1. Arithmetic Operators:

main.py

```python
#Arithmetic operators
a= 56
b= 4

print("For a=",a,"and b=",b,"\nCalculations:")

print("1. Addition:",a+b)
print("2. Subtraction:",a-b)
print("3. Subtraction:",a*b)
print("4. Subtraction:",a/b)
print("5. Subtraction:",a//b)
print("6. Subtraction:",a%b)
print("7. Subtraction:",a**b)
```

Output

```
For a= 56 and b= 4
Calculations:
1. Addition: 60
2. Subtraction: 52
3. Subtraction: 224
4. Subtraction: 14.0
5. Subtraction: 14
6. Subtraction: 0
7. Subtraction: 9834496

=== Code Execution Successful ===
```

2. Comparison operators:

main.py

```python
#Comparision operators
a = 56
b = 4

print("For a =", a, "and b =", b,"\nCheck the following:")

print('1. Two numbers are equal or not:', a == b)
print('2. Two numbers are not equal or not:', a != b)
print('3. a is less than or equal to b:', a <= b)
print('4. a is greater than or equal to b:', a >= b)
print('5. a is greater b:', a > b)
print('6. a is less than b:', a < b)
```

Output

```
For a = 56 and b = 4
Check the following:
1. Two numbers are equal or not: False
2. Two numbers are not equal or not: True
3. a is less than or equal to b: False
4. a is greater than or equal to b: True
5. a is greater b: True
6. a is less than b: False

=== Code Execution Successful ===
```

3. Assignment operators:

main.py

```python
#Assignment operators
a = 56
b = 4

# printing the different results
print('a += b:', a + b)
print('a -= b:', a - b)
print('a *= b:', a * b)
print('a /= b:', a / b)
print('a %= b:', a % b)
print('a **= b:', a ** b)
print('a //= b:', a // b)
```

Output

```
a += b: 60
a -= b: 52
a *= b: 224
a /= b: 14.0
a %= b: 0
a **= b: 9834496
a //= b: 14

=== Code Execution Successful ===
```

4. Bitwise operators:

main.py

```python
#Bitwise operators
a = 7
b = 8

# printing different results
print('a & b :', a & b)
print('a | b :', a | b)
print('a ^ b :', a ^ b)
print('~a :', ~a)
print('a << b :', a << b)
print('a >> b :', a >> b)
```

Output

```
a & b : 0
a | b : 15
a ^ b : 15
~a : -8
a << b : 1792
a >> b : 0

=== Code Execution Successful ===
```

5. Logical operators:

main.py Output

```python
#Logical operators
a = 7

# printing different results
print("For a = 7, checking whether the following conditions are True or False:")
print('\"a > 5 and a < 7\" =>', a > 5 and a < 7)
print('\"a > 5 or a < 7\" =>', a > 5 or a < 7)
print('\"not (a > 5 and a < 7)\" =>', not(a > 5 and a < 7))
```

main.py Output

```
For a = 7, checking whether the following conditions are True or False:
"a > 5 and a < 7" => False
"a > 5 or a < 7" => True
"not (a > 5 and a < 7)" => True

=== Code Execution Successful ===
```

6. Membership operators:

main.py

```python
#Membership operators
myList = [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]

x = 31
y = 28

print("Given List:", myList)

# checking if x is present in the list or not
if (x not in myList):
    print("x =", x,"is NOT present in the given list.")
else:
    print("x =", x,"is present in the given list.")

# checking if y is present in the list or not
if (y in myList):
    print("y =", y,"is present in the given list.")
else:
    print("y =", y,"is NOT present in the given list.")
```

Output

```
Given List: [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
x = 31 is NOT present in the given list.
y = 28 is present in the given list.

=== Code Execution Successful ===
```

