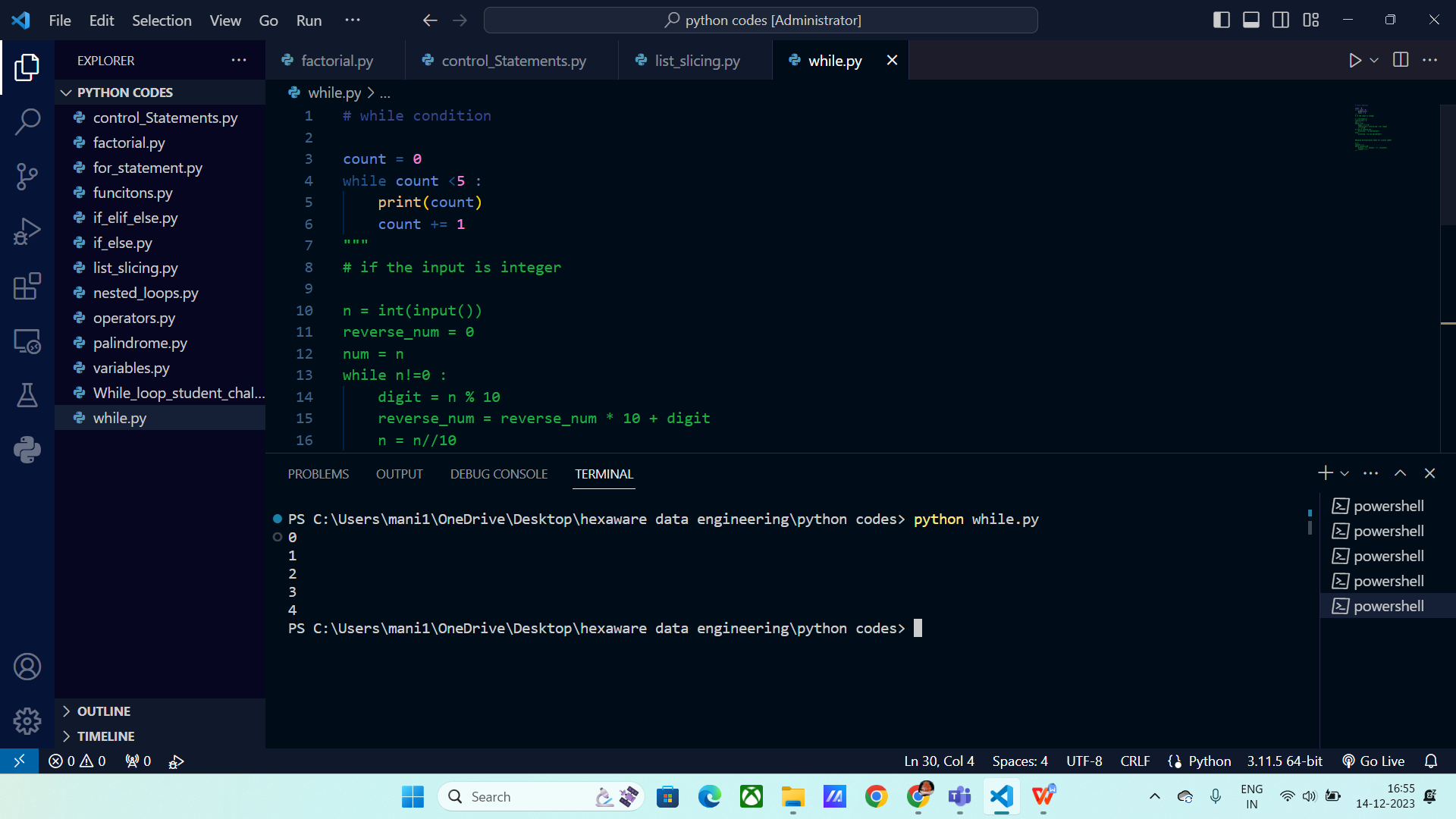


**Continue statement :**

It will returns the control to beginning of the loop.

