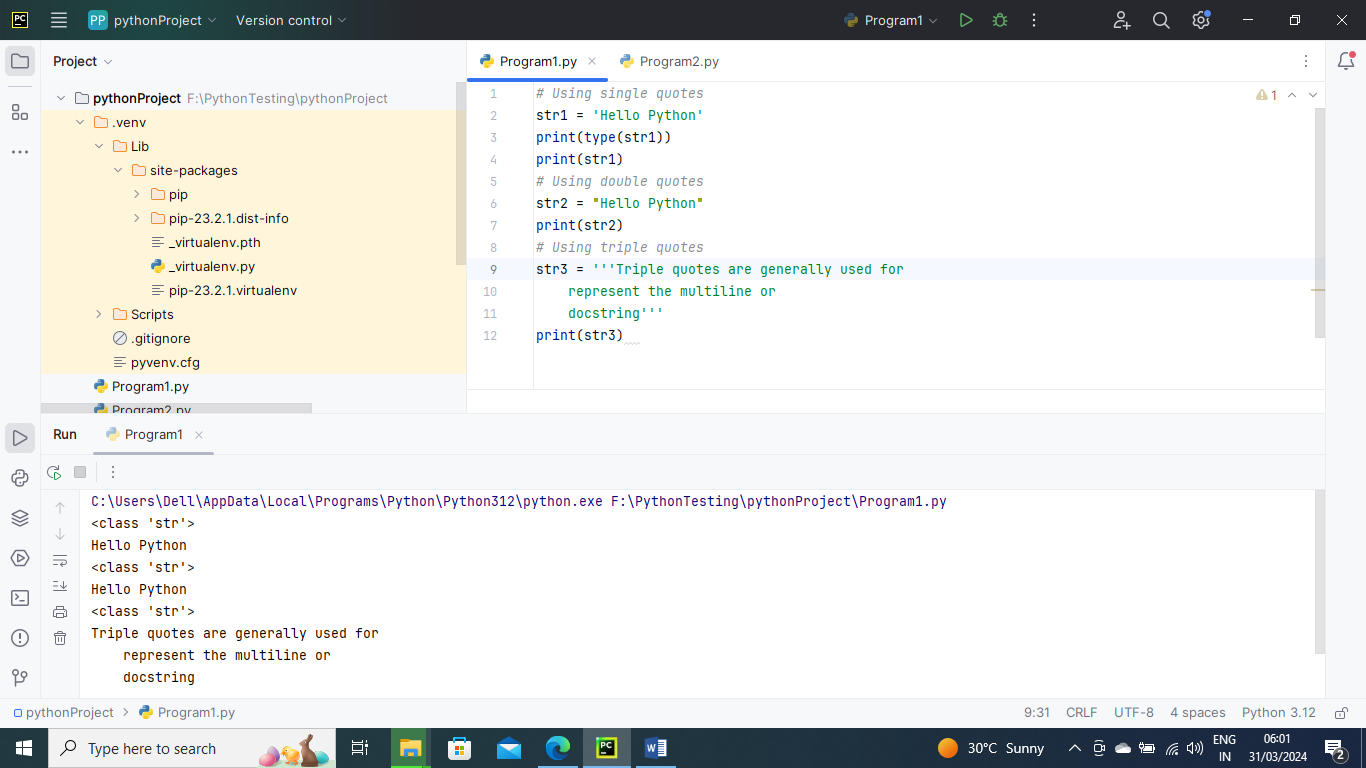
String –

Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes.

Below is an example where we are assigned value to string and getting its type and output of value

*# Using single quotes*str1 = 'Hello Python'  
print(type(str1))  
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)



String Indexing –

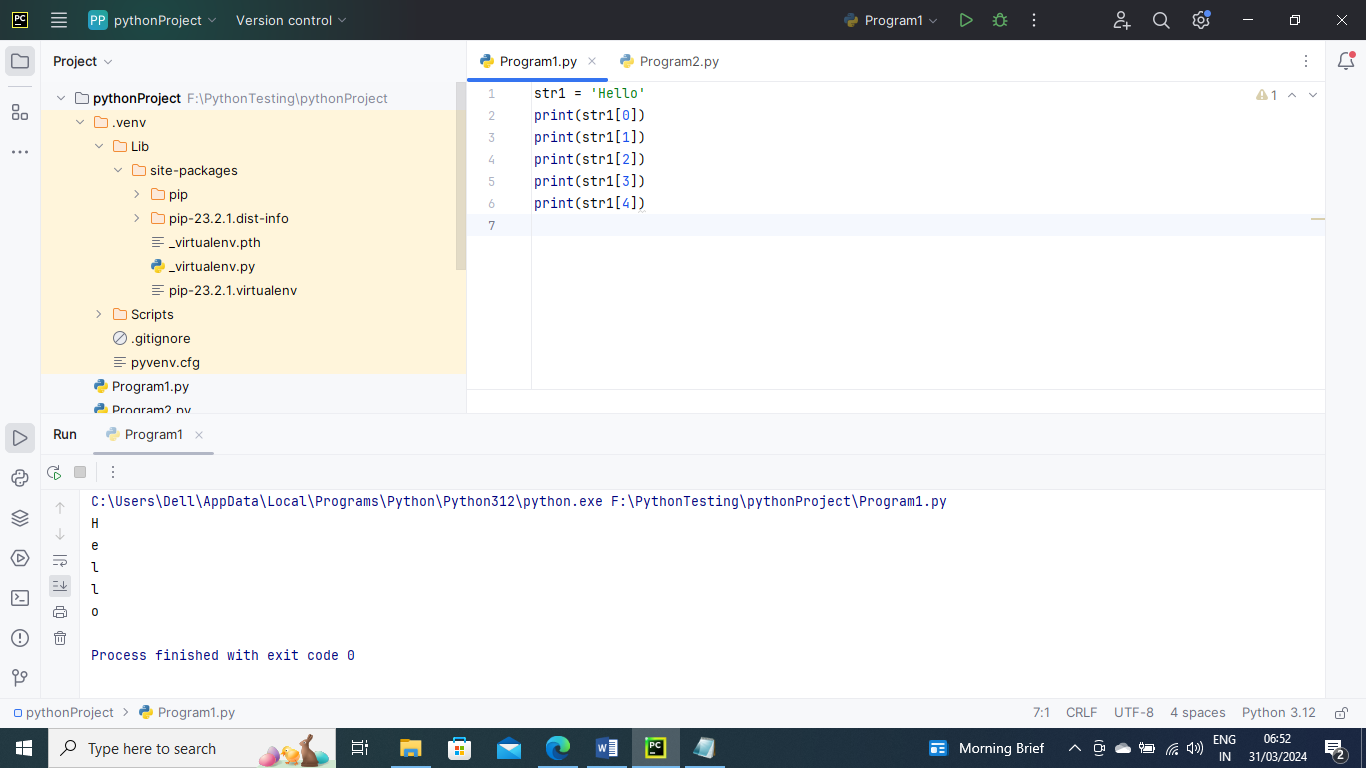
Like other languages, the indexing of the Python strings starts from 0. For example, The string "HELLO" is indexed as given in the below figure.