7. Identity operators:

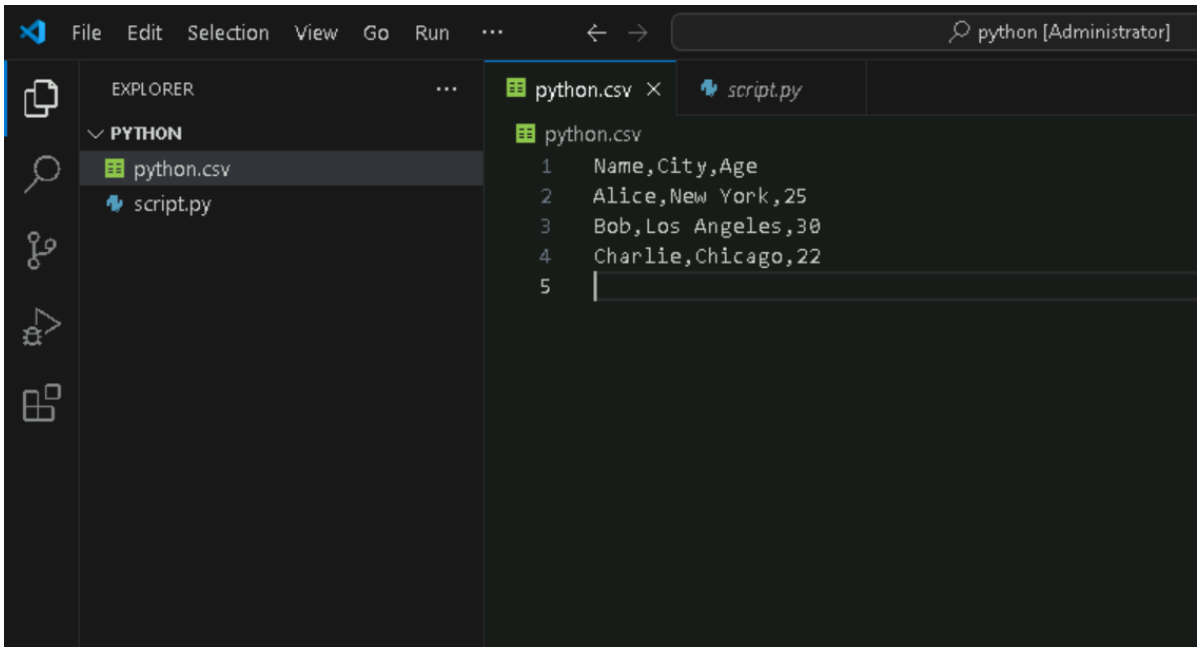
main.py

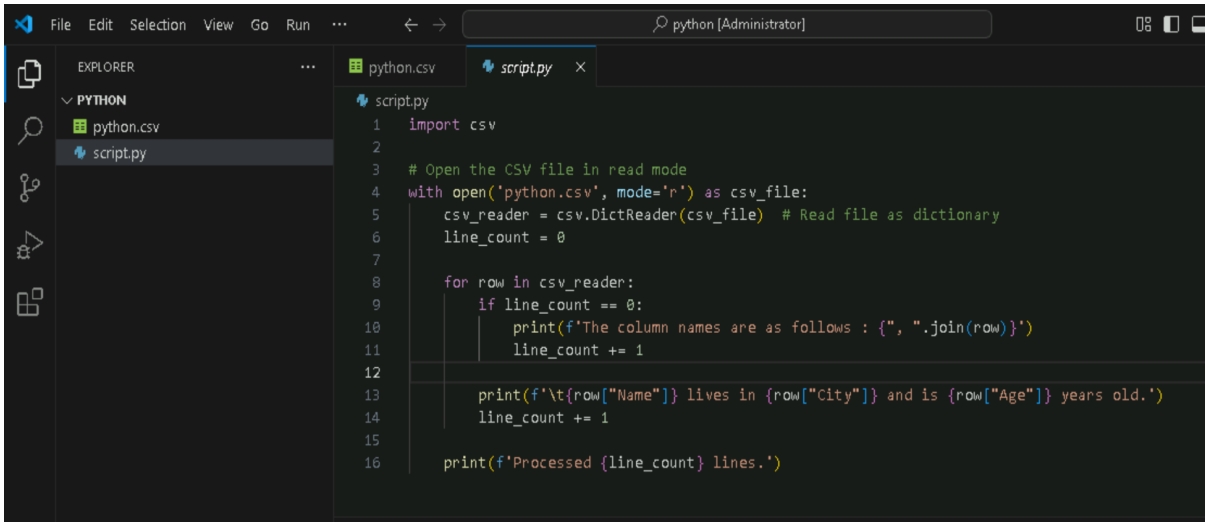
```python
#Identity operators

a = ["Rose", "Lotus"]
b = ["Rose", "Lotus"]

c = a

# printing the different results
print("a is c => ", a is c)
print("a is not c => ", a is not c)
print("a is b => ", a is b)
print("a is not b => ", a is not b)
print("a == b => ", a == b)
print("a != b => ", a != b)
```

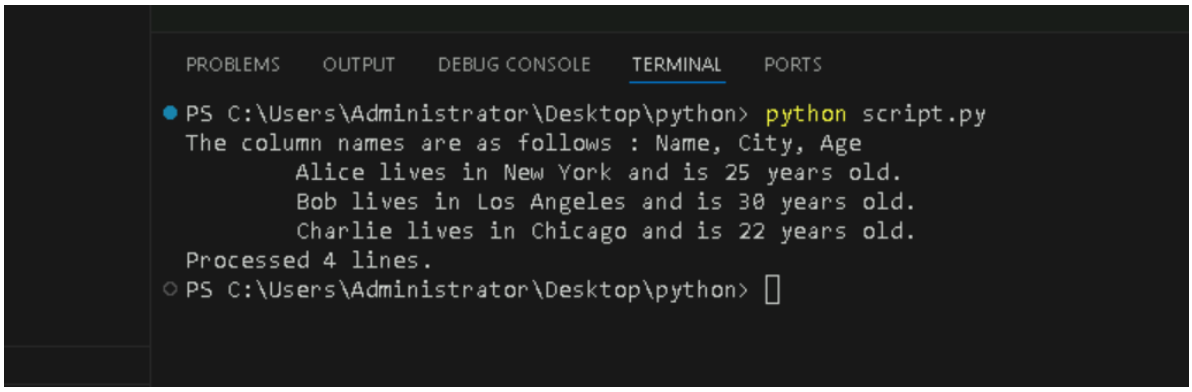
Output

```
a is c =>  True
a is not c =>  False
a is b =>  False
a is not b =>  True
a == b =>  True
a != b =>  False

=== Code Execution Successful ===
```

How to read CSV file in Python?

