



# Python Class

**#Python Notes**

# Functions in Python



# Function in Python

- Function is a group of related statements that perform a specific task.

- **Syntax :-**

```
def function_name(parameters):  
    statement(s)  
    return [expression]
```

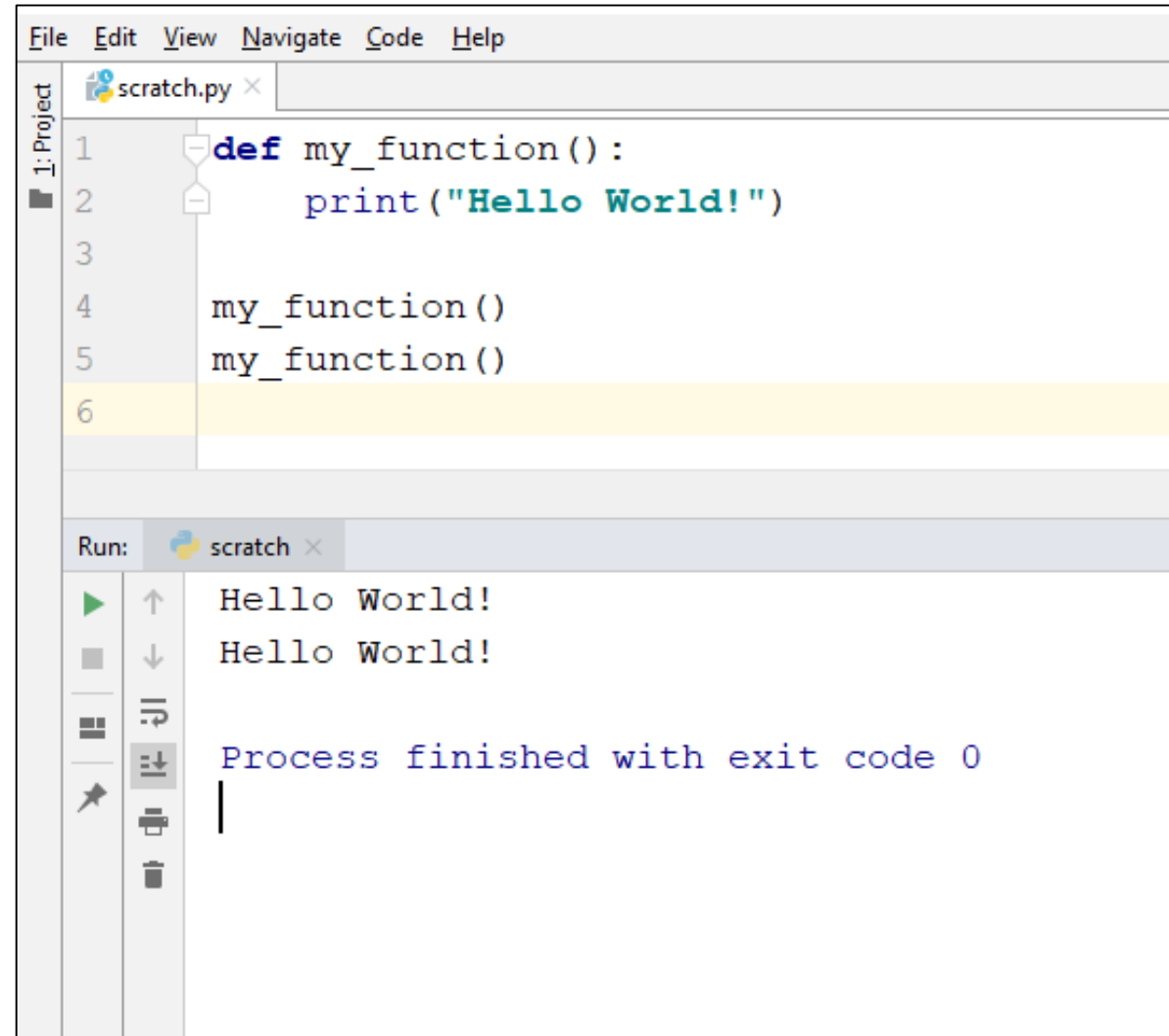


## Cont...

- Here,
- **def** keyword marks the start of function header.
- A **function name** to uniquely identify it.
- A **colon (:)** to mark the end of function header.
- The **function body**.
- An optional **return statement** to return a value from the function.



# Example



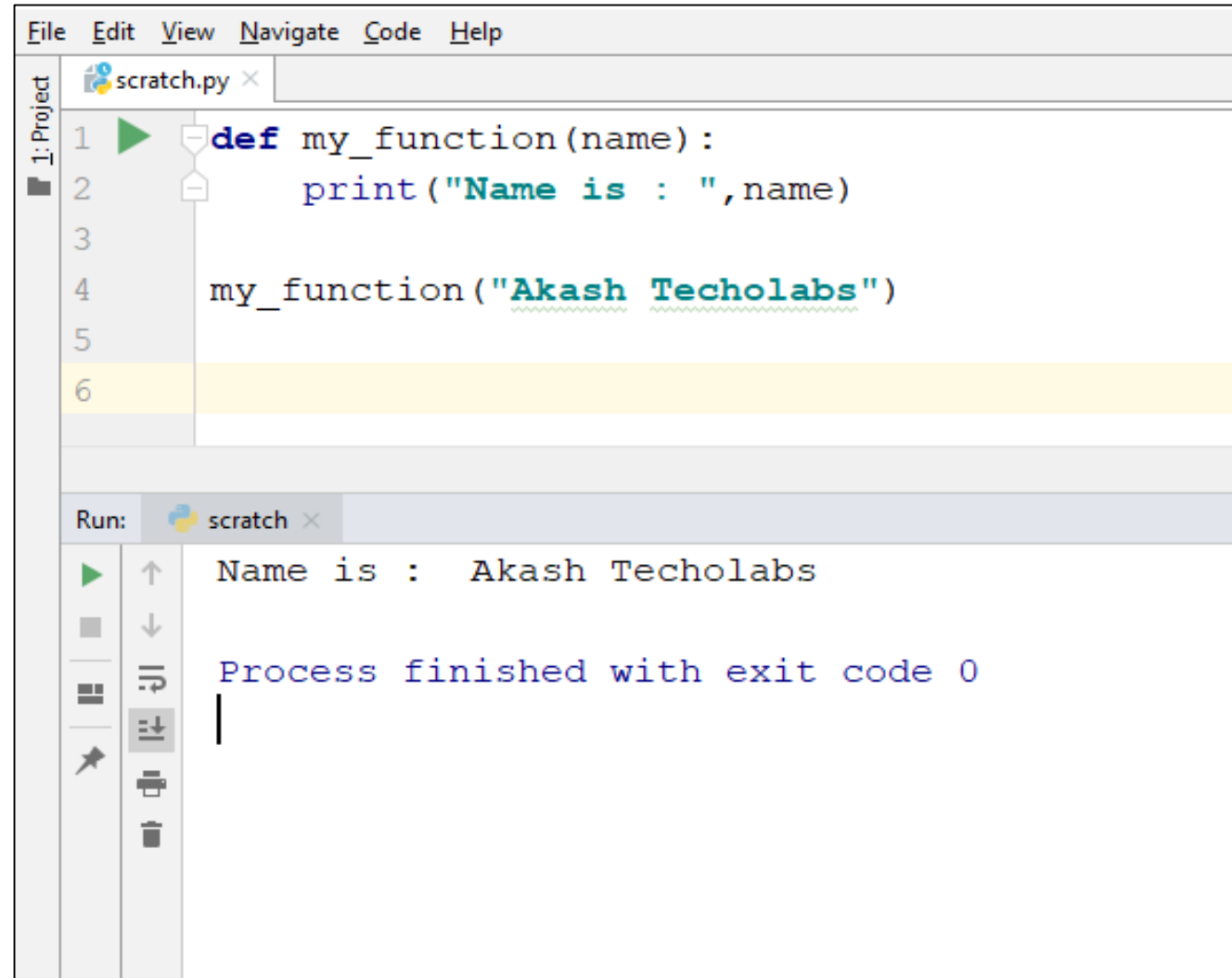
The screenshot shows a Python IDE with a menu bar (File, Edit, View, Navigate, Code, Help) and a tab for 'scratch.py'. The code editor contains the following Python code:

```
1 def my_function():
2     print("Hello World!")
3
4 my_function()
5 my_function()
6
```

Line 6 is highlighted in yellow. Below the code editor is a 'Run' panel with a tab for 'scratch'. It shows the output of the program:

```
▶ ↑ Hello World!
■ ↓ Hello World!
⌵ ↻
⌵ ⌵ Process finished with exit code 0
⌵ ⌵ |
⌵ ⌵
```

# Example With Argument



The screenshot shows a Python IDE with a menu bar (File, Edit, View, Navigate, Code, Help) and a toolbar. The main editor window displays a file named 'scratch.py' with the following code:

```
1 def my_function(name):  
2     print("Name is : ", name)  
3  
4 my_function("Akash Techolabs")  
5  
6
```

The code defines a function `my_function` that takes a parameter `name` and prints a message. It then calls the function with the argument `"Akash Techolabs"`. The output window, titled 'Run: scratch', shows the execution results:

```
Name is : Akash Techolabs  
  
Process finished with exit code 0
```