Str1=”HELLO”

Str1[0]=’H’, Str1[1]=’E’, Str1[2]=’L’, Str1[3]=’L’, Str1[4]=’O’

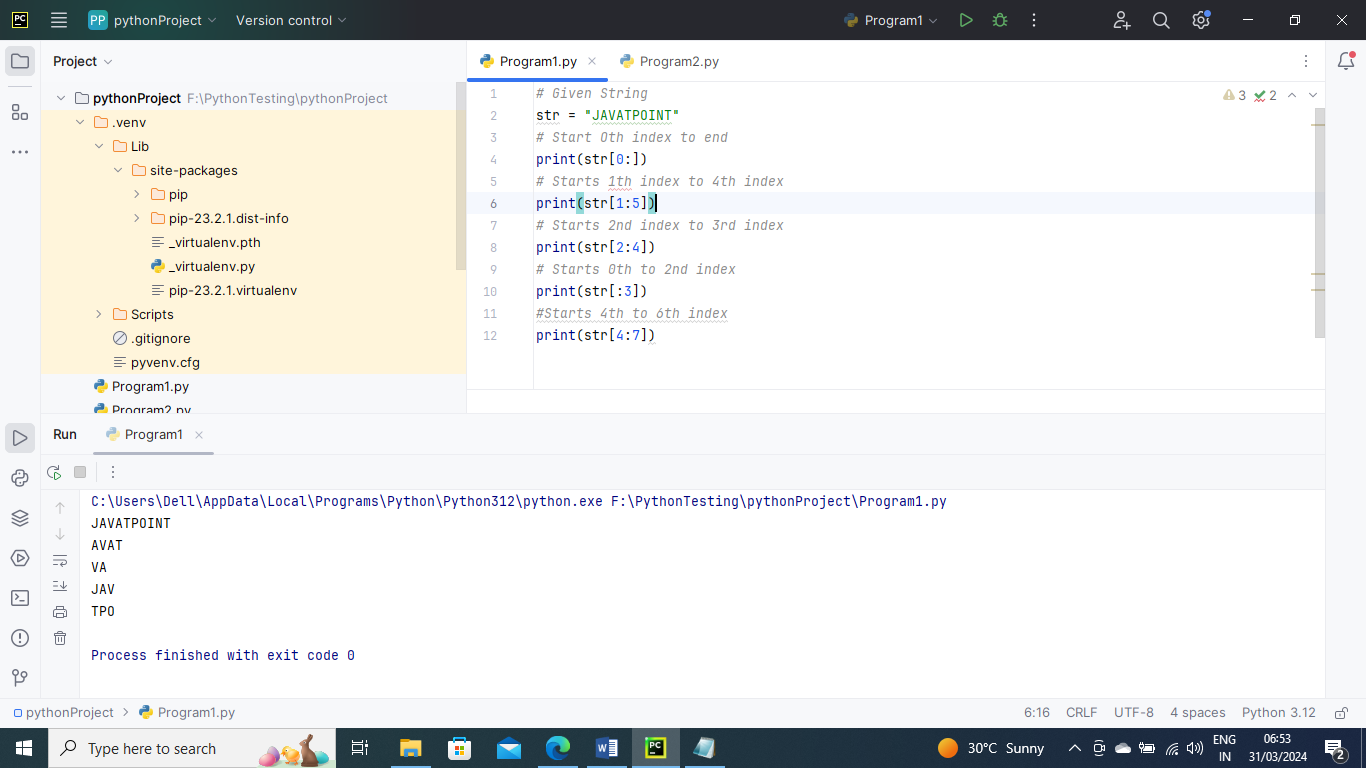
Below is example for string indexing

str1 = 'Hello'  
print(str1[0])  
print(str1[1])  
print(str1[2])  
print(str1[3])  
print(str1[4])

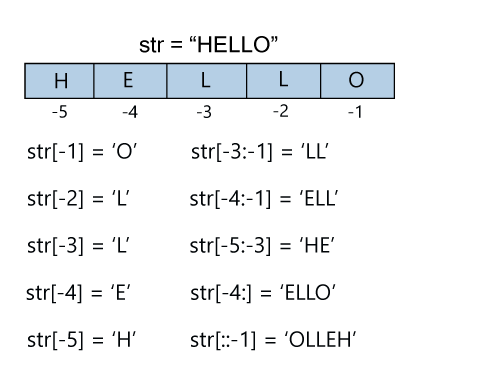


Below is an example of indexing “from position” to “to position”

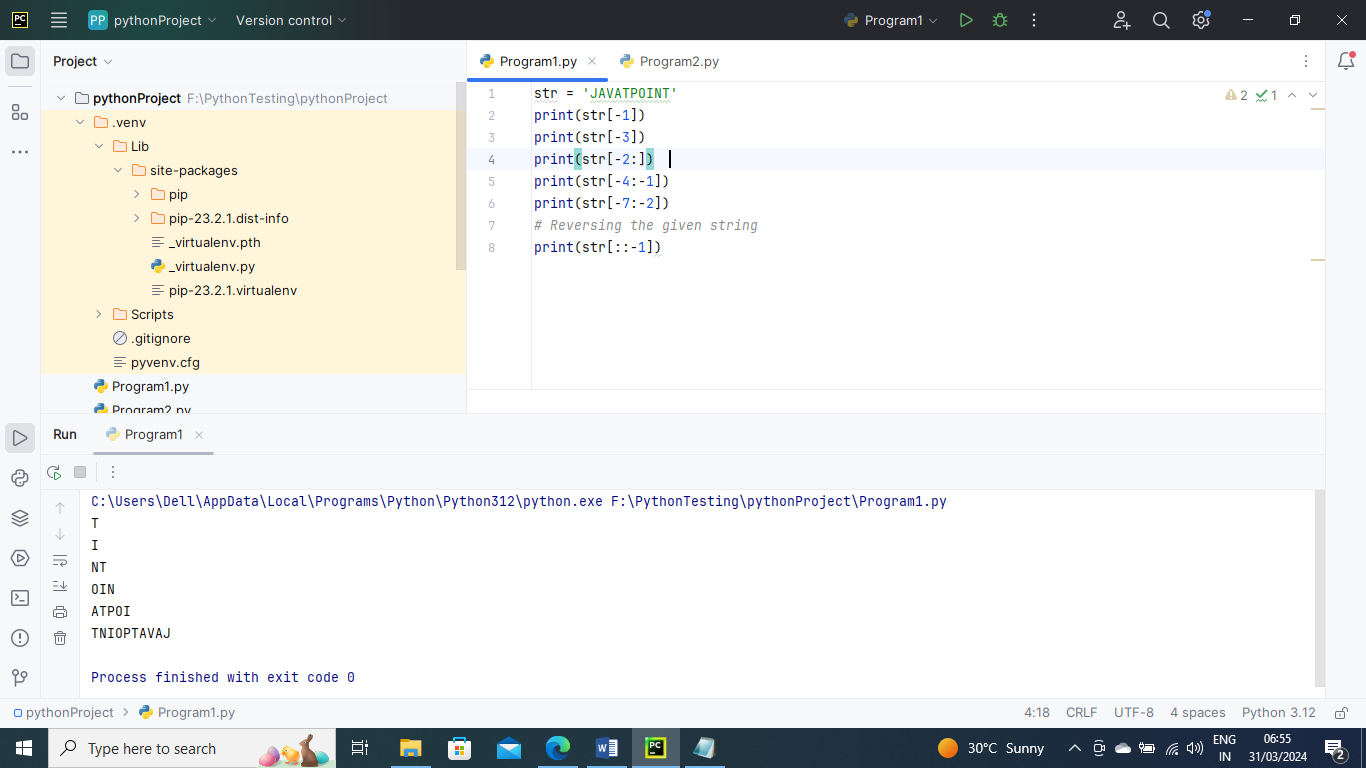
*# Given String*str = "JAVATPOINT"  
*# Start Oth index to end*print(str[0:])  
*# Starts 1th index to 4th index*print(str[1:5])  
*# Starts 2nd index to 3rd index*print(str[2:4])  
*# Starts 0th to 2nd index*print(str[:3])  
*#Starts 4th to 6th index*print(str[4:7])