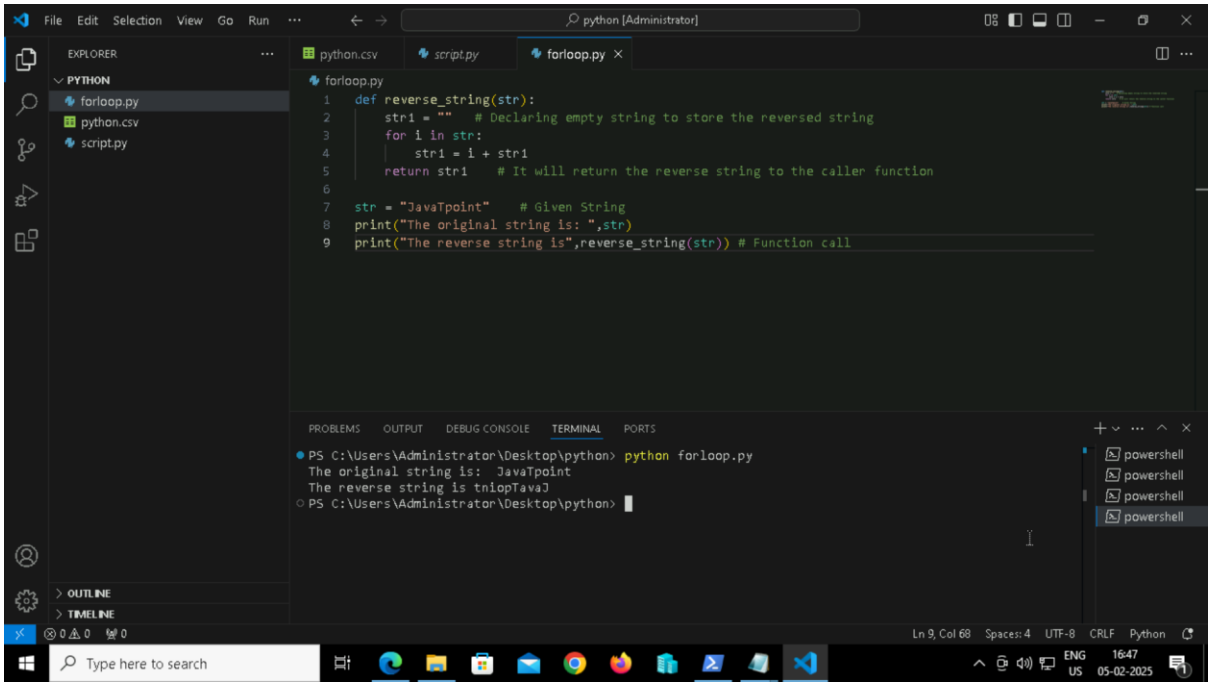




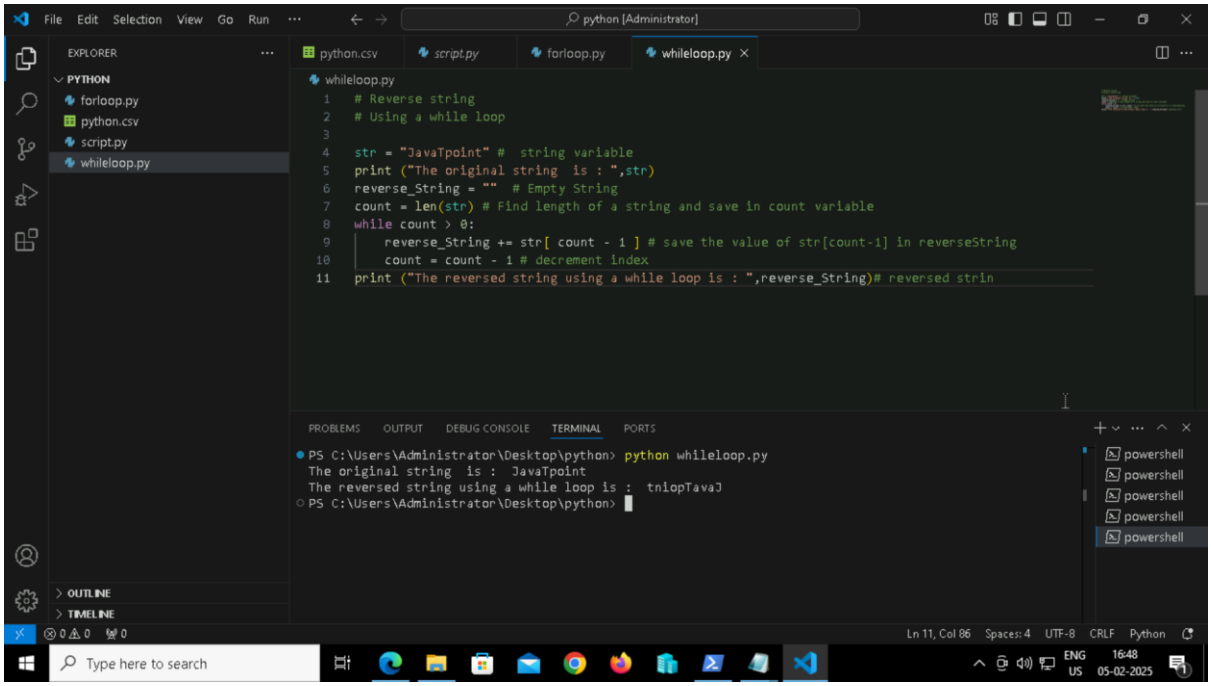


REVERSE OF A STRING