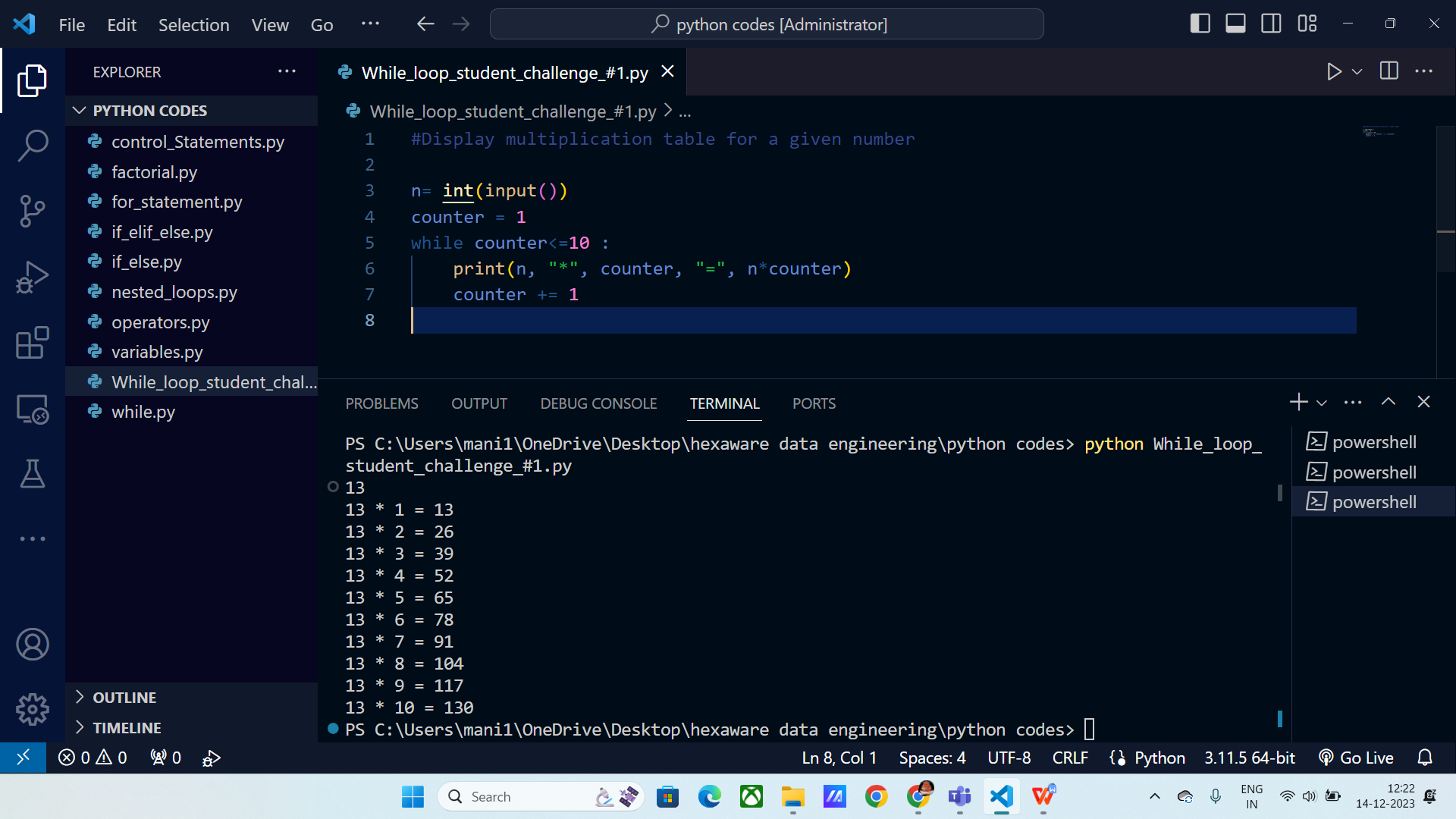


**While :**

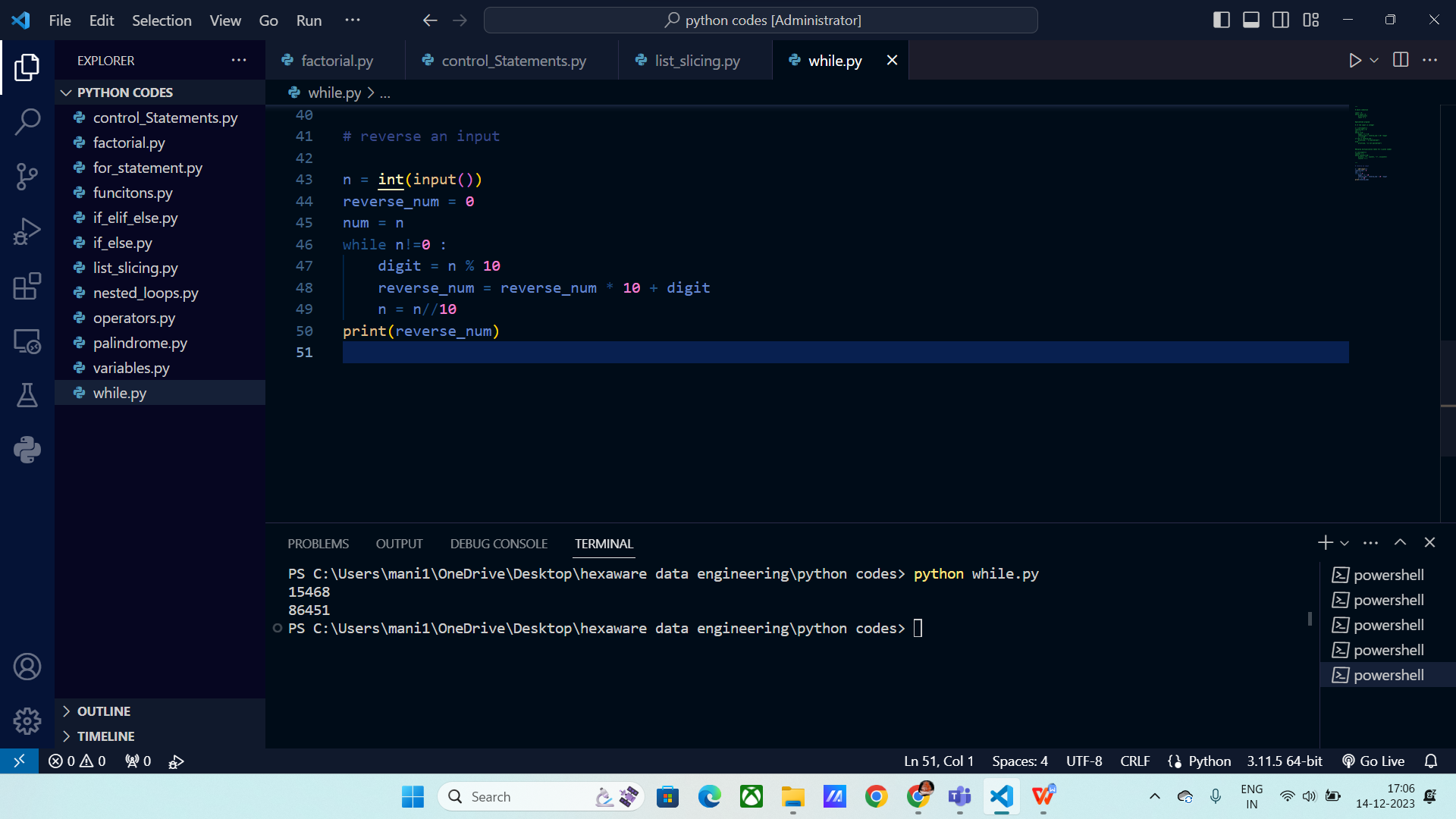


**Displaying table using While loop for given number :**

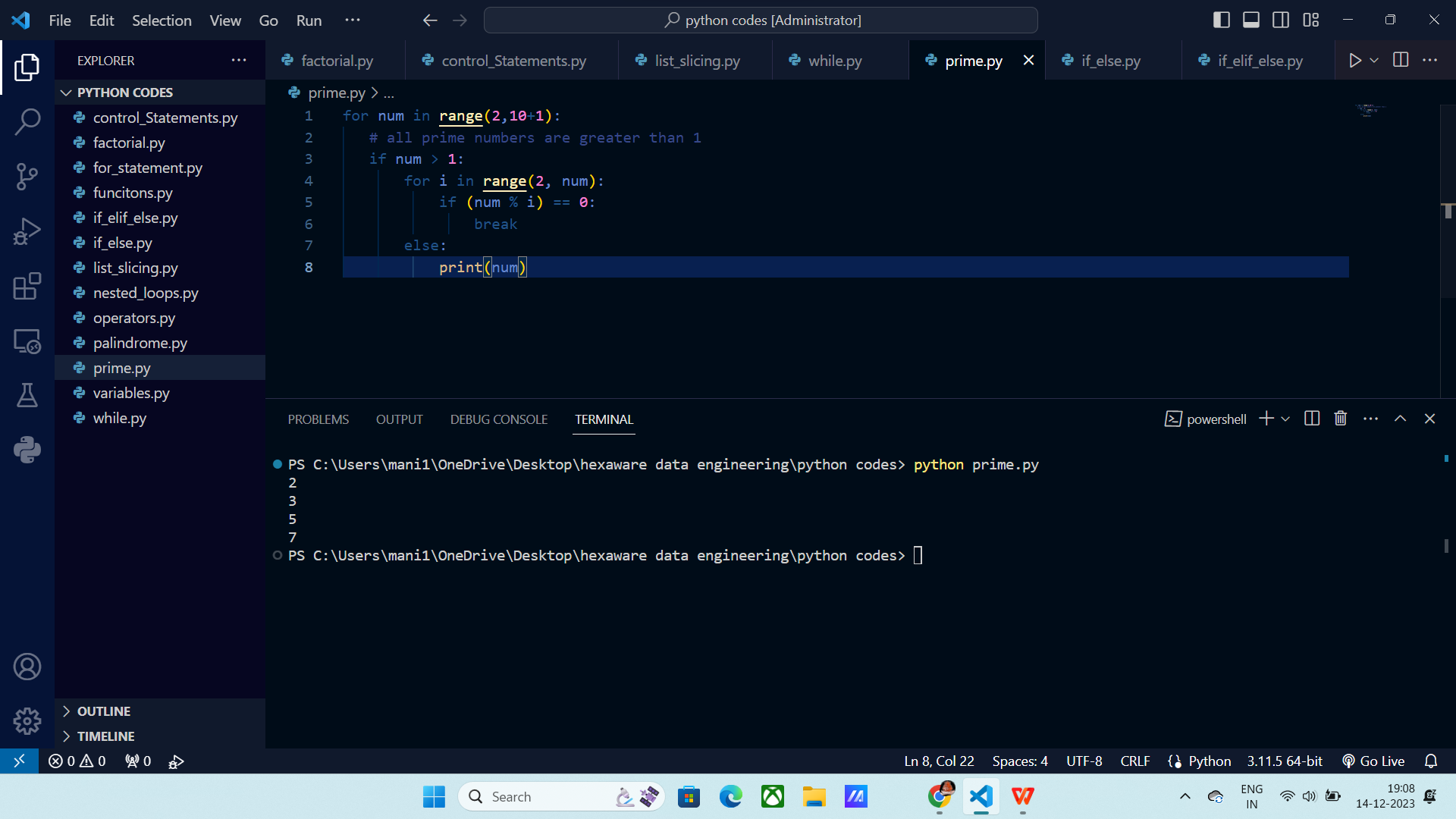
Every time, the output is printed we will increase the counter variable to move to next number.