Below is an example where we can do the negative slicing in the string; it starts from the rightmost character, which is indicated as -1. The second rightmost index indicates -2, and so on



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])

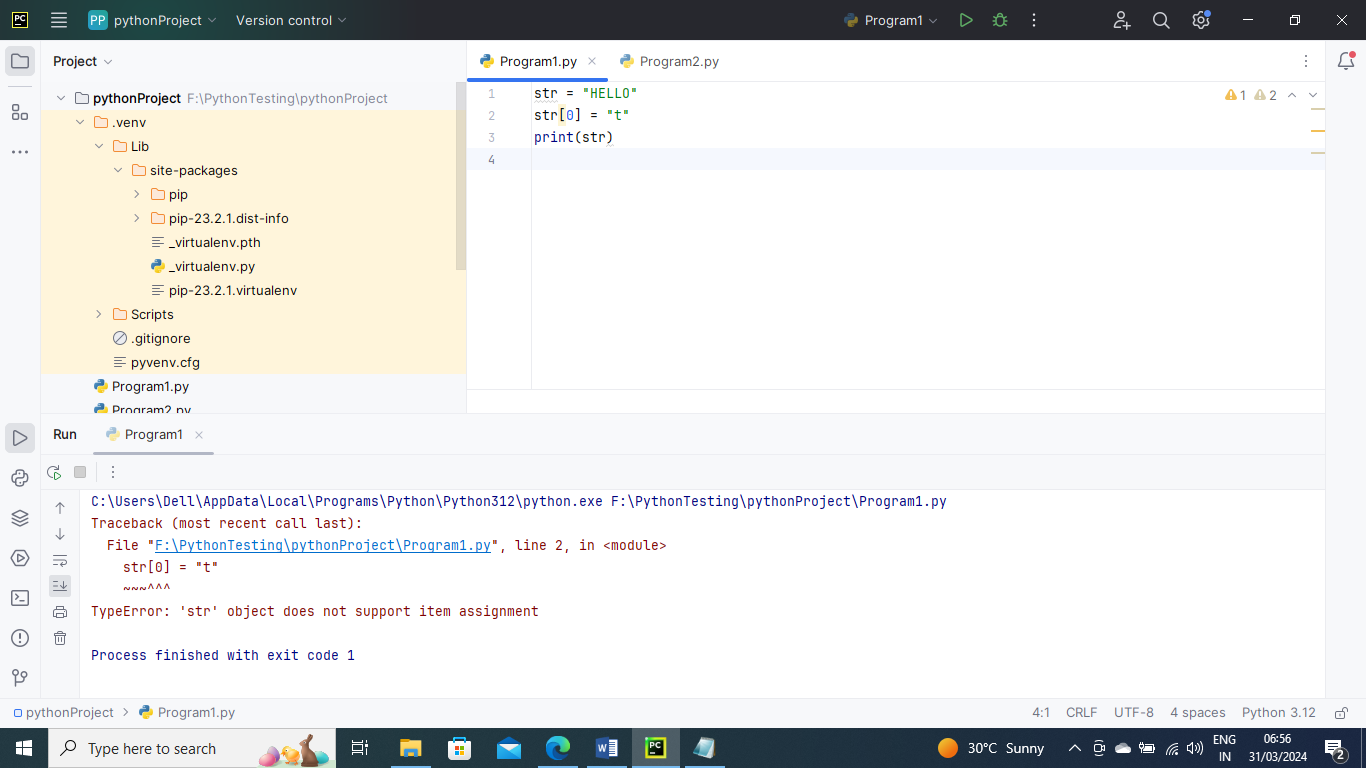


String reassignment-

The string object doesn't support item assignment i.e., A string can only be replaced with new string since its content cannot be partially replaced. Strings are immutable in Python.

Below is an example

str = "HELLO"  
str[0] = "t"  
print(str)