## Example with return Statement

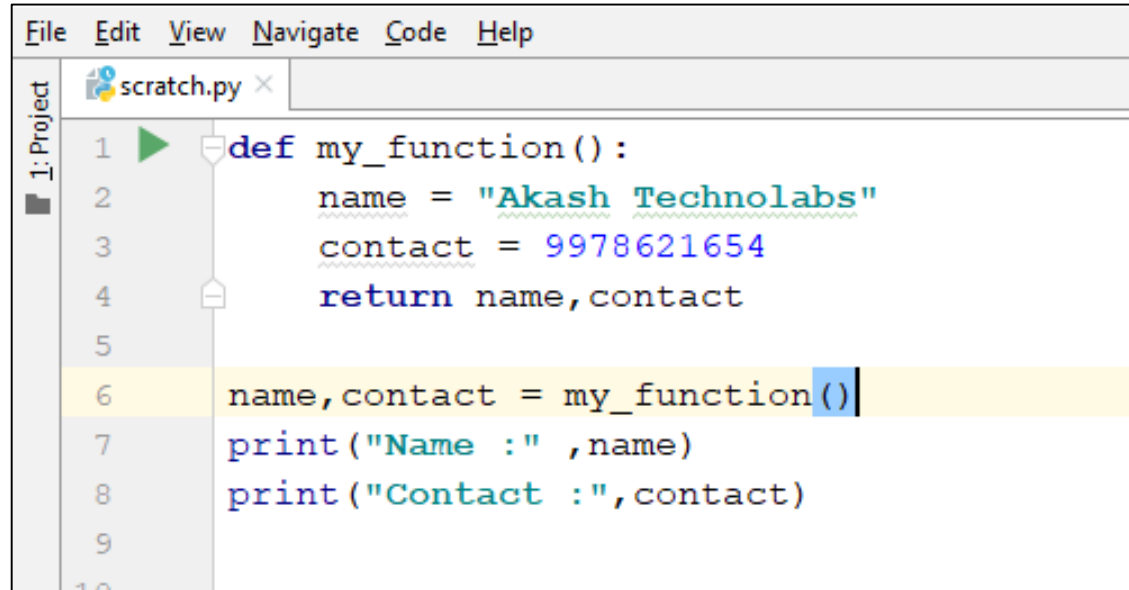
The screenshot shows a code editor window with a menu bar at the top containing "File", "Edit", "View", "Navigate", "Code", and "Help". Below the menu bar, there's a tab labeled "scratch.py" with a close button. The main area displays a Python script:

```
1 def my_function(name):  
2     return name  
3  
4 name = my_function("Akash Techolabs")  
5 print(name)  
6
```

Line 3 is highlighted in yellow. To the left of the code, there's a vertical toolbar with icons for project view, run, and other functions. Below the code editor, there's a "Run:" section with a tab labeled "scratch". It contains a green play button icon, a vertical toolbar with navigation and editing icons, and the output of the program:

```
Akash Techolabs  
  
Process finished with exit code 0  
|
```

# Example With Multiple Return Statement



```
File Edit View Navigate Code Help
scratch.py x
1: Project
1  def my_function():
2     name = "Akash Technolabs"
3     contact = 9978621654
4     return name,contact
5
6  name,contact = my_function()
7  print("Name :" ,name)
8  print("Contact :",contact)
9
10
```

Name : Akash Technolabs  
Contact : 9978621654



# Python function arguments

- There are three types of Python function arguments using which we can call a function.
  - Default Arguments
  - Keyword Arguments
  - Variable-length Arguments



# Python Default Arguments

- Sometimes we may want to use parameters in a function that takes default values in case the user doesn't want to provide a value for them.



# Example

```
File Edit View Navigate Code Help
scratch.py x
1: Project
1  def sum(a=5, b=7):
2      """ This function will print sum of two numbers
3          if the arguments are not supplied
4          it will add the default value """
5      print (a+b)
6
7      sum(10,20) #calling with arguments
8      sum( )    #calling without arguments
9
```

30

12

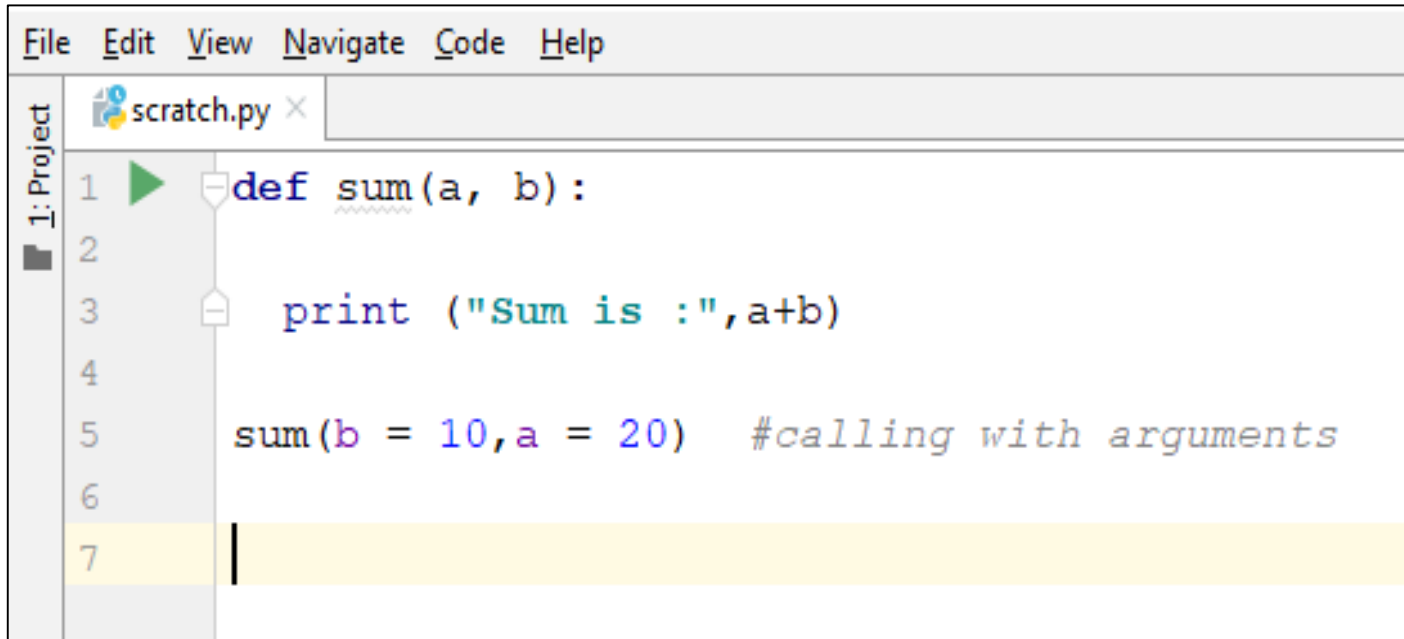


# Python Keyword Arguments

- In function, the values passed through arguments are assigned to parameters in order, by their position.
- With Keyword arguments, we can use the name of the parameter irrespective of its position while calling the function to supply the values.



# Example



```
File Edit View Navigate Code Help
scratch.py x
1 def sum(a, b):
2
3     print ("Sum is :",a+b)
4
5     sum(b = 10,a = 20)  #calling with arguments
6
7
```

Sum is : 30

# Variable-length Arguments

- Sometimes you may need more arguments to process function then you mentioned in the definition.
- If we don't know in advance about the arguments needed in function, we can use variable-length arguments also called arbitrary arguments.

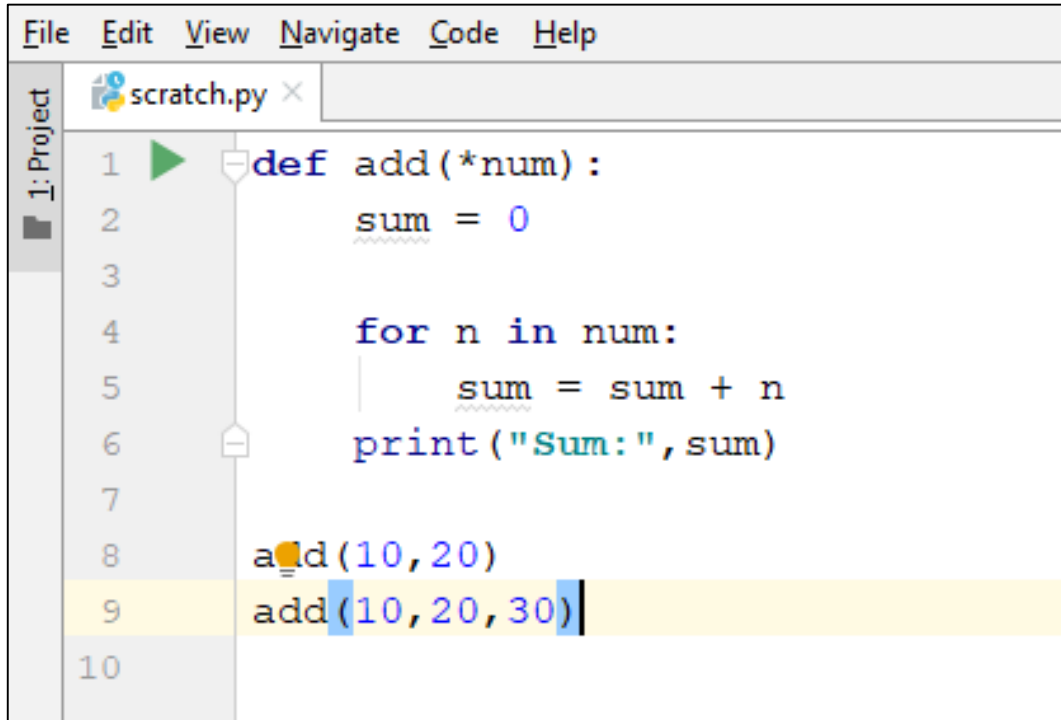


## Cont..

- We can pass a variable number of arguments to a function using special symbols. There are two special symbols:
  - \* (Non Keyword Arguments)
  - \*\* (Keyword Arguments)
- An asterisk (\*) is placed before a parameter in function definition which can hold non-keyworded variable-length arguments.
- A double asterisk (\*\*) is placed before a parameter in function which can hold keyworded variable-length arguments.