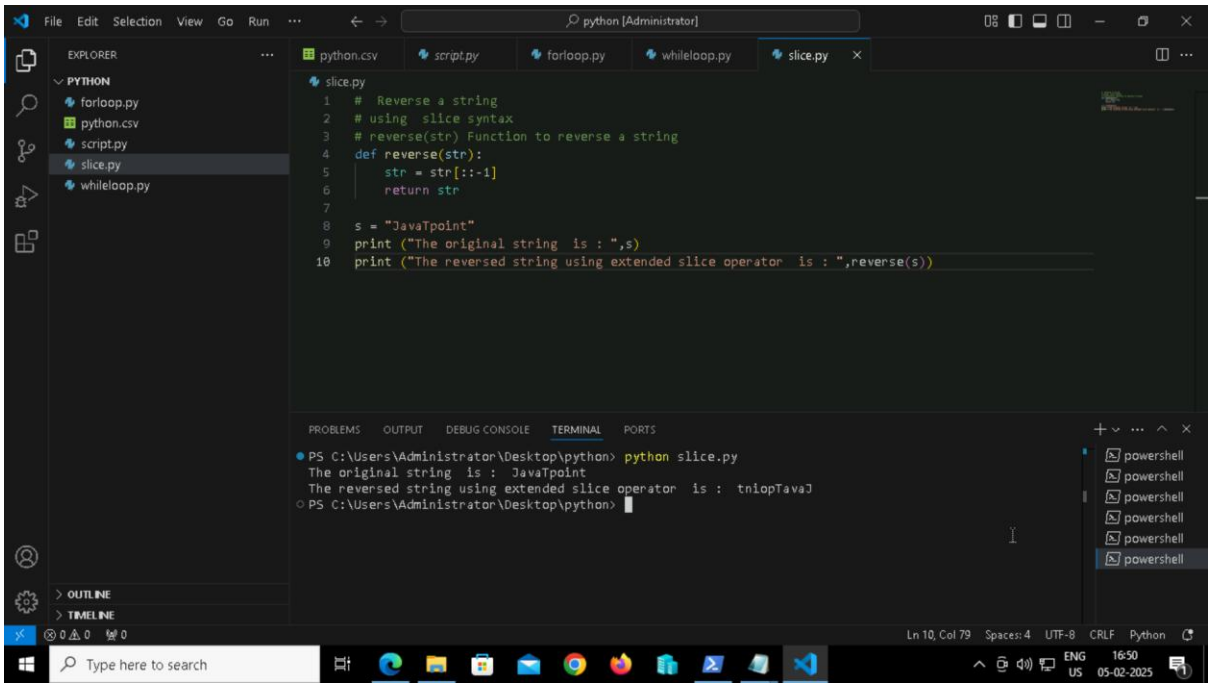
Using For loop:



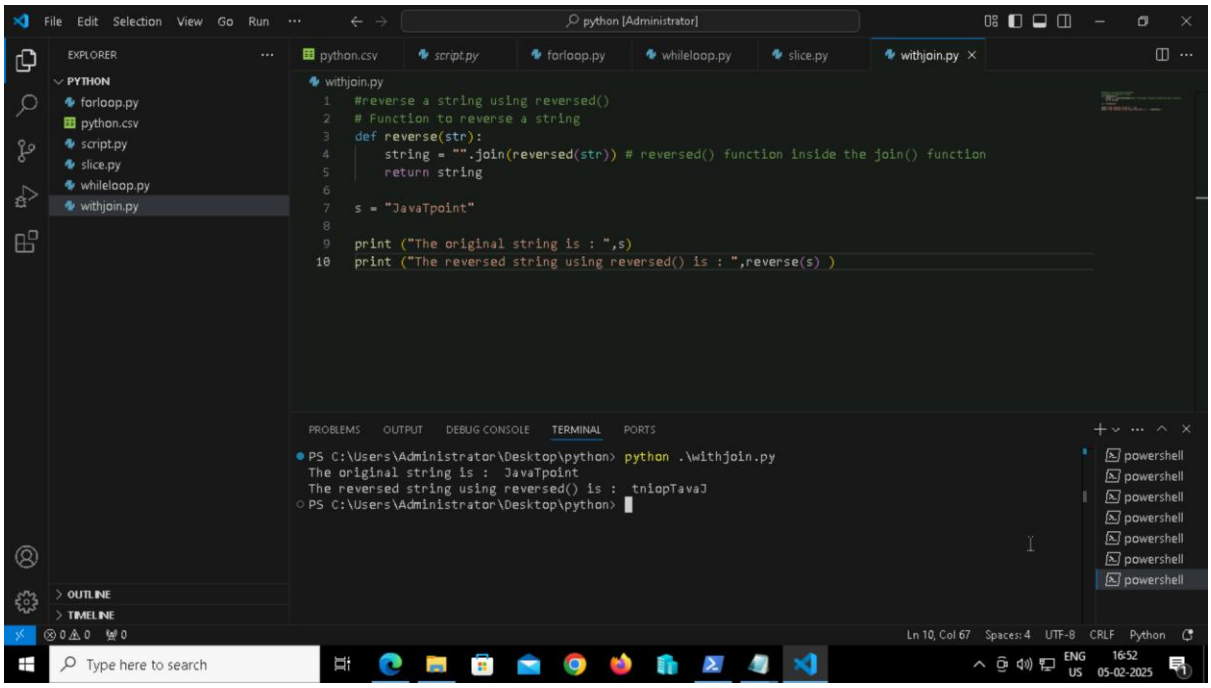
Using while loop:



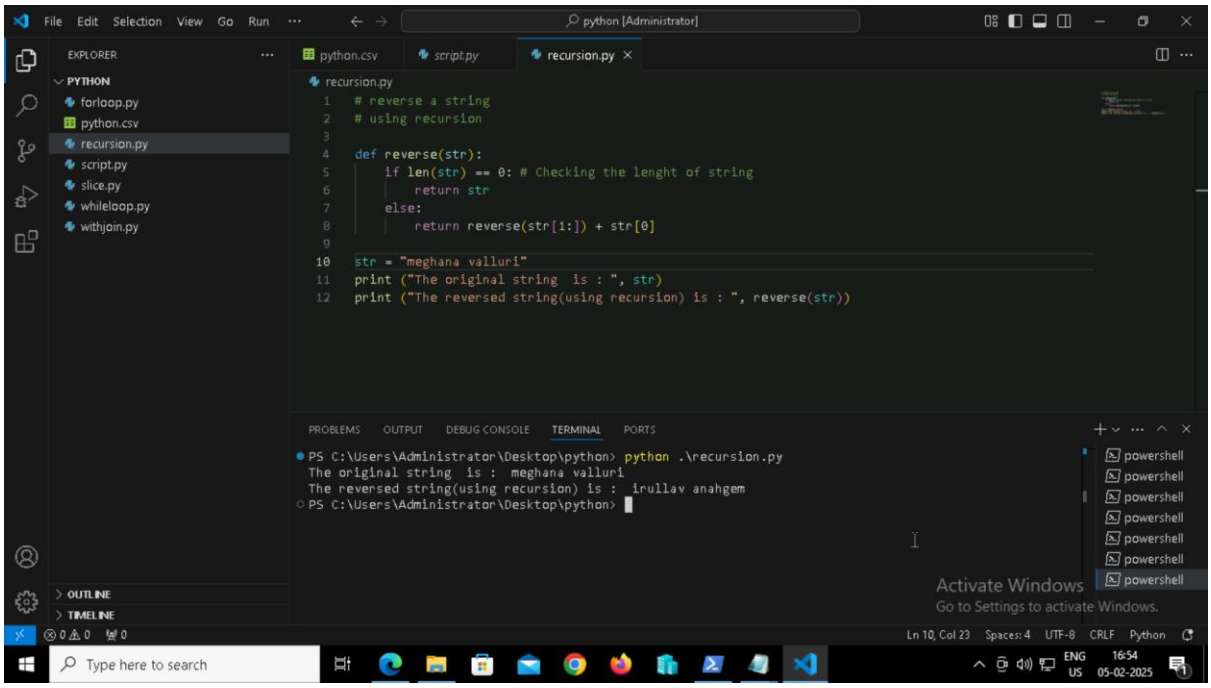
Using slice operator



Using reverse function with join



Using recursion



If statement:

Example 1:

main.py | Output

```
#if statement
num = int(input("enter the number:"))
if num%2 == 0:
    print("The Given number is an even number")
```

main.py | Output

```
enter the number:30
The Given number is an even number

=== Code Execution Successful ===
```

Example 2: Program to print the largest of the three numbers.

main.py Output

```
a = int (input("Enter a: "));
b = int (input("Enter b: "));
c = int (input("Enter c: "));
if a>b and a>c:
# Here, we are checking the condition. If the condition is true, we will enter the block
    print ("From the above three numbers given a is largest");
if b>a and b>c:
# Here, we are checking the condition. If the condition is true, we will enter the block
    print ("From the above three numbers given b is largest");
if c>a and c>b:
# Here, we are checking the condition. If the condition is true, we will enter the block
    print ("From the above three numbers given c is largest");
```

main.py Output

```
Enter a: 3
Enter b: 887
Enter c: 567
From the above three numbers given b is largest

=== Code Execution Successful ===
```

The if-else statement

Example 1:

main.py Output

```
age = int (input("Enter your age: "))
if age>=18:
    print("You are eligible to vote !!");
else:
    print("Sorry! you have to wait !!");
```

main.py Output

```
Enter your age: 22
You are eligible to vote !!

=== Code Execution Successful ===
```