**Reversing the number:**



**Prime or not :**



**Operators :**

Operators are used to perform operations on variables and values.

Operators are categorized into the following:

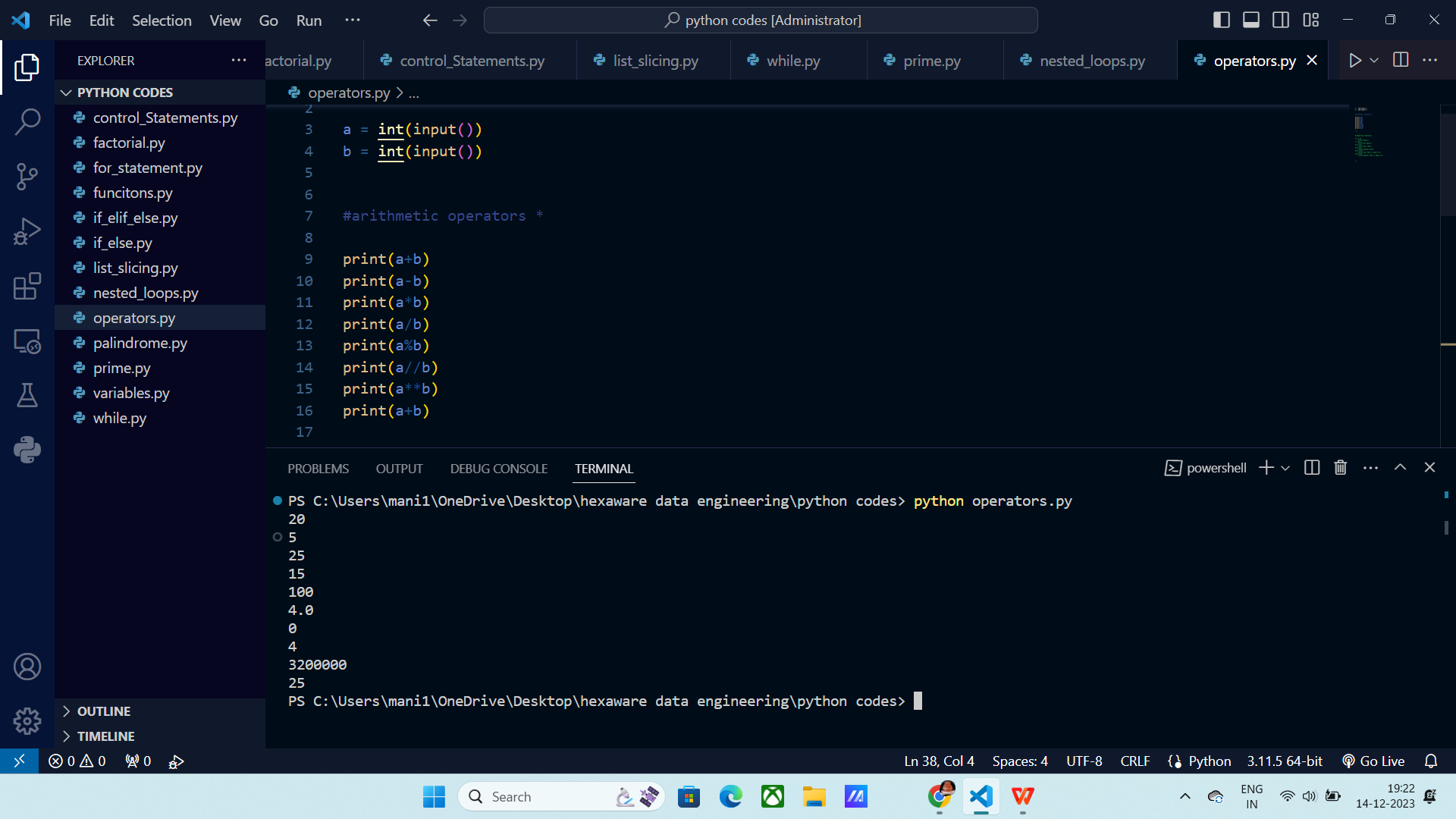
* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