# Example (Non Keyword Arguments)



```
File Edit View Navigate Code Help
scratch.py x
1 def add(*num):
2     sum = 0
3
4     for n in num:
5         sum = sum + n
6     print("Sum:", sum)
7
8 add(10, 20)
9 add(10, 20, 30)
10
```

Sum: 30

Sum: 60



# Example (Keyword Arguments)

```
scratch.py x
1 def my_func(**arg):
2     for i, j in arg.items():
3         print(i, j)
4
5 my_func(Name='Akash', Lastname='Padhiyar')
6
```

```
Name Akash
Lastname Padhiyar
```

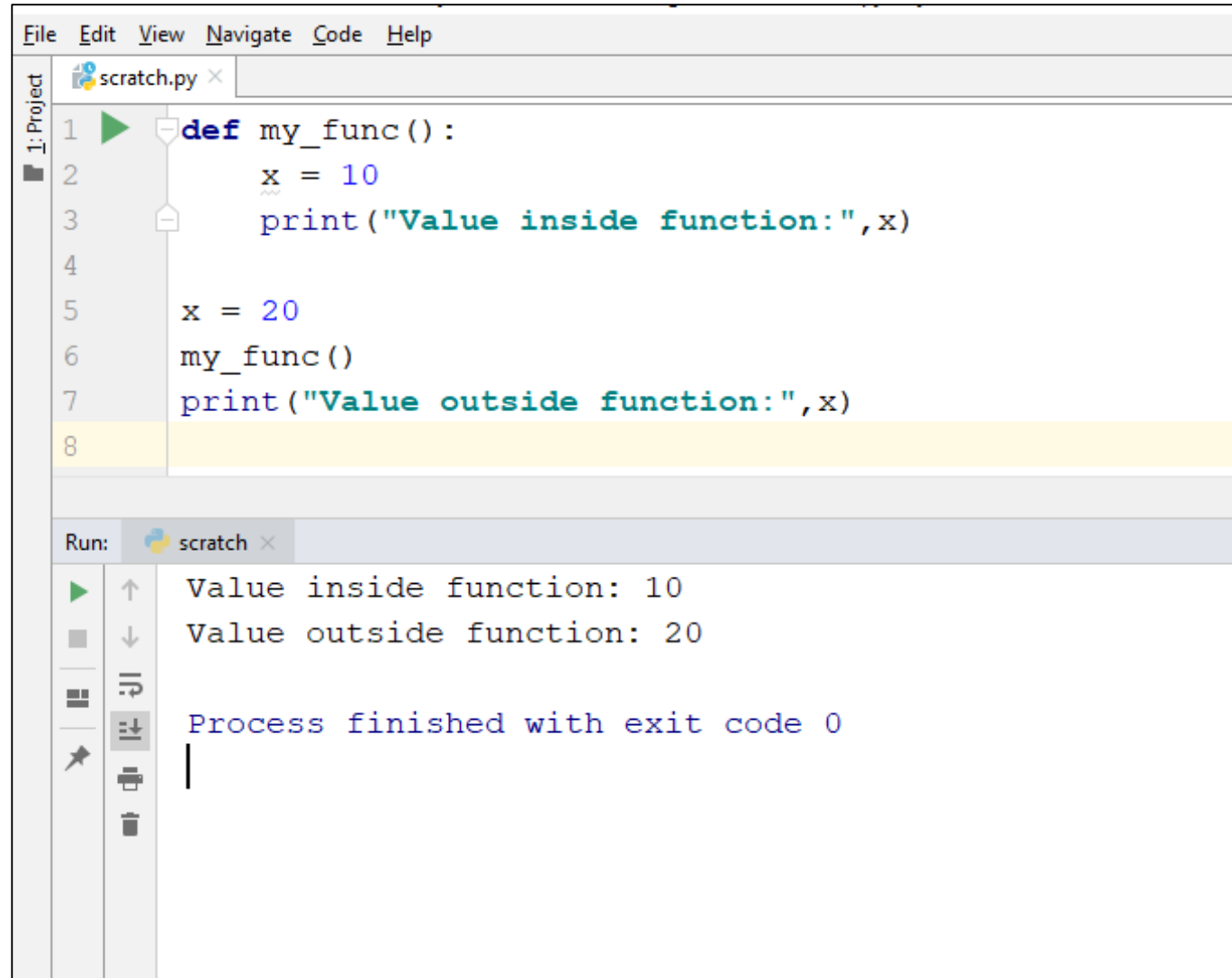


# Scope of Variables

- There are two basic scopes of variables in Python –
  1. Global variables
  2. Local variables
- Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.



# Example



The screenshot shows a Python IDE with a menu bar (File, Edit, View, Navigate, Code, Help) and a tab for 'scratch.py'. The code editor contains the following Python code:

```
1 def my_func():  
2     x = 10  
3     print("Value inside function:", x)  
4  
5     x = 20  
6     my_func()  
7     print("Value outside function:", x)  
8
```

Below the code editor is a 'Run' panel with a tab for 'scratch'. It shows the output of the program:

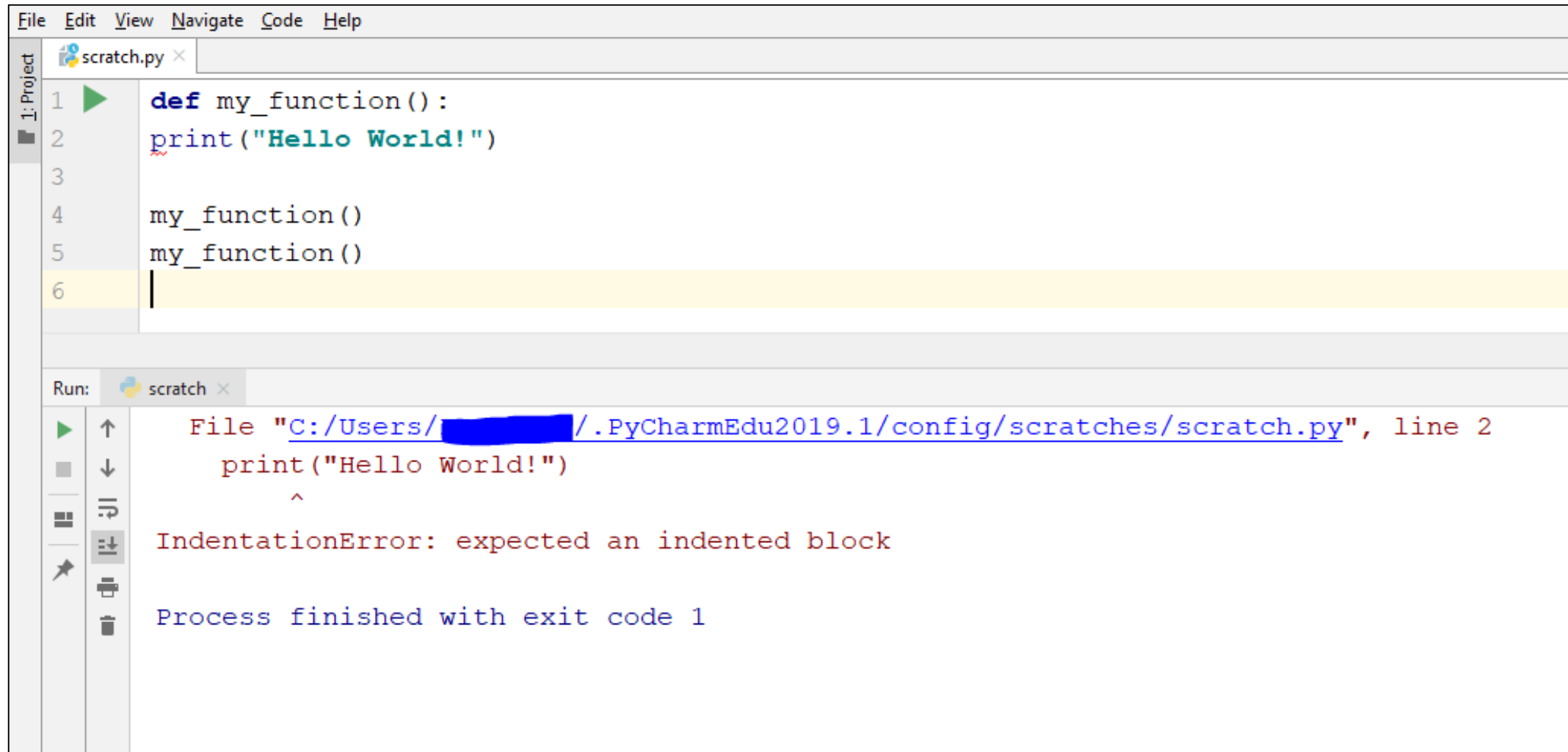
```
Value inside function: 10  
Value outside function: 20  
  
Process finished with exit code 0
```

# Significance of Indentation (Space) in Python

- We take a simple example with "print" command. When we write "print" function right below the `def my_function ()`:
- It will show an **"indentation error: expected an indented block"**.