Example 2: Program to check whether a number is even or not.

main.py Output

```python
num = int(input("enter the number:"))
if num%2 == 0:
    print("The Given number is an even number")
else:
    print("The Given Number is an odd number")
```

main.py Output

```
enter the number:45
The Given Number is an odd number

=== Code Execution Successful ===
```

The elif statement

Example 1:

main.py | Output

```python
number = int(input("Enter the number:"))
if number==10:
    print("The given number is equals to 10")
elif number==50:
    print("The given number is equal to 50");
elif number==100:
    print("The given number is equal to 100");
else:
    print("The given number is not equal to 10, 50 or 100");
```

main.py | Output

```
Enter the number:50
The given number is equal to 50

=== Code Execution Successful ===
```

Example 2

main.py | Output

```python
marks = int(input("Enter the marks: "))
if marks > 85 and marks <= 100:
    print("Congrats ! you scored grade A ...")
elif marks > 60 and marks <= 85:
    print("You scored grade B + ...")
elif marks > 40 and marks <= 60:
    print("You scored grade B ...")
elif (marks > 30 and marks <= 40):
    print("You scored grade C ...")
else:
    print("Sorry you are fail ?")
```

main.py Output

```
Enter the marks: 89
Congrats ! you scored grade A ...

=== Code Execution Successful ===
```

Python Loops

The for Loop:

main.py Output

```python
numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]

square = 0

squares = []

for value in numbers:
    square = value ** 2
    squares.append(square)
print("The list of squares is", squares)
```

main.py Output

```
The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]

=== Code Execution Successful ===
```

Using else Statement with for Loop

Example 1:

main.py Output

```python
string = "Python Loop"

# Initiating a loop
for s in string:
    # giving a condition in if block
    if s == "o":
        print("If block")
    # if condition is not satisfied then else block will be executed

    else:
        print(s)
```

main.py Output

```
P
y
t
h
If block
n

L
If block
If block
p

=== Code Execution Successful ===
```

Example 2:

main.py Output

```python
tuple_ = (3, 4, 6, 8, 9, 2, 3, 8, 9, 7)

# Initiating the loop
for value in tuple_:
    if value % 2 != 0:
        print(value)
# giving an else statement
else:
    print("These are the odd numbers present in the tuple")
```

main.py Output

```
3
9
3
9
7
These are the odd numbers present in the tuple

=== Code Execution Successful ===
```

The range() Function

main.py Output

```python
print(range(15))

print(list(range(15)))

print(list(range(4, 9)))

print(list(range(5, 25, 4)))
```

main.py Output

```
range(0, 15)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[4, 5, 6, 7, 8]
[5, 9, 13, 17, 21]

=== Code Execution Successful ===
```

To iterate over a sequence with the help of indexing

main.py Output

```python
tuple_ = ("Python", "Loops", "Sequence", "Condition", "Range")

# iterating over tuple_ using range() function
for iterator in range(len(tuple_)):
    print(tuple_[iterator].upper())
```

main.py Output

```
PYTHON
LOOPS
SEQUENCE
CONDITION
RANGE

=== Code Execution Successful ===
```

While Loop

Example 1:

main.py Output

```python
counter = 0

while counter < 10: # giving the condition
    counter = counter + 3
    print("Python Loops")
```

main.py Output

```
Python Loops
Python Loops
Python Loops
Python Loops

=== Code Execution Successful ===
```

Using else Statement with while Loops

main.py Output

```python
counter = 0

while (counter < 10):
    counter = counter + 3
    print("Python Loops")
else:
    print("Code block inside the else statement")
```

main.py Output

```
Python Loops
Python Loops
Python Loops
Python Loops
Code block inside the else statement

=== Code Execution Successful ===
```

to write a single statement while loop

main.py Output

```python
count = 0
while (count < 3):
    print("Python Loops")
```

main.py Output

```
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
```

Loop Control Statements

Continue Statement

main.py Output

```python
for string in "Python Loops":
    if string == "o" or string == "p" or string == "t":
        continue
    print('Current Letter:', string)
```

main.py Output

```
Current Letter: P
Current Letter: y
Current Letter: h
Current Letter: n
Current Letter:
Current Letter: L
Current Letter: s

=== Code Execution Successful ===
```

Break Statement

main.py Output

```
for string in "Python Loops":
    if string == 'L':
        break
    print('Current Letter: ', string)
```

main.py Output

```
Current Letter:  P
Current Letter:  y
Current Letter:  t
Current Letter:  h
Current Letter:  o
Current Letter:  n
Current Letter:

=== Code Execution Successful ===
```

Pass Statement

main.py Output

```
for string in "Python Loops":
    pass
print( 'Last Letter:', string)
```

main.py Output

```
Last Letter: s

=== Code Execution Successful ===
```

Python String

Creating String in Python

main.py Output

```python
str1 = 'Hello Python'
print(str1)
#Using double quotes
str2 = "Hello Python"
print(str2)

#Using triple quotes
str3 = '''''Triple quotes are generally used for
    represent the multiline or
    docstring'''
print(str3)
```

main.py Output

```
Hello Python
Hello Python
''Triple quotes are generally used for
    represent the multiline or
    docstring
```