**Arithmetic operators:**

**+, -, \*, /, %, //, \*\*,**

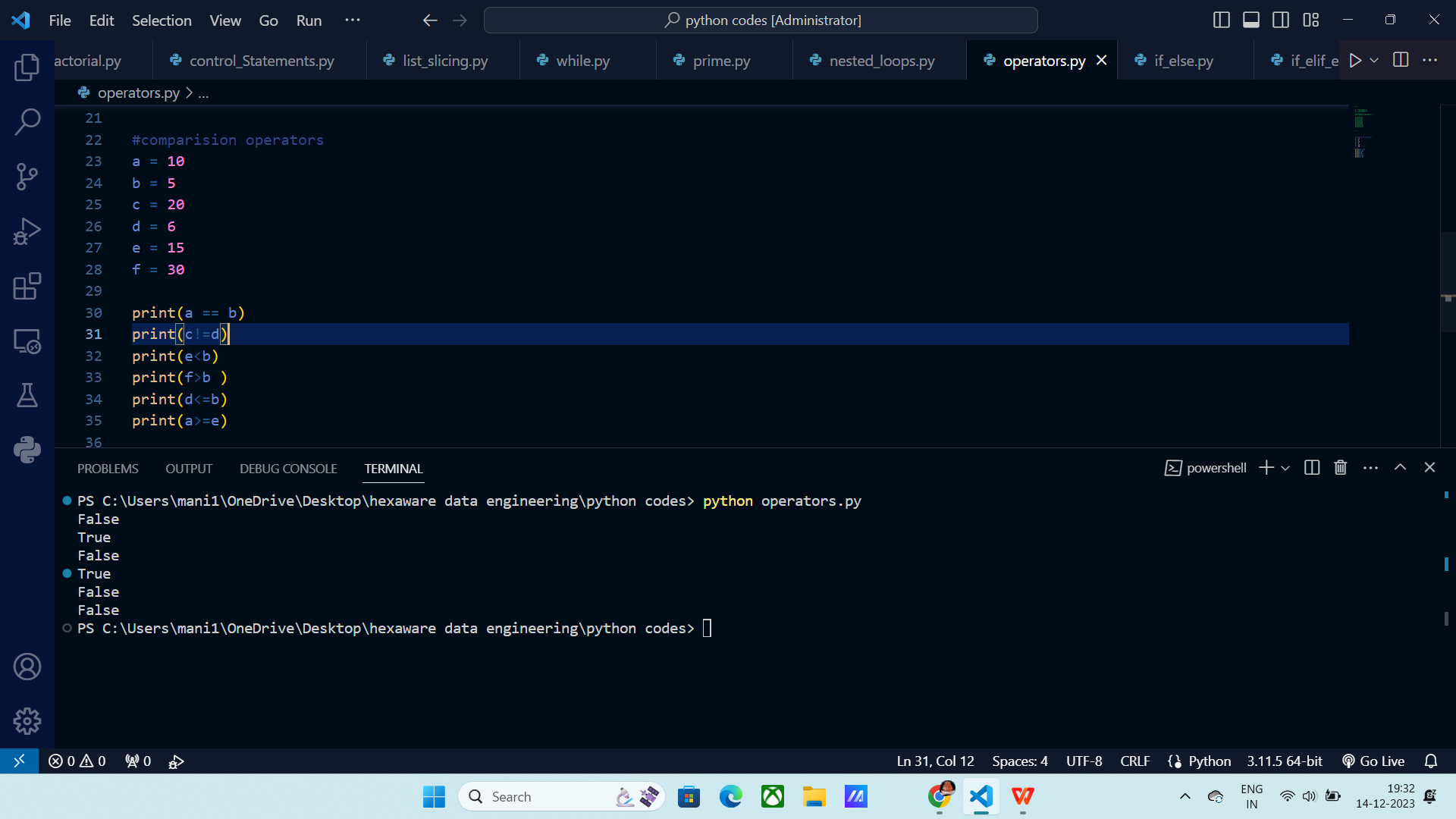
% :gives the remainder

\*\* : power of number



**Comparison operators:**

**==, !=, <, >, <=, >=**

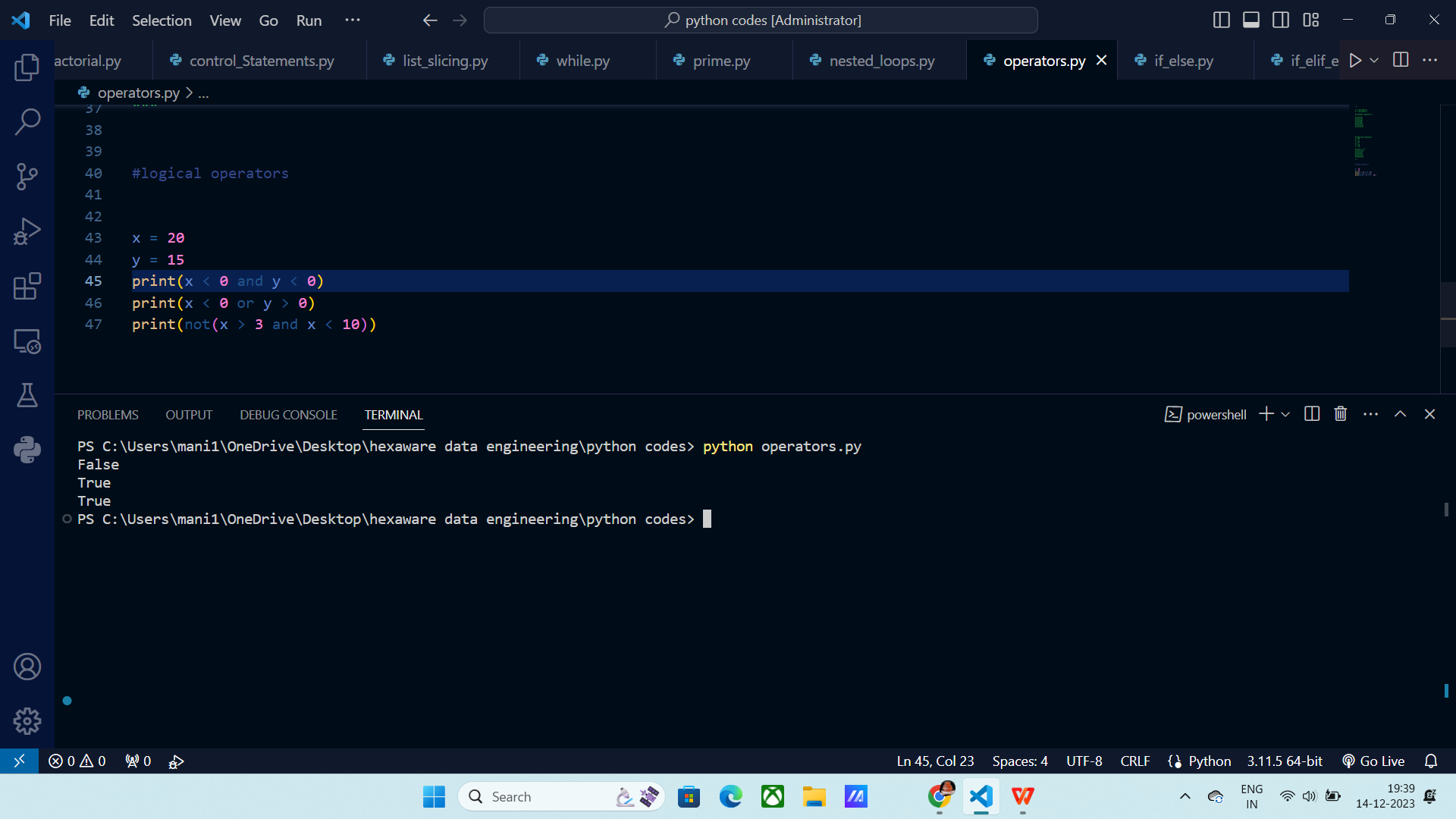


Logical Operators:

**And :** Both the conditions should be right then only it will give True, else False.

**or :** Any one of the condition should be True it will give True

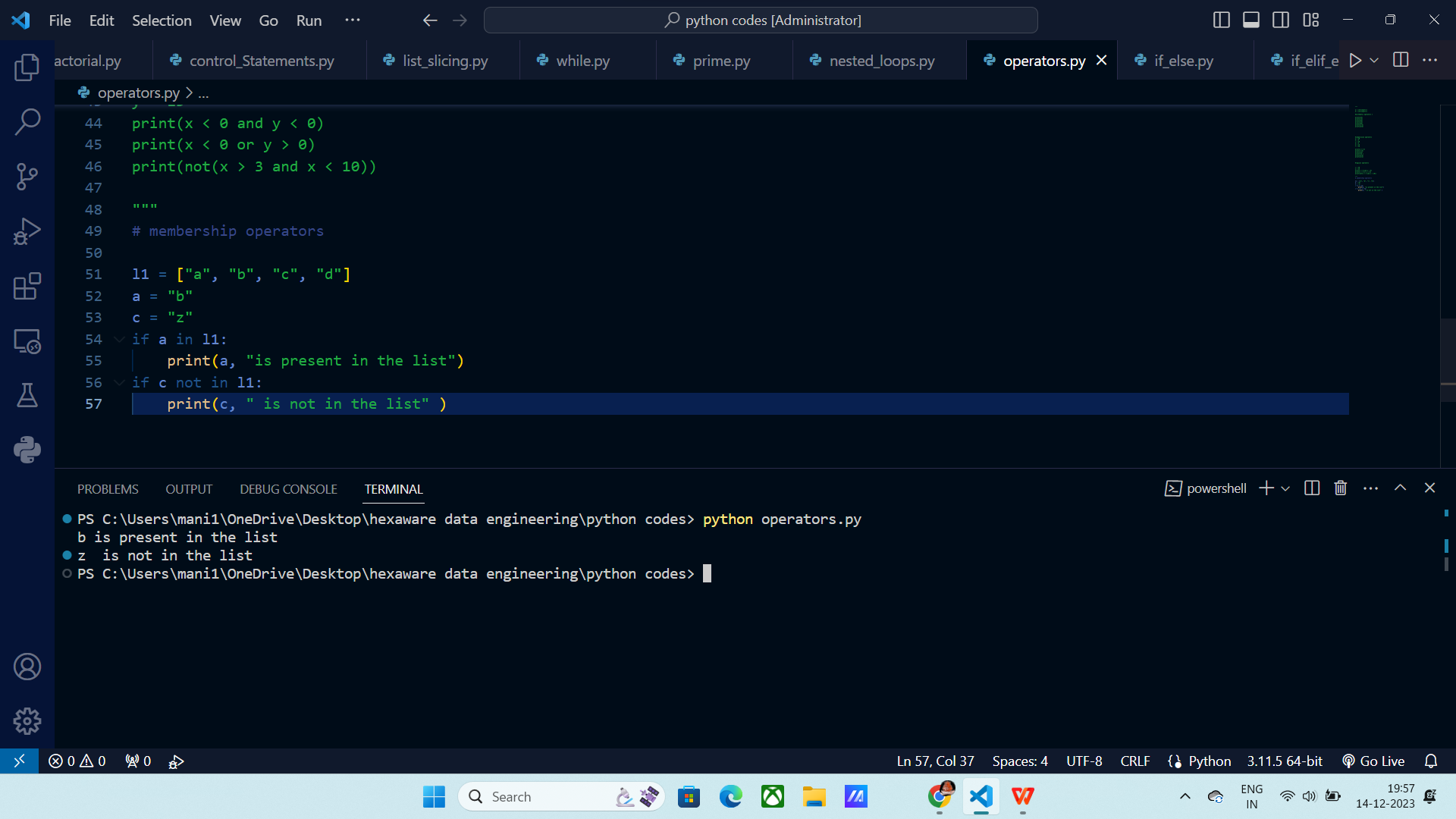
**Not :** It makes True to false and false to True.