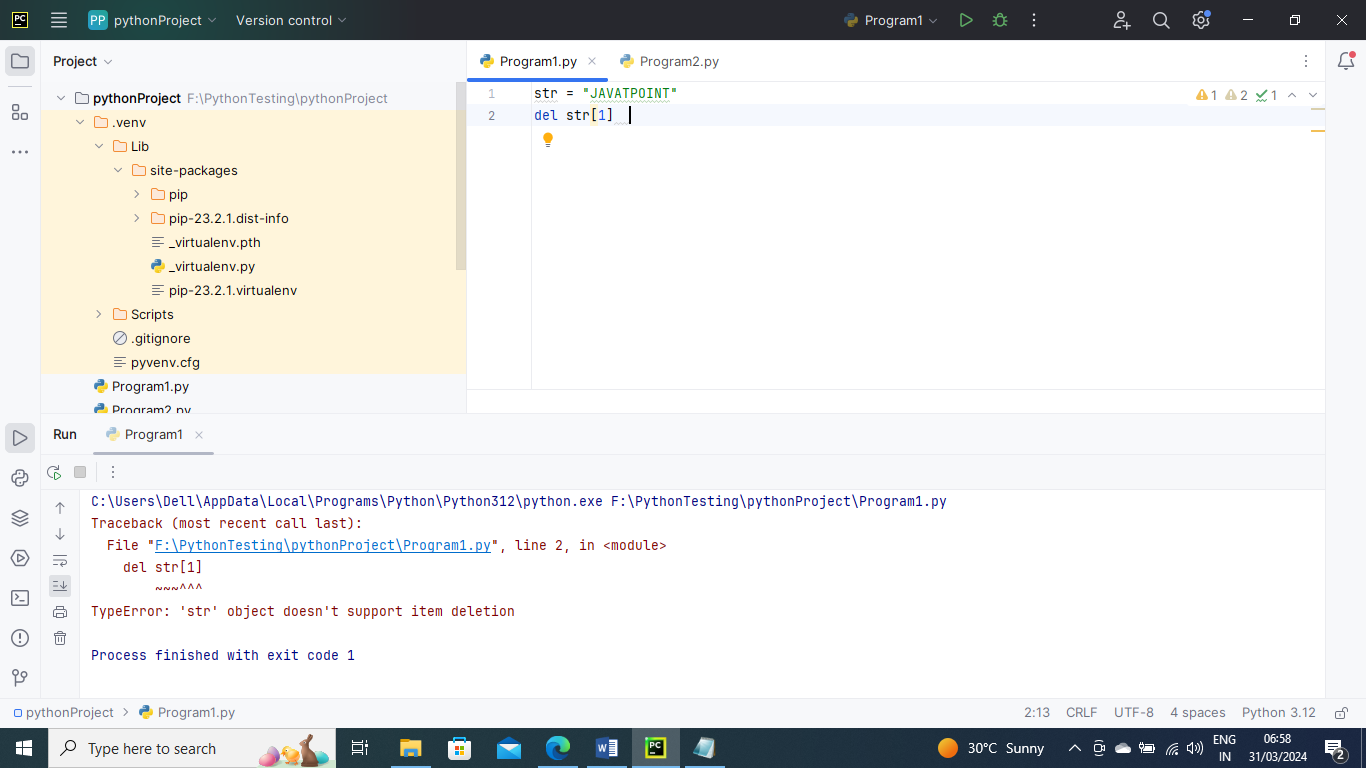


Deleting String

As we know that strings are immutable. We cannot delete or remove the characters from the string.  But we can delete the entire string using the **del** keyword.

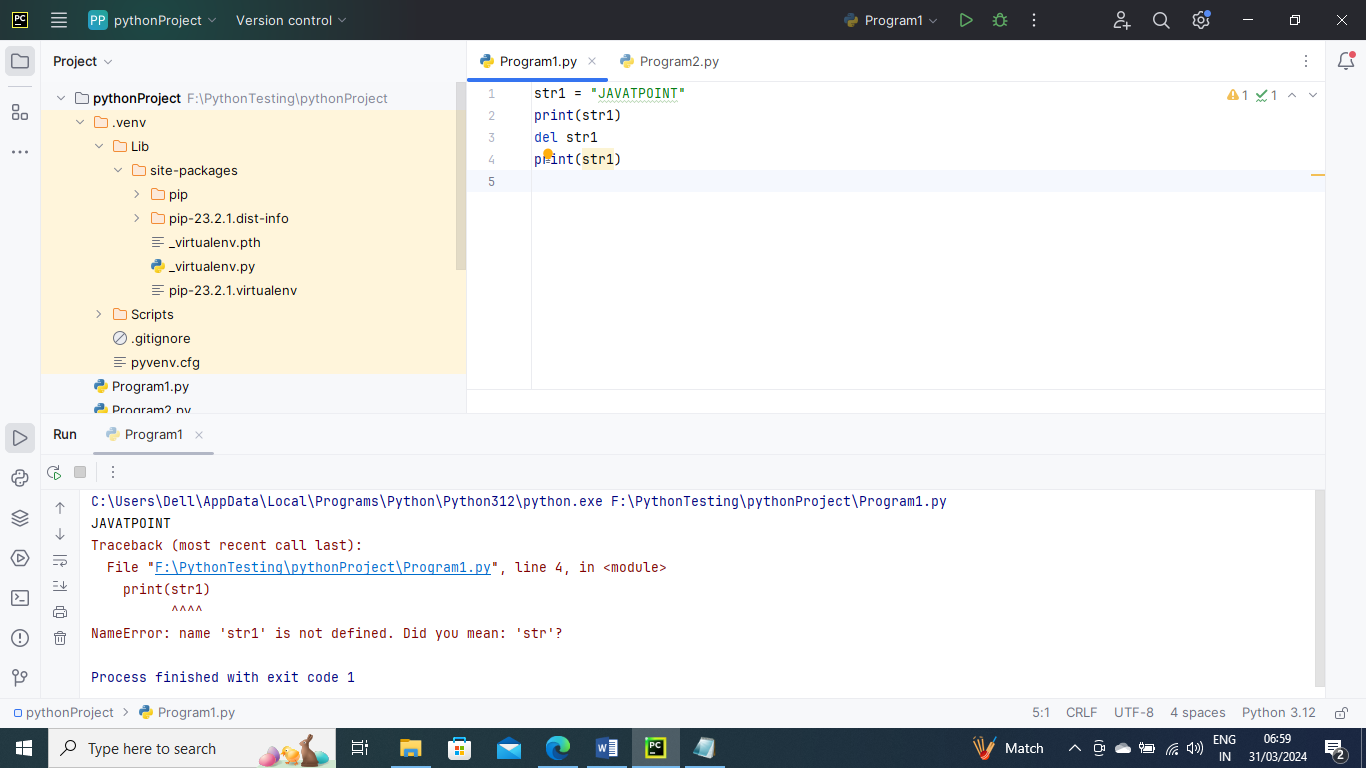
Below is an example where we cannot delete a single item in string

str = "JAVATPOINT"  
del str[1]

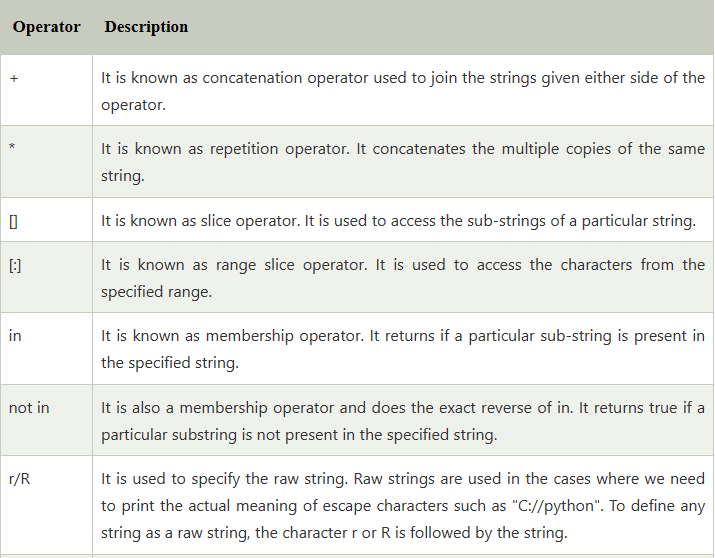


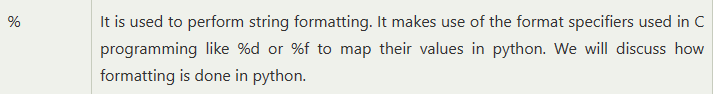
Below is an example where you can delete entire string. As you can see after deleting String at line 4 it gives error saying Str1 is not defined

str1 = "JAVATPOINT"  
print(str1)  
del str1  
print(str1)



