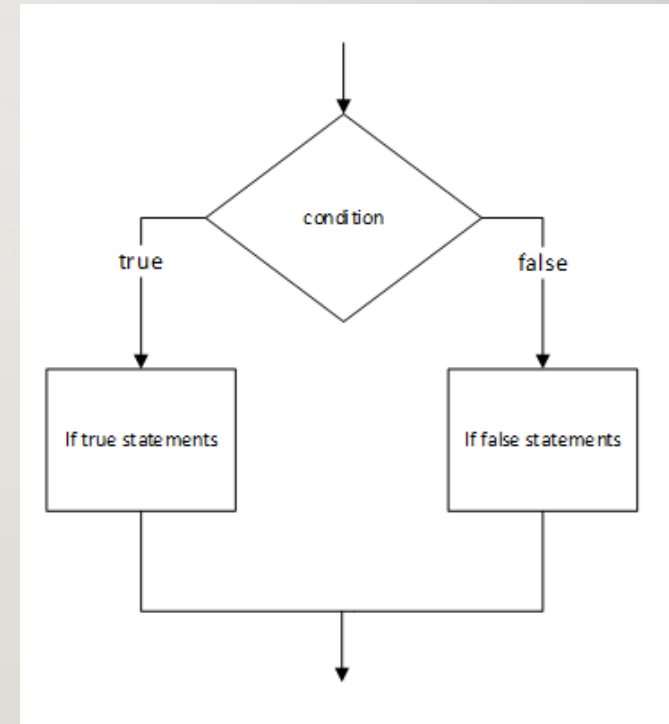


METHODS AND FUNCTION IN PYTHON

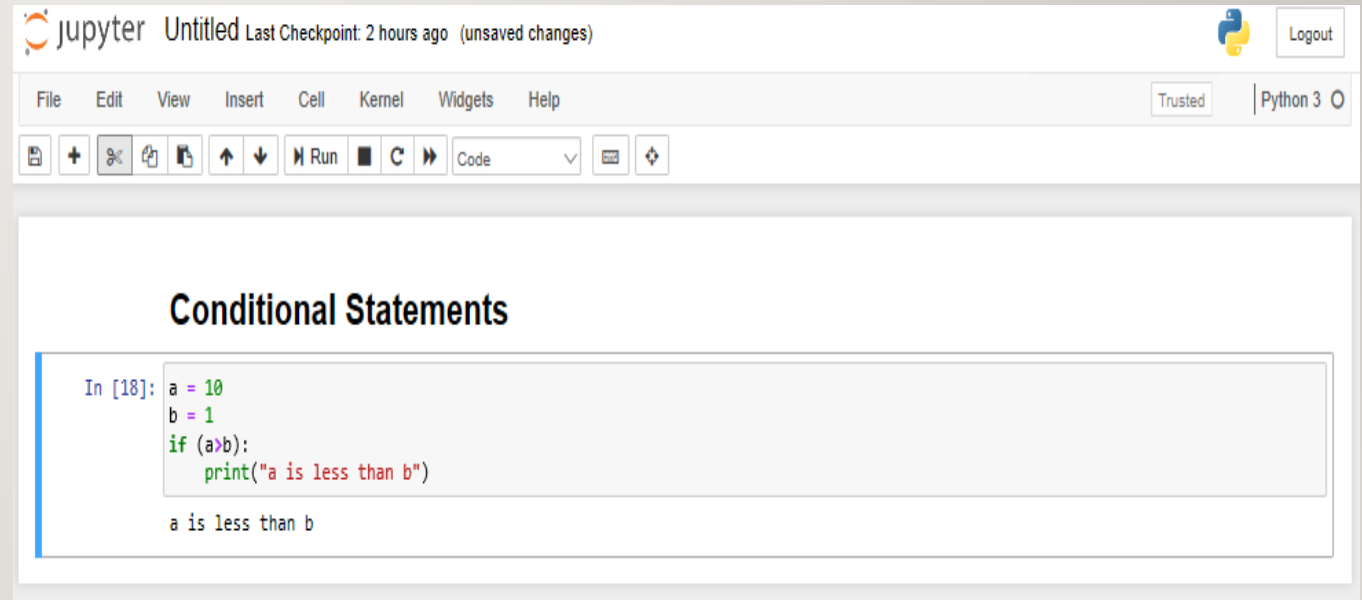
CONDITIONAL STATEMENTS IN PYTHON(1/5)

- The conditional statements are used to control the flow of the program depending upon the Boolean answer of the condition.