Membership Operators :

IN : It returns True, if the item is present.

Not in : It checks whether an item is not present and returns True.



Bit wise operators: These returns the equilant numbers using truth tables.

**& table**:

1 0 | 0

0 1 | 0

0 0 | 0

1 1 | 1

| ( or ) table

0 0 | 0

0 1 | 1

1 0 | 1

1 1 | 1

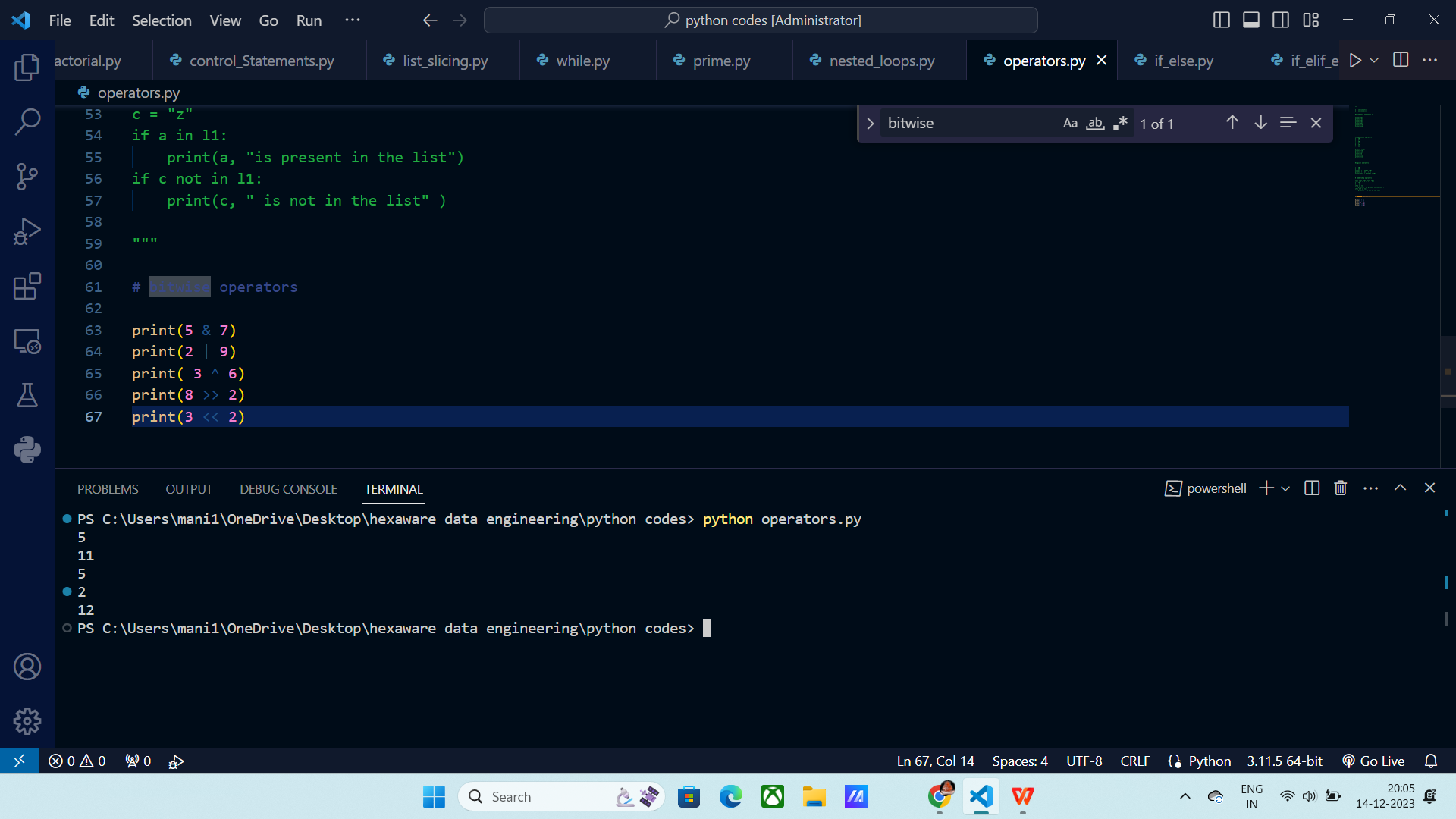
^ (xor) table:

0 0 | 0

1 0 | 1

0 1 | 1

1 1 | 0



**Slicing :**

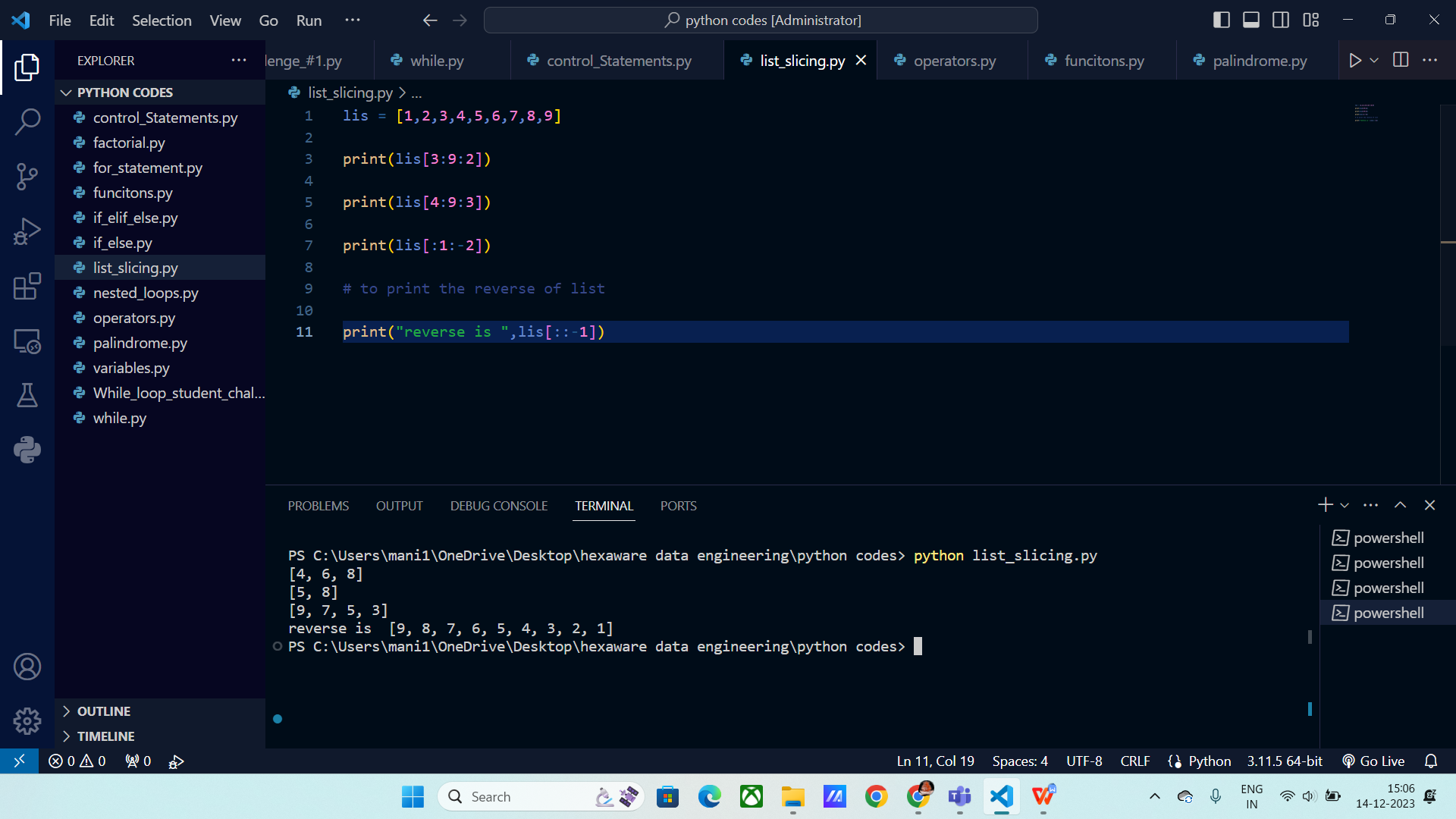
It will give you sub string from a string.