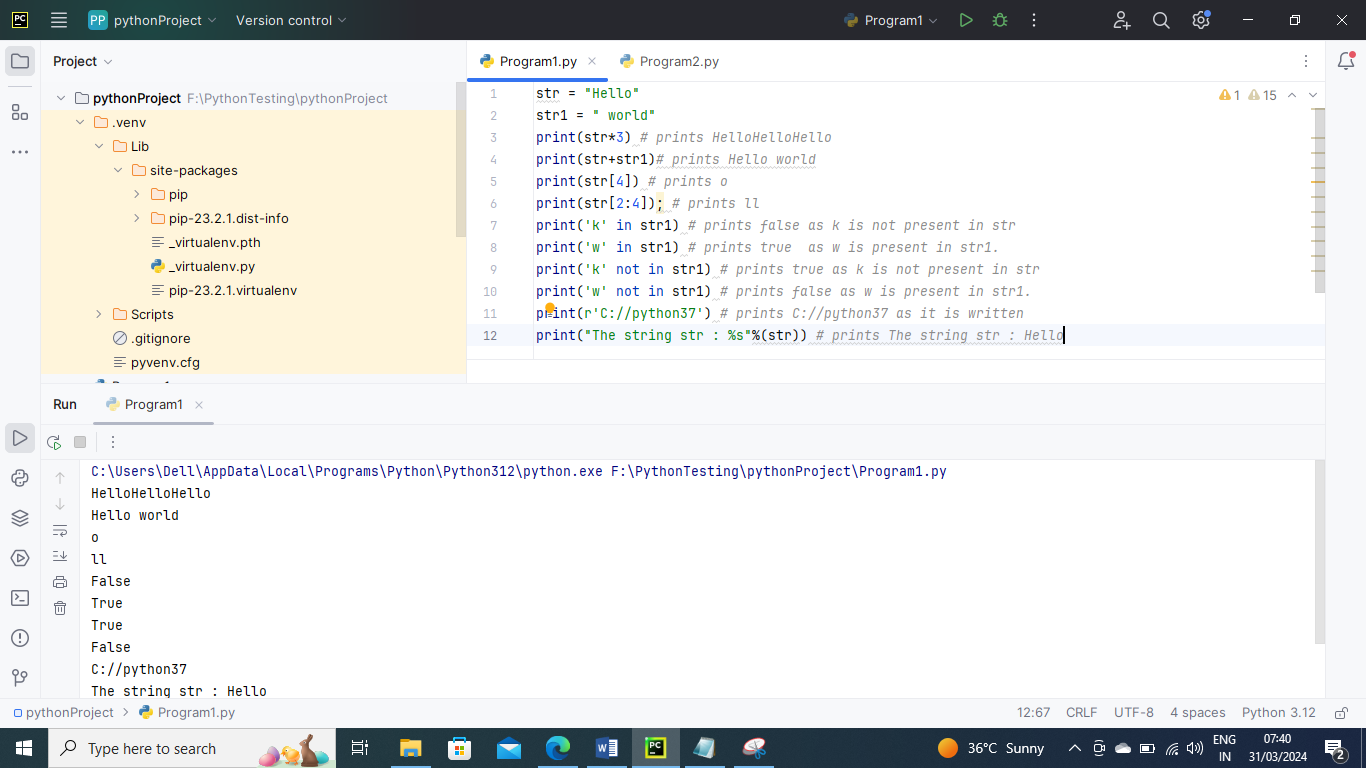
String Operators





Below is an example of operators

str = "Hello"  
str1 = " world"  
print(str\*3) *# prints HelloHelloHello*print(str+str1)*# prints Hello world*print(str[4]) *# prints o*print(str[2:4]); *# prints ll*print('k' in str1) *# prints false as k is not present in str*print('w' in str1) *# prints true as w is present in str1.*print('k' not in str1) *# prints true as k is not present in str*print('w' not in str1) *# prints false as w is present in str1.*print(r'C://python37') *# prints C://python37 as it is written*print("The string str : %s"%(str)) *# prints The string str : Hello*



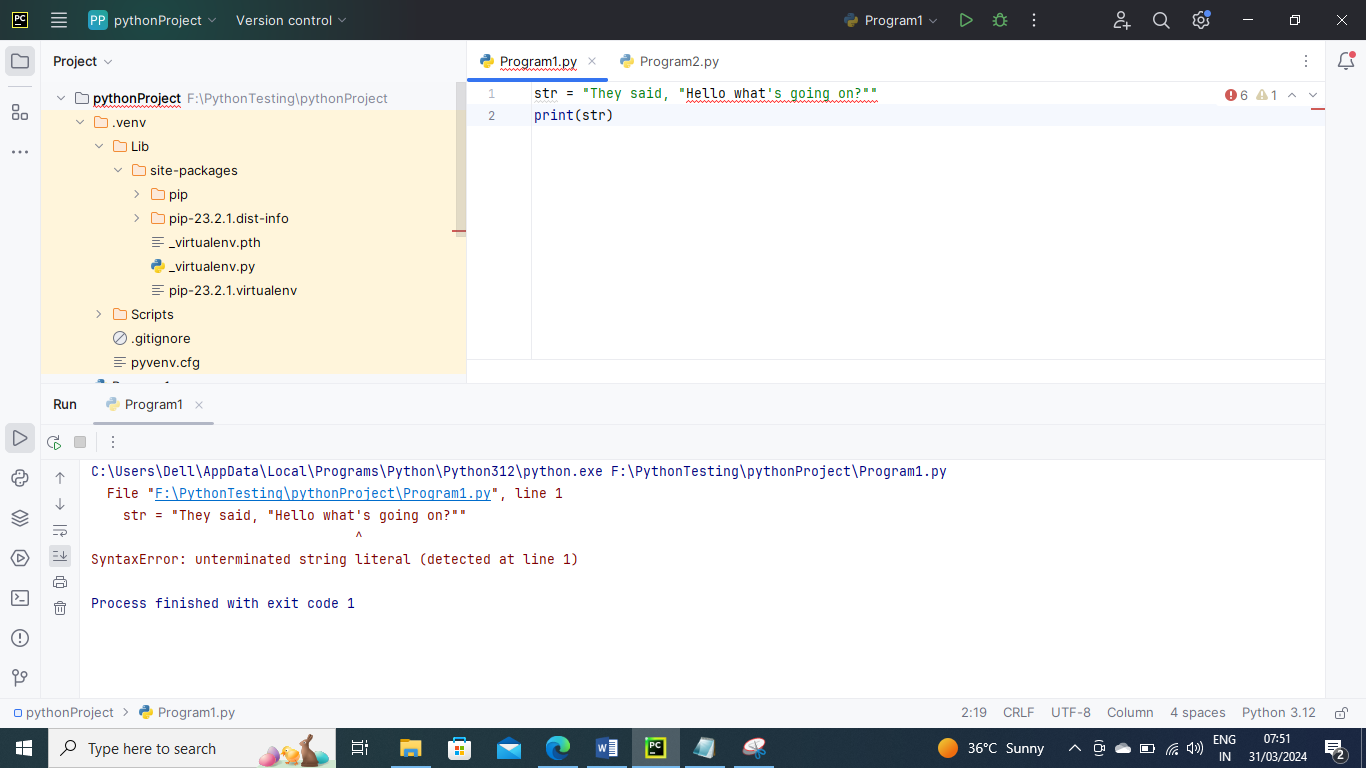
String Formatting

Escape Sequence

Let's suppose we need to write the text as - They said, "Hello what's going on?"- the given statement can be written in single quotes or double quotes but it will raise the **SyntaxError** as it contains both single and double-quotes.

Below is example

str = "They said, "Hello what's going on?""  
print(str)

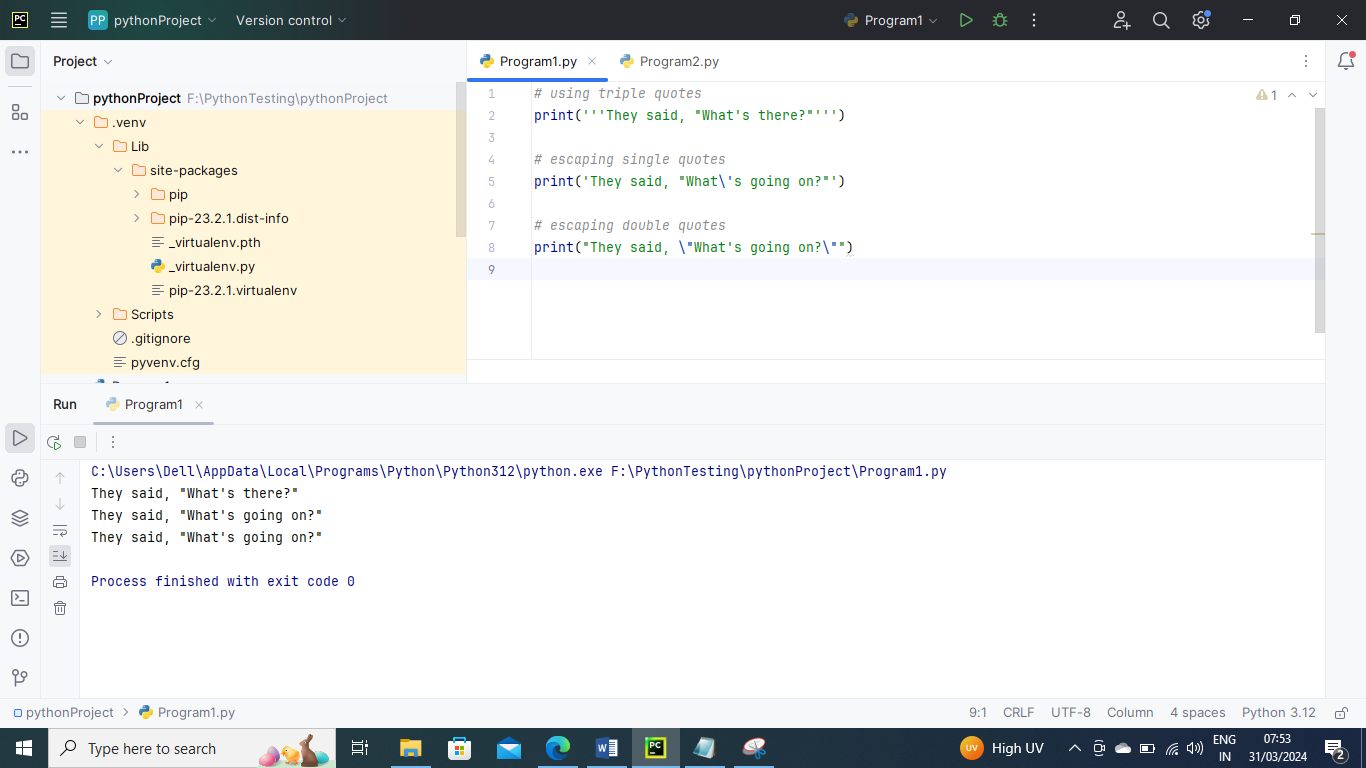


We can use the triple quotes to accomplish this problem but Python provides the escape sequence.

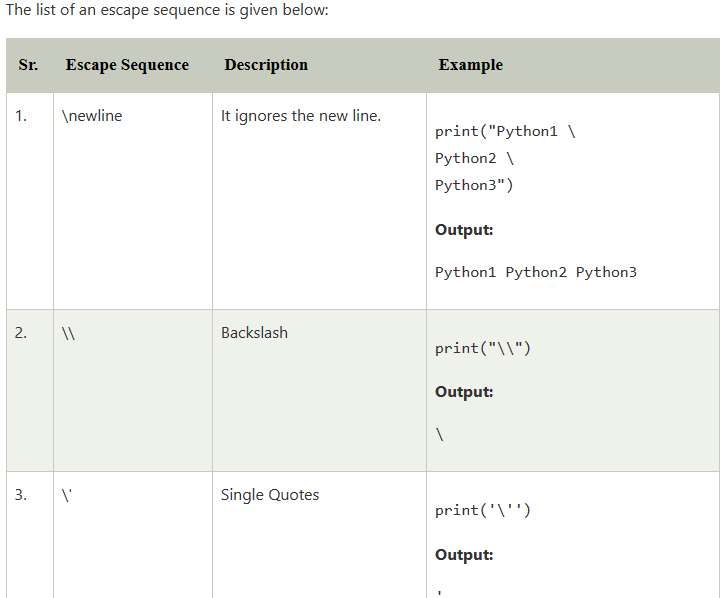
Another way is the backslash(/) symbol denotes the escape sequence. The backslash can be followed by a special character and it interpreted differently. The single quotes inside the string must be escaped. We can apply the same as in the double quotes.

Below is an example

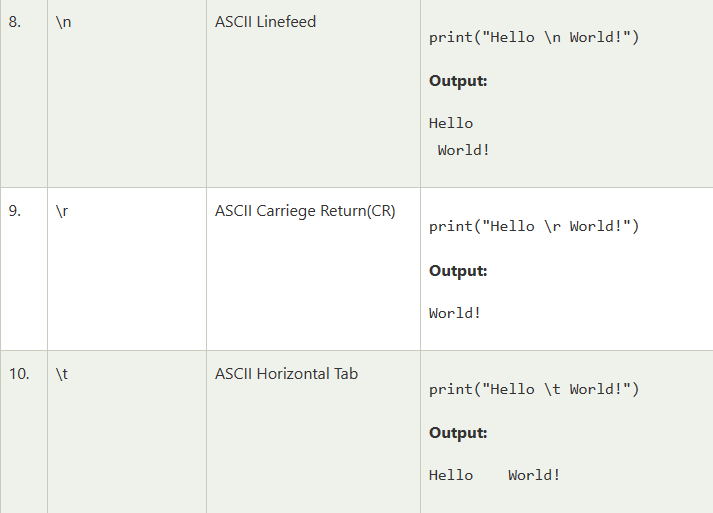
*# using triple quotes*print('''They said, "What's there?"''')  
  
*# escaping single quotes*print('They said, "What\'s going on?"')  
  
*# escaping double quotes*print("They said, \"What's going on?\"")