IF() METHOD(2/5)

- The if() method is the most basic single condition checker.
- Syntax:
 If (test condition):
 expression

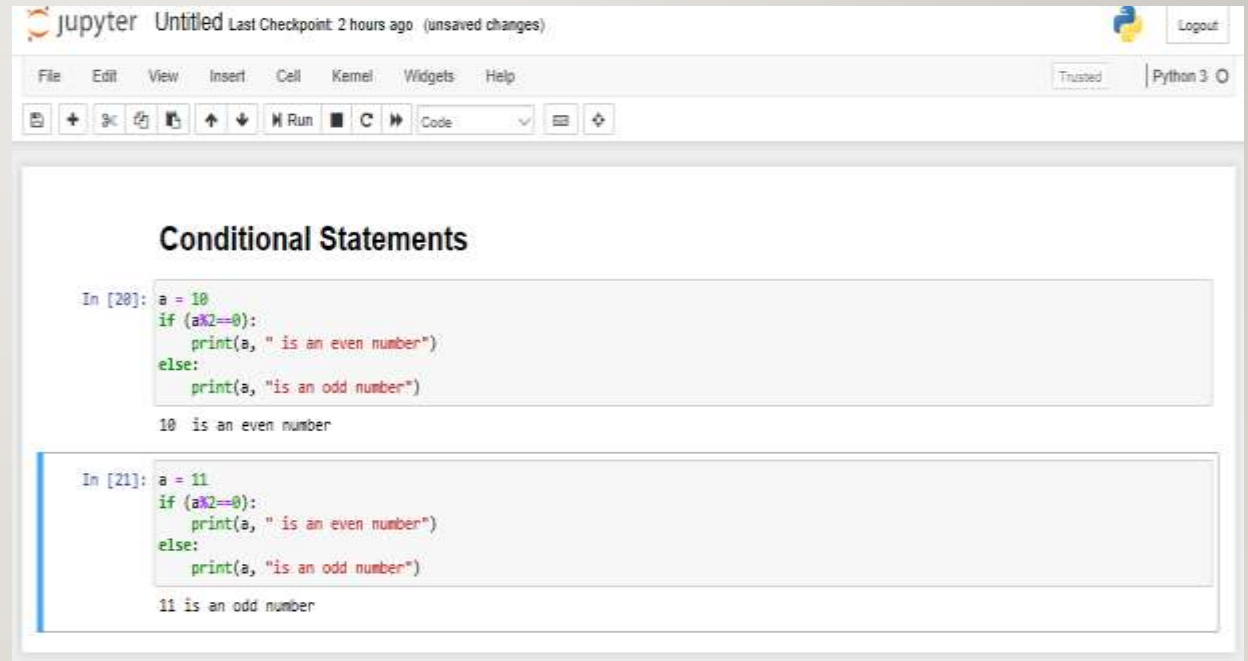
A screenshot of a Jupyter Notebook interface. The title bar says "jupyter Untitled" with a timestamp "Last Checkpoint: 2 hours ago (unsaved changes)" and a "Logout" button. The menu bar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar contains icons for file operations, running, and other notebook functions. The main area has a heading "Conditional Statements" and a code cell. The code cell contains the following Python code:

```
In [18]: a = 10  
b = 1  
if (a>b):  
    print("a is less than b")
```

The output of the code cell is "a is less than b".

IF ELSE() METHOD(3/5)

- If else controls the flow of the program by selecting one of the two options depending upon conditions used.
- Syntax:
 if (test expression):
 expression
 else:
 expression



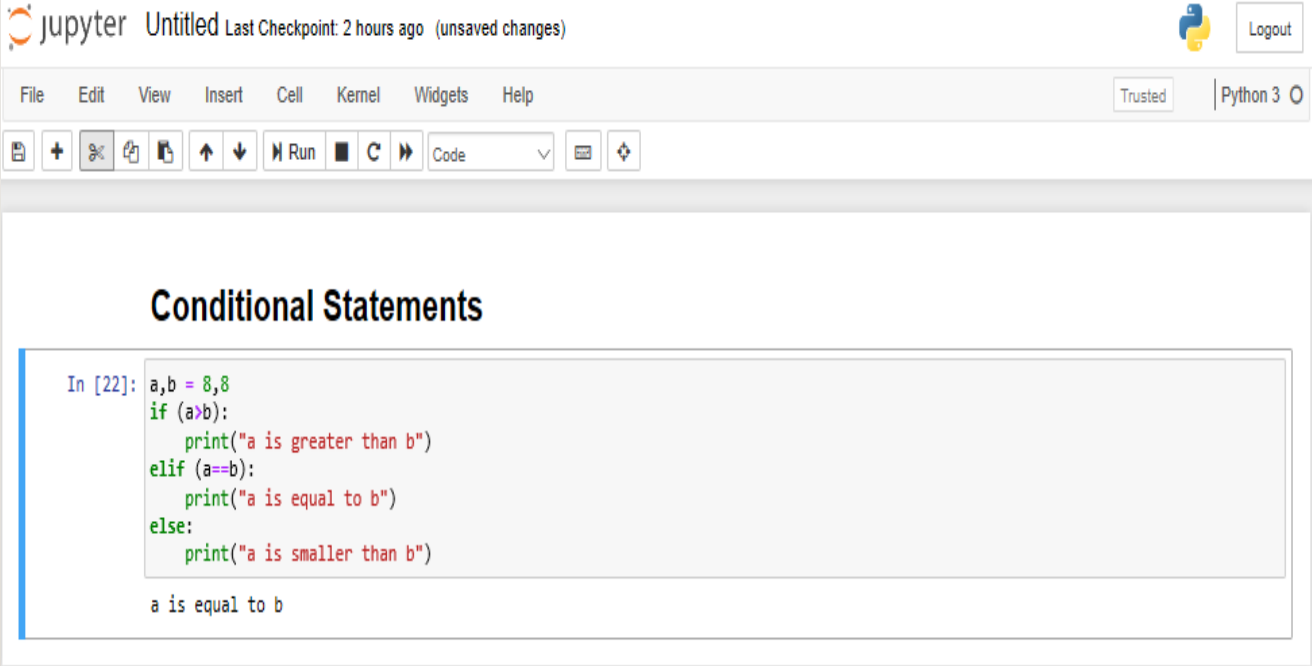
The screenshot shows a Jupyter Notebook interface with a title bar indicating 'Untitled' and 'Last Checkpoint: 2 hours ago (unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for file operations, running, and code execution. The notebook content is titled 'Conditional Statements' and displays two code cells. The first cell, labeled 'In [20]:', contains a Python script that checks if a number 'a' is even or odd using a modulo operation. It prints '10 is an even number'. The second cell, labeled 'In [21]:', uses the same logic for the number 11 and prints '11 is an odd number'.

```
jupyter Untitled Last Checkpoint: 2 hours ago (unsaved changes) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [20]: a = 10
         if (a%2==0):
             print(a, " is an even number")
         else:
             print(a, "is an odd number")
10 is an even number

In [21]: a = 11
         if (a%2==0):
             print(a, " is an even number")
         else:
             print(a, "is an odd number")
11 is an odd number
```

ELIF() METHOD(4/5)

- If the number of conditions increase more than two we tend to use the 'ELIF()' method.
- Syntax:
 - if (test condition):
 expression
 - elif (test condition):
 expression
 - else:
 expression