Strings indexing and splitting

Example 1:

main.py Output

```python
str = "HELLO"
print(str[0])
print(str[1])
print(str[2])
print(str[3])
print(str[4])
# It returns the IndexError because 6th index doesn't exist
print(str[6])
```

main.py Output

```
H
E
L
L
O
```

Example 2:

main.py Output

```python
# Given String
str = "JAVATPOINT"
# Start 0th index to end
print(str[0:])
# Starts 1th index to 4th index
print(str[1:5])
# Starts 2nd index to 3rd index
print(str[2:4])
# Starts 0th to 2nd index
print(str[:3])
#Starts 4th to 6th index
print(str[4:7])
```

main.py Output

```
JAVATPOINT
AVAT
VA
JAV
TPO

=== Code Execution Successful ===
```

Example 3:

main.py Output

```python
str = 'JAVATPOINT'
print(str[-1])
print(str[-3])
print(str[-2:])
print(str[-4:-1])
print(str[-7:-2])
# Reversing the given string
print(str[::-1])
print(str[-12])
```

main.py Output

```
T
I
NT
OIN
ATPOI
TNIOPTAVAJ
```

Reassigning Strings

main.py Output

```python
str = "HELLO"
print(str)
str = "hello"
print(str)
```

```
HELLO
hello

=== Code Execution Successful ===
```

The format() method

main.py Output

```python
# Using Curly braces
print("{} and {} both are the best friend".format("Devansh"
    ,"Abhishek"))

#Positional Argument
print("{1} and {0} best players ".format("Virat","Rohit"))

#Keyword Argument
print("{a},{b},{c}".format(a = "James", b = "Peter", c = "Ricky"))
```

main.py Output

```
Devansh and Abhishek both are the best friend
Rohit and Virat best players
James,Peter,Ricky

=== Code Execution Successful ===
```

Python String Formatting Using % Operator

main.py Output

```python
Integer = 10;
Float = 1.290
String = "Devansh"
print("Hi I am Integer ... My value is %d\nHi I am float ... My value is %f\nHi I am string ... My value is %s"%(Integer,Float,String))
```

main.py Output

```
Hi I am Integer ... My value is 10
Hi I am float ... My value is 1.290000
Hi I am string ... My value is Devansh

=== Code Execution Successful ===
```

Python List

List Declaration

main.py Output

```python
# a simple list
list1 = [1, 2, "Python", "Program", 15.9]
list2 = ["Amy", "Ryan", "Henry", "Emma"]

# printing the list
print(list1)
print(list2)

# printing the type of list
print(type(list1))
print(type(list2))
```

main.py Output

```
[1, 2, 'Python', 'Program', 15.9]
['Amy', 'Ryan', 'Henry', 'Emma']
<class 'list'>
<class 'list'>

=== Code Execution Successful ===
```

Ordered List Checking

Example 1:

main.py Output

```
# example
a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6 ]
b = [ 1, 2, 5, "Ram", 3.50, "Rahul", 6 ]
print(a == b)
```

main.py Output

```
False

=== Code Execution Successful ===
```

Example 2:

main.py Output

```
# example
a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
b = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
print(a == b)
```

main.py Output

```
True

=== Code Execution Successful ===
```

List Indexing and Splitting

main.py | Output

```python
list = [1,2,3,4,5,6,7]
print(list[0])
print(list[1])
print(list[2])
print(list[3])
# Slicing the elements
print(list[0:6])
# By default, the index value is 0 so its starts from the 0th element and go for index -1.
print(list[:])
print(list[2:5])
print(list[1:6:2])
```

main.py | Output

```
1
2
3
4
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 7]
[3, 4, 5]
[2, 4, 6]

=== Code Execution Successful ===
```

Updating List Values

main.py | Output

```python
# updating list values
list = [1, 2, 3, 4, 5, 6]
print(list)
# It will assign value to the value to the second index
list[2] = 10
print(list)
# Adding multiple-element
list[1:3] = [89, 78]
print(list)
# It will add value at the end of the list
list[-1] = 25
print(list)
```

main.py Output

```
[1, 2, 3, 4, 5, 6]
[1, 2, 10, 4, 5, 6]
[1, 89, 78, 4, 5, 6]
[1, 89, 78, 4, 5, 25]

=== Code Execution Successful ===
```

List and Tuple Syntax Differences

main.py Output

```python
list_ = [4, 5, 7, 1, 7]
tuple_ = (4, 1, 8, 3, 9)

print("List is: ", list_)
print("Tuple is: ", tuple_)
```

main.py Output

```
List is:  [4, 5, 7, 1, 7]
Tuple is:  (4, 1, 8, 3, 9)

=== Code Execution Successful ===
```