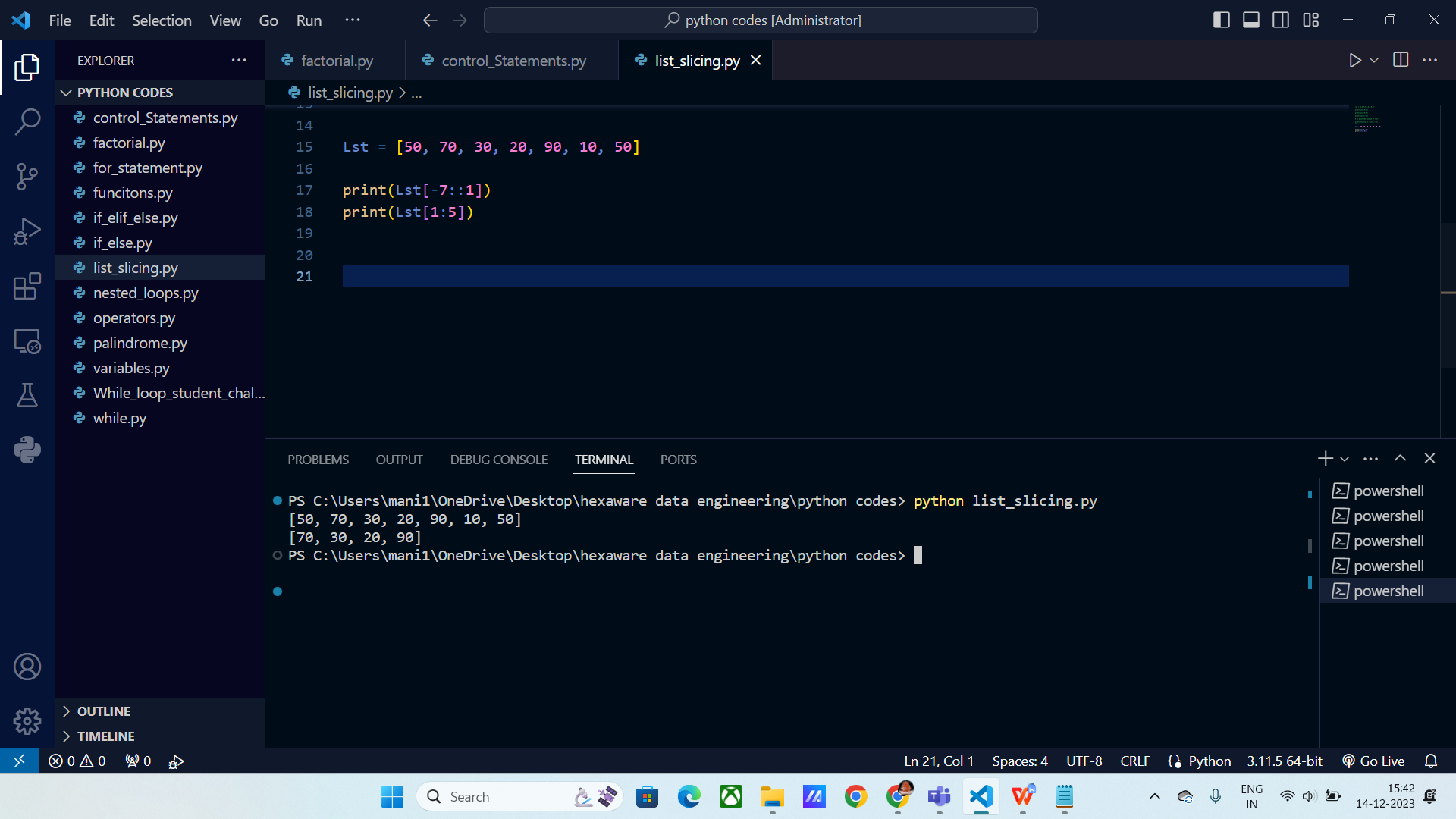
Syntax:

[start : stop : step count ]

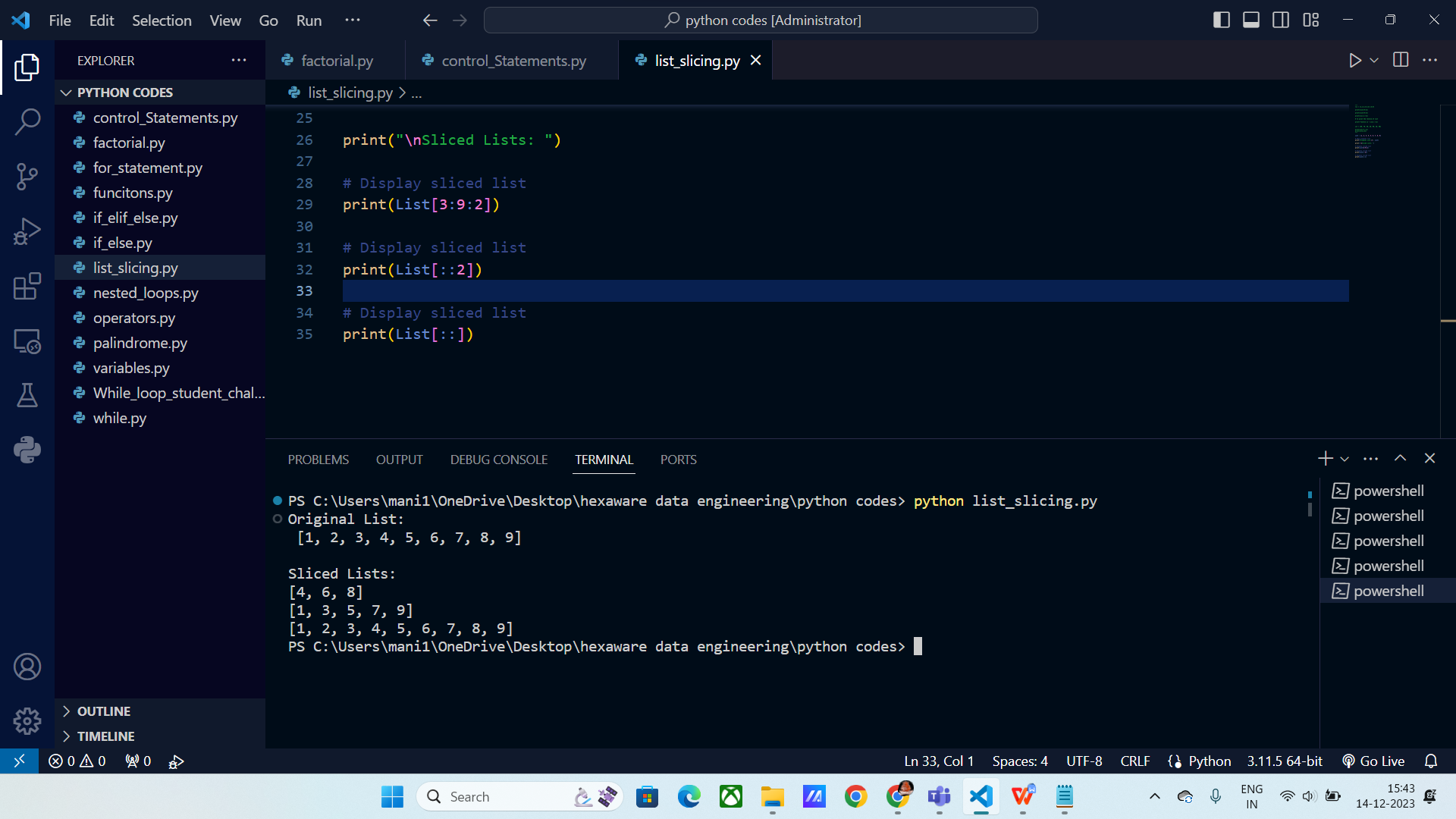
Indexing starts with 0,1,2….N

Negative indexing starts with -1,-2, …





**List Slicing :**



Modifying the lists:

