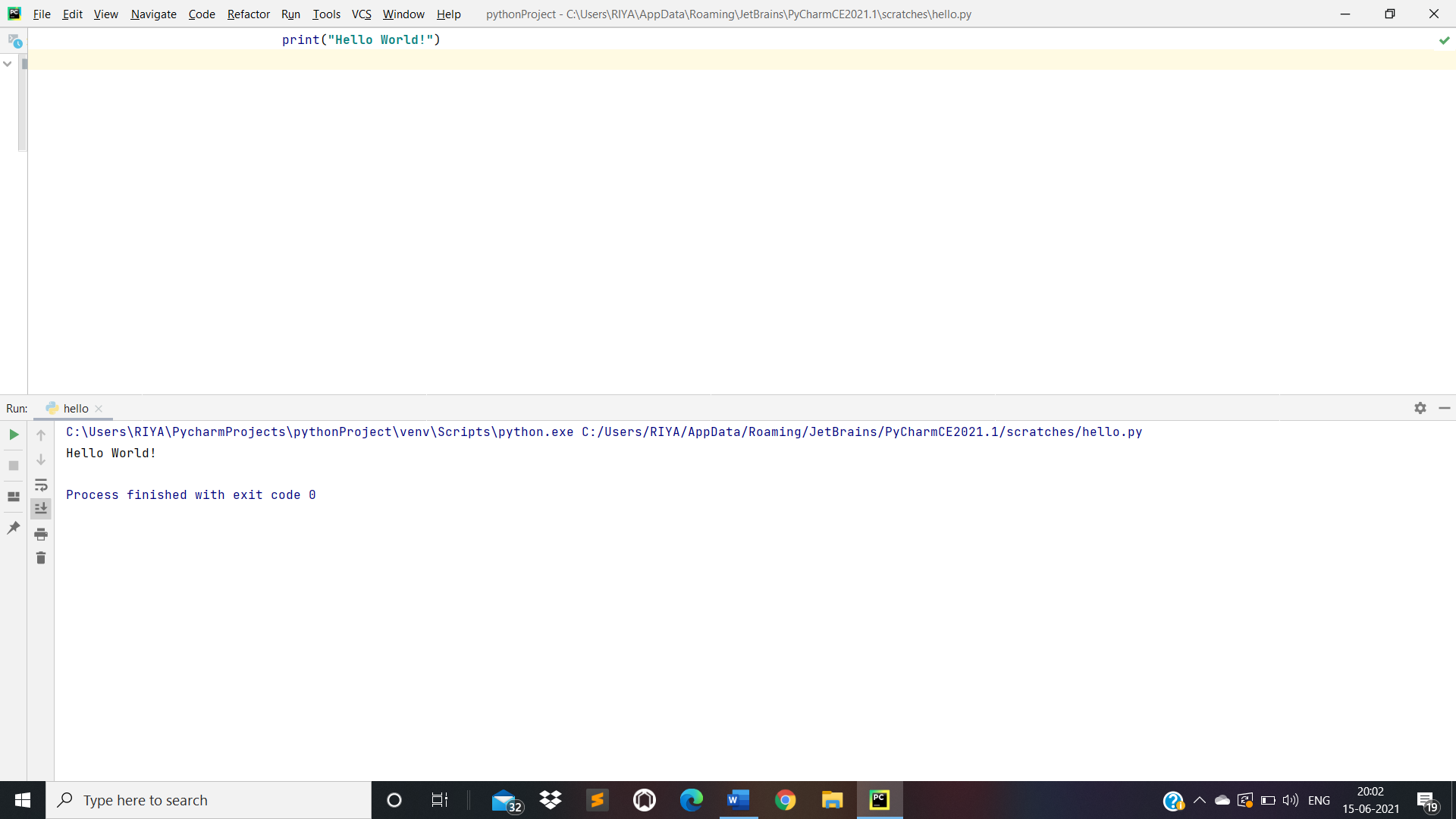
Python Programs

Daily work

Day 1:

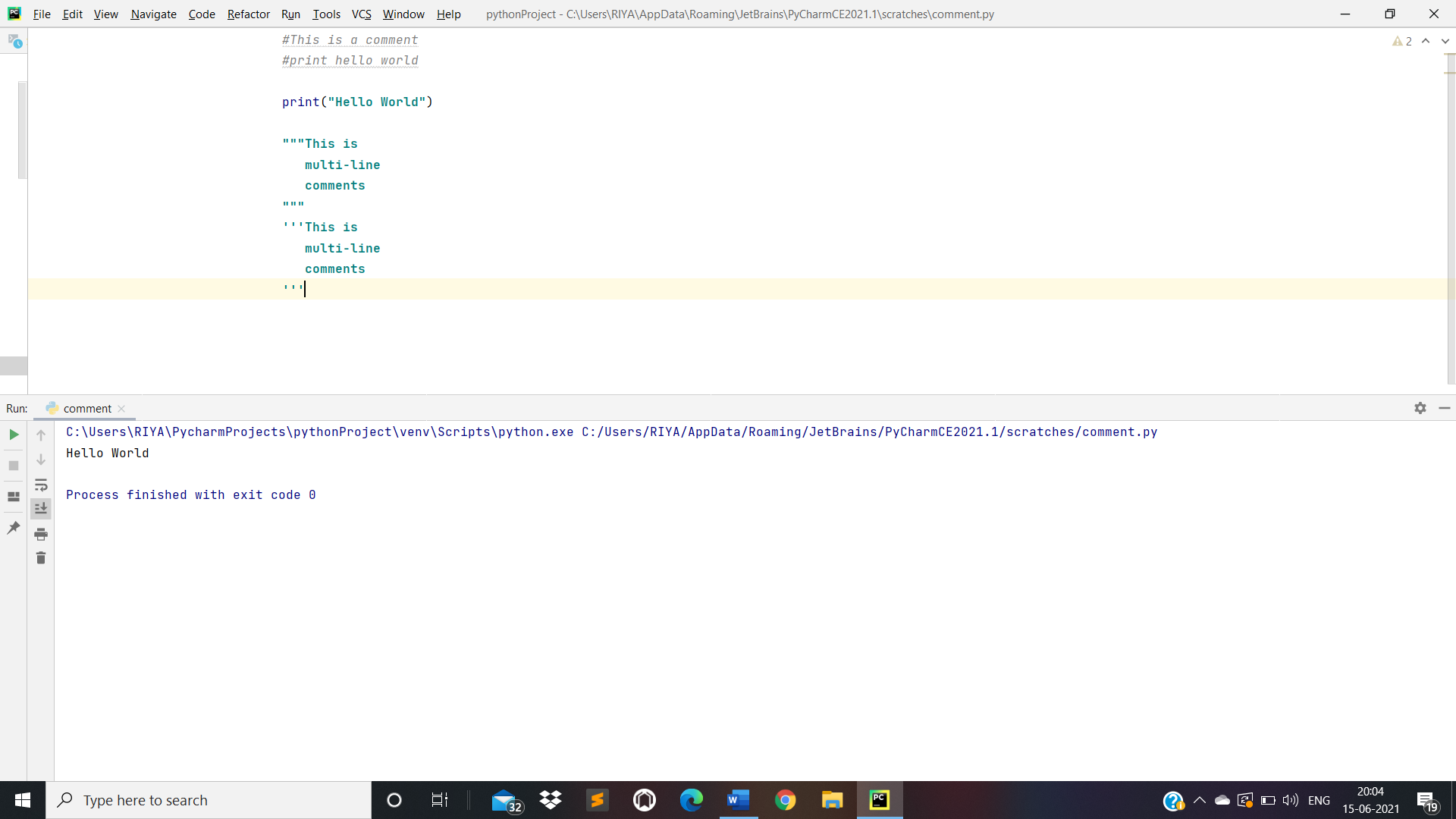
print(**"Hello World!"**)



Day 2:

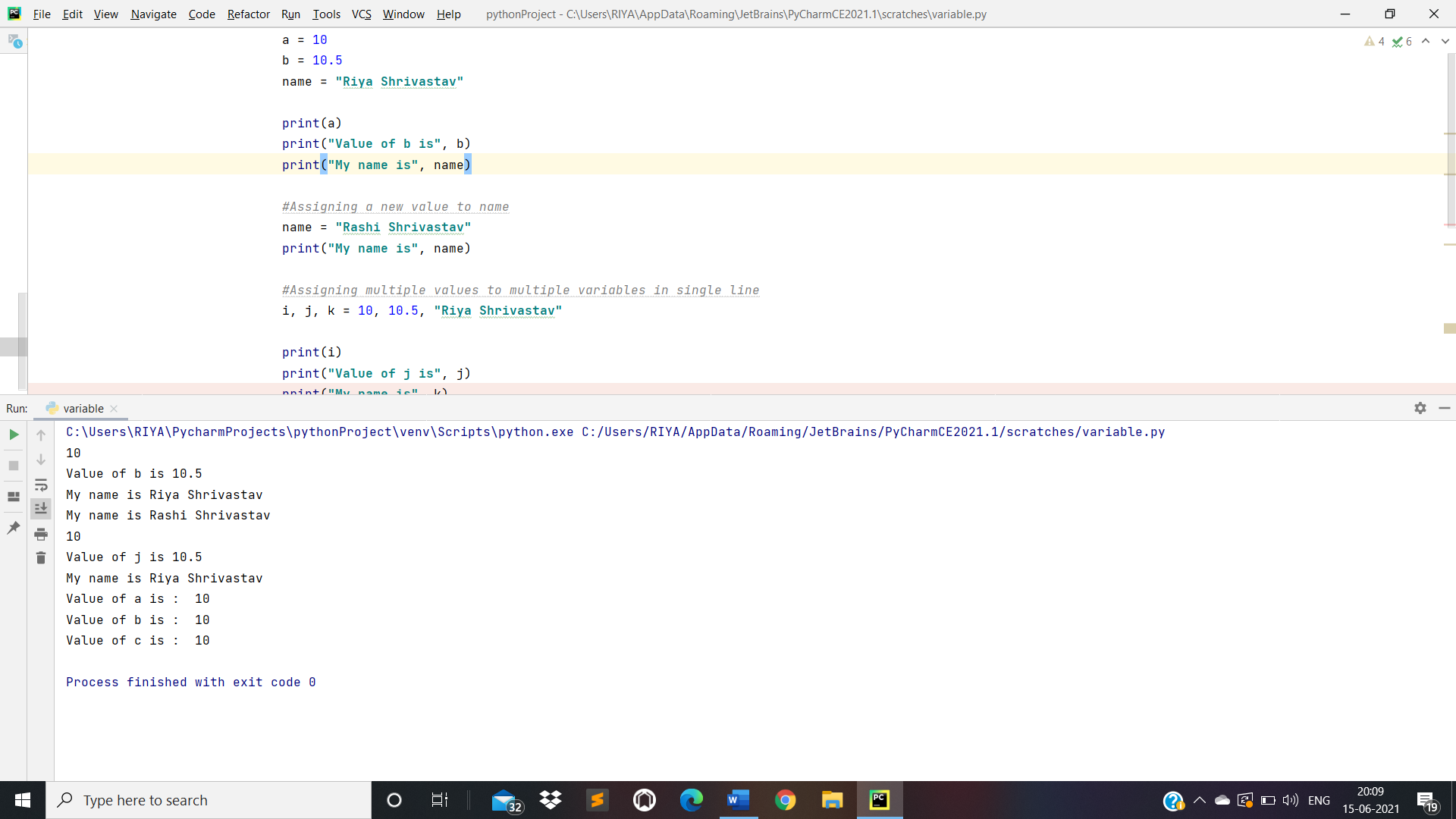
* Comments in Python:

*#This is a comment  
#print hello world*print(**"Hello World"**)  
  
**"""This is  
 multi-line   
 comments  
"""  
'''This is  
 multi-line  
 comments  
'''**



* Variables in Python:

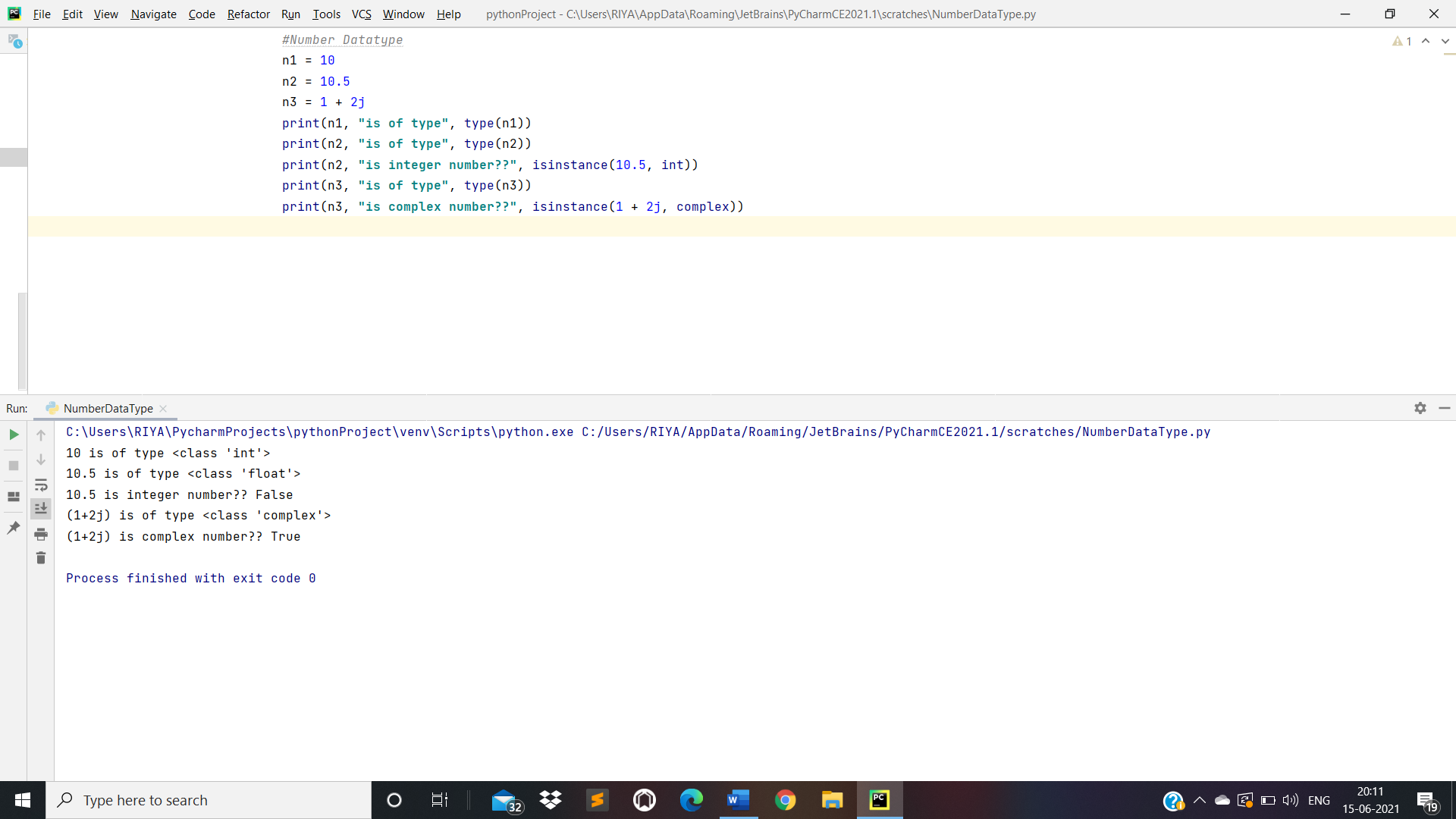
a = 10  
b = 10.5  
name = **"Riya Shrivastav"**print(a)  
print(**"Value of b is"**, b)  
print(**"My name is"**, name)  
  
*#Assigning a new value to name*name = **"Rashi Shrivastav"**print(**"My name is"**, name)  
  
*#Assigning multiple values to multiple variables in single line*i, j, k = 10, 10.5, **"Riya Shrivastav"**print(i)  
print(**"Value of j is"**, j)  
print(**"My name is"**, k)  
  
*#Assign the same value to multiple variables in single line*m = n = o = 10  
print(**"Value of a is : "**, m)  
print(**"Value of b is : "**, n)  
print(**"Value of c is : "**, o)



* Data Types in Python:

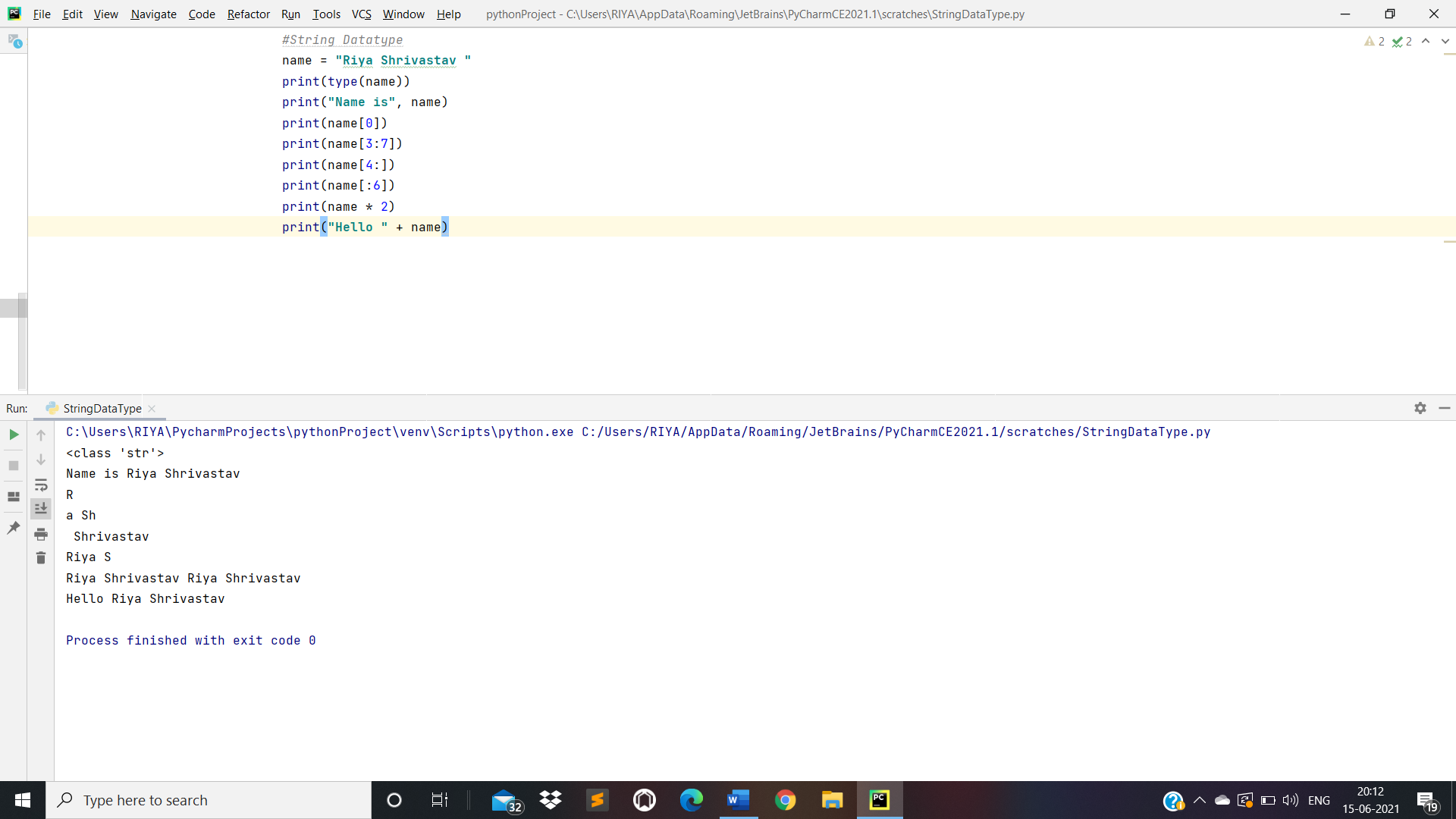
1. Number Datatype:

*#Number Datatype*n1 = 10  
n2 = 10.5  
n3 = 1 + 2j  
print(n1, **"is of type"**, type(n1))  
print(n2, **"is of type"**, type(n2))  
print(n2, **"is integer number??"**, isinstance(10.5, int))  
print(n3, **"is of type"**, type(n3))  
print(n3, **"is complex number??"**, isinstance(1 + 2j, complex))



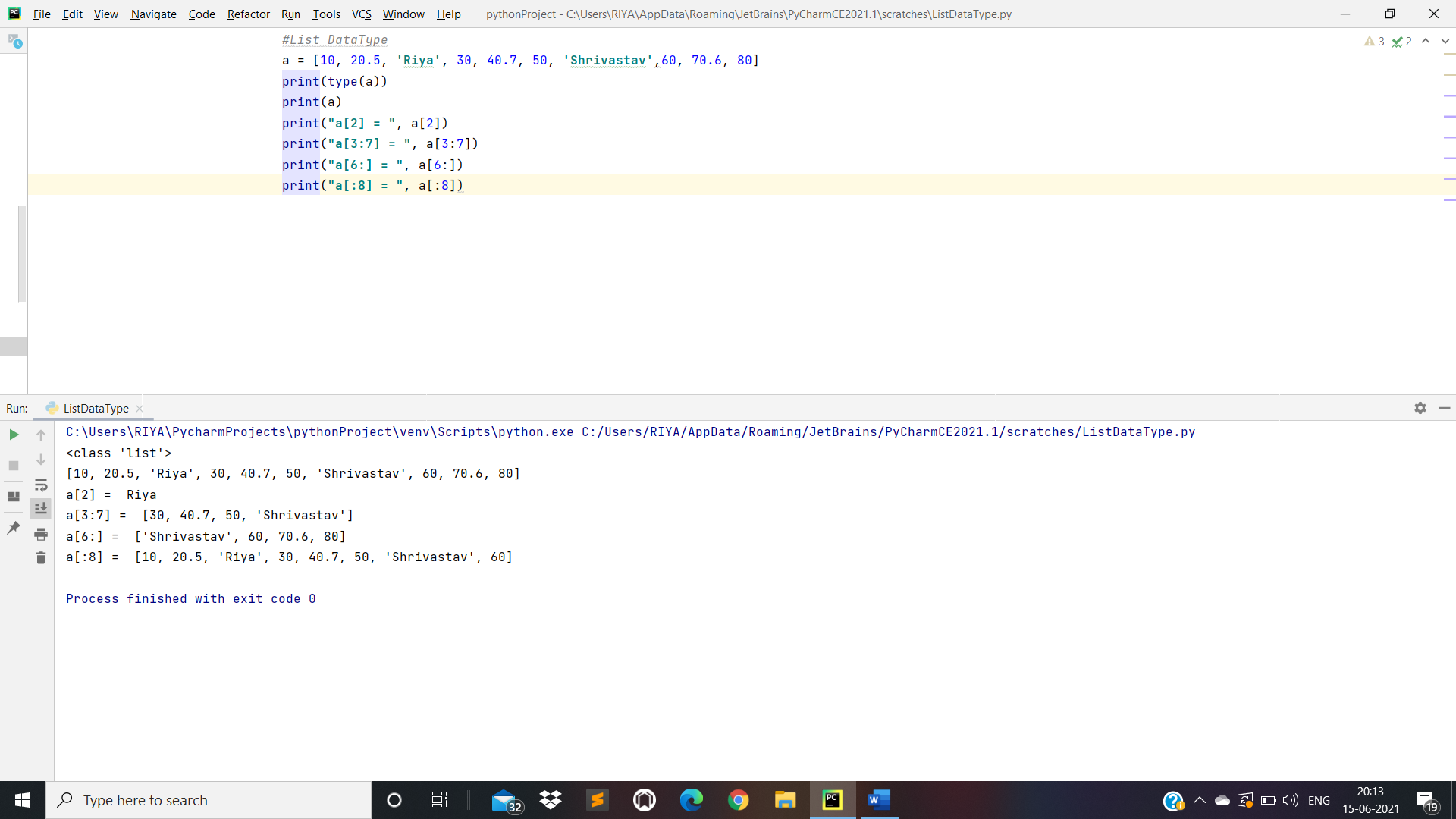
1. String Datatype:

*#String Datatype*name = **"Riya Shrivastav "**print(type(name))  
print(**"Name is"**, name)  
print(name[0])  
print(name[3:7])  
print(name[4:])  
print(name[:6])  
print(name \* 2)  
print(**"Hello "** + name)



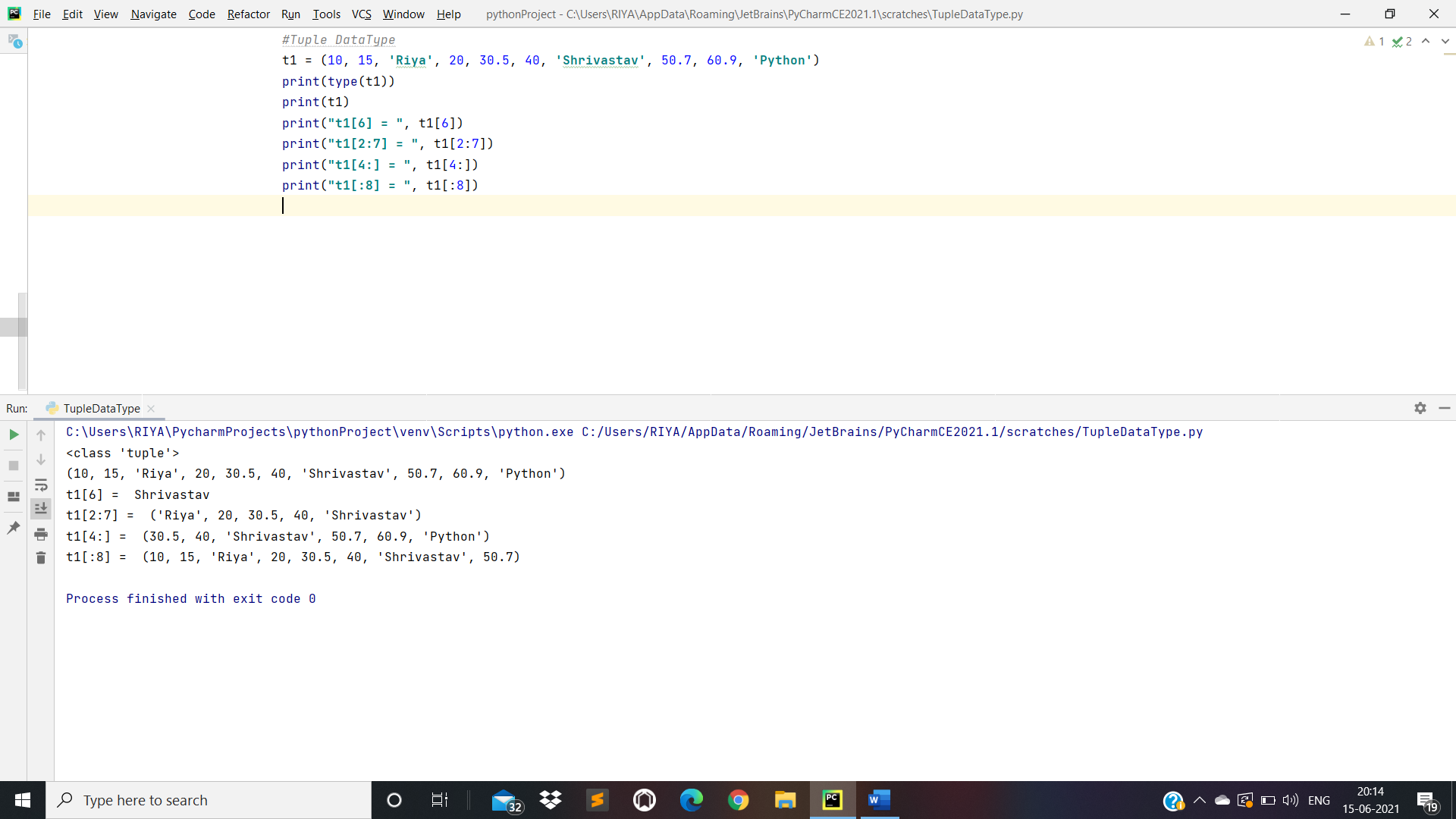
1. List Datatype:

*#List DataType*a = [10, 20.5, **'Riya'**, 30, 40.7, 50, **'Shrivastav'**,60, 70.6, 80]  
print(type(a))  
print(a)  
print(**"a[2] = "**, a[2])  
print(**"a[3:7] = "**, a[3:7])  
print(**"a[6:] = "**, a[6:])  
print(**"a[:8] = "**, a[:8])



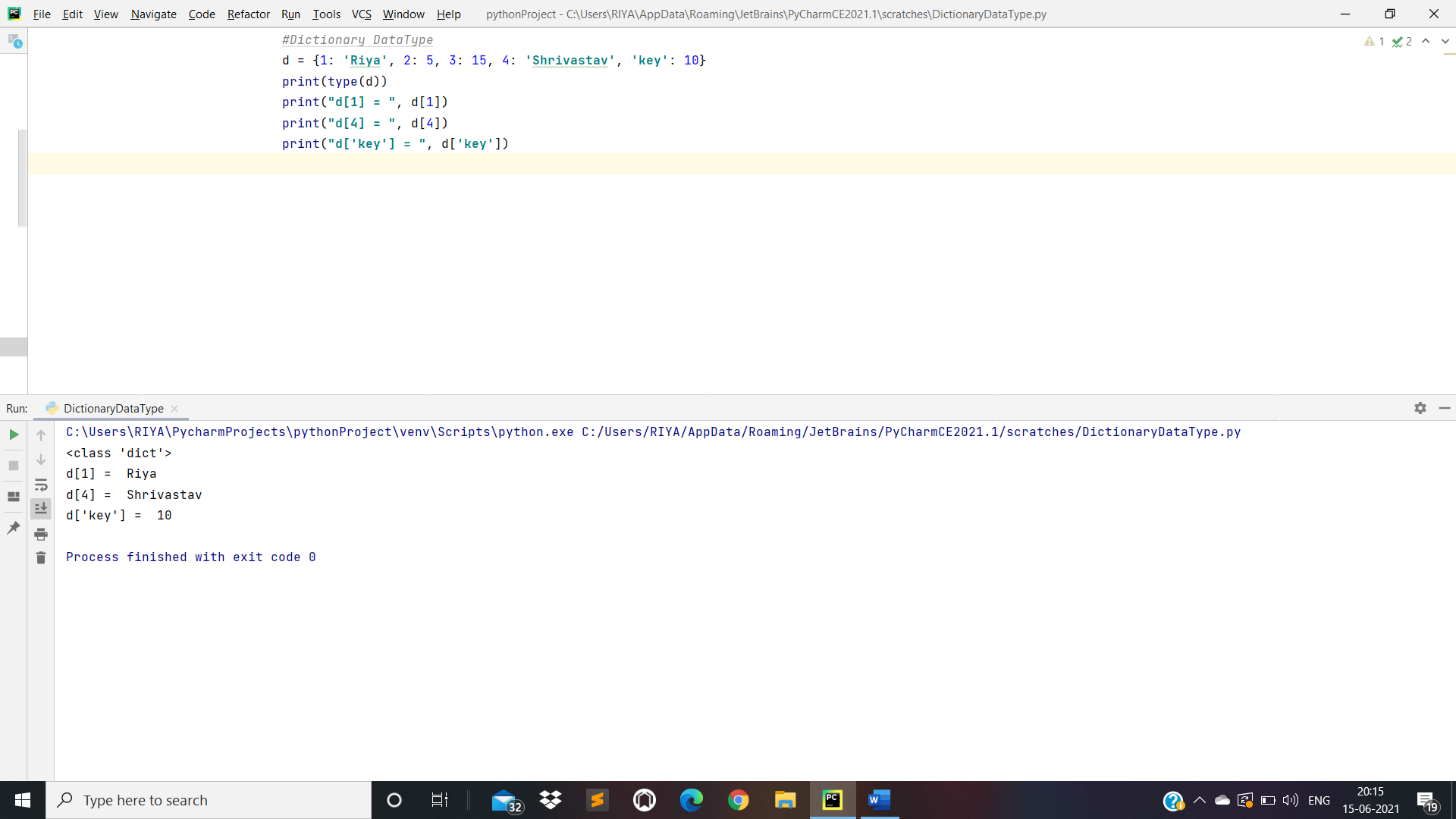
1. Tuple Datatype:

*#Tuple DataType*t1 = (10, 15, **'Riya'**, 20, 30.5, 40, **'Shrivastav'**, 50.7, 60.9, **'Python'**)  
print(type(t1))  
print(t1)  
print(**"t1[6] = "**, t1[6])  
print(**"t1[2:7] = "**, t1[2:7])  
print(**"t1[4:] = "**, t1[4:])  
print(**"t1[:8] = "**, t1[:8])



1. Dictionary Datatype:

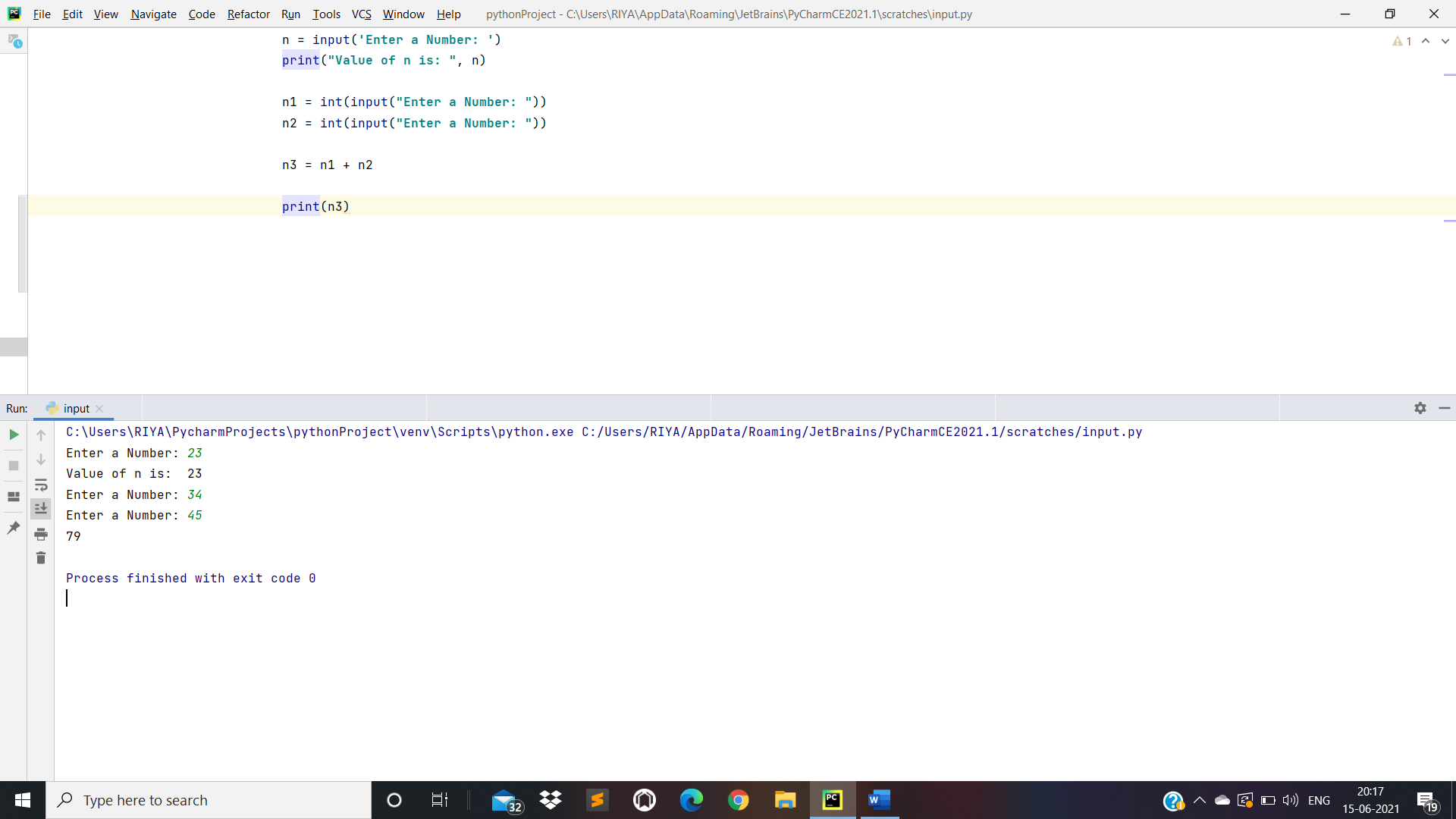
*#Dictionary DataType*d = {1: **'Riya'**, 2: 5, 3: 15, 4: **'Shrivastav'**, **'key'**: 10}  
print(type(d))  
print(**"d[1] = "**, d[1])  
print(**"d[4] = "**, d[4])  
print(**"d['key'] = "**, d[**'key'**])



Day 3: I/O Function and Conditional Statements

* Input/Output

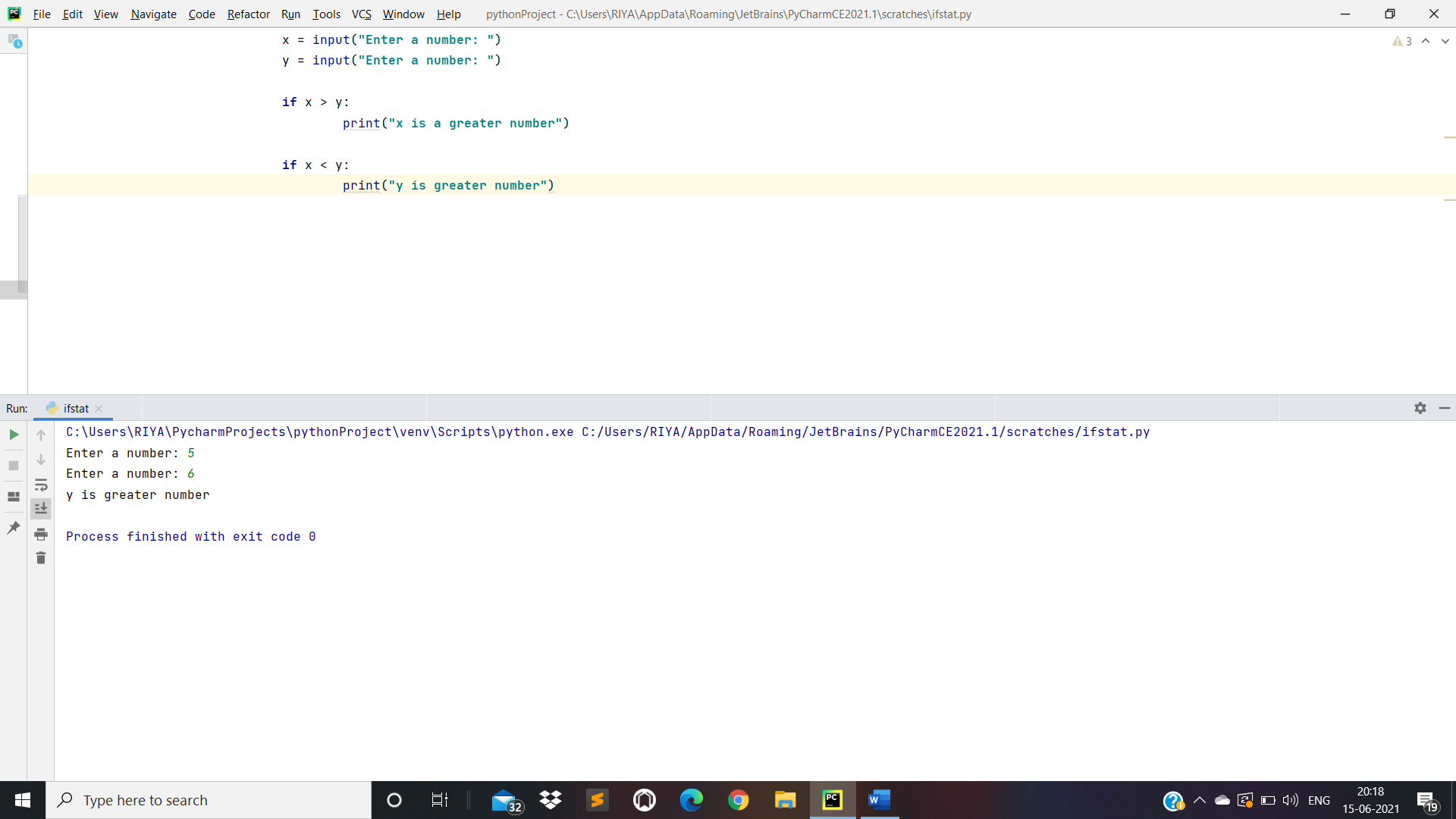
n = input(**'Enter a Number: '**)  
print(**"Value of n is: "**, n)  
  
n1 = int(input(**"Enter a Number: "**))  
n2 = int(input(**"Enter a Number: "**))  
  
n3 = n1 + n2  
  
print(n3)



* Conditional Statements:

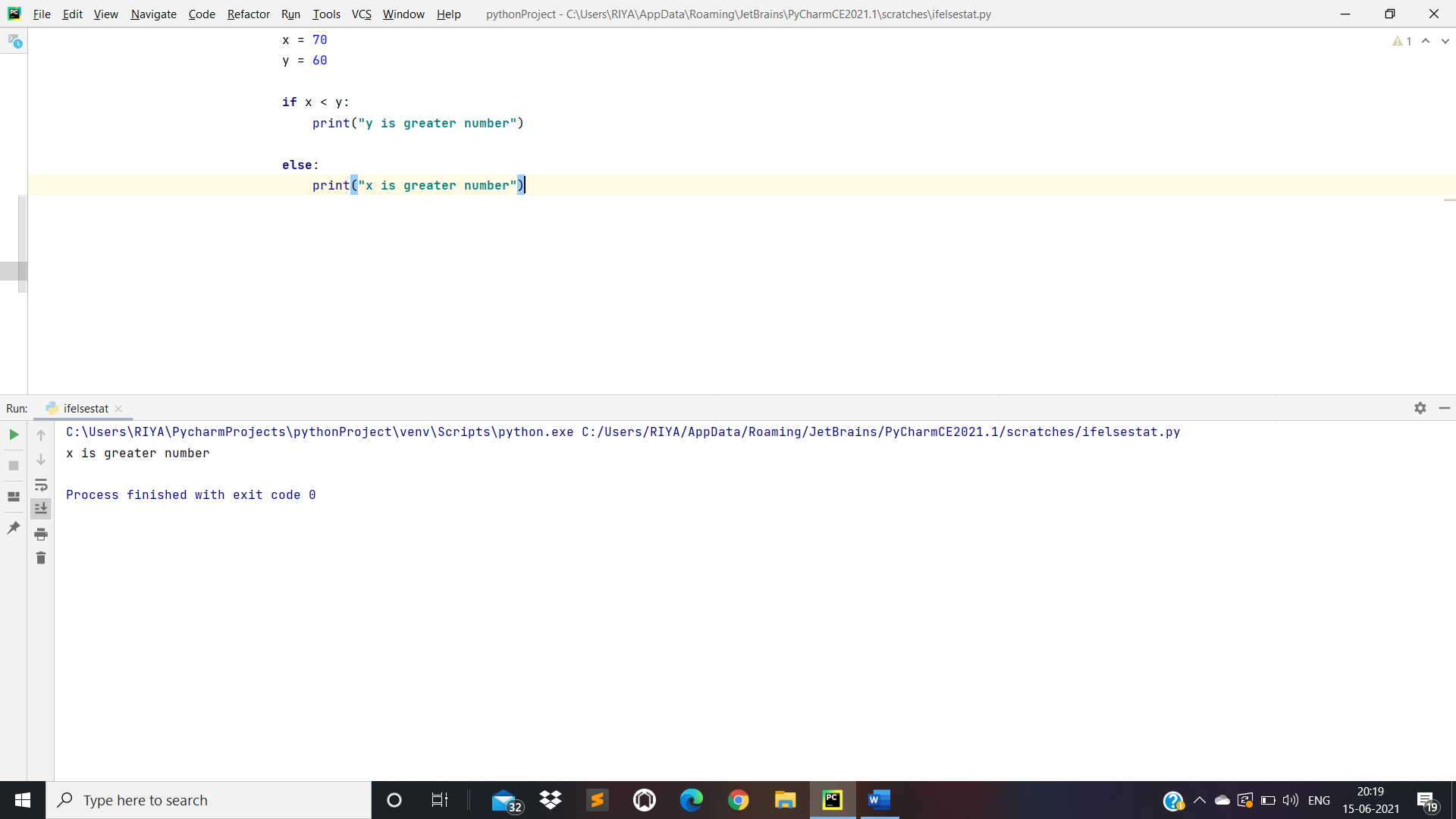
1. If statement:

x = input(**"Enter a number: "**)  
y = input(**"Enter a number: "**)  
  
**if** x > y:  
 print(**"x is a greater number"**)  
  
**if** x < y:  
 print(**"y is greater number"**)



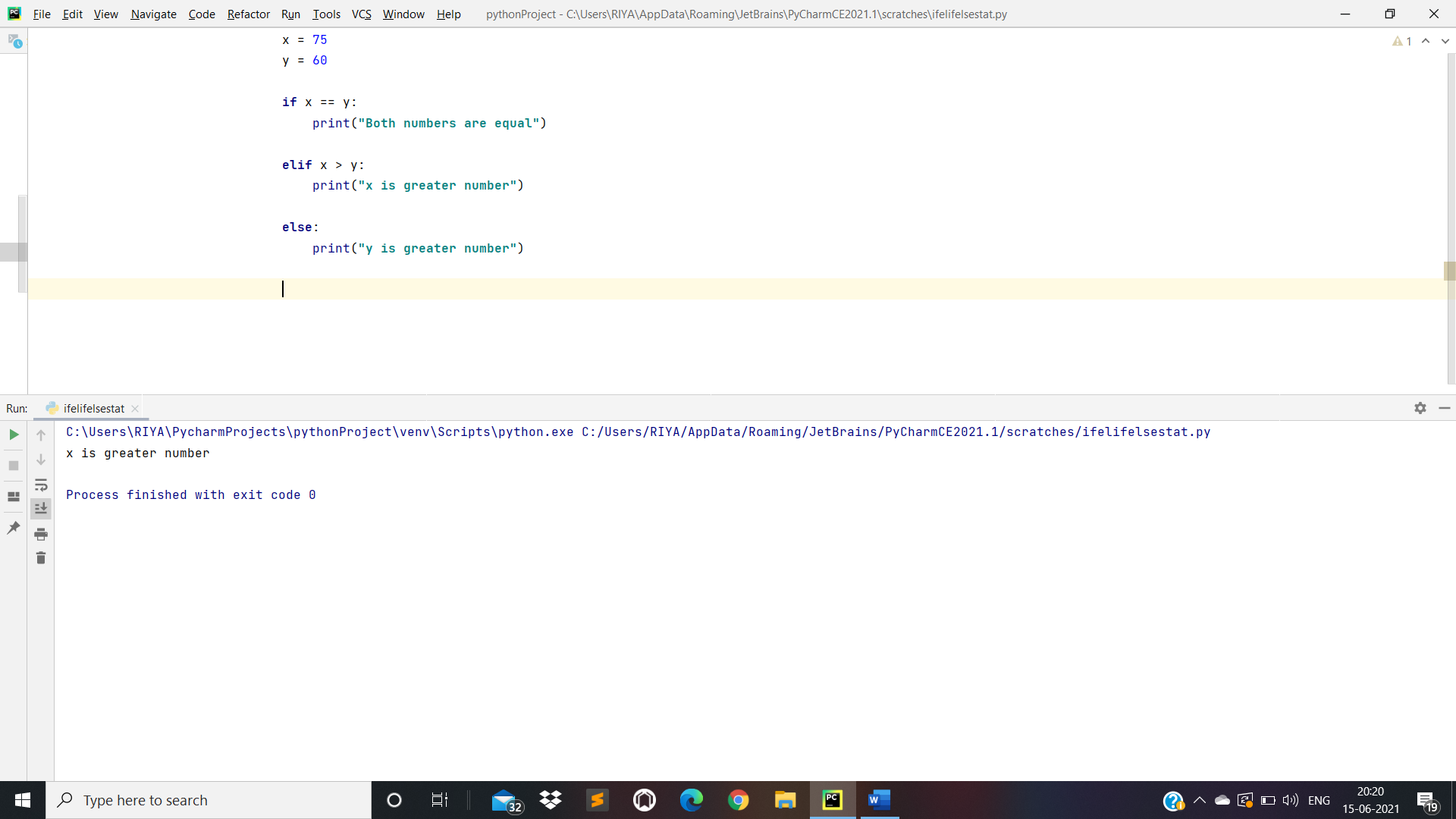
1. If…Else Statement:

x = 70  
y = 60  
  
**if** x < y:  
 print(**"y is greater number"**)  
  
**else**:  
 print(**"x is greater number"**)



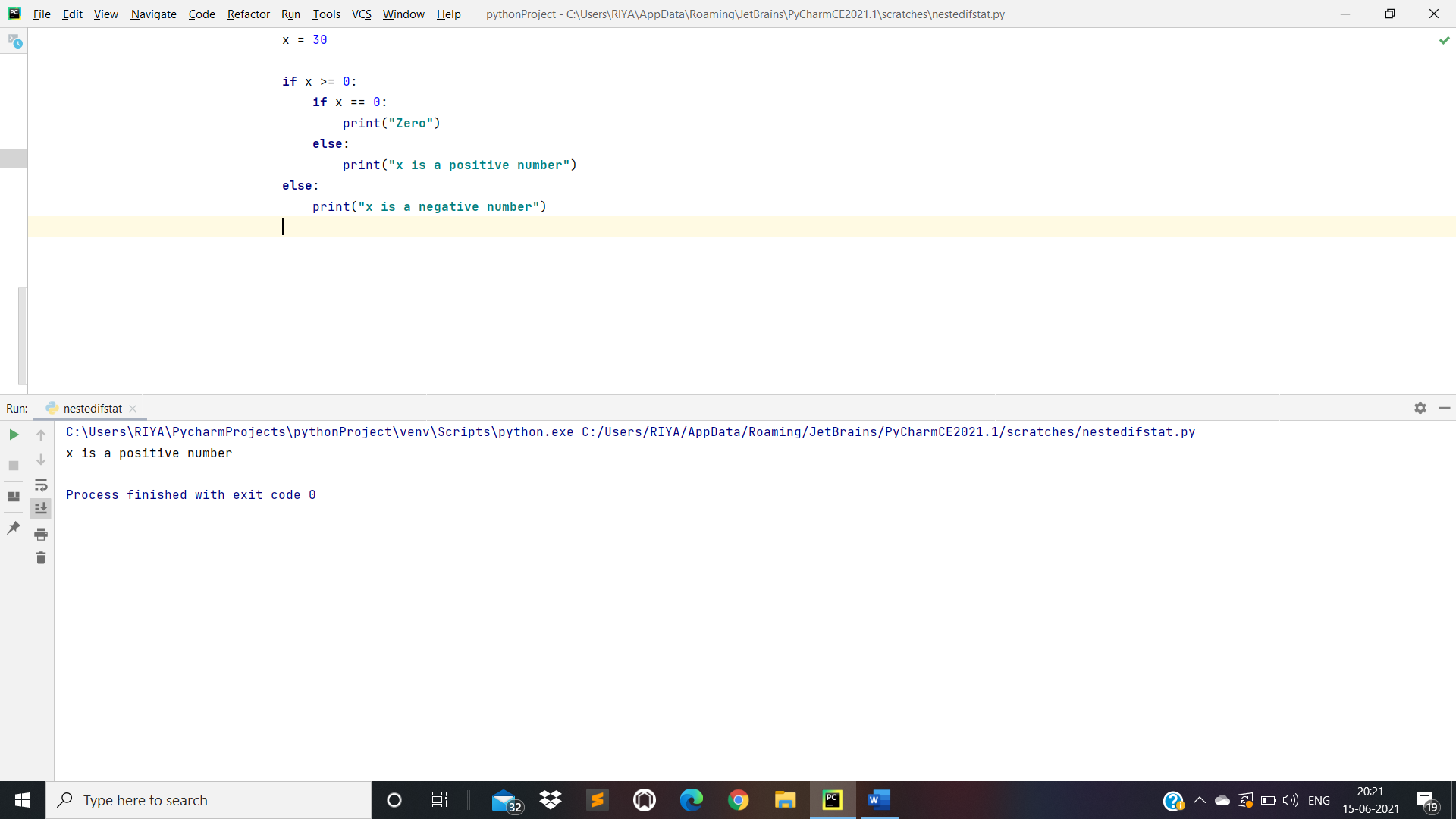
1. If…Elif…Else Statement:

x = 75  
y = 60  
  
**if** x == y:  
 print(**"Both numbers are equal"**)  
  
**elif** x > y:  
 print(**"x is greater number"**)  
  
**else**:  
 print(**"y is greater number"**)



1. Nested If Statements:

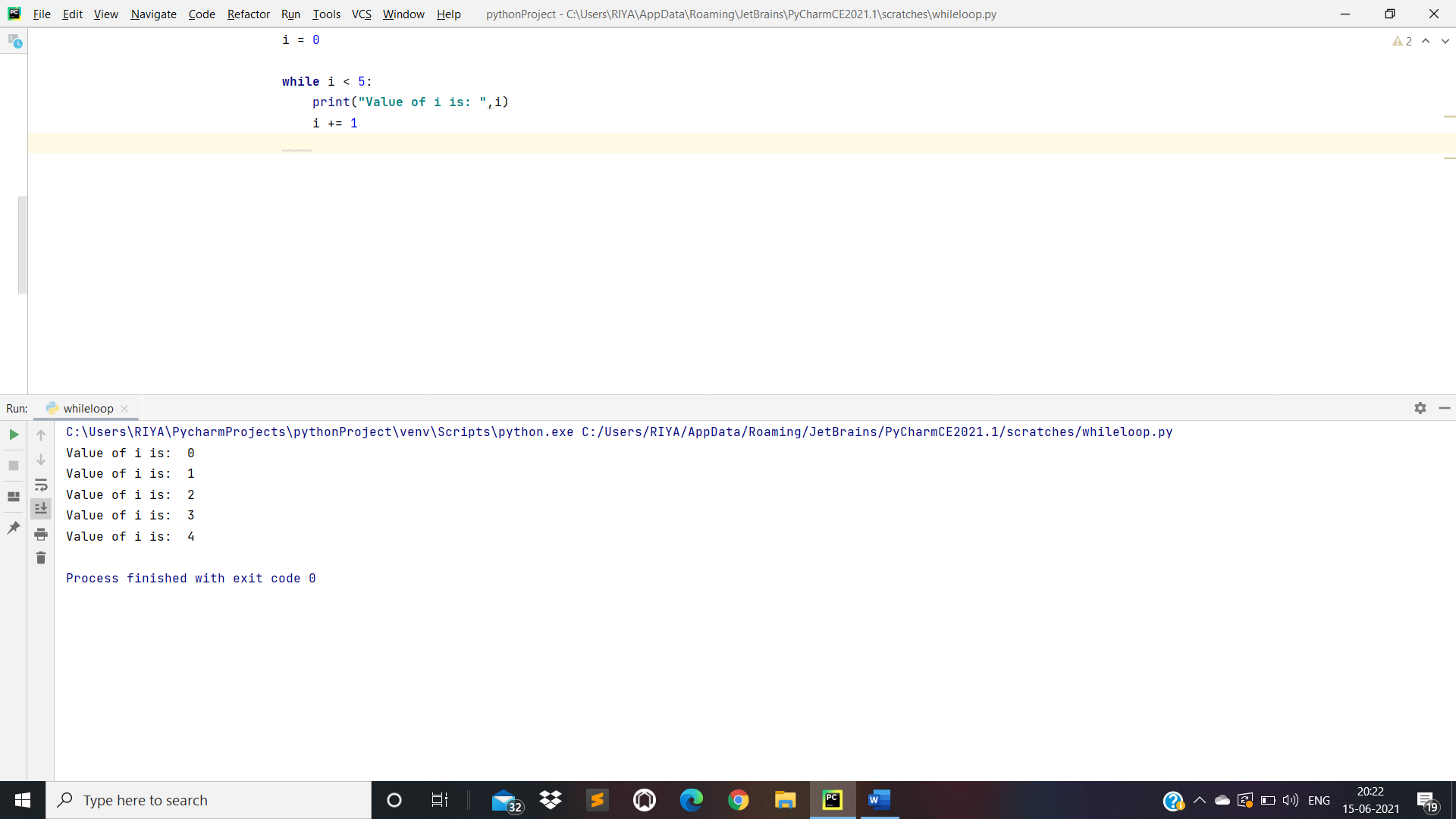
x = 30  
  
**if** x >= 0:  
 **if** x == 0:  
 print(**"Zero"**)  
 **else**:  
 print(**"x is a positive number"**)  
**else**:  
 print(**"x is a negative number"**)



* Loops:

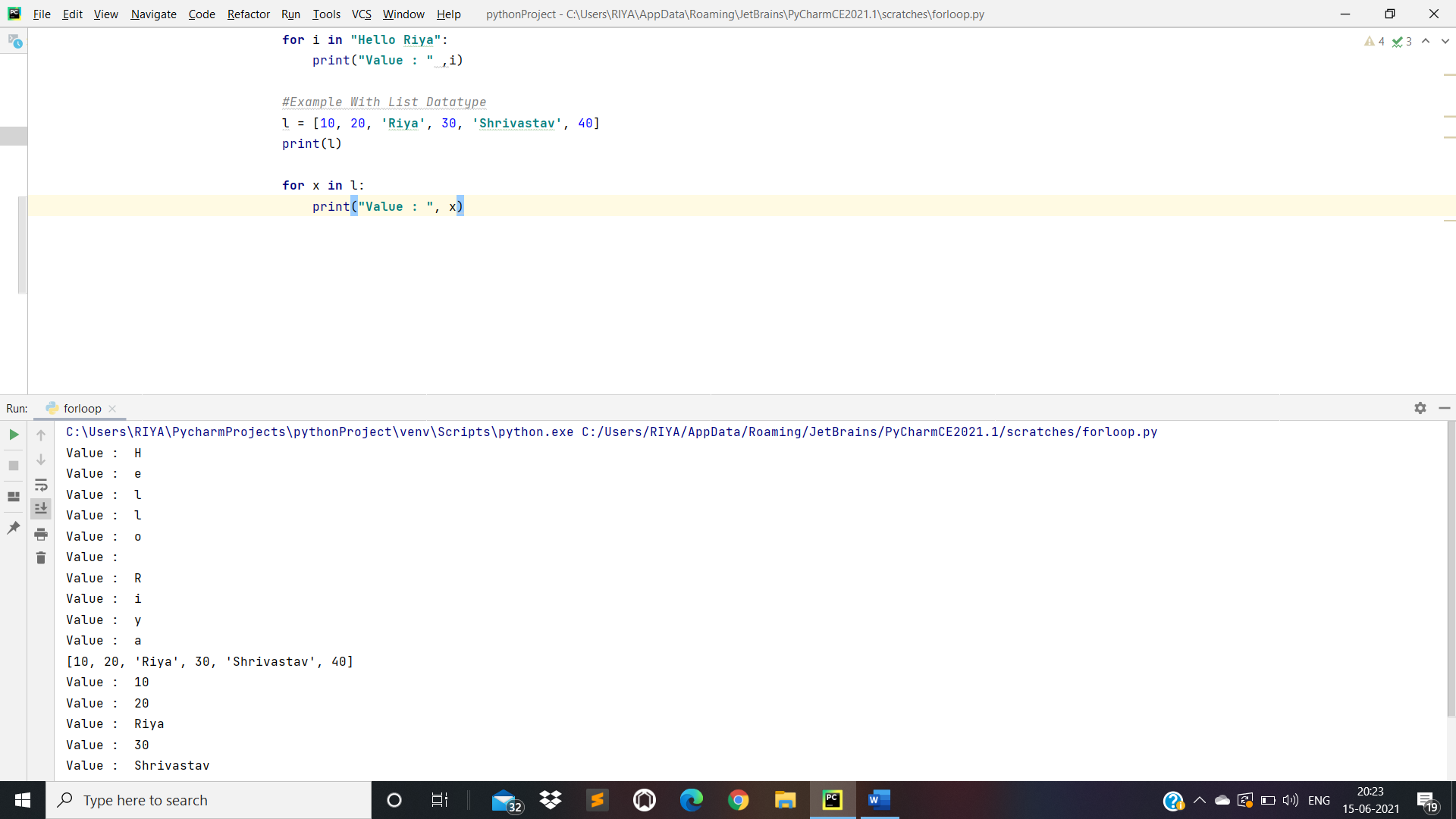
1. While Loop Statements:

i = 0  
  
**while** i < 5:  
 print(**"Value of i is: "**,i)  
 i += 1



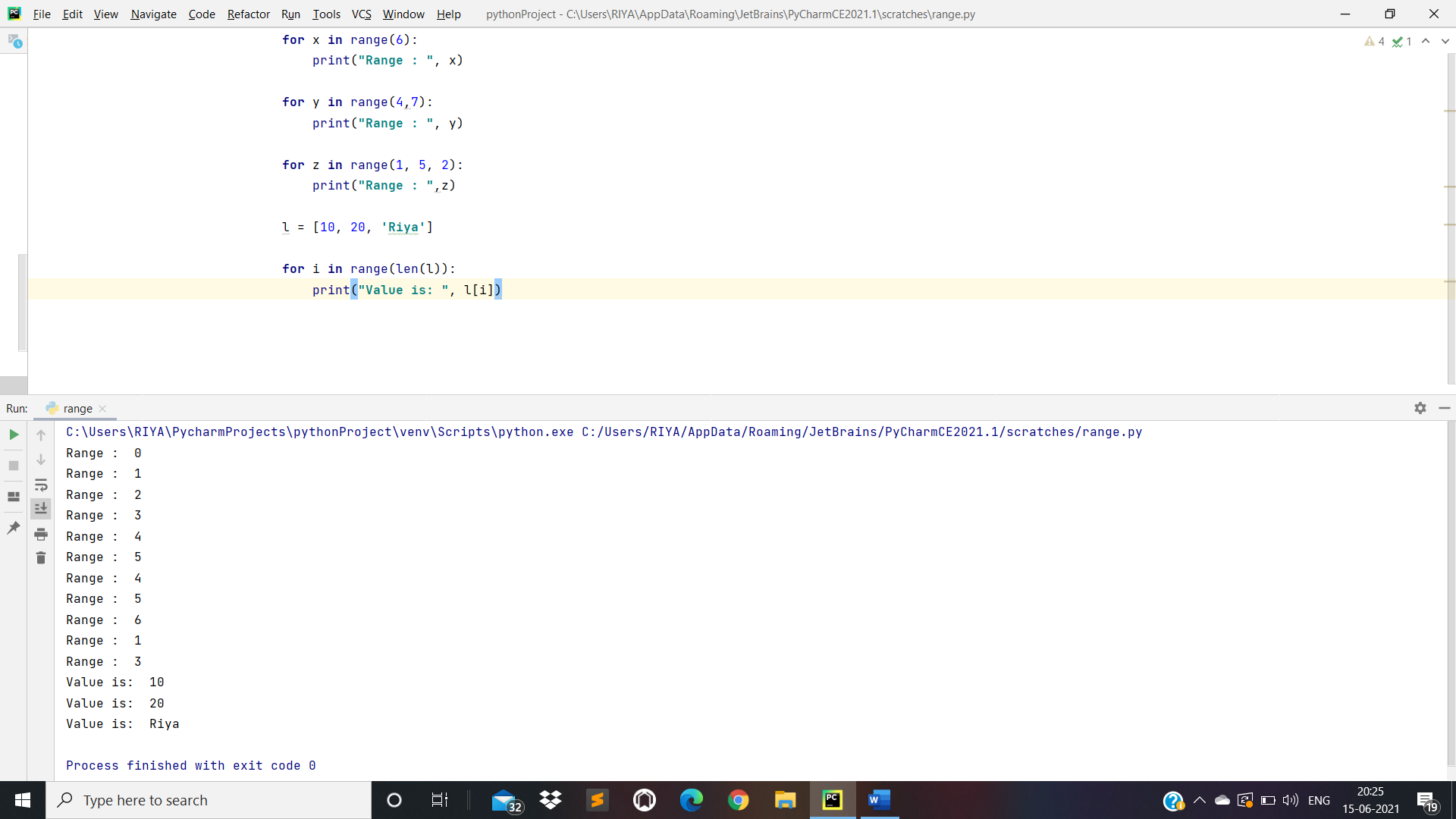
1. For Loop Statements:

**for** i **in "Hello Riya"**:  
 print(**"Value : "** ,i)  
  
*#Example With List Datatype*l = [10, 20, **'Riya'**, 30, **'Shrivastav'**, 40]  
print(l)  
  
**for** x **in** l:  
 print(**"Value : "**, x)



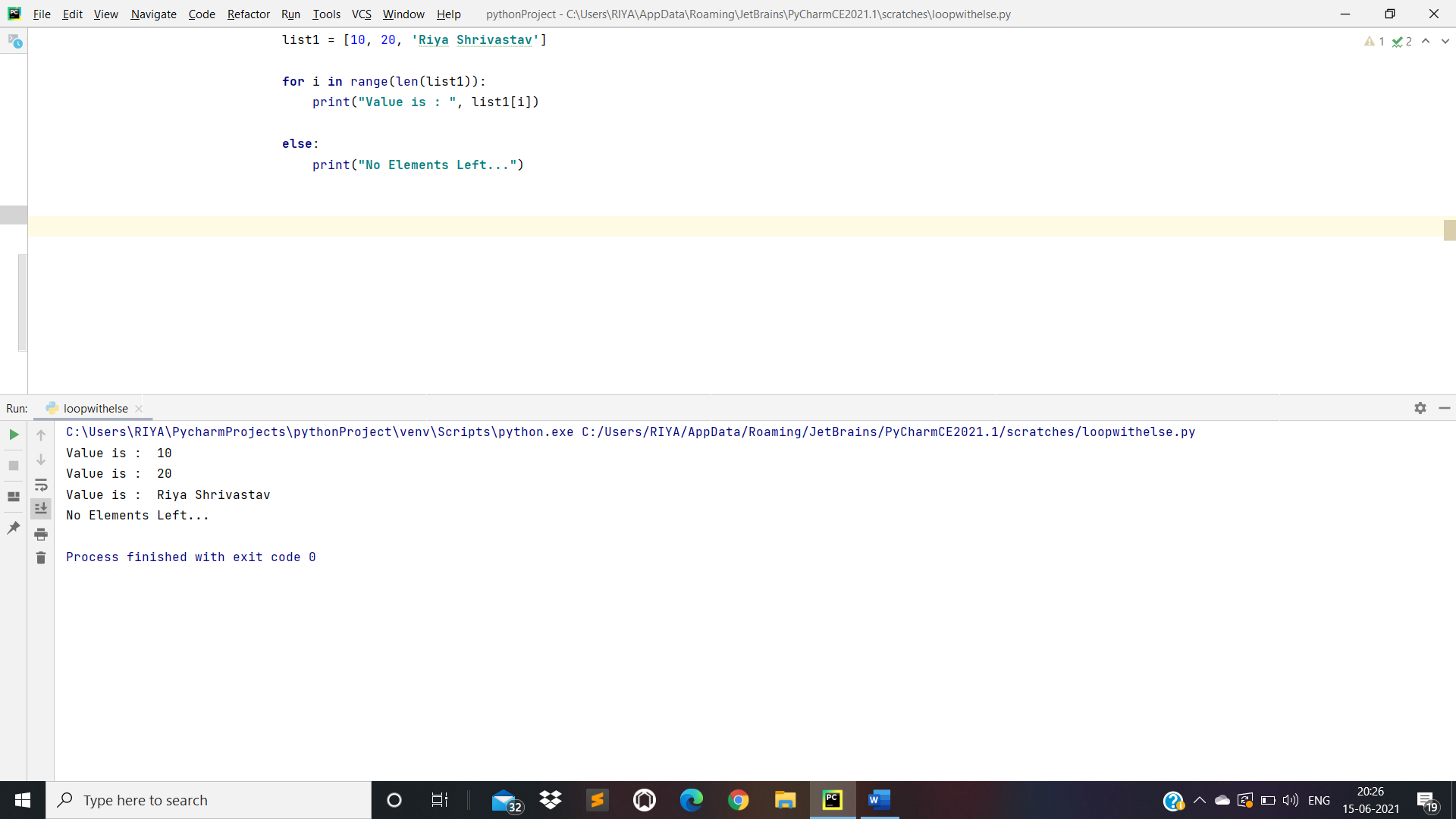
* The range() Function:

**for** x **in** range(6):  
 print(**"Range : "**, x)  
  
**for** y **in** range(4,7):  
 print(**"Range : "**, y)  
  
**for** z **in** range(1, 5, 2):  
 print(**"Range : "**,z)  
  
l = [10, 20, **'Riya'**]  
  
**for** i **in** range(len(l)):  
 print(**"Value is: "**, l[i])



* Loop with Else:

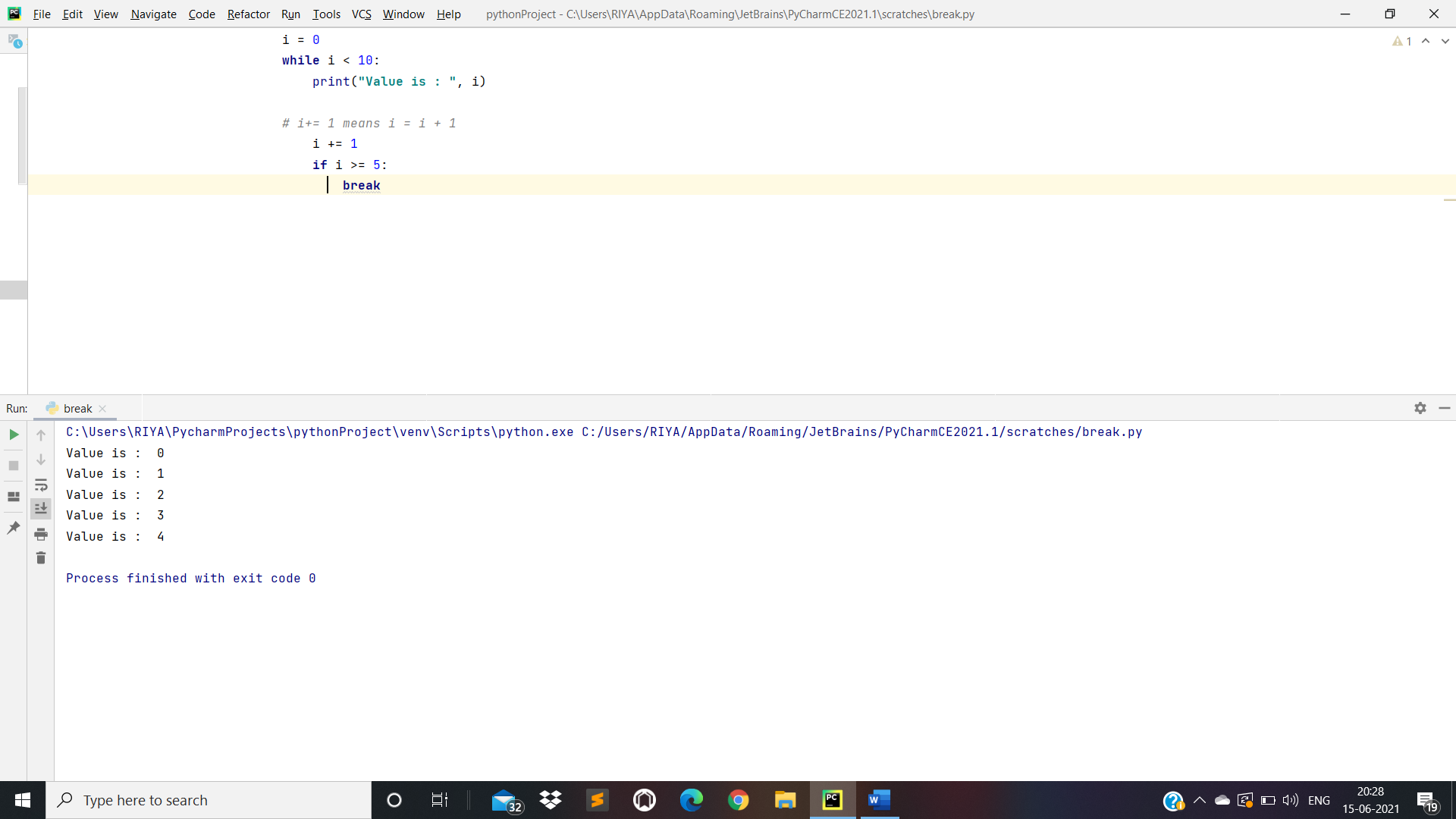
list1 = [10, 20, **'Riya Shrivastav'**]  
  
**for** i **in** range(len(list1)):  
 print(**"Value is : "**, list1[i])  
  
**else**:  
 print(**"No Elements Left..."**)



* ‘‘Break’’ and “Continue” Statement:

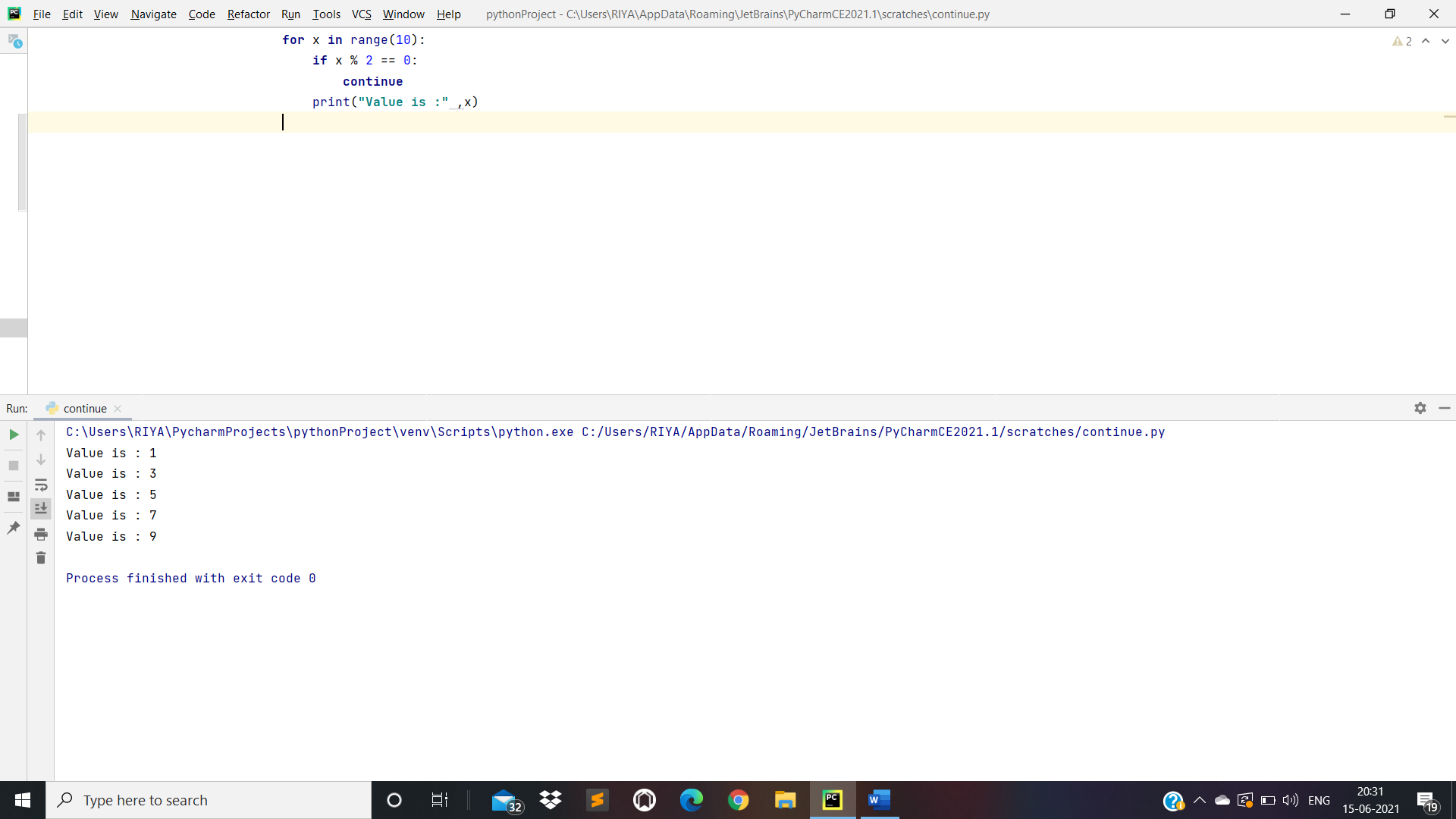
Example of Break Statement:

i = 0  
**while** i < 10:  
 print(**"Value is : "**, i)  
  
*# i+= 1 means i = i + 1* i += 1  
 **if** i >= 5:  
 **break**



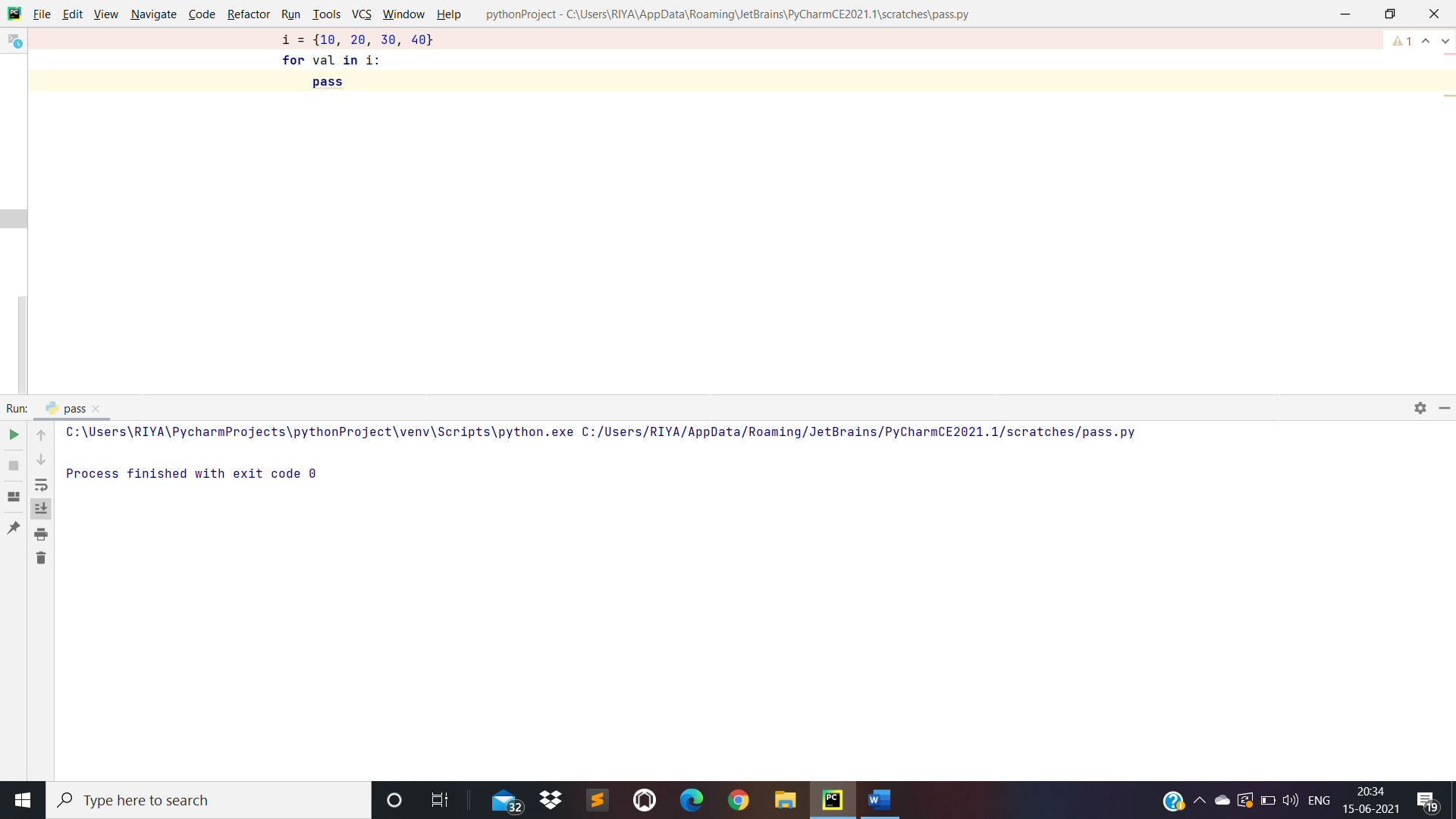
Example of Continue Statement:

**for** x **in** range(10):  
 **if** x % 2 == 0:  
 **continue** print(**"Value is :"** ,x)



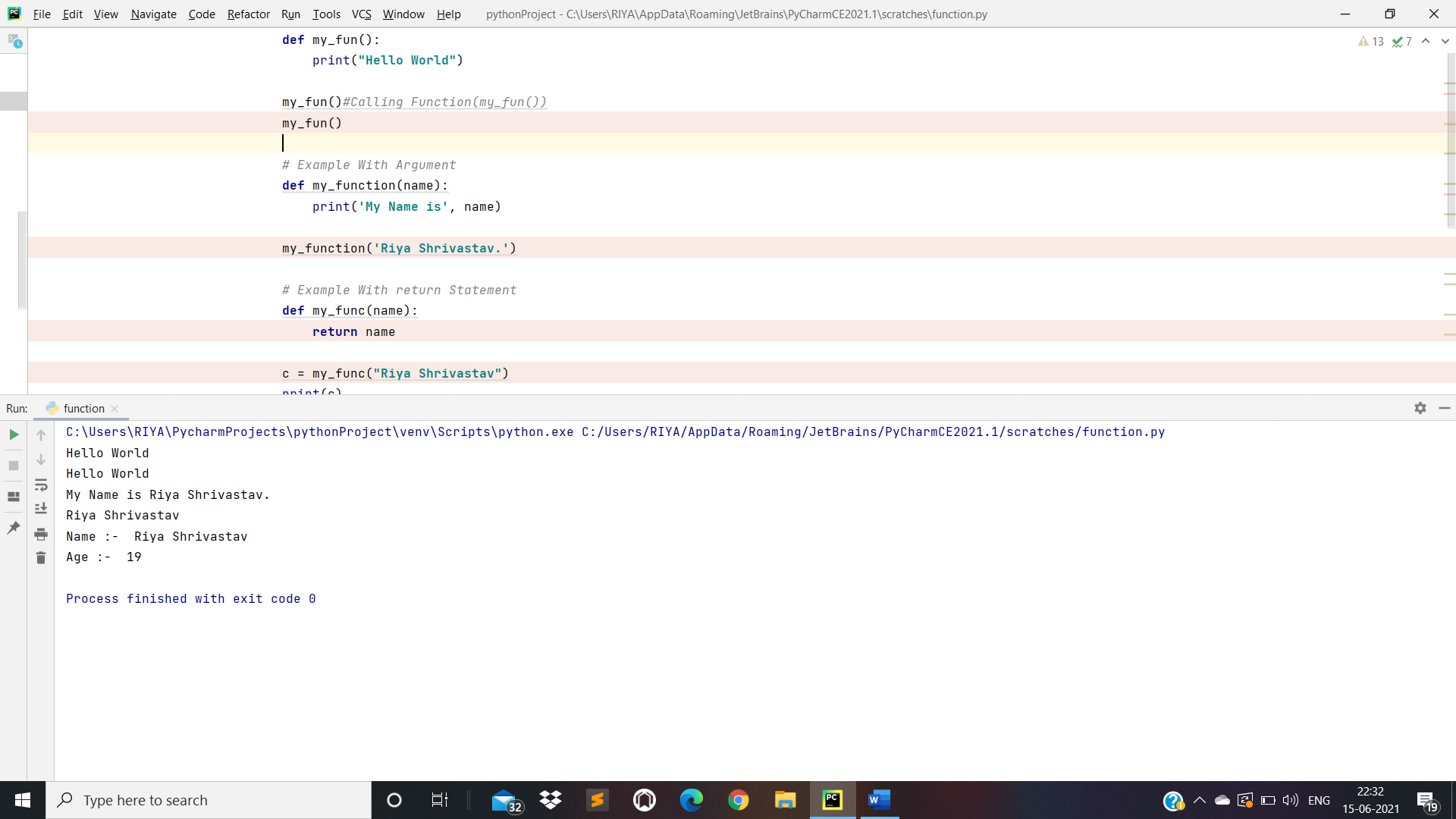
* Pass Statement in Python:

*# pass is just a placeholder for  
#functionality to be added later.*i = {10, 20, 30, 40}  
**for** val **in** i:  
 **pass**



Day 4: Functions in Python

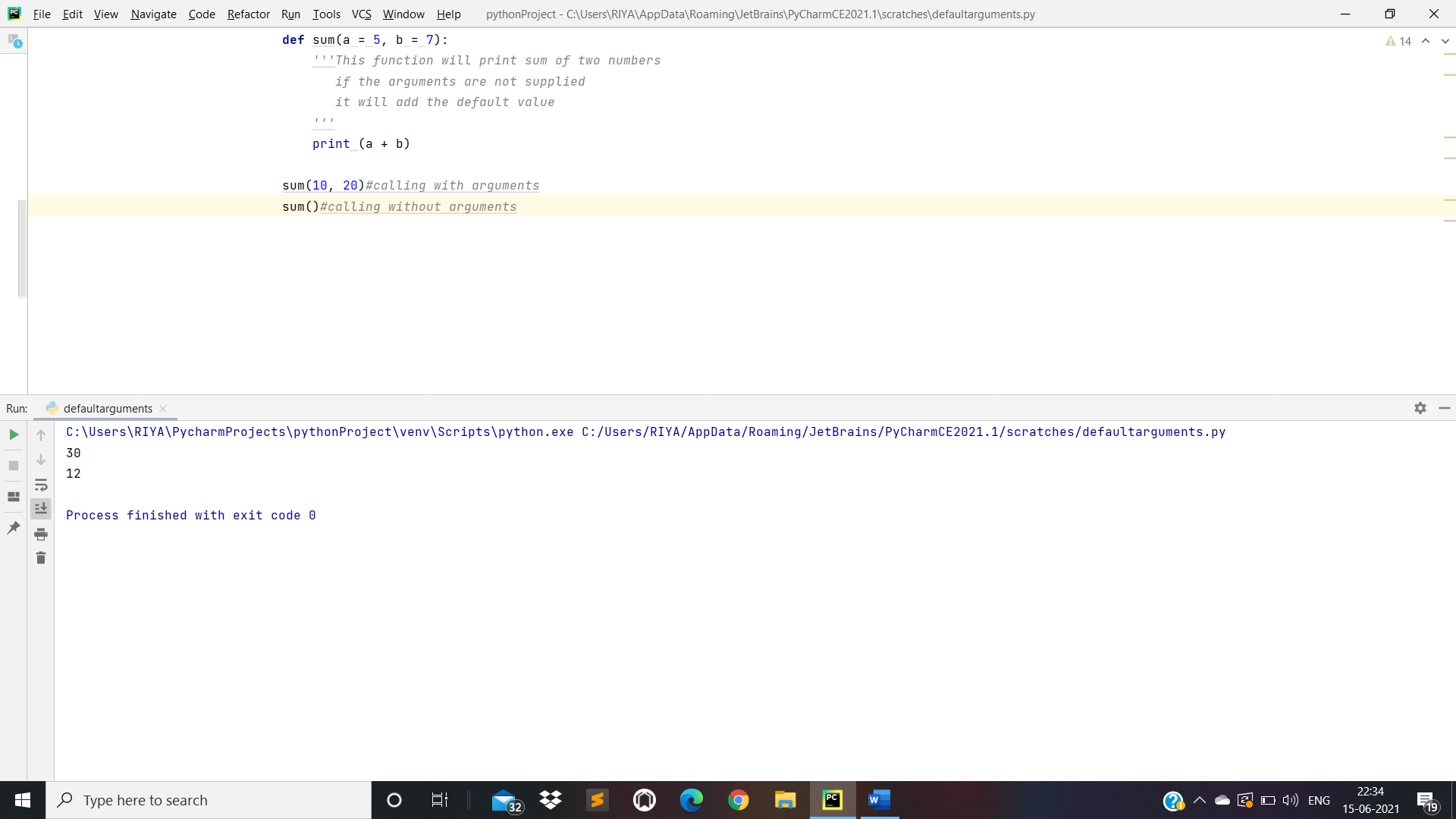
**def** my\_fun():  
 print(**"Hello World"**)  
  
my\_fun() *#Calling Function(my\_fun())*my\_fun()  
  
*# Example With Argument***def** my\_function(name):  
 print(**'My Name is'**, name)  
  
my\_function(**'Riya Shrivastav.'**)  
  
*# Example With return Statement***def** my\_func(name):  
 **return** name  
  
c = my\_func(**"Riya Shrivastav"**)  
print(c)  
  
  
*# Example With Multiple return Statement***def** my\_funct():  
 name = **"Riya Shrivastav"** age = 19  
 **return** name, age  
  
name, age = my\_funct()  
print(**"Name :- "**, name)  
print(**"Age :- "**, age)



* Python Function Arguments:

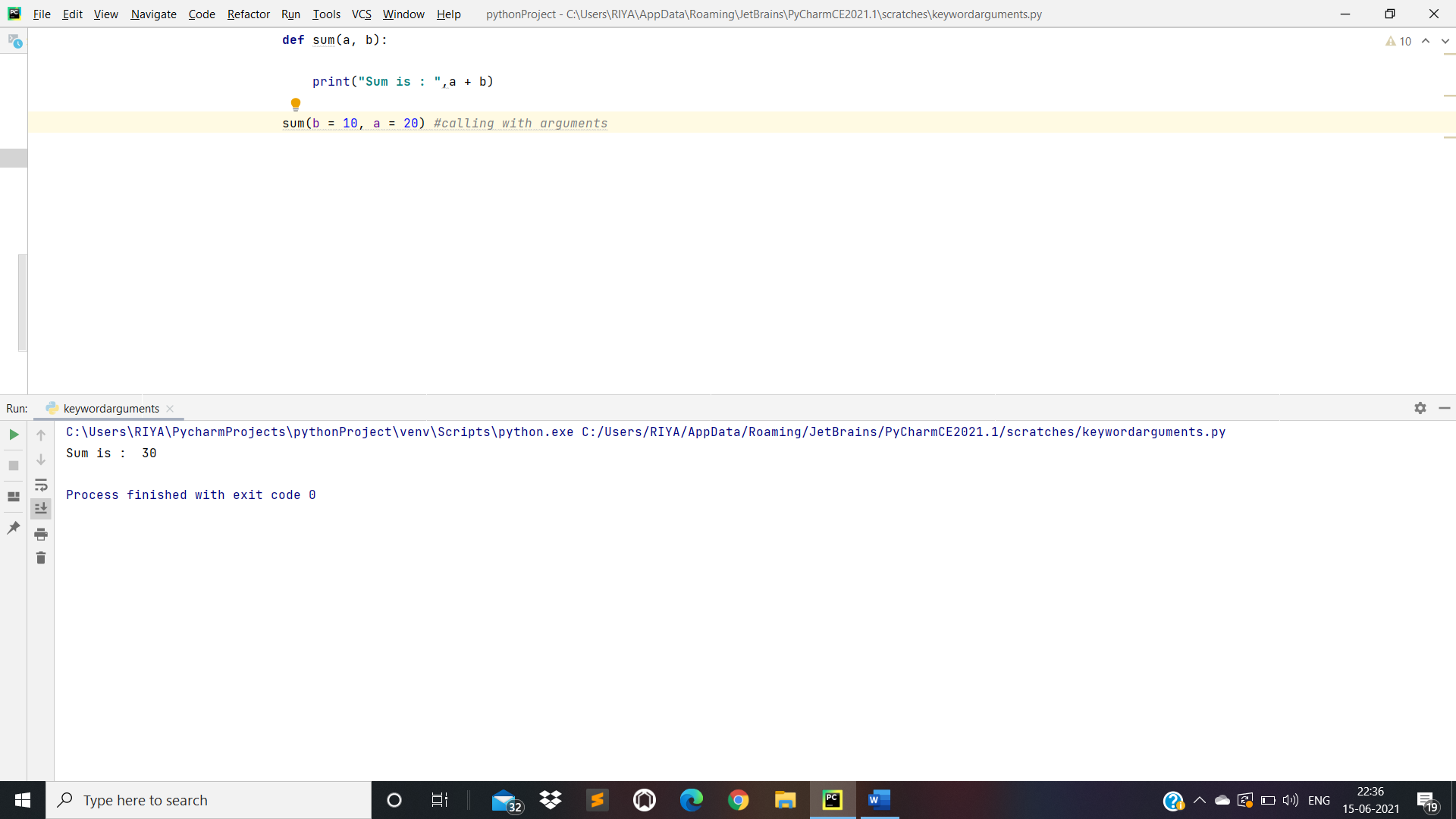
1. Python Default Arguments:

**def** sum(a = 5, b = 7):  
 *'''This function will print sum of two numbers  
 if the arguments are not supplied  
 it will add the default value  
 '''* print (a + b)  
  
sum(10, 20)*#calling with arguments*sum()*#calling without arguments*



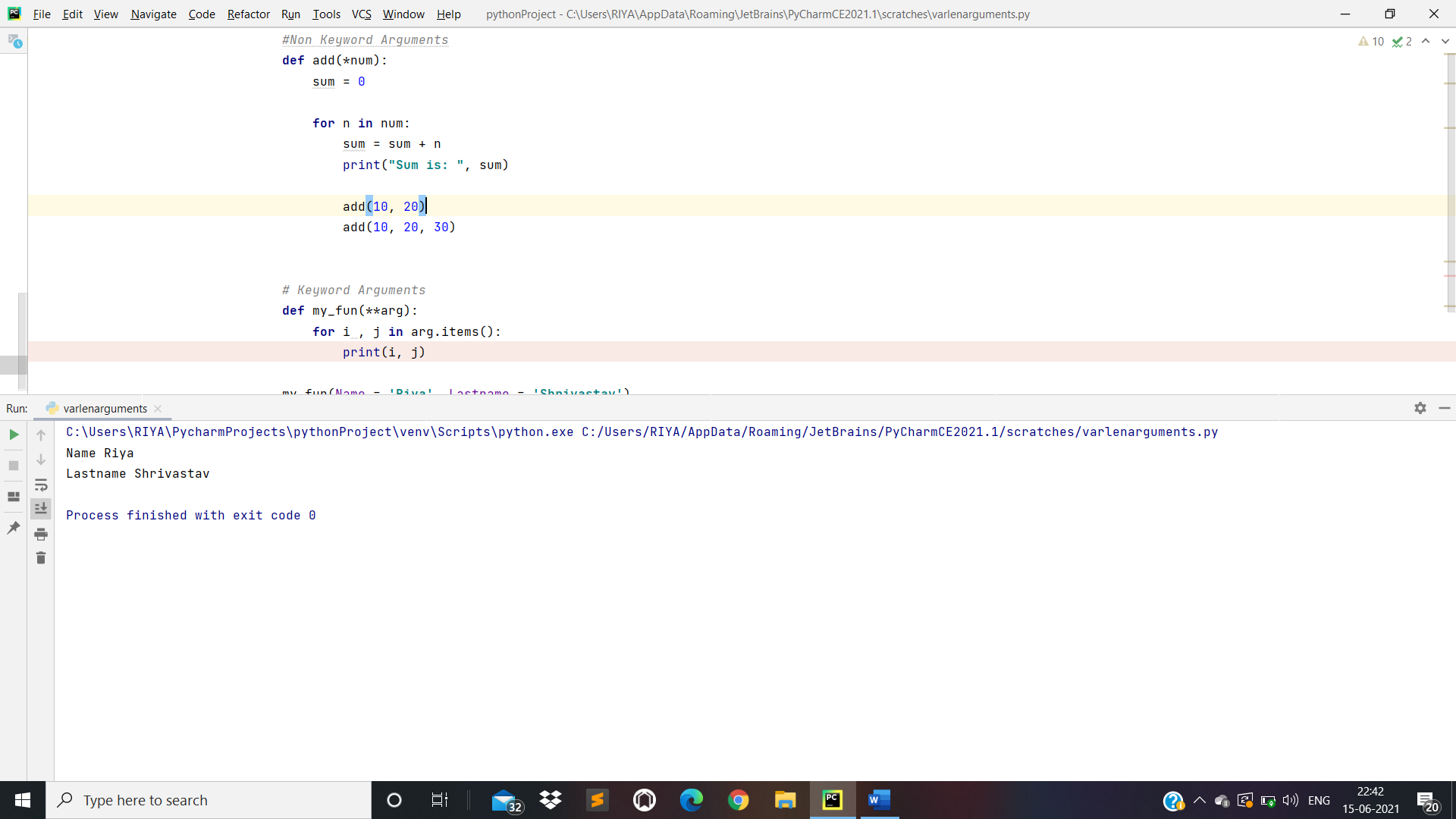
1. Python Keyword Arguments:

**def** sum(a, b):  
  
 print(**"Sum is : "**,a + b)  
  
sum(b = 10, a = 20) *#calling with arguments*



1. Variable-length Arguments:

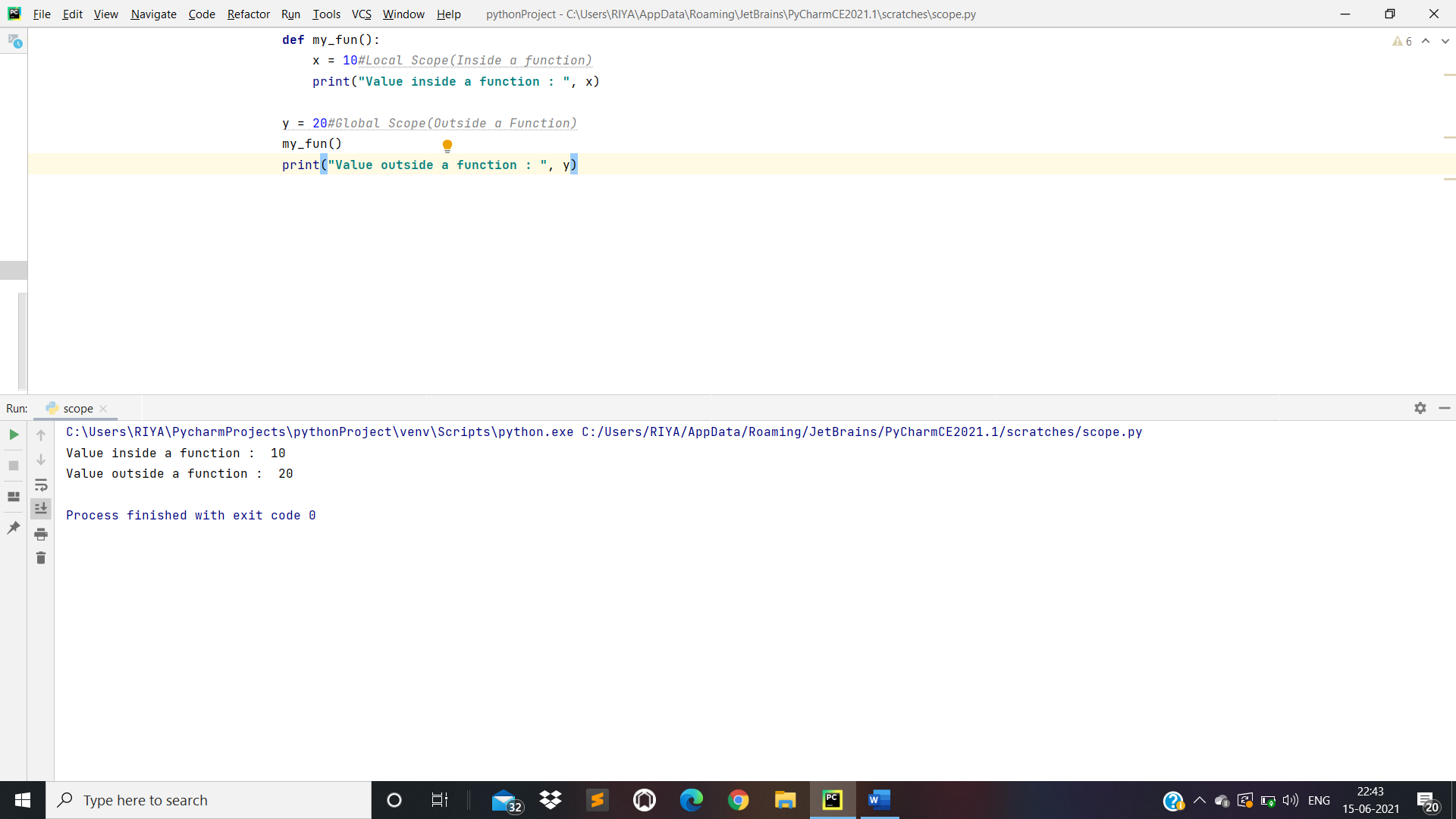
*# Non - Keyword Arguments***def** add(\*num):  
 sum = 0  
  
 **for** x **in** num:  
 sum = sum + x  
 print(**"Sum : "**, sum)  
  
 add(10, 20)  
 add(10, 20, 30)  
  
*# Keyword Arguments***def** my\_fun(\*\*arg):  
 **for** i , j **in** arg.items():  
 print(i, j)  
  
my\_fun(Name = **'Riya'**, Lastname = **'Shrivastav'**)



* Scope of Variables:

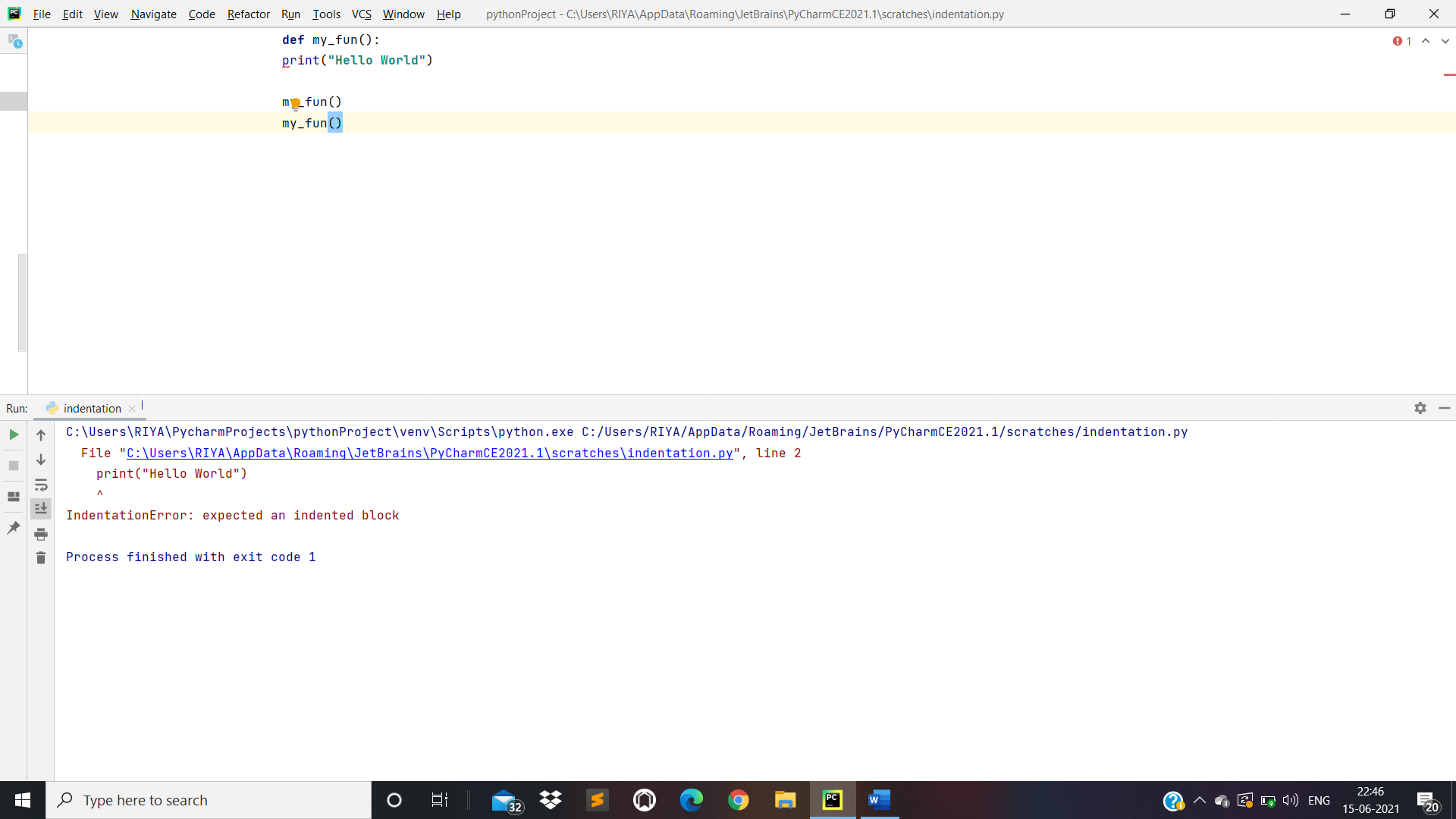
1. Global Variables:
2. Local Variables:

**def** my\_fun():  
 x = 10*#Local Scope(Inside a function)* print(**"Value inside a function : "**, x)  
  
y = 20*#Global Scope(Outside a Function)*my\_fun()  
print(**"Value outside a function : "**, y)



* Significance of Indentation(Space) in Python:

**def** my\_fun():  
print(**"Hello World"**) my\_fun()  
my\_fun()



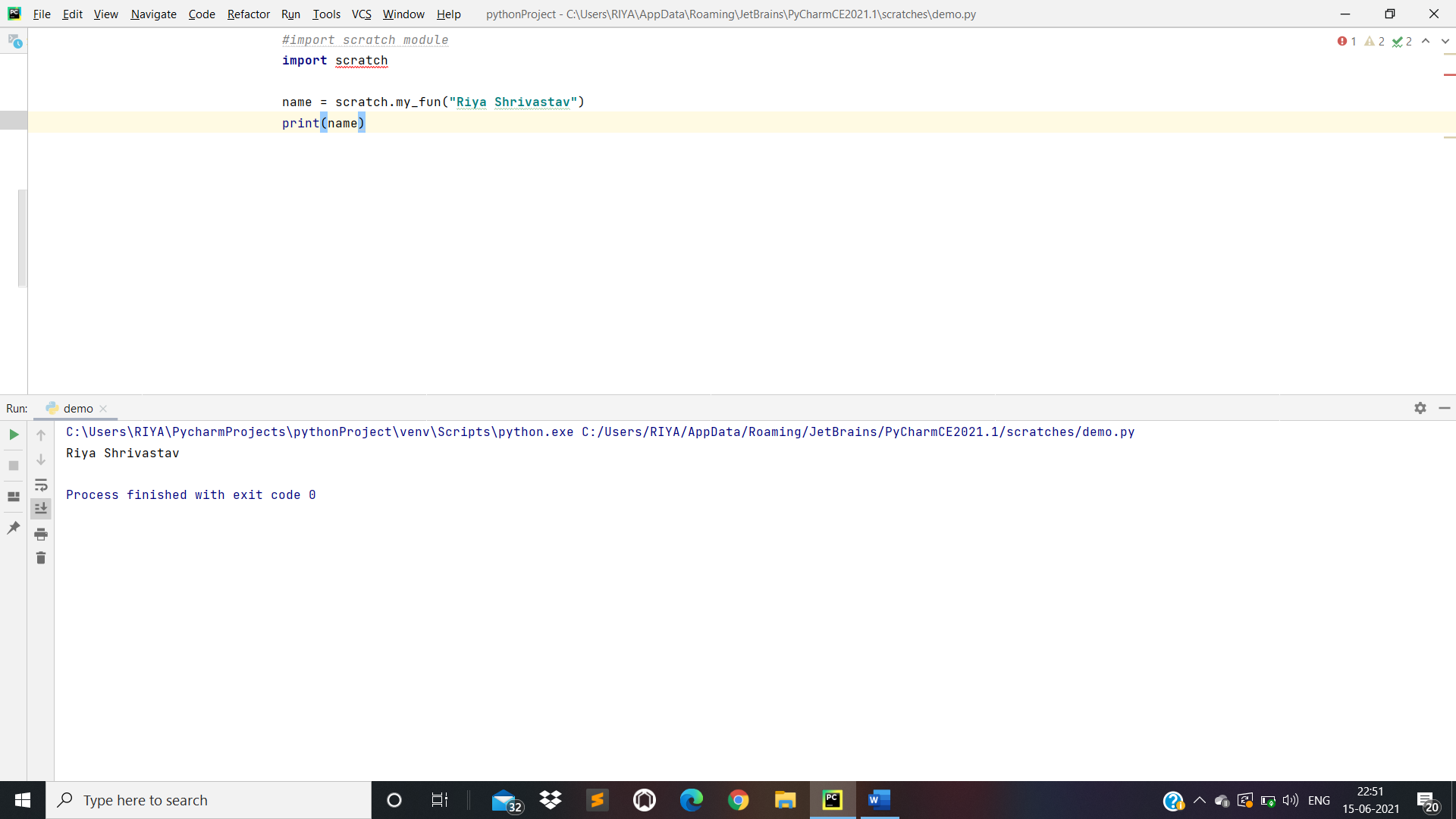
* Modules in Python:

Scratch.py(1st File):

**def** my\_fun(name):  
 **return** name

demo.py(2nd File):