# Example



The screenshot shows a Python IDE with a file named `scratch.py`. The code in the editor is as follows:

```
1 def my_function():
2     print("Hello World!")
3
4 my_function()
5 my_function()
6
```

The IDE has a menu bar with `File`, `Edit`, `View`, `Navigate`, `Code`, and `Help`. A sidebar on the left shows the project structure. Below the editor, the `Run` console displays the following error message:

```
File "C:/Users/[redacted]/.PyCharmEdu2019.1/config/scratches/scratch.py", line 2
    print("Hello World!")
    ^
IndentationError: expected an indented block

Process finished with exit code 1
```

# Modules in Python

- A file containing Python code, for example : `scratch.py`, is called a **module** and its module name would be `scratch`.
- We use modules to break down large programs into small manageable and organized files.
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.



# The import Statement

- The ***import*** keyword is used to import module to another module
- **Syntax :-**  
`import module_name`



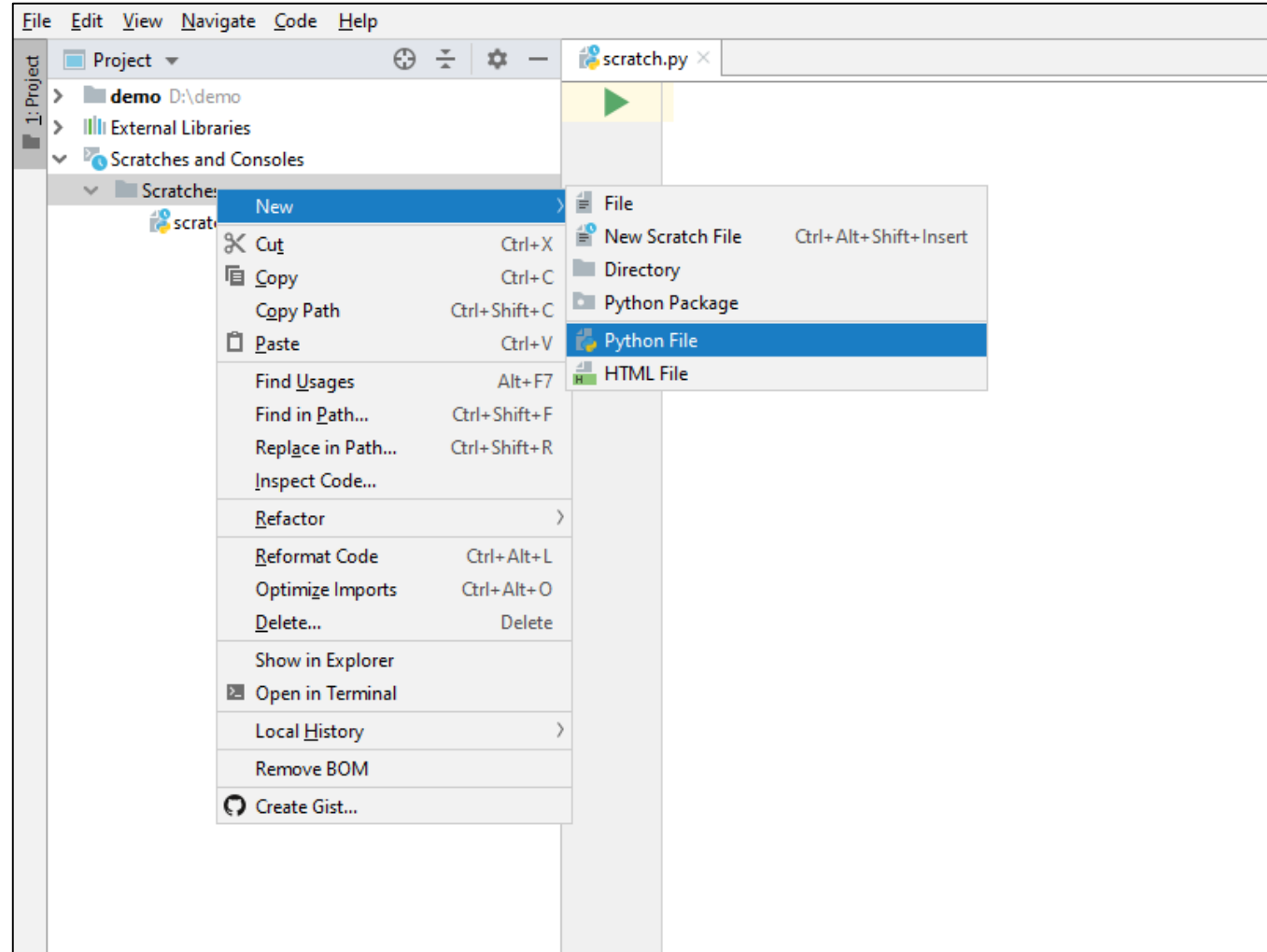
# Example

- In the following example we create 2 files :-
  - scratch.py
  - demo.py
- Scratch.py file contain user define function my\_function() that we will import in demo.py file.

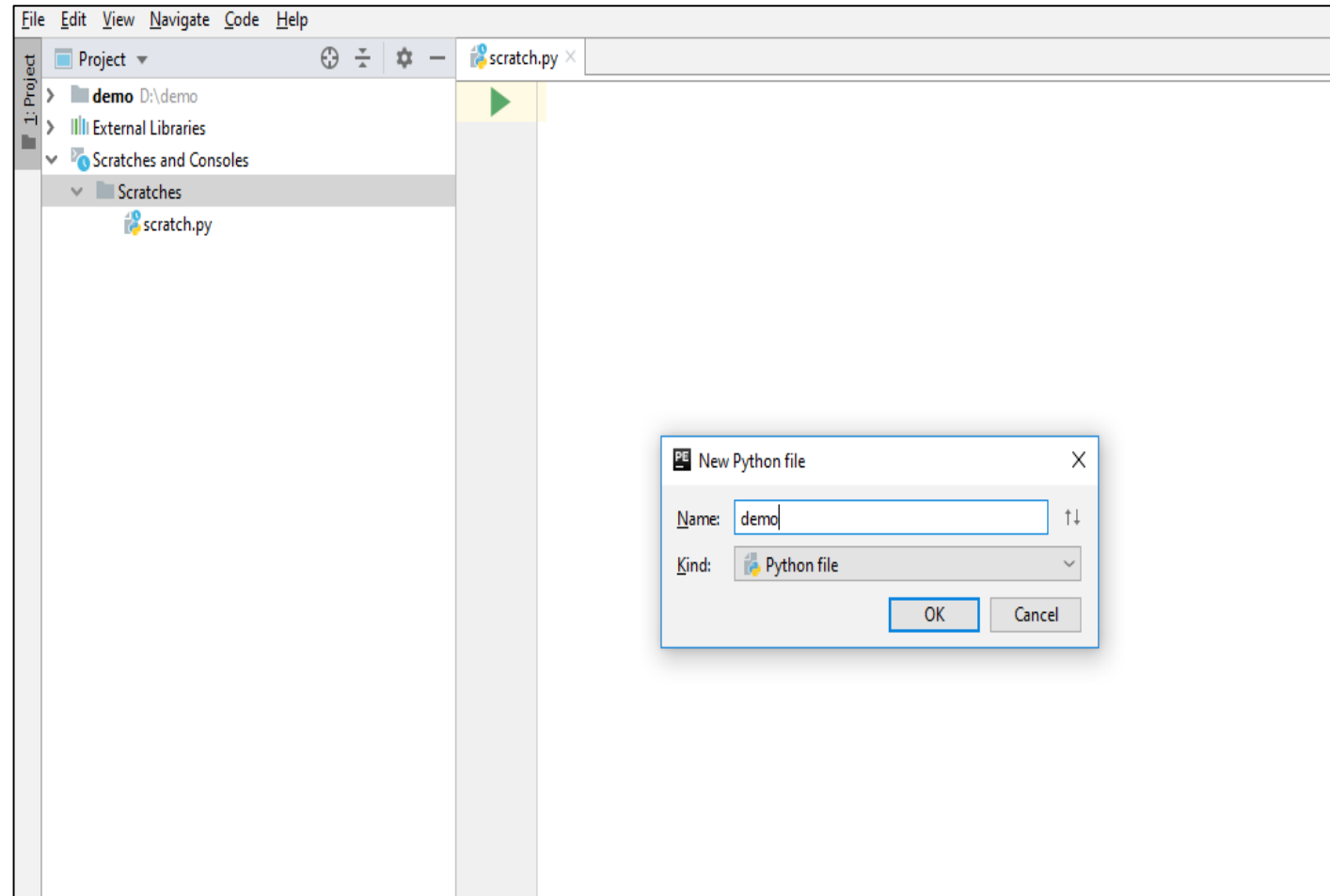




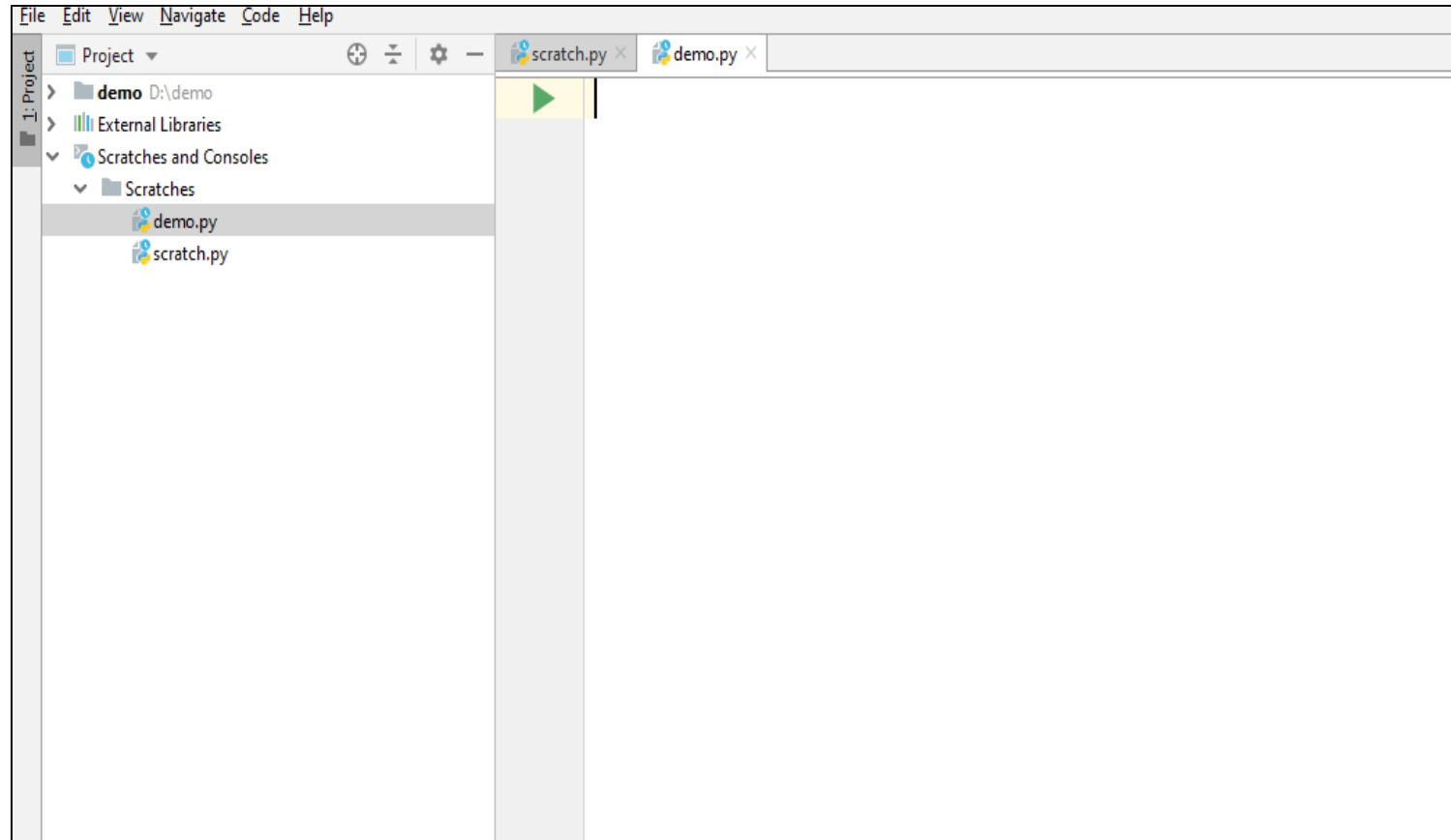
# Create a new file by right clicking on Scratches folder



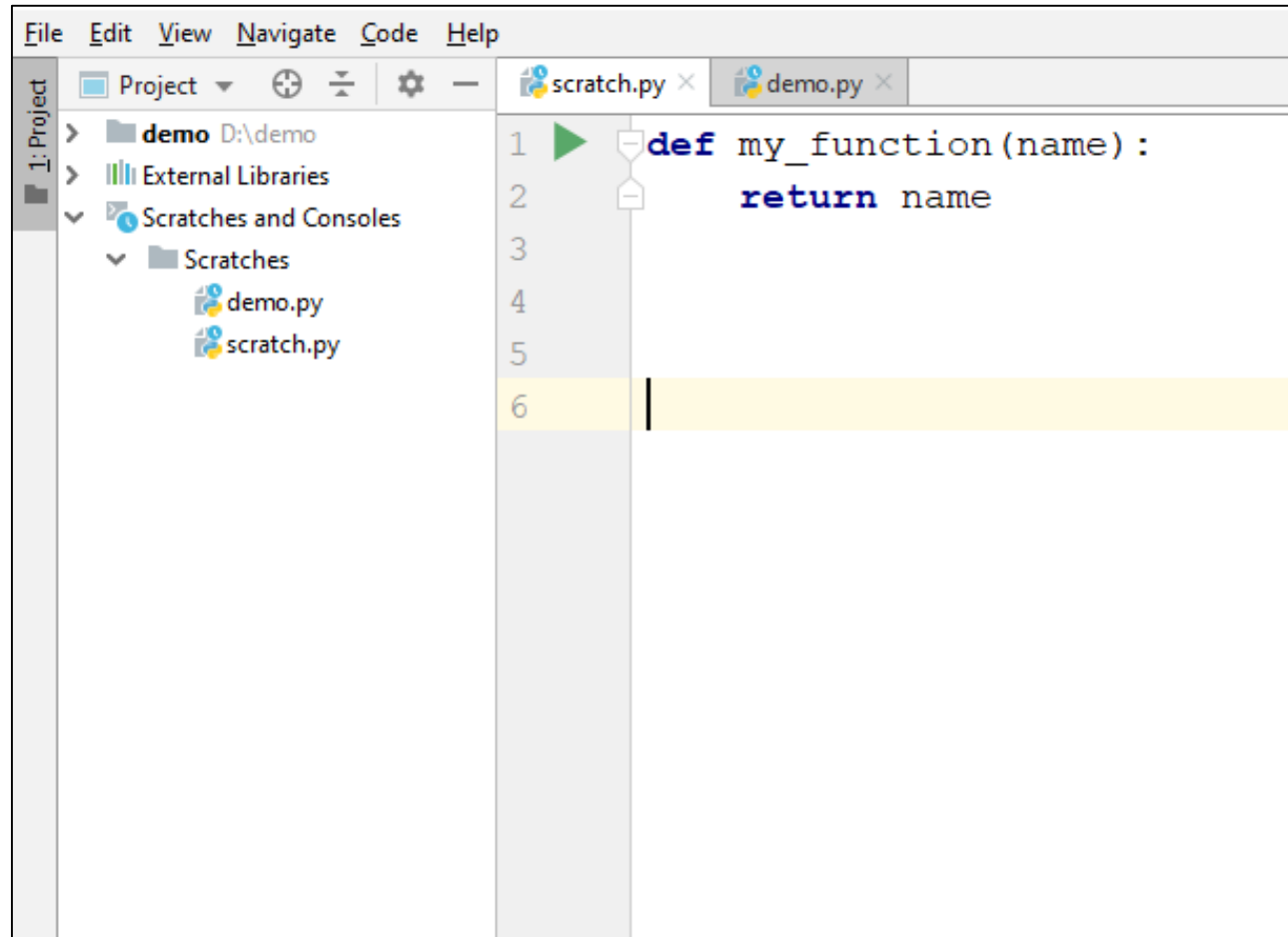
# Give file name



# File is created



# scratch.py



The screenshot shows an IDE window with a menu bar (File, Edit, View, Navigate, Code, Help) and a toolbar. The left sidebar displays a project tree with the following structure:

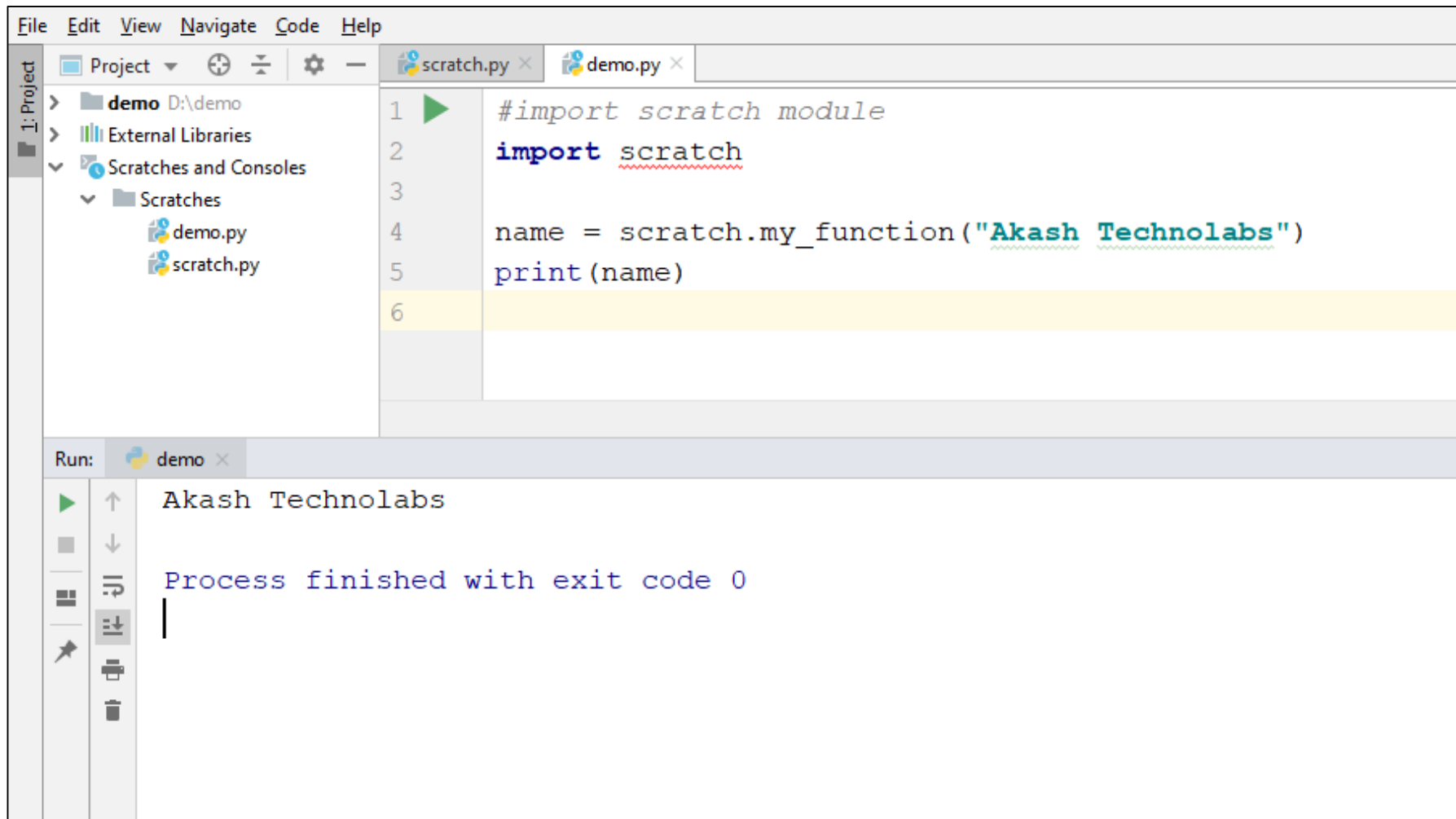
- 1: Project
  - > demo D:\demo
  - > External Libraries
  - > Scratches and Consoles
    - Scratches
      - demo.py
      - scratch.py

The main editor area shows the content of `scratch.py` with line numbers 1 through 6 on the left. The code is as follows:

```
1 def my_function(name):  
2     return name  
3  
4  
5  
6
```

Line 6 is highlighted in yellow, and a vertical cursor is positioned at the end of the line.

# demo.py



The screenshot shows an IDE window with a menu bar (File, Edit, View, Navigate, Code, Help) and a toolbar. The left sidebar displays a project tree with a folder named 'demo' at 'D:\demo', containing 'demo.py' and 'scratch.py'. The main editor area shows the code in 'demo.py' with line numbers 1 through 6. The code imports a 'scratch' module and calls its 'my\_function' with the argument 'Akash Technolabs'. The bottom 'Run' panel shows the output 'Akash Technolabs' and 'Process finished with exit code 0'.

```
1 #import scratch module
2 import scratch
3
4 name = scratch.my_function("Akash Technolabs")
5 print(name)
6
```

Run: demo x

Akash Technolabs

Process finished with exit code 0

# Operators in Python



# Operators in Python

- Arithmetic Operators
- Comparison operators
- Logical operators
- Assignment operators
- Membership Operators
- Identity Operators



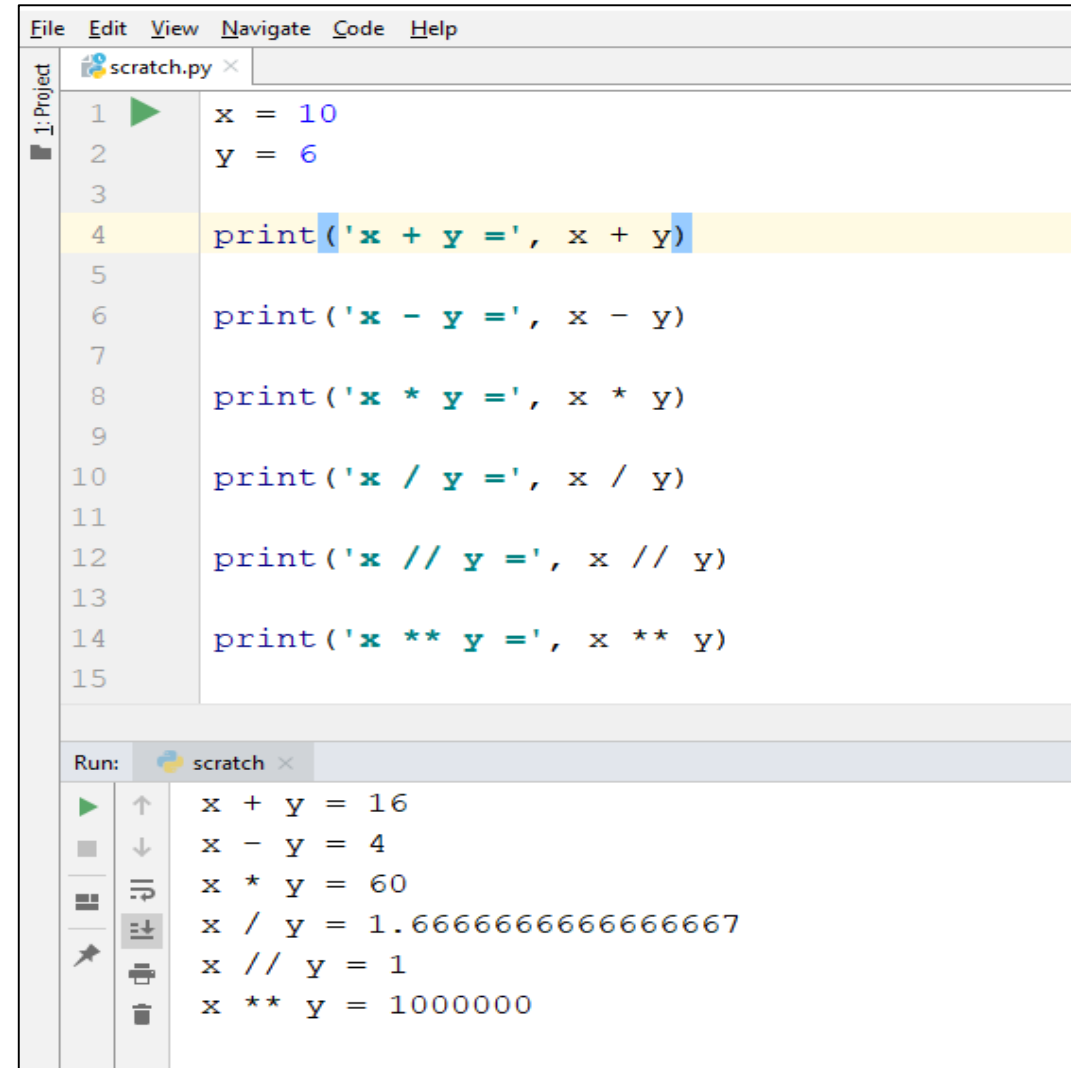
# Arithmetic Operators

Operator	Meaning
+	Add two operands or unary plus
-	Subtract right operand from the left or unary minus
*	Multiply two operands
/	Divide left operand by the right one (always results into float)
%	Modulus - remainder of the division of left operand by the right
//	Floor division - division that results into whole number adjusted to the left in the number line
**	Exponent - left operand raised to the power of right(x to the power y)





# Example



```
File Edit View Navigate Code Help
scratch.py x
1 x = 10
2 y = 6
3
4 print('x + y =', x + y)
5
6 print('x - y =', x - y)
7
8 print('x * y =', x * y)
9
10 print('x / y =', x / y)
11
12 print('x // y =', x // y)
13
14 print('x ** y =', x ** y)
15

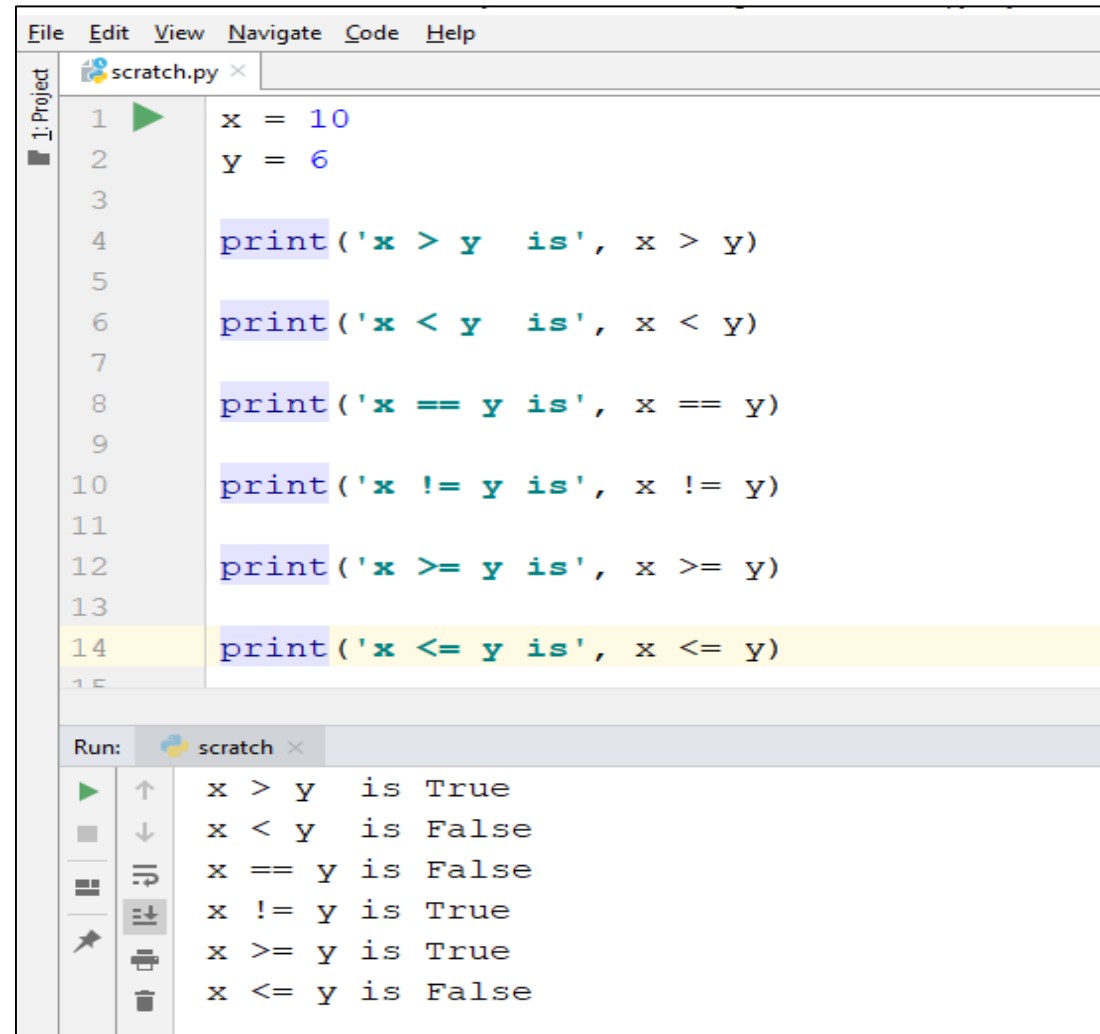
Run: scratch x
x + y = 16
x - y = 4
x * y = 60
x / y = 1.6666666666666667
x // y = 1
x ** y = 1000000
```

# Comparison operators

Operator	Meaning	Example
>	Greater than	<code>x &gt; y</code>
<	Less than	<code>x &lt; y</code>
==	Equal to	<code>x == y</code>
!=	Not equal to	<code>x != y</code>
>=	Greater than or equal to	<code>x &gt;= y</code>
<=	Less than or equal to	<code>x &lt;= y</code>



# Example



```
File Edit View Navigate Code Help
scratch.py x
1 x = 10
2 y = 6
3
4 print('x > y is', x > y)
5
6 print('x < y is', x < y)
7
8 print('x == y is', x == y)
9
10 print('x != y is', x != y)
11
12 print('x >= y is', x >= y)
13
14 print('x <= y is', x <= y)
15

Run: scratch x
x > y is True
x < y is False
x == y is False
x != y is True
x >= y is True
x <= y is False
```

# Logical operators

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x



# Example Of and

Scrach.py ×

C: > Users > Devanshi > Desktop > Scrach.py > ...

```
1  n1=10
2  n2=20
3  n3=30
4
5  if n1 > n2 and n1 > n3:
6      print("n1 is the largest number")
7
8  if n2 > n1 and n2 > n3:
9      print("n2 is the largest number.")
10
11 if n3 > n1 and n3 > n2:
12     print("n3 is the largest number.")
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Devanshi> & python c:/Users/Devanshi/Desktop/Scrach.py

n3 is the largest number.

PS C:\Users\Devanshi> █



# Example of or

Scrach.py ×

C: > Users > Devanshi > Desktop > Scrach.py > ...

```
1 ch = input("Enter a character: ")
2
3 if(ch=='A' or ch=='a' or ch=='E' or ch=='e' or ch=='I'
4    or ch=='i' or ch=='O' or ch=='o' or ch=='U' or ch=='u'):
5     print(ch, "is a Vowel")
6 else:
7     print(ch, "is a Consonant")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: Python

PS C:\Users\Devanshi> & python c:/Users/Devanshi/Desktop/Scrach.py

Enter a character: a

a is a Vowel

PS C:\Users\Devanshi> █



# Assignment operators

Operator	Example	Equivalent to
=	<code>x = y</code>	<code>x = 5</code>
<code>+=</code>	<code>x += 5</code>	<code>x = x + 5</code>
<code>-=</code>	<code>x -= 5</code>	<code>x = x - 5</code>
<code>*=</code>	<code>x *= 5</code>	<code>x = x * 5</code>
<code>/=</code>	<code>x /= 5</code>	<code>x = x / 5</code>
<code>%=</code>	<code>x %= 5</code>	<code>x = x % 5</code>
<code>//=</code>	<code>x //= 5</code>	<code>x = x // 5</code>
<code>**=</code>	<code>x **= 5</code>	<code>x = x ** 5</code>



# Membership Operators

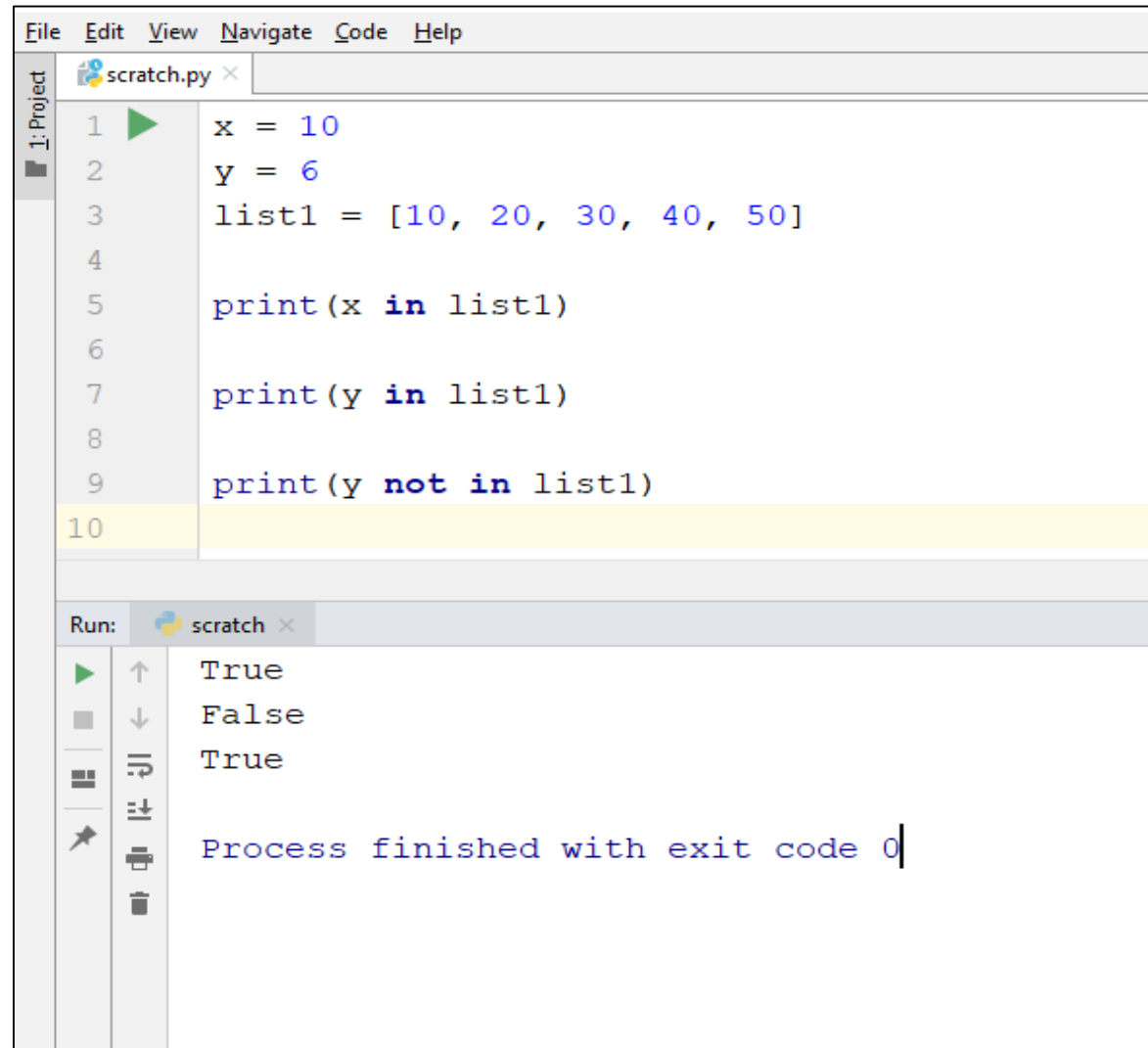
- There are two membership operators that are used in Python.
- They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

Operator	Meaning
in	True if value/variable is found in the sequence
not in	True if value/variable is not found in the sequence





# Example



The screenshot shows a Python IDE with a menu bar (File, Edit, View, Navigate, Code, Help) and a tab for 'scratch.py'. The code editor contains the following Python code:

```
1 x = 10
2 y = 6
3 list1 = [10, 20, 30, 40, 50]
4
5 print(x in list1)
6
7 print(y in list1)
8
9 print(y not in list1)
10
```

Below the code editor is a 'Run' panel with a tab for 'scratch'. It shows the output of the script:

```
True
False
True

Process finished with exit code 0
```

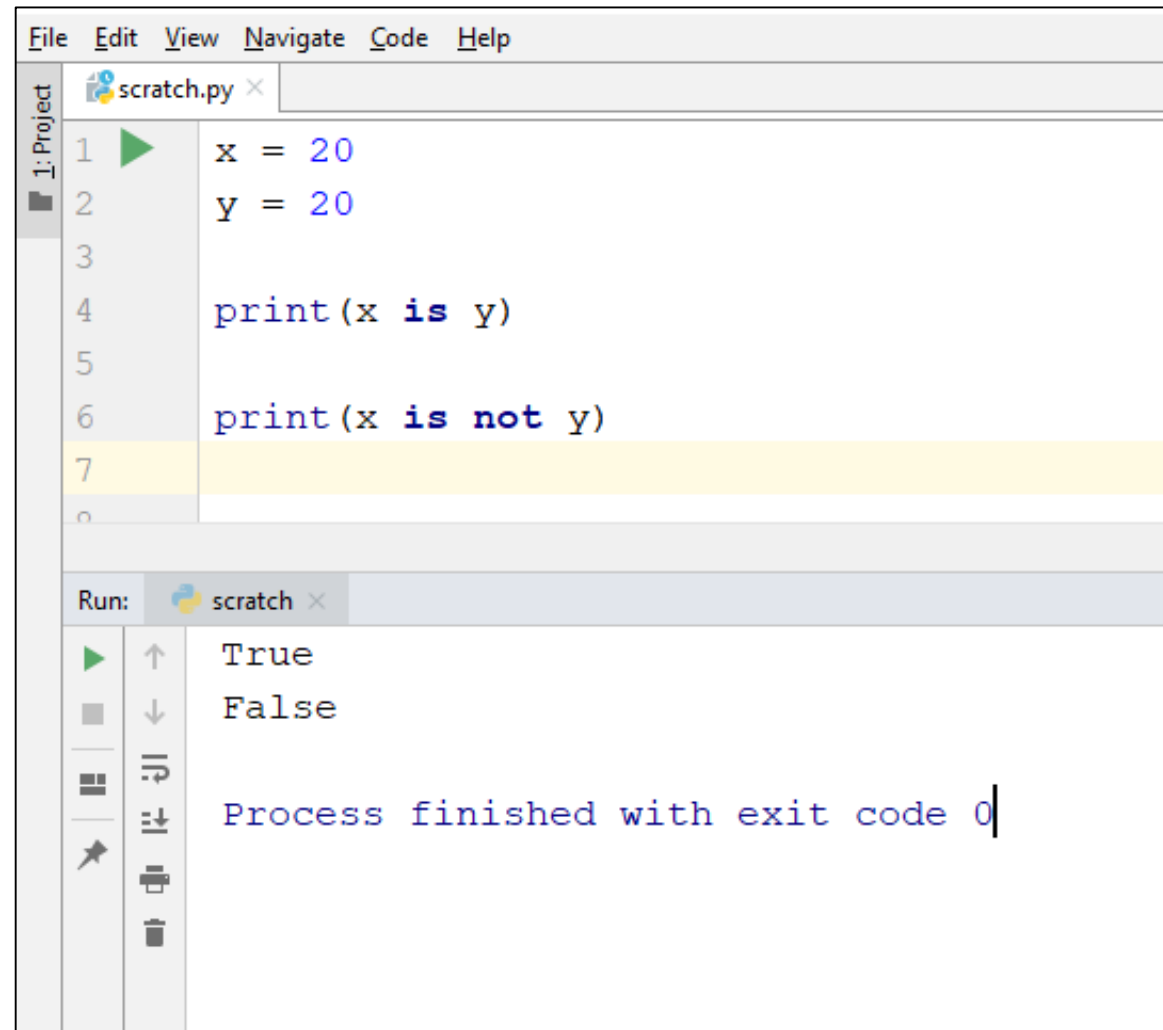
# Identity Operators

- To compare the memory location of two objects, Identity Operators are used. The two identify operators used in Python.

Operator	Meaning
is	True if the operands are identical (refer to the same object)
is not	True if the operands are not identical (do not refer to the same object)



# Example



The screenshot shows a Python IDE window with a menu bar (File, Edit, View, Navigate, Code, Help) and a tab for 'scratch.py'. The code editor contains the following Python code:

```
1 x = 20
2 y = 20
3
4 print(x is y)
5
6 print(x is not y)
7
```

Below the code editor is a 'Run' panel with a tab for 'scratch'. It displays the output of the script:

```
True
False

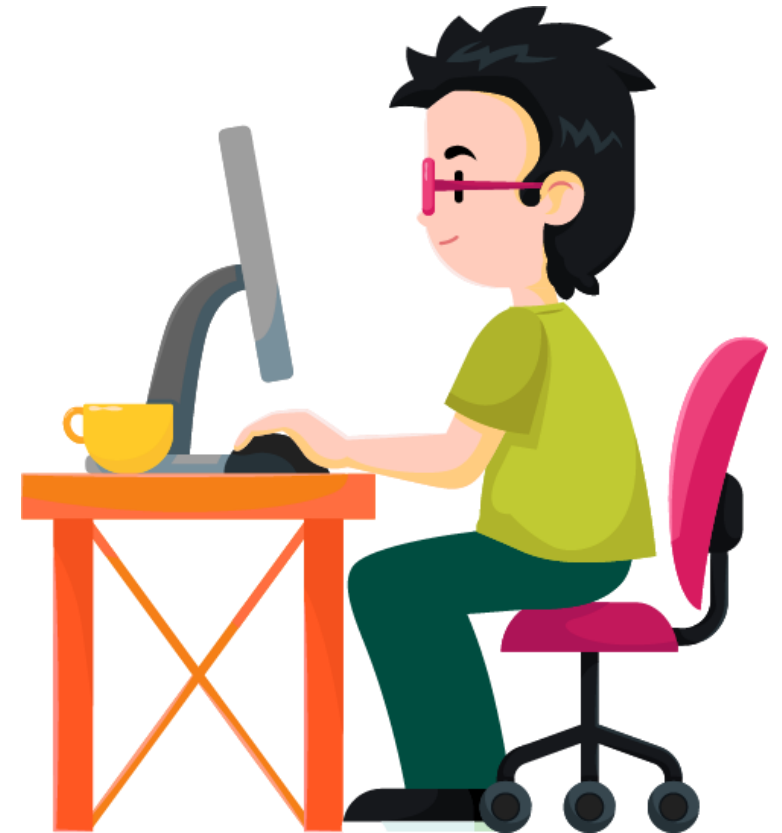
Process finished with exit code 0
```

# Get Exclusive Video Tutorials



[www.apptutorials.com](http://www.apptutorials.com)

<https://www.youtube.com/user/Akashtips>





Get More Details

[www.akashsir.com](http://www.akashsir.com)



# If You Liked It !

## Rating Us Now



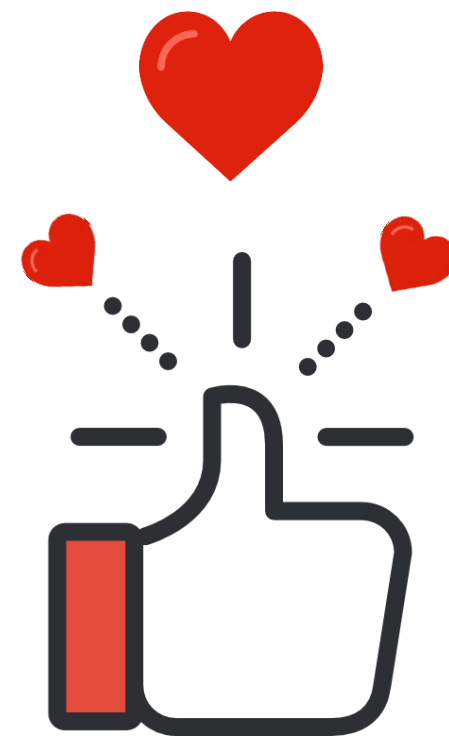
**Just Dial**

[https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4\\_BZDET](https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET)



**Sulekha**

<https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad>



# Connect With Me



Akash Padhiyar  
#AkashSir

[www.akashsir.com](http://www.akashsir.com)

[www.akashtechlabs.com](http://www.akashtechlabs.com)

[www.akashpadhiyar.com](http://www.akashpadhiyar.com)

[www.aptutorials.com](http://www.aptutorials.com)

## # Social Info



Akash.padhiyar



Akashpadhiyar



Akash\_padhiyar



+91 99786-21654



#Akashpadhiyar  
#aptutorials