Mutable List vs. Immutable Tuple

main.py Output

```python
list_ = ["Python", "Lists", "Tuples", "Differences"]
tuple_ = ("Python", "Lists", "Tuples", "Differences")

# modifying the last string in both data structures
list_[3] = "Mutable"
print( list_ )
try:
    tuple_[3] = "Immutable"
    print( tuple_ )
except TypeError:
    print( "Tuples cannot be modified because they are immutable" )
```

main.py Output

```
['Python', 'Lists', 'Tuples', 'Mutable']
Tuples cannot be modified because they are immutable

=== Code Execution Successful ===
```

Size Difference

main.py Output

```python
list_ = ["Python", "Lists", "Tuples", "Differences"]
tuple_ = ("Python", "Lists", "Tuples", "Differences")
# printing sizes
print("Size of tuple: ", tuple_.__sizeof__())
print("Size of list: ", list_.__sizeof__())
```

main.py Output

```
Size of tuple:  56
Size of list:  72

=== Code Execution Successful ===
```

Python Functions:

Illustration of a User-Defined Function

main.py Output

```python
def square( num ):
    """
    This function computes the square of the number.
    """
    return num**2
object_ = square(6)
print( "The square of the given number is: ", object_ )
```

main.py Output

```
The square of the given number is:  36

=== Code Execution Successful ===
```

Calling a Function

main.py Output

```python
def a_function( string ):
    "This prints the value of length of string"
    return len(string)

# Calling the function we defined
print( "Length of the string Functions is: ", a_function( "Functions" ) )
print( "Length of the string Python is: ", a_function( "Python" ) )
```

main.py Output

```
Length of the string Functions is:  9
Length of the string Python is:  6
```

Pass by Reference vs. Pass by Value

main.py Output

```python
def square( item_list ):
    '''''''This function will find the square of items in the list'''
    squares = [ ]
    for l in item_list:
        squares.append( l**2 )
    return squares

# calling the defined function
my_list = [17, 52, 8];
my_result = square( my_list )
print( "Squares of the list are: ", my_result )
```

main.py Output

```
Squares of the list are:  [289, 2704, 64]
```

Function Arguments

1. Default arguments

main.py | Output

```python
def function( n1, n2 = 20 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)


# Calling the function and passing only one argument
print( "Passing only one argument" )
function(30)

# Now giving two arguments to the function
print( "Passing two arguments" )
function(50,30)
```

main.py | Output

```
Passing only one argument
number 1 is:  30
number 2 is:  20
Passing two arguments
number 1 is:  50
number 2 is:  30
```

2. Keyword arguments

main.py | Output

```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)

# Calling function and passing arguments without using keyword
print( "Without using keyword" )
function( 50, 30)

# Calling function and passing arguments using keyword
print( "With using keyword" )
function( n2 = 50, n1 = 30)
```

main.py | Output

```
Without using keyword
number 1 is:  50
number 2 is:  30
With using keyword
number 1 is:  30
number 2 is:  50
```

3. Required arguments

main.py Output

```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)

# Calling function and passing two arguments out of order, we need num1 to be 20 and num2 to be 30
print( "Passing out of order arguments" )
function( 30, 20 )

# Calling function and passing only one argument
print( "Passing only one argument" )
try:
    function( 30 )
except:
    print( "Function needs two positional arguments" )
```

main.py Output

```
Passing out of order arguments
number 1 is:  30
number 2 is:  20
Passing only one argument
Function needs two positional arguments
```

4. Variable-length arguments

main.py Output

```python
def function( *args_list ):
    ans = []
    for l in args_list:
        ans.append( l.upper() )
    return ans
# Passing args arguments
object = function('Python', 'Functions', 'tutorial')
print( object )

# defining a function
def function( **kargs_list ):
    ans = []
    for key, value in kargs_list.items():
        ans.append([key, value])
    return ans
# Paasing kwargs arguments
object = function(First = "Python", Second = "Functions", Third = "Tutorial")
print(object)
```

main.py Output

```
['PYTHON', 'FUNCTIONS', 'TUTORIAL']
[['First', 'Python'], ['Second', 'Functions'], ['Third', 'Tutorial']]
```

return Statement

main.py Output

```python
def square( num ):
    return num**2

# Calling function and passing arguments.
print( "With return statement" )
print( square( 52 ) )

# Defining a function without return statement
def square( num ):
    num**2

# Calling function and passing arguments.
print( "Without return statement" )
print( square( 52 ) )
```

main.py Output

```
With return statement
2704
Without return statement
None
```

The Anonymous Functions

main.py Output

```python
lambda_ = lambda argument1, argument2: argument1 + argument2;

# Calling the function and passing values
print( "Value of the function is : ", lambda_( 20, 30 ) )
print( "Value of the function is : ", lambda_( 40, 50 ) )
```

main.py Output

```
Value of the function is :  50
Value of the function is :  90
```

Scope and Lifetime of Variables

main.py Output

```python
def number( ):
    num = 50
    print( "Value of num inside the function: ", num)
num = 10
number()
print( "Value of num outside the function:", num)
```

main.py Output

```
Value of num inside the function:  50
Value of num outside the function: 10
```

Python Capability inside Another Capability

main.py Output

```python
def word():
    string = 'Python functions tutorial'
    x = 5
    def number():
        print( string )
        print( x )
    number()
word()
```

main.py Output

```
Python functions tutorial
5
```