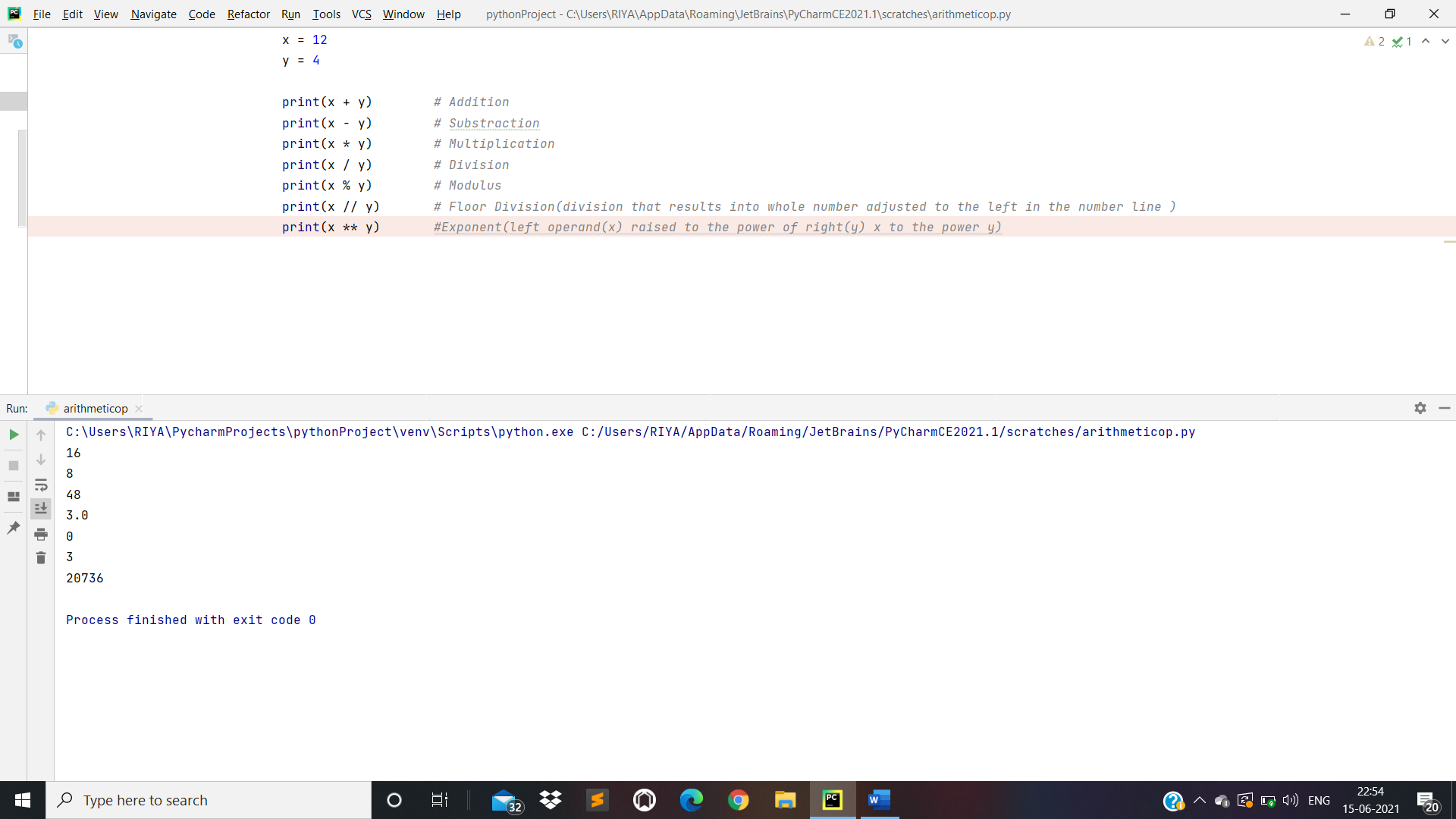
*#import scratch module***import** scratch  
  
name = scratch.my\_fun(**"Riya Shrivastav"**)  
print(name)



* Operators in Python

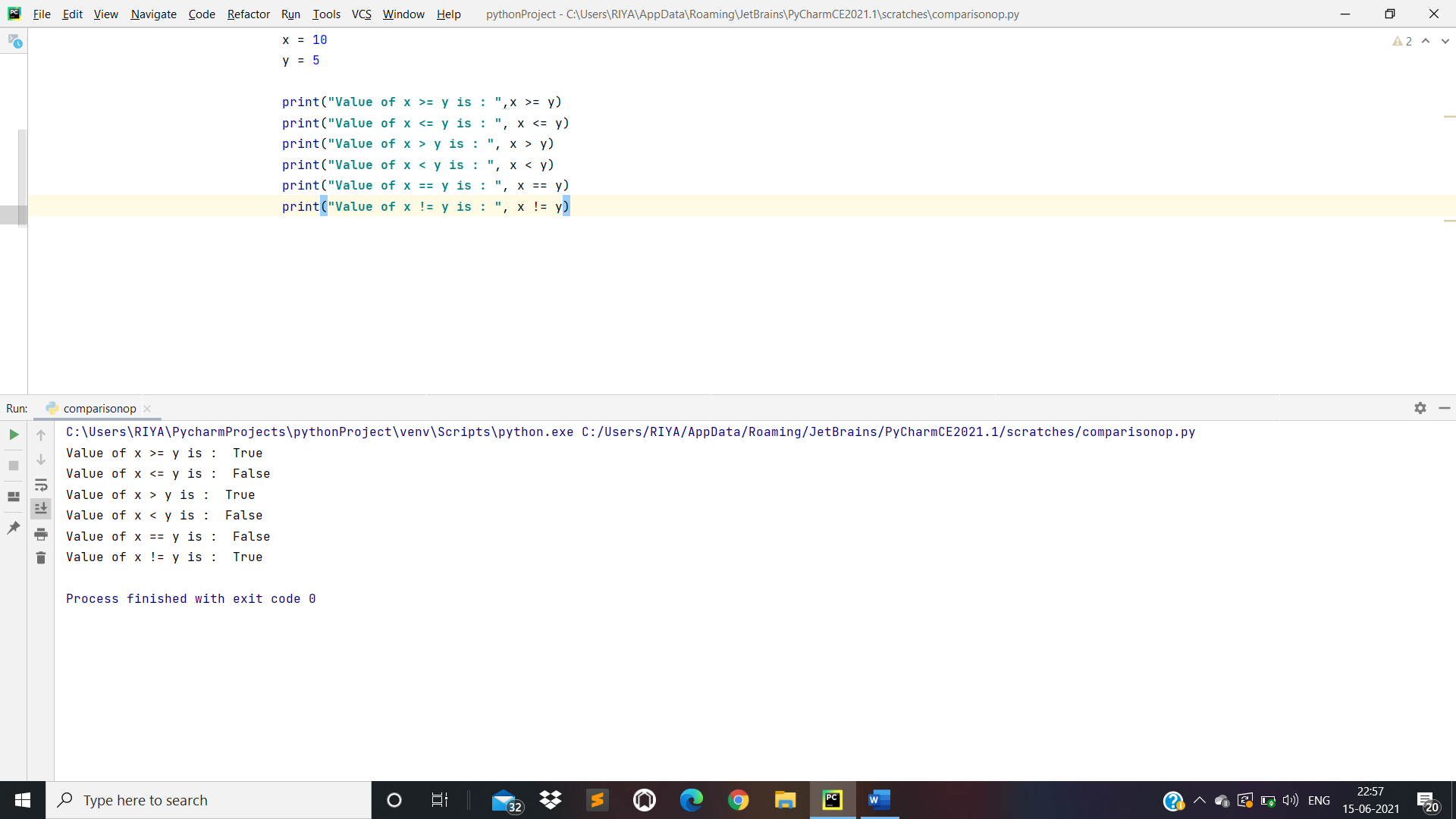
1. Arithmetic Operators:

x = 12  
y = 4  
  
print(x + y) *# Addition*print(x - y) *# Substraction*print(x \* y) *# Multiplication*print(x / y) *# Division*print(x % y) *# Modulus*print(x // y) *# Floor Division(division that results into whole number adjusted to the left in the number line )*print(x \*\* y) *#Exponent(left operand(x) raised to the power of right(y) x to the power y)*



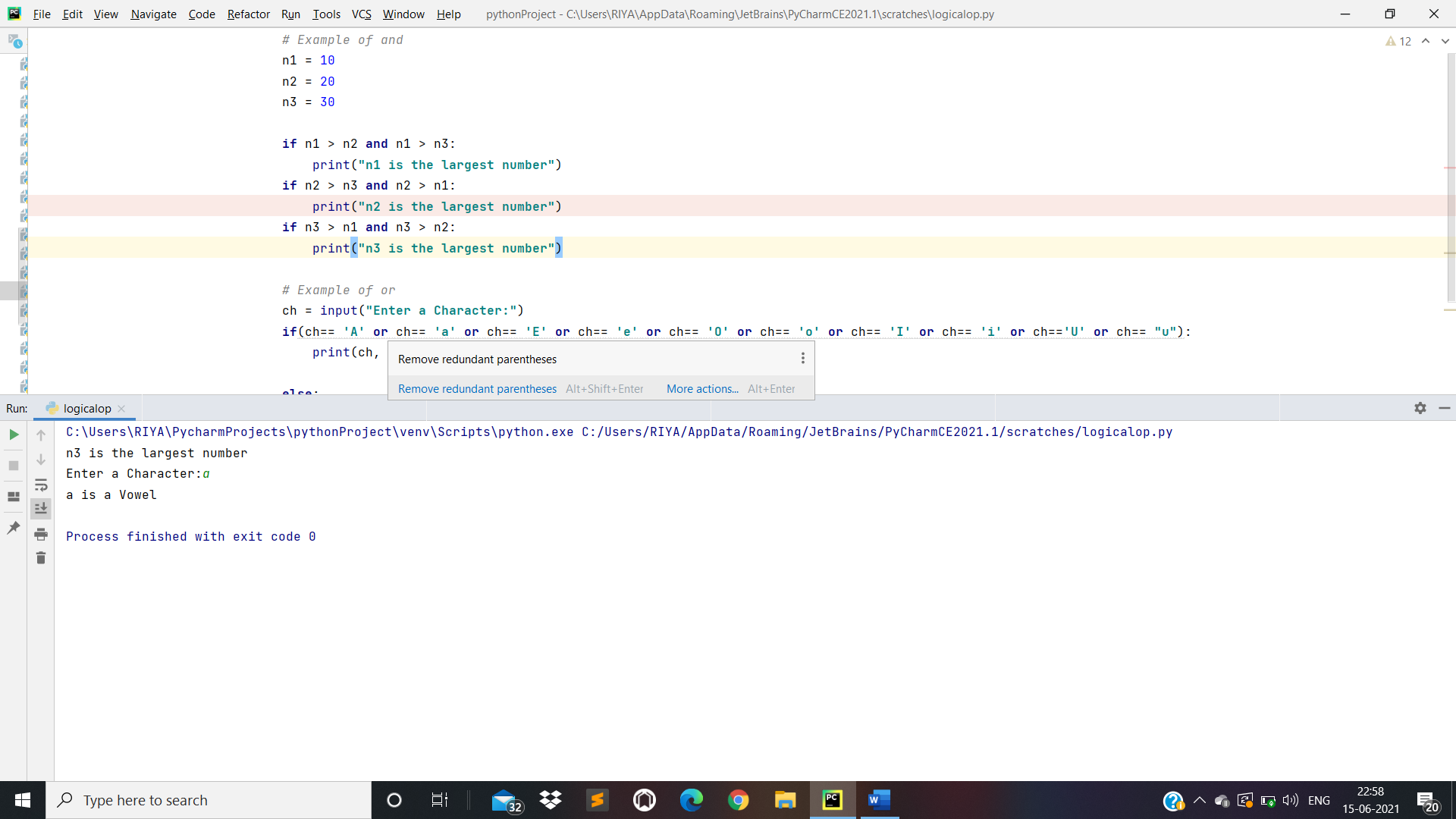
1. Comparison Operators:

x = 10  
y = 5  
  
print(**"Value of x >= y is : "**,x >= y)  
print(**"Value of x <= y is : "**, x <= y)  
print(**"Value of x > y is : "**, x > y)  
print(**"Value of x < y is : "**, x < y)  
print(**"Value of x == y is : "**, x == y)  
print(**"Value of x != y is : "**, x != y)



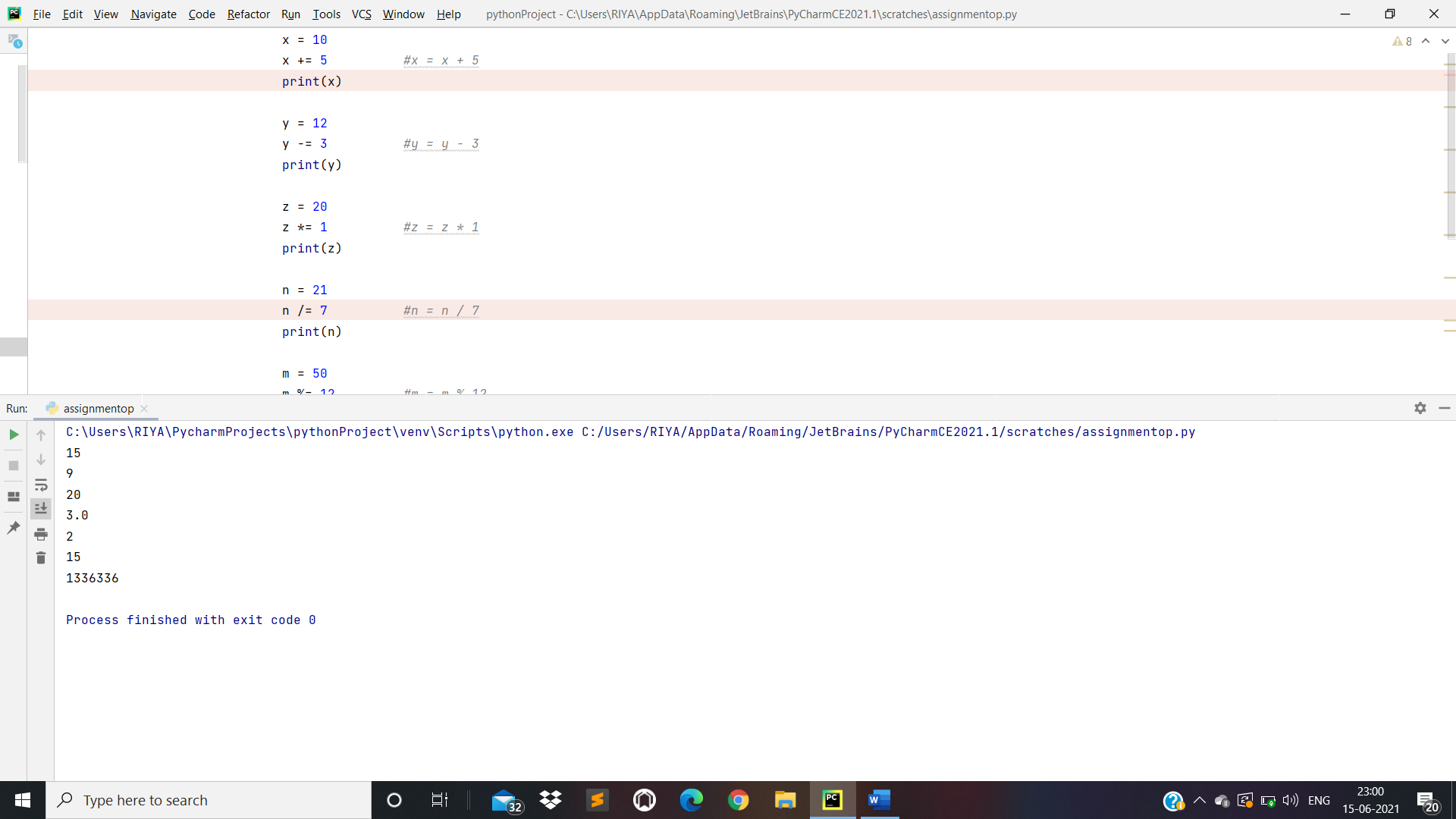
1. Logical Operators:

*# Example of and*n1 = 10  
n2 = 20  
n3 = 30  
  
**if** n1 > n2 **and** n1 > n3:  
 print(**"n1 is the largest number"**)  
**if** n2 > n3 **and** n2 > n1:  
 print(**"n2 is the largest number"**)  
**if** n3 > n1 **and** n3 > n2:  
 print(**"n3 is the largest number"**)  
  
*# Example of or*ch = input(**"Enter a Character:"**)  
**if**(ch== **'A' or** ch== **'a' or** ch== **'E' or** ch== **'e' or** ch== **'O' or** ch== **'o' or** ch== **'I' or** ch== **'i' or** ch==**'U' or** ch== **"u"**):  
 print(ch, **"is a Vowel"**)  
  
**else**:  
 print(ch, **"is a Consonant"**)



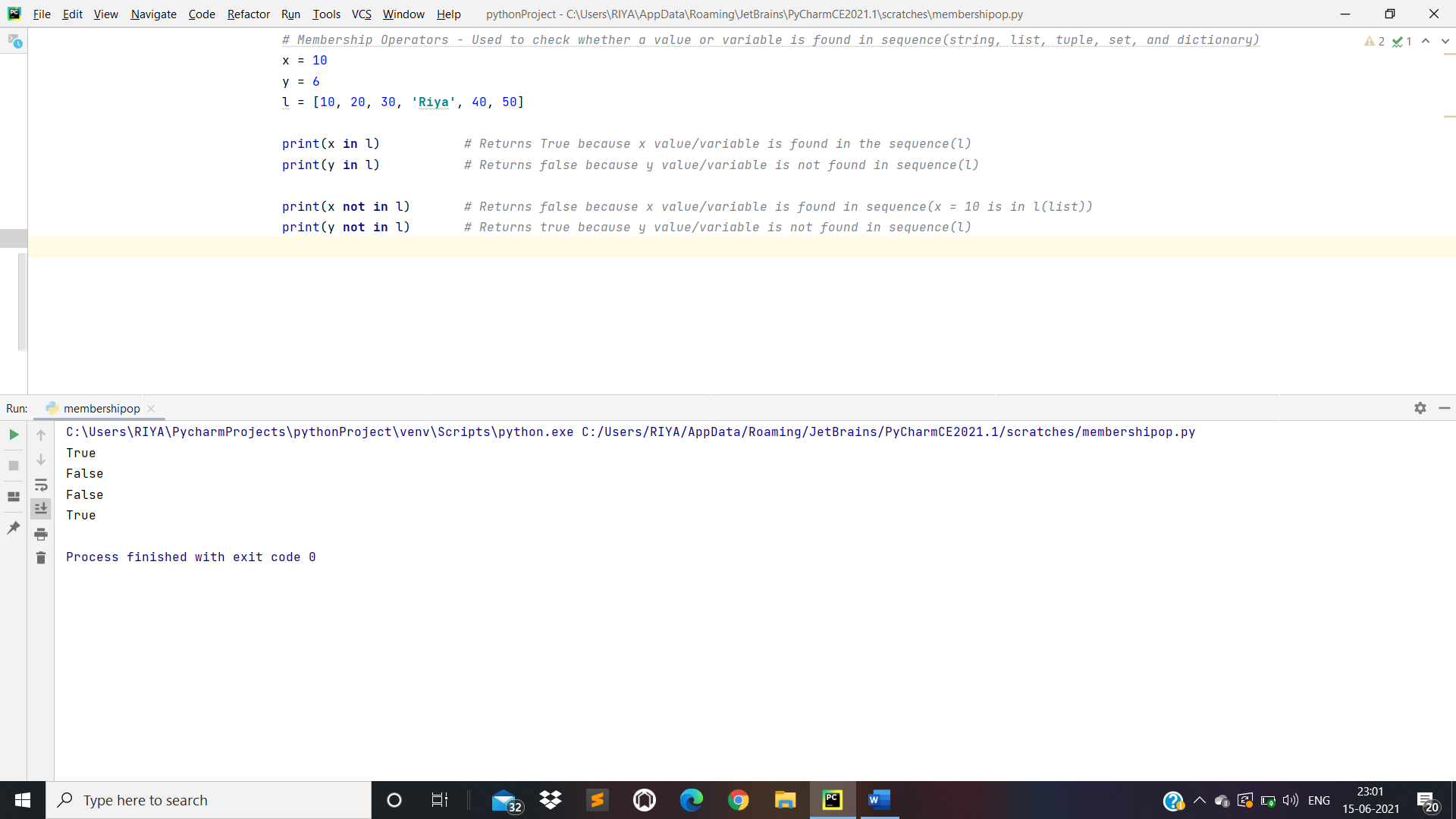
1. Assignment Operators:

x = 10  
x += 5 *#x = x + 5*print(x)  
  
y = 12  
y -= 3 *#y = y - 3*print(y)  
  
z = 20  
z \*= 1 *#z = z \* 1*print(z)  
  
n = 21  
n /= 7 *#n = n / 7*print(n)  
  
m = 50  
m %= 12 *#m = m % 12*print(m)  
  
o = 45  
o //= 3 *#o = o // 3*print(o)  
  
p = 34  
p \*\*= 4 *#p = p \*\* 4*print(p)



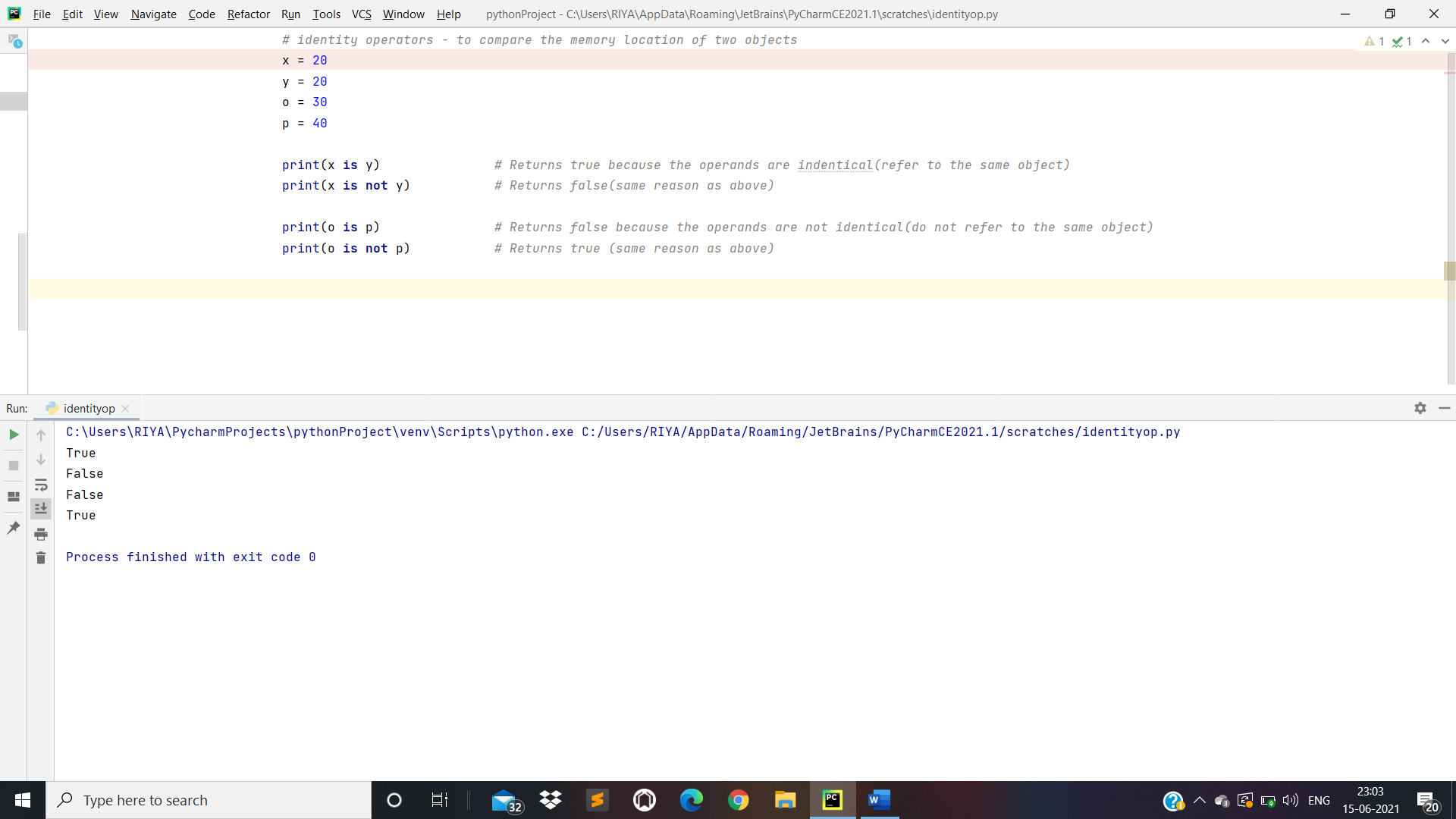
1. Membership Operators:

*# Membership Operators - Used to check whether a value or variable is found in sequence(string, list, tuple, set, and dictionary)*x = 10  
y = 6  
l = [10, 20, 30, **'Riya'**, 40, 50]  
  
print(x **in** l) *# Returns True because x value/variable is found in the sequence(l)*print(y **in** l) *# Returns false because y value/variable is not found in sequence(l)*print(x **not in** l) *# Returns false because x value/variable is found in sequence(x = 10 is in l(list))*print(y **not in** l) *# Returns true because y value/variable is not found in sequence(l)*



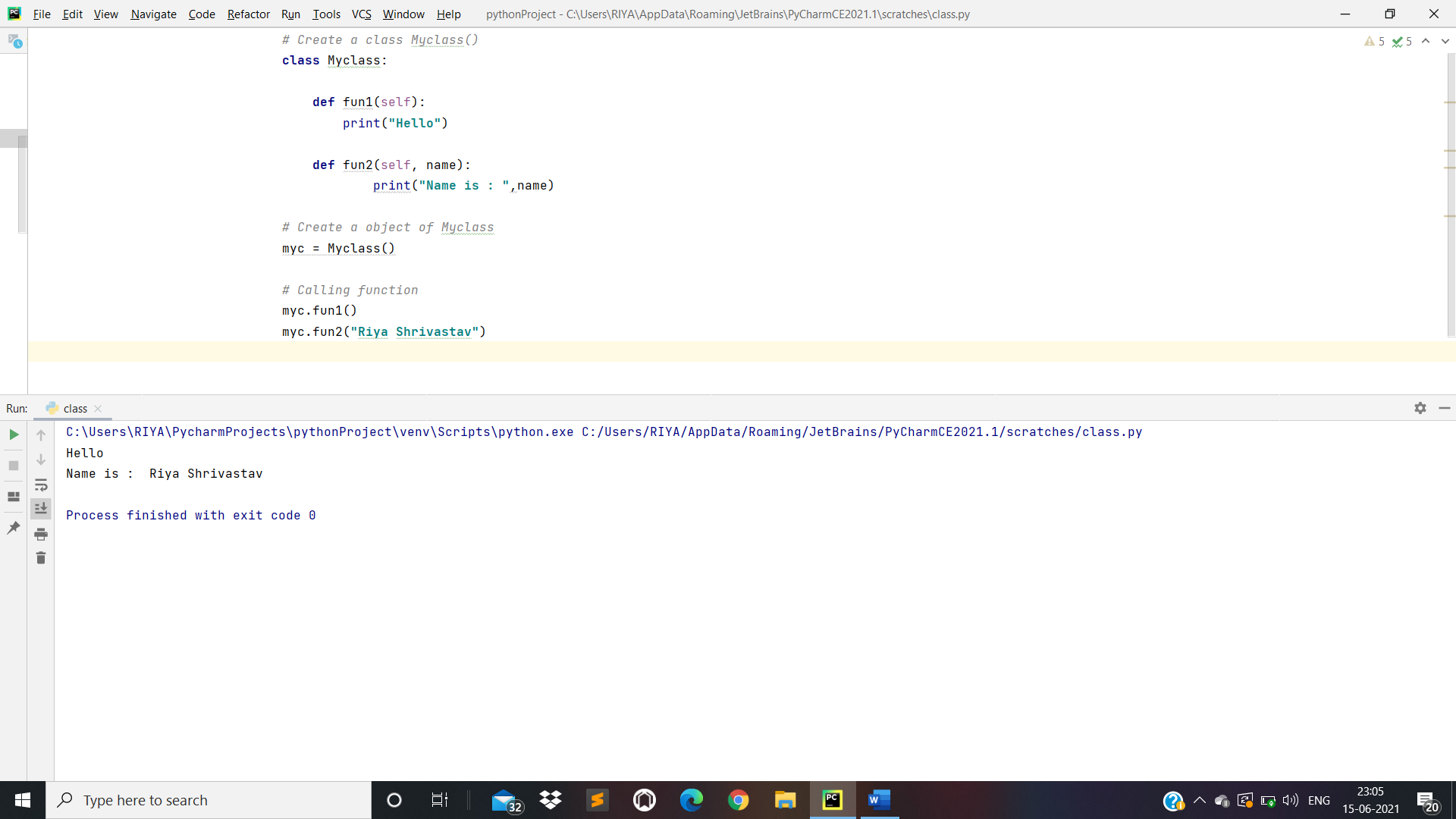
1. Identity Operators:

*# identity operators - to compare the memory location of two objects*x = 20  
y = 20  
o = 30  
p = 40  
  
print(x **is** y) *# Returns true because the operands are indentical(refer to the same object)*print(x **is not** y) *# Returns false(same reason as above)*print(o **is** p) *# Returns false because the operands are not identical(do not refer to the same object)*print(o **is not** p) *# Returns true (same reason as above)*

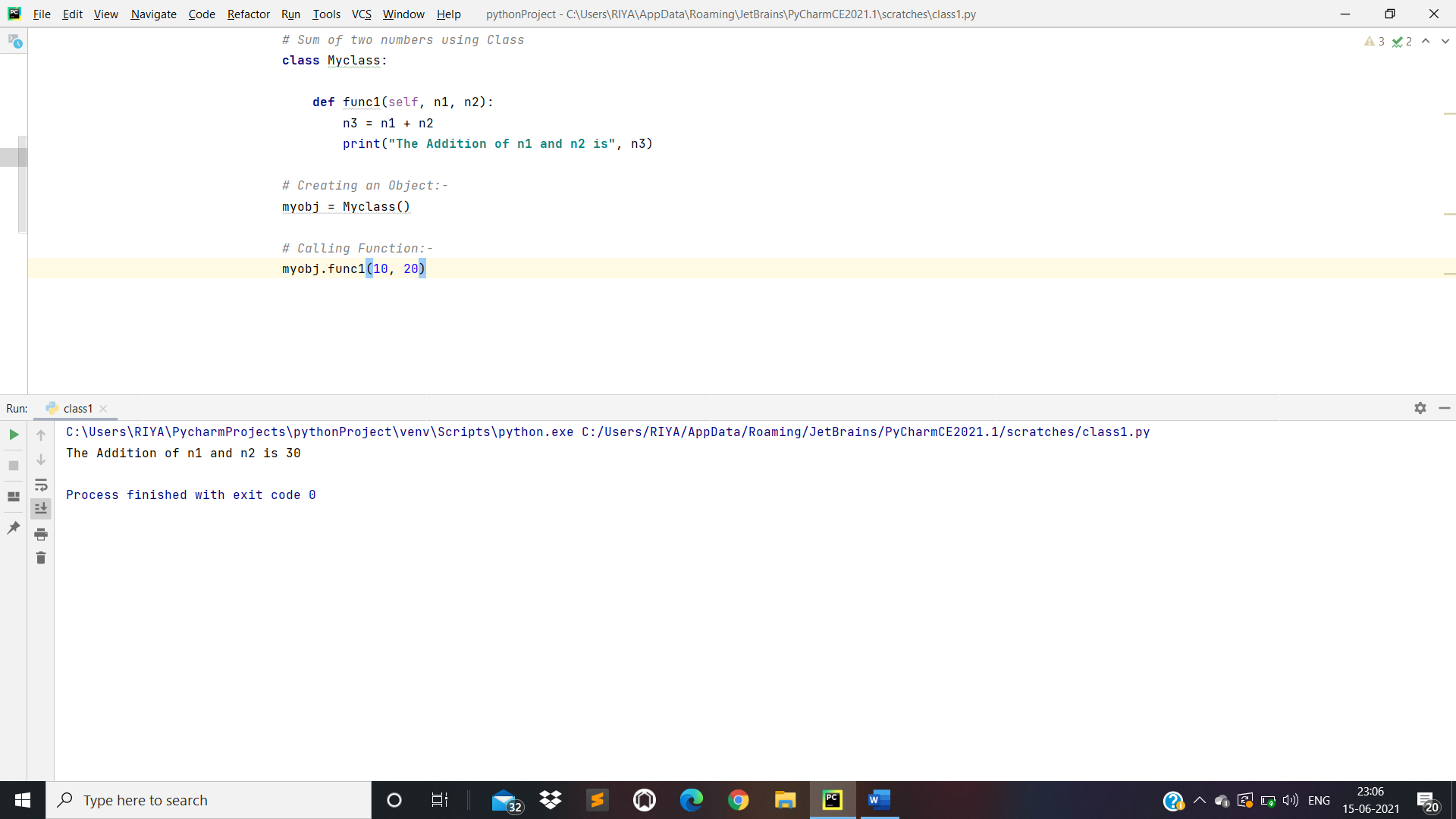


Day 5: OOP’s Concept:

*# Create a class Myclass()***class** Myclass:  
  
 **def** fun1(self):  
 print(**"Hello"**)  
  
 **def** fun2(self, name):  
 print(**"Name is : "**,name)  
  
*# Create a object of Myclass*myc = Myclass()  
  
*# Calling function*myc.fun1()  
myc.fun2(**"Riya Shrivastav"**)



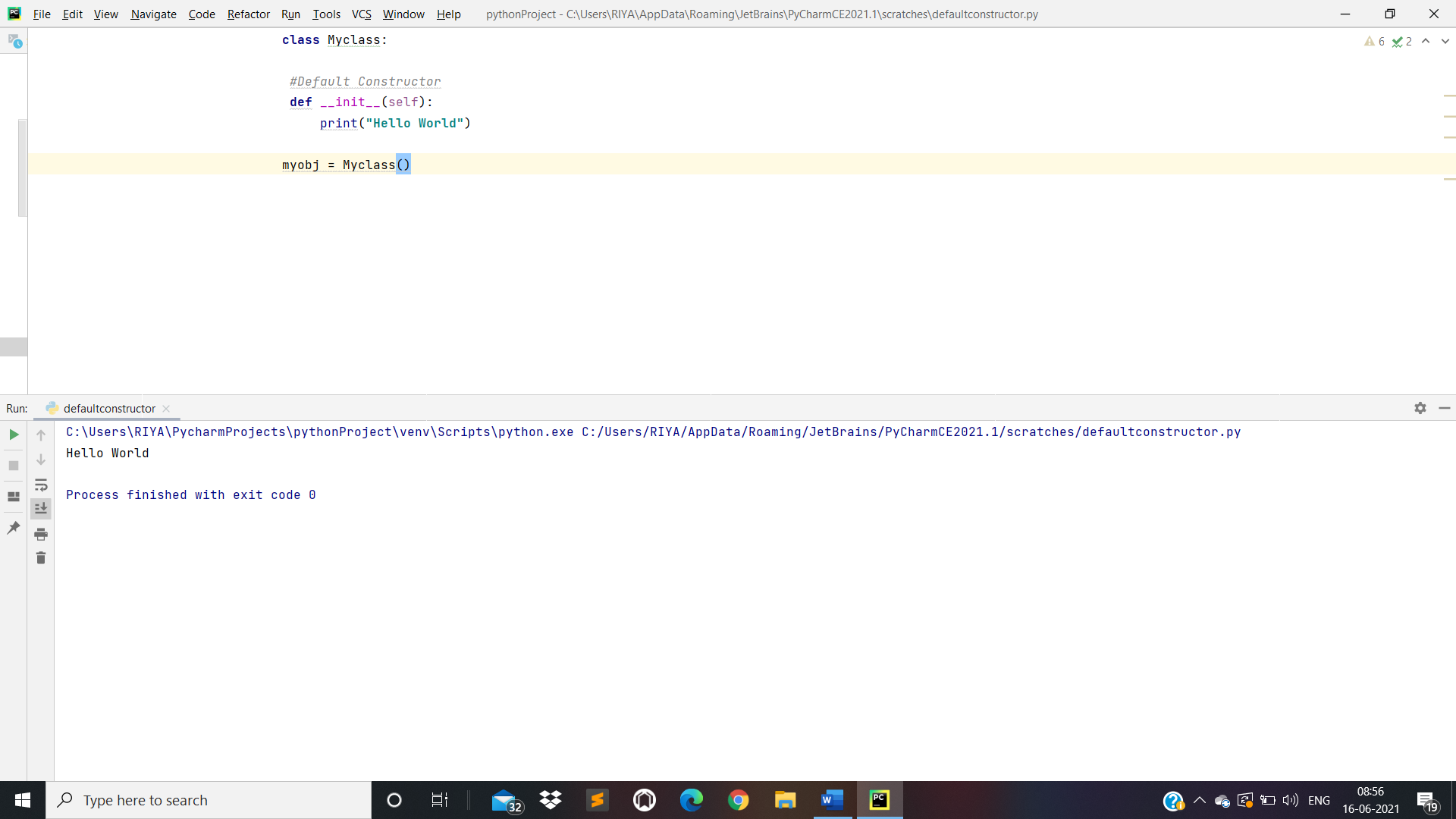
*# Sum of two numbers using Class***class** Myclass:  
  
 **def** func1(self, n1, n2):  
 n3 = n1 + n2  
 print(**"The Addition of n1 and n2 is"**, n3)  
  
*# Creating an Object:-*myobj = Myclass()  
  
*# Calling Function:-*myobj.func1(10, 20)



* Python Constructors

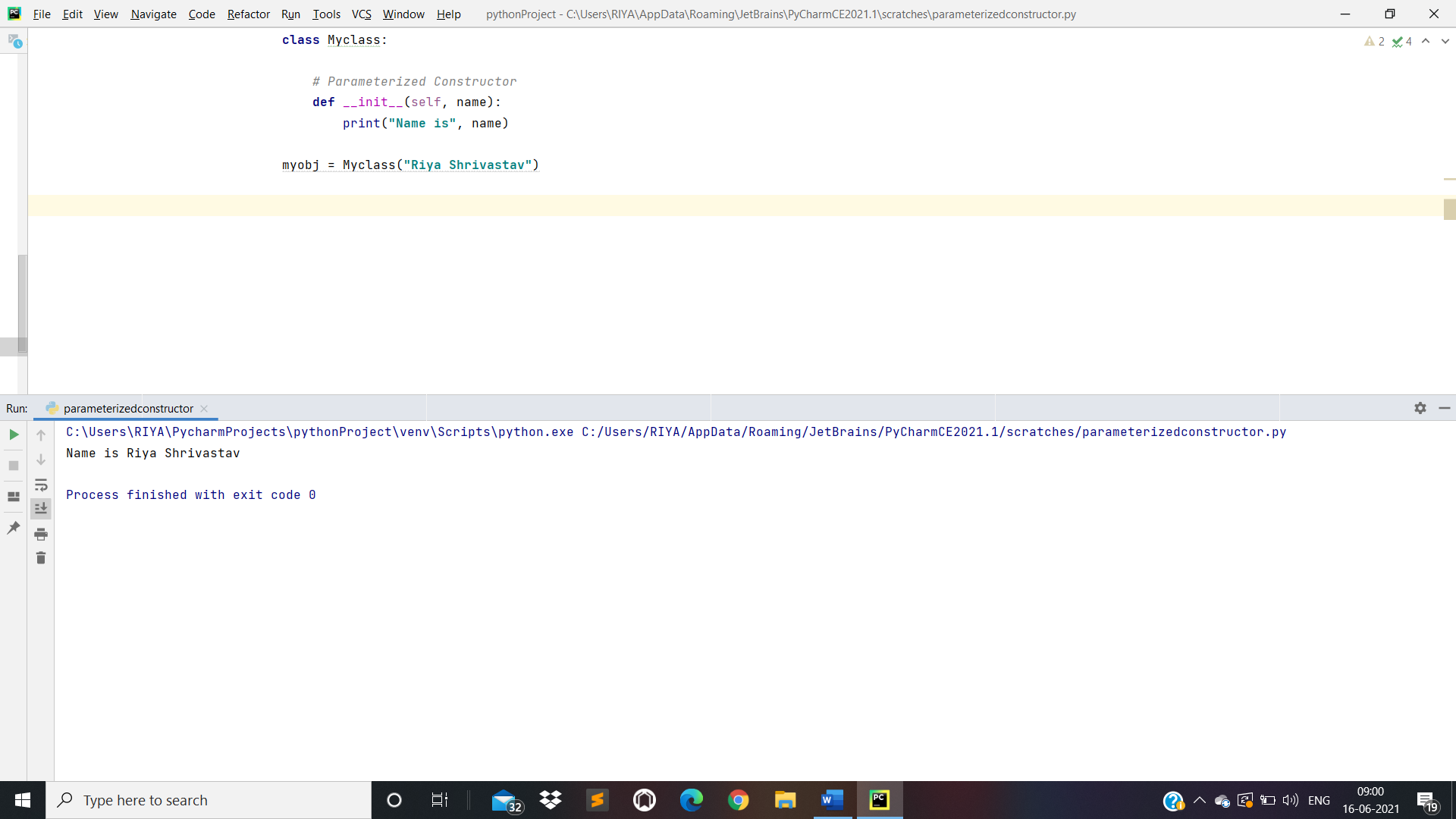
1. Default Constructor:

**class** Myclass:  
  
 *#Default Constructor* **def** \_\_init\_\_(self):  
 print(**"Hello World"**)  
  
myobj = Myclass()



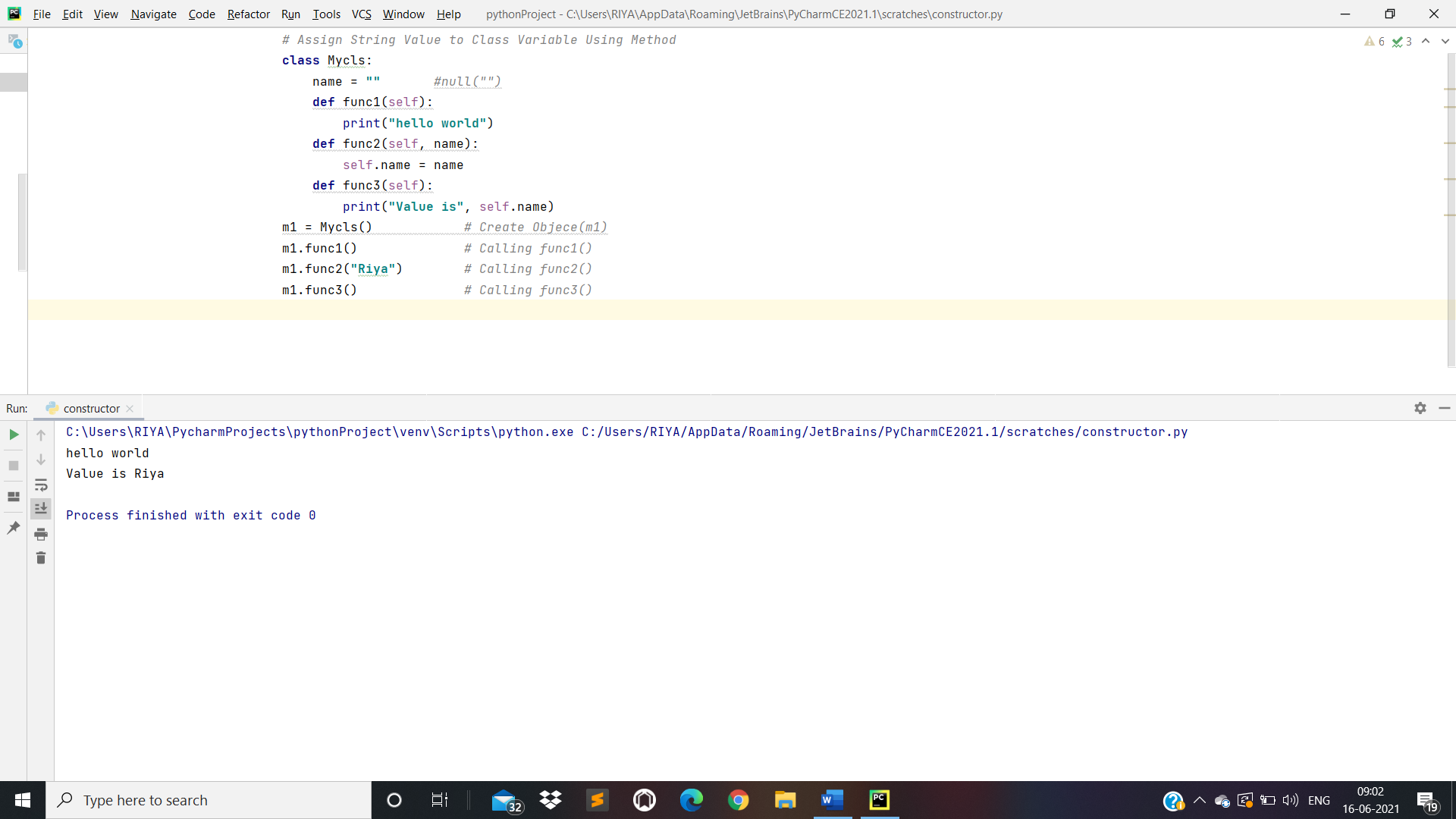
1. Parameterized Constructor:

**class** Myclass:  
  
 *# Parameterized Constructor* **def** \_\_init\_\_(self, name):  
 print(**"Name is"**, name)  
  
myobj = Myclass(**"Riya Shrivastav"**)



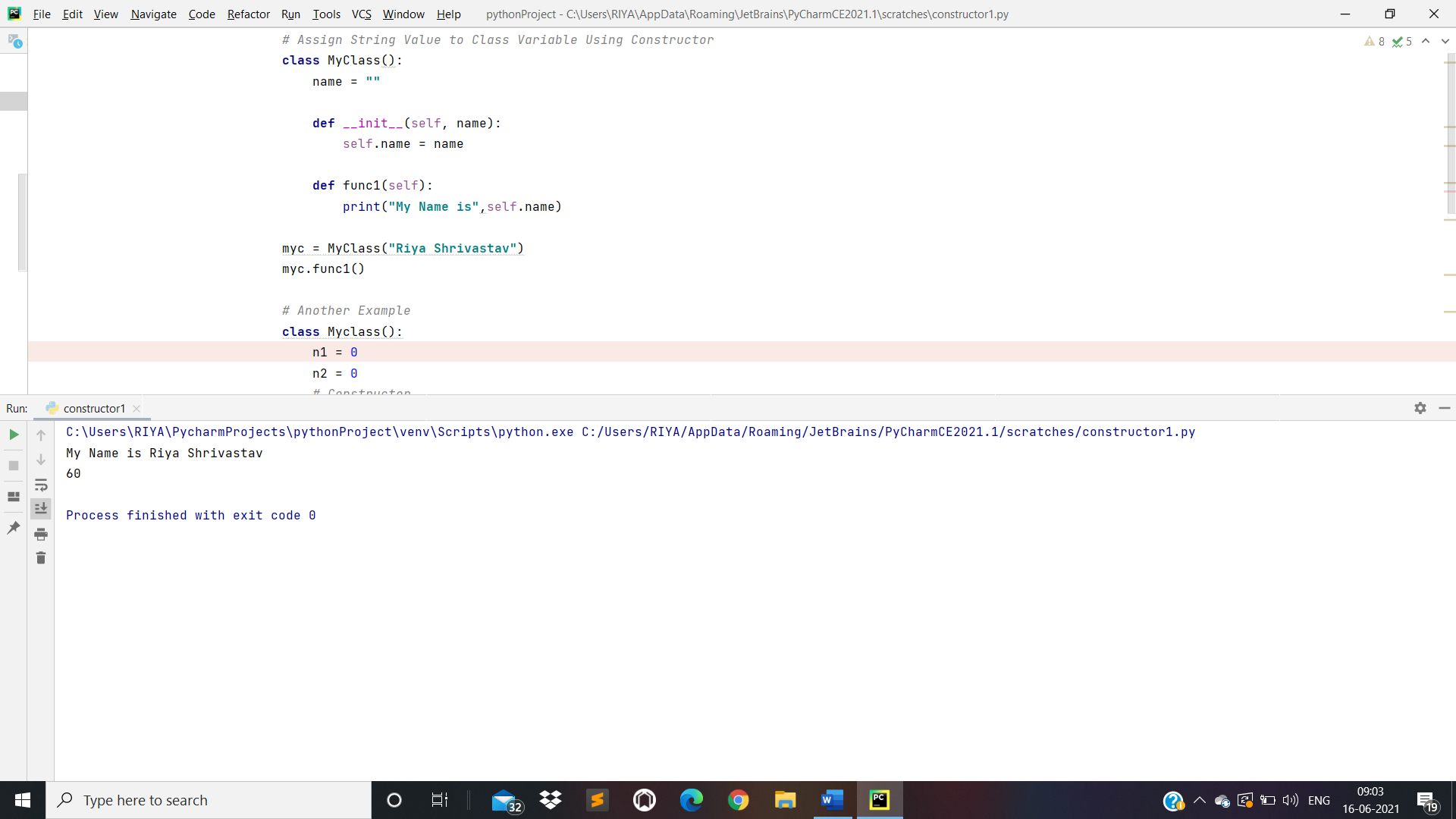
* Assign String Value to Class Variable Using Method:

*# Assign String Value to Class Variable Using Method***class** Mycls:  
 name = **""** *#null("")* **def** func1(self):  
 print(**"hello world"**)  
 **def** func2(self, name):  
 self.name = name  
 **def** func3(self):  
 print(**"Value is"**, self.name)  
m1 = Mycls() *# Create Objece(m1)*m1.func1() *# Calling func1()*m1.func2(**"Riya"**) *# Calling func2()*m1.func3() *# Calling func3()*



* Assign String Value to Class Variable Using Constructor

*# Assign String Value to Class Variable Using Constructor***class** MyClass():  
 name = **""  
  
 def** \_\_init\_\_(self, name):  
 self.name = name  
  
 **def** func1(self):  
 print(**"My Name is"**,self.name)  
  
myc = MyClass(**"Riya Shrivastav"**)  
myc.func1()  
  
*# Another Example***class** Myclass():  
 n1 = 0  
 n2 = 0  
 *# Constructor* **def** \_\_init\_\_(self, n1, n2):  
 self.n1 = n1  
 self.n2 = n2  
  
 *# Function* **def** func(self):  
 n3 = self.n1 + self.n2  
 print(n3)  
  
*# Creating Object of Myclass()*myobj = Myclass(20, 40)  
*# Calling Function*myobj.func()



* Inheritance:

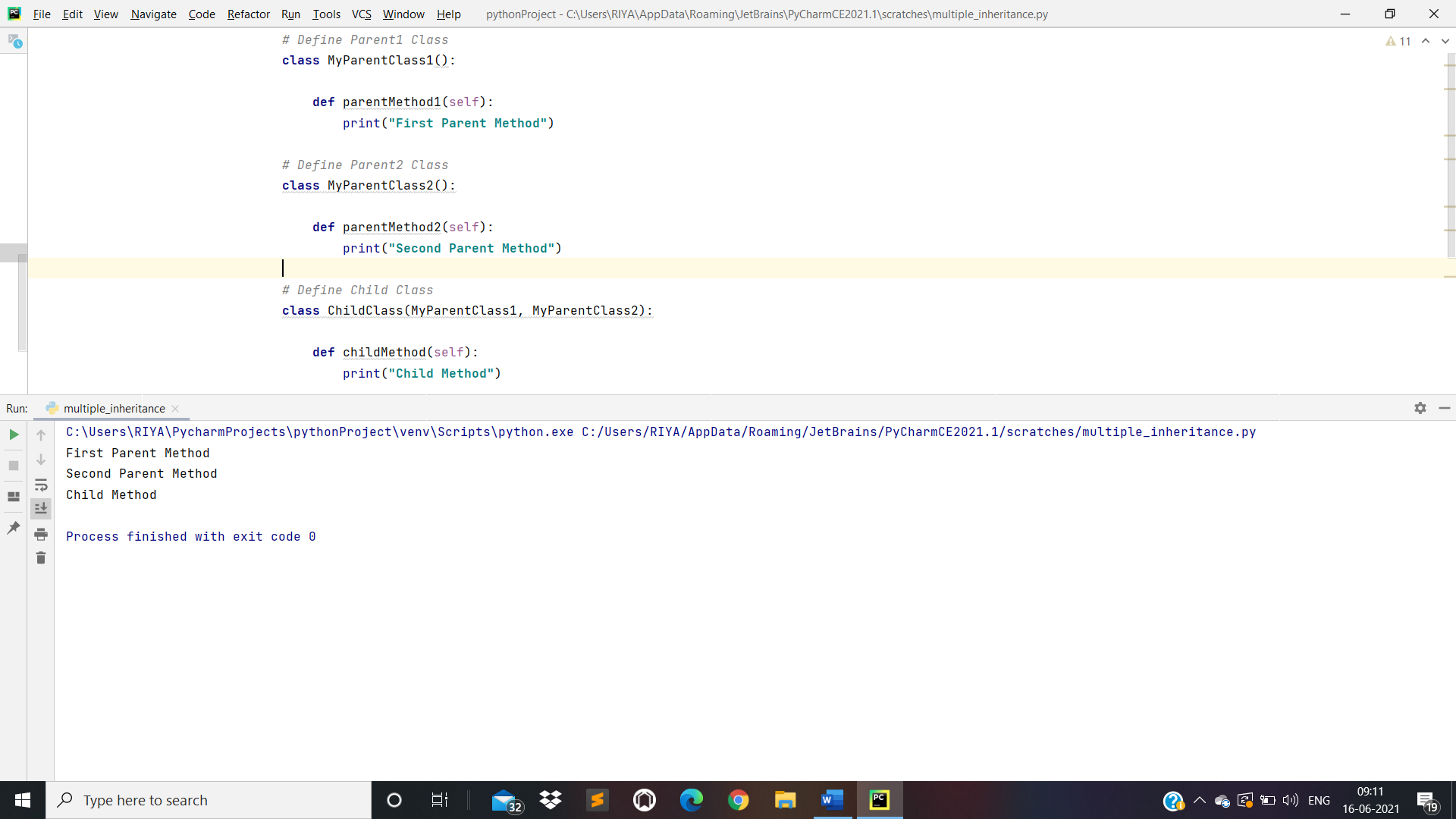
1. Single Level and Multi Level Inheritance:

*# Single and Multi-level Inheritance  
# Define Parent Class***class** Parent:  
  
 **def** \_\_init\_\_(self):  
 print(**"Parent Constructor"**)  
  
 **def** parentMethod(self):  
 print(**"Parent Method"**)  
  
*# Define Child Class***class** Child(Parent):  
  
 **def** \_\_init\_\_(self):  
 print(**"Child Constructor"**)  
  
 **def** childMethod(self):  
 print(**"Child Method"**)  
  
*# Define Subchild Class***class** Subchild(Child):  
  
 **def** \_\_init\_\_(self):  
 print(**"Subchild Constructor"**)  
  
 **def** subchildMethod(self):  
 print(**"Subchild Method"**)  
  
*# Creating Object:***'''myobj = Child() # Create Child Object  
myobj.childMethod() # Calling Child class Method(childMethod)  
myobj.parentMethod() # Calling Parent Method(parentMethod)'''**sc = Subchild()  
sc.subchildMethod()  
sc.childMethod()  
sc.parentMethod()



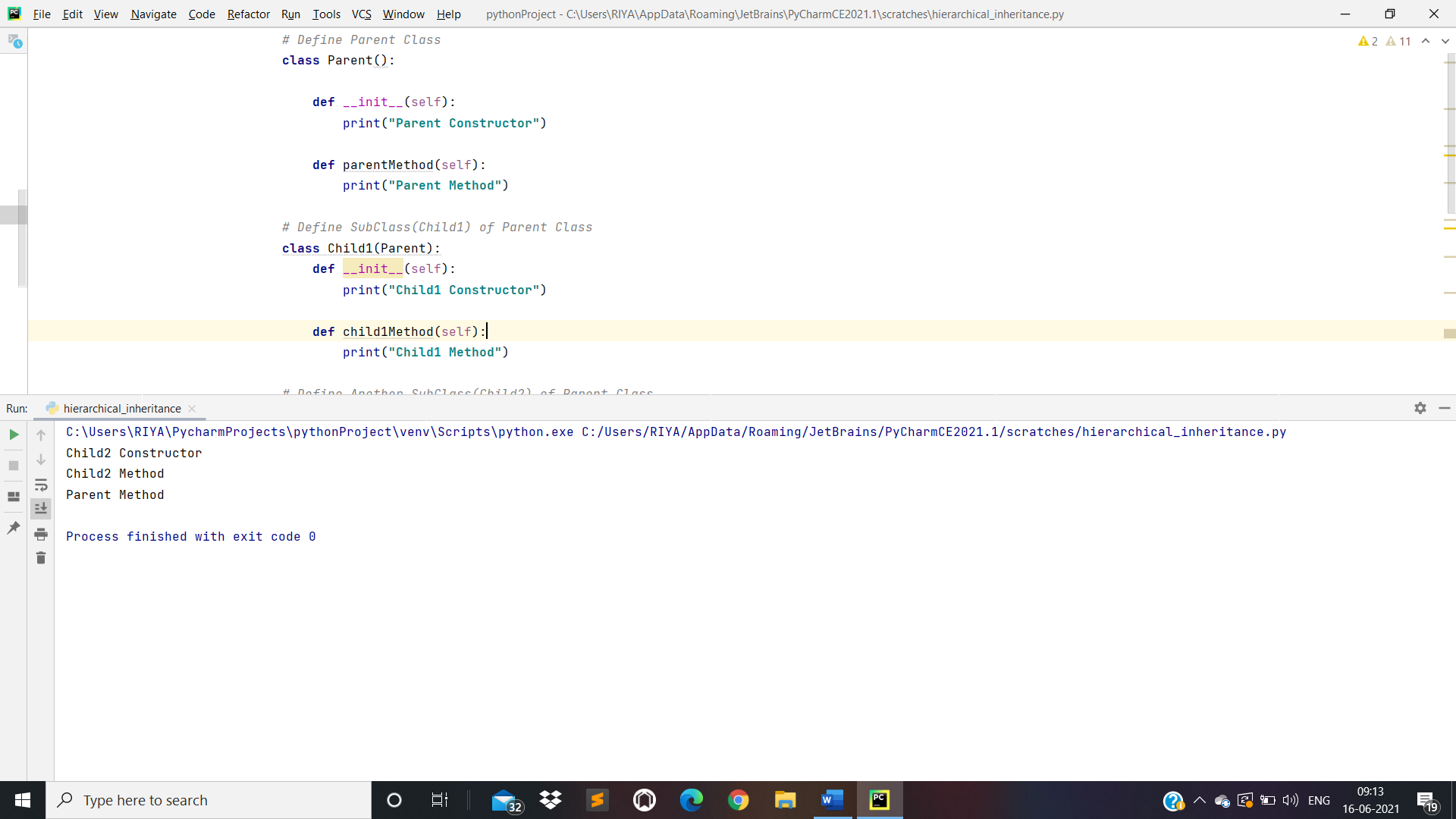
1. Multiple Inheritance:

*# Define Parent1 Class***class** MyParentClass1():  
  
 **def** parentMethod1(self):  
 print(**"First Parent Method"**)  
  
*# Define Parent2 Class***class** MyParentClass2():  
  
 **def** parentMethod2(self):  
 print(**"Second Parent Method"**)  
  
*# Define Child Class***class** ChildClass(MyParentClass1, MyParentClass2):  
  
 **def** childMethod(self):  
 print(**"Child Method"**)  
  
*# Creating object of class(ChildClass)*c = ChildClass()  
c.parentMethod1()  
c.parentMethod2()  
c.childMethod()



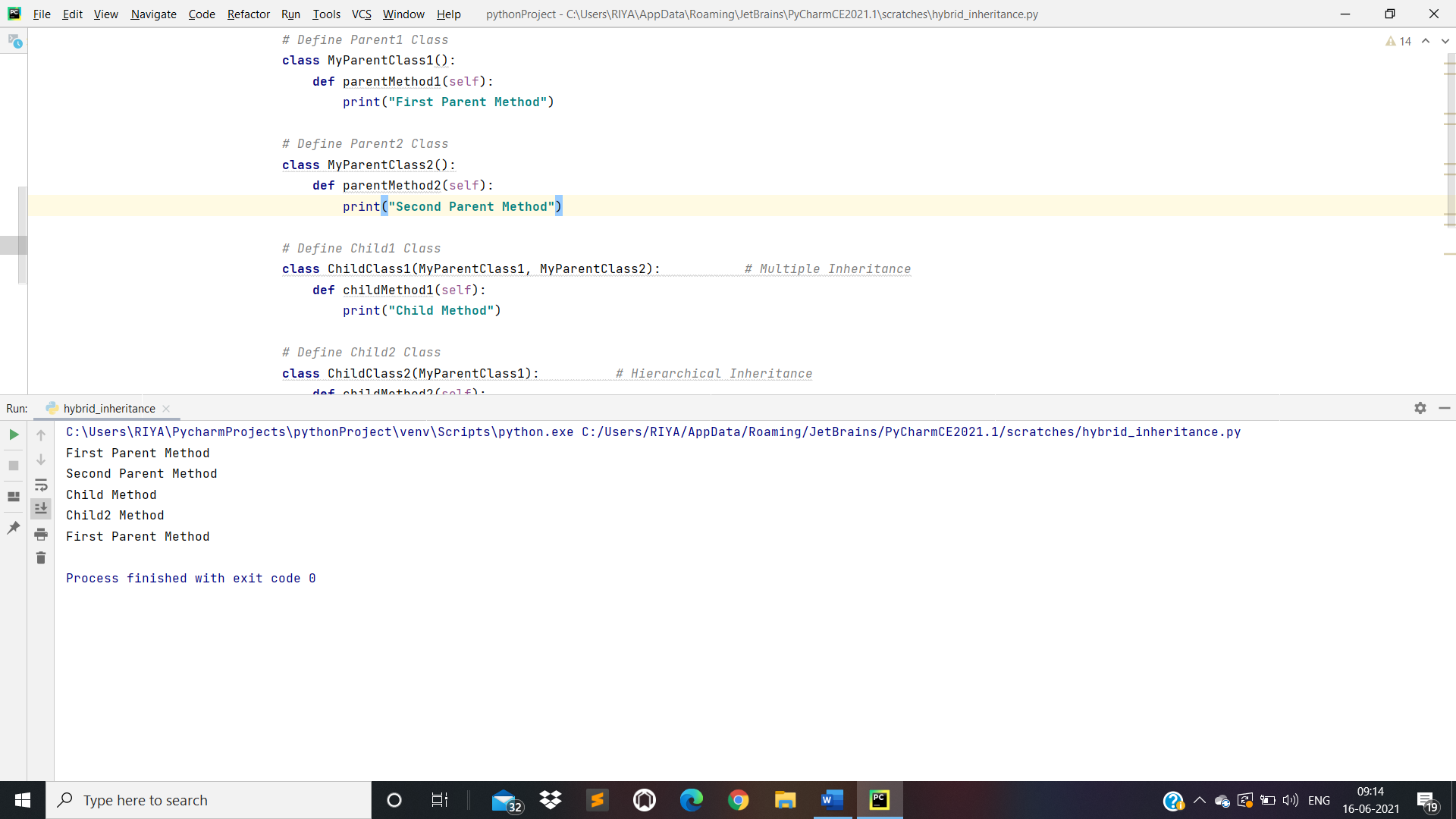
1. Hierarchical Inheritance:

*# Define Parent Class***class** Parent():  
  
 **def** \_\_init\_\_(self):  
 print(**"Parent Constructor"**)  
  
 **def** parentMethod(self):  
 print(**"Parent Method"**)  
  
*# Define SubClass(Child1) of Parent Class***class** Child1(Parent):  
 **def** \_\_init\_\_(self):  
 print(**"Child1 Constructor"**)  
  
 **def** child1Method(self):  
 print(**"Child1 Method"**)  
  
*# Define Another SubClass(Child2) of Parent Class***class** Child2(Parent):  
 **def** \_\_init\_\_(self):  
 print(**"Child2 Constructor"**)  
  
 **def** child2Method(self):  
 print(**"Child2 Method"**)  
  
*# Creating Object*c = Child2()  
c.child2Method()  
c.parentMethod()



1. Hybrid Inheritance:

*# Define Parent1 Class***class** MyParentClass1():  
 **def** parentMethod1(self):  
 print(**"First Parent Method"**)  
  
*# Define Parent2 Class***class** MyParentClass2():  
 **def** parentMethod2(self):  
 print(**"Second Parent Method"**)  
  
*# Define Child1 Class***class** ChildClass1(MyParentClass1, MyParentClass2): *# Multiple Inheritance* **def** childMethod1(self):  
 print(**"Child Method"**)  
  
*# Define Child2 Class***class** ChildClass2(MyParentClass1): *# Hierarchical Inheritance* **def** childMethod2(self):  
 print(**"Child2 Method"**)  
  
c = ChildClass1()  
c.parentMethod1()  
c.parentMethod2()  
c.childMethod1()  
  
c2 = ChildClass2()  
c2.childMethod2()  
c2.parentMethod1()



1. Polymorphism:
2. Overriding Methods:

*# Define Parent Class***class** ParentClass():  
  
 **def** func1(self):  
 print(**"Parent Method"**)  
  
*# Define Child Class***class** ChildClass(ParentClass):  
  
 **def** func1(self):  
 print(**"Child Method"**)  
  
*# Object of ChildClass()*c = ChildClass()  
c.func1()  
  
*# Object of ParentClass()*p = ParentClass()  
p.func1()



1. Overloading Methods:

*# Define Class***class** MyClass():  
  
  
 **def** sum(self,n1,n2,n3):  
 ans=n1+n2+n3  
 print(ans)  
  
 **def** sum(self, n1, n2):  
 ans = n1 + n2  
 print(ans)  
  
  
*# Creating object*p = MyClass()  
p.sum(10, 20)