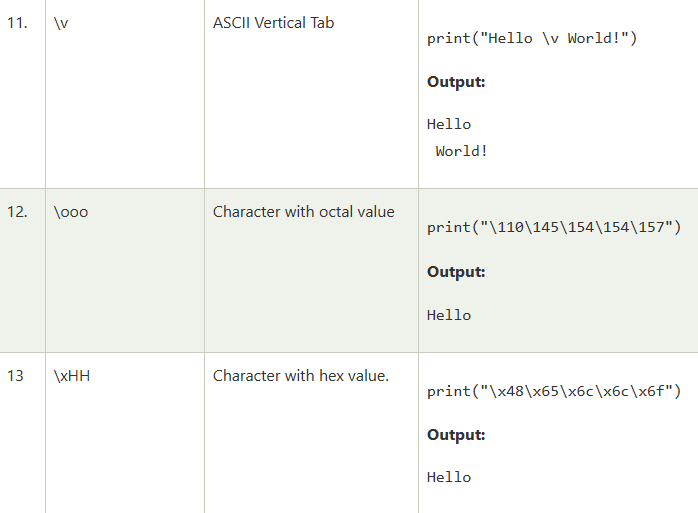


Below is the list of escape sequence



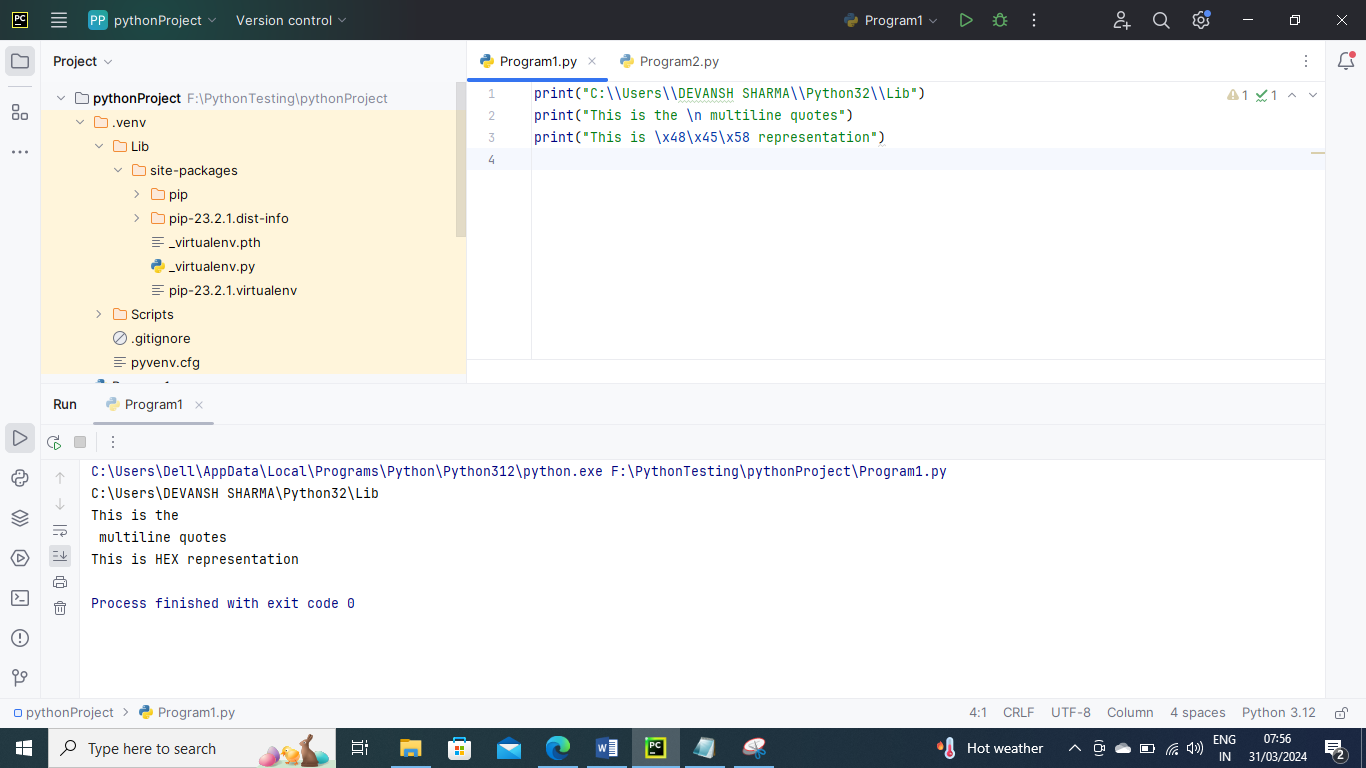






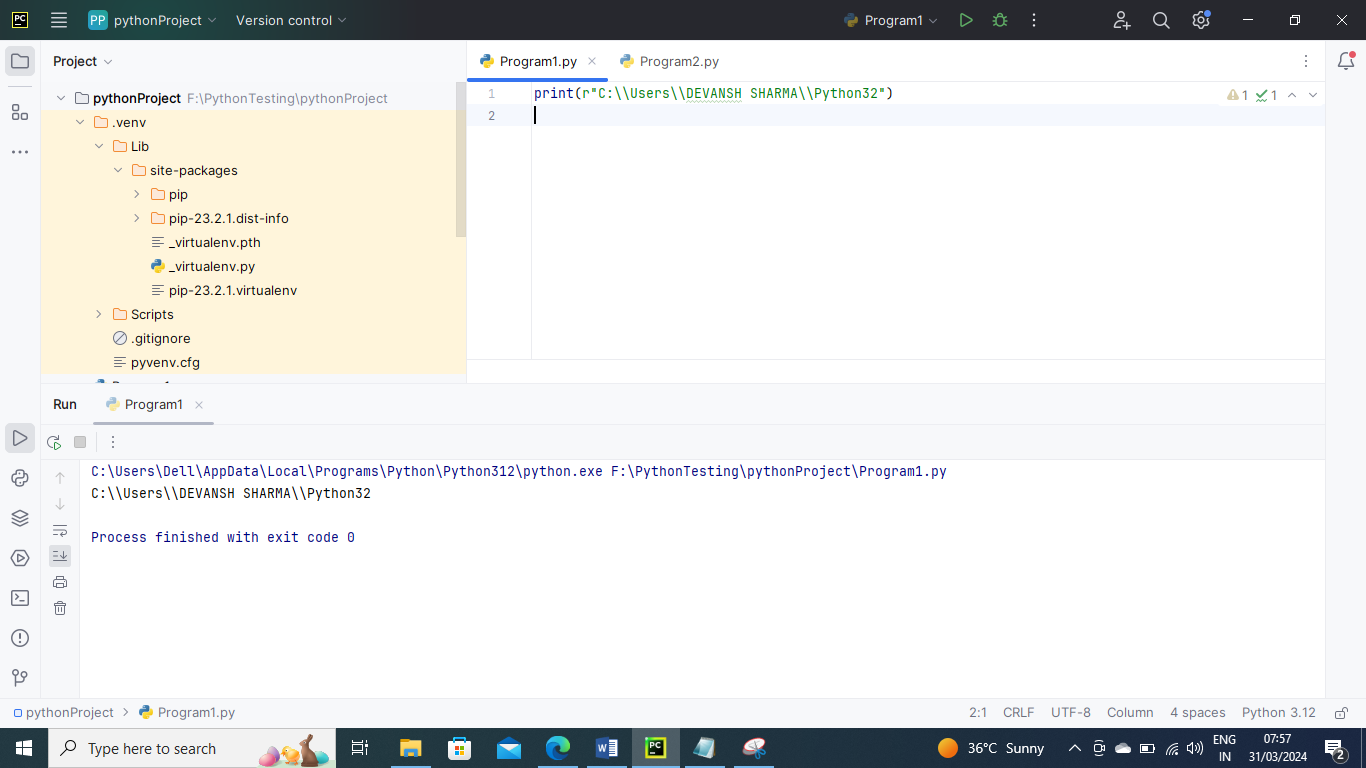
Below is an example of escape sequence

print("C:\\Users\\DEVANSH SHARMA\\Python32\\Lib")  
print("This is the \n multiline quotes")  
print("This is \x48\x45\x58 representation")



We can ignore the escape sequence from the given string by using the raw string. We can do this by writing **r** or **R** in front of the string. Consider the following example.

print(r"C:\\Users\\DEVANSH SHARMA\\Python32")

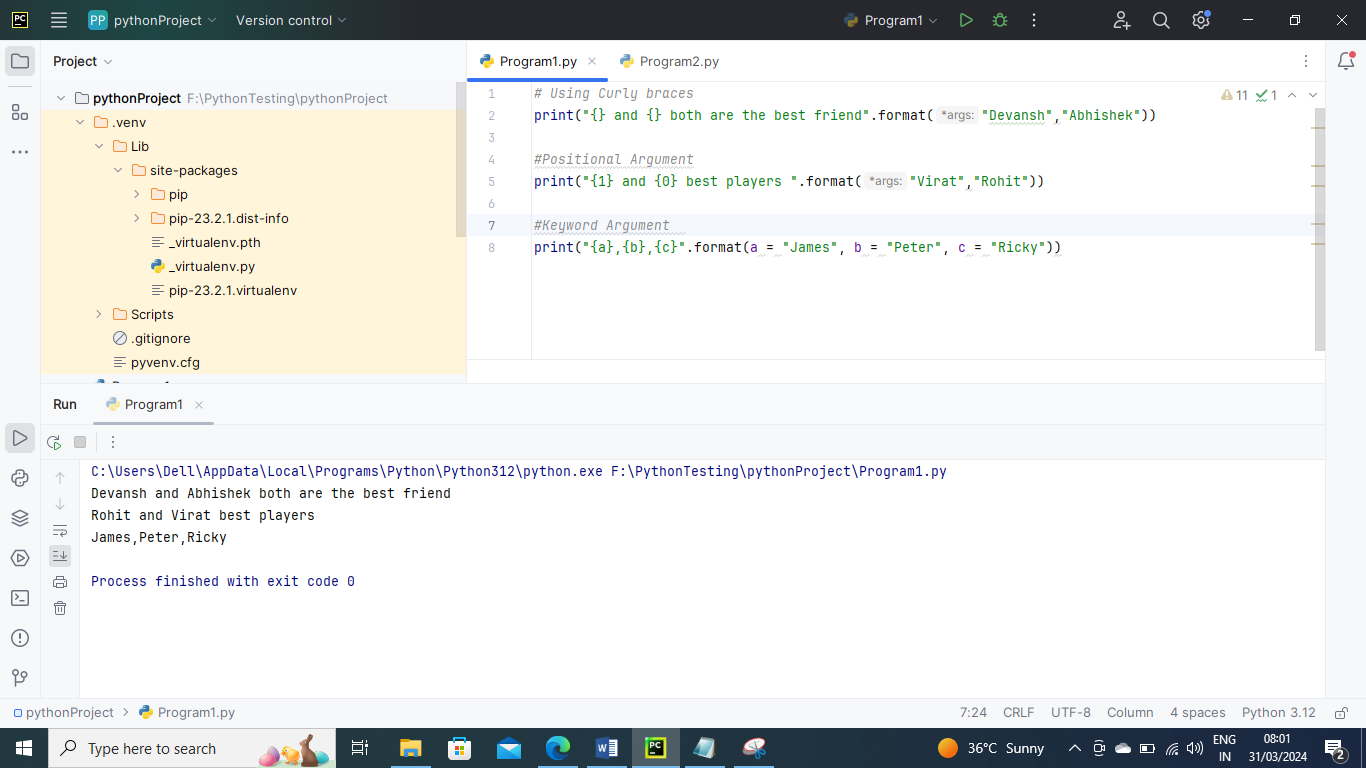


format() method-

The **format()** method is the most flexible and useful method in formatting strings. The curly braces {} are used as the placeholder in the string and replaced by the **format()** method argument.

Below is an example

*# 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"))



Formatting using % Operator –

Python provides an additional operator %, which is used as an interface between the format specifiers and their values.

Below is an example

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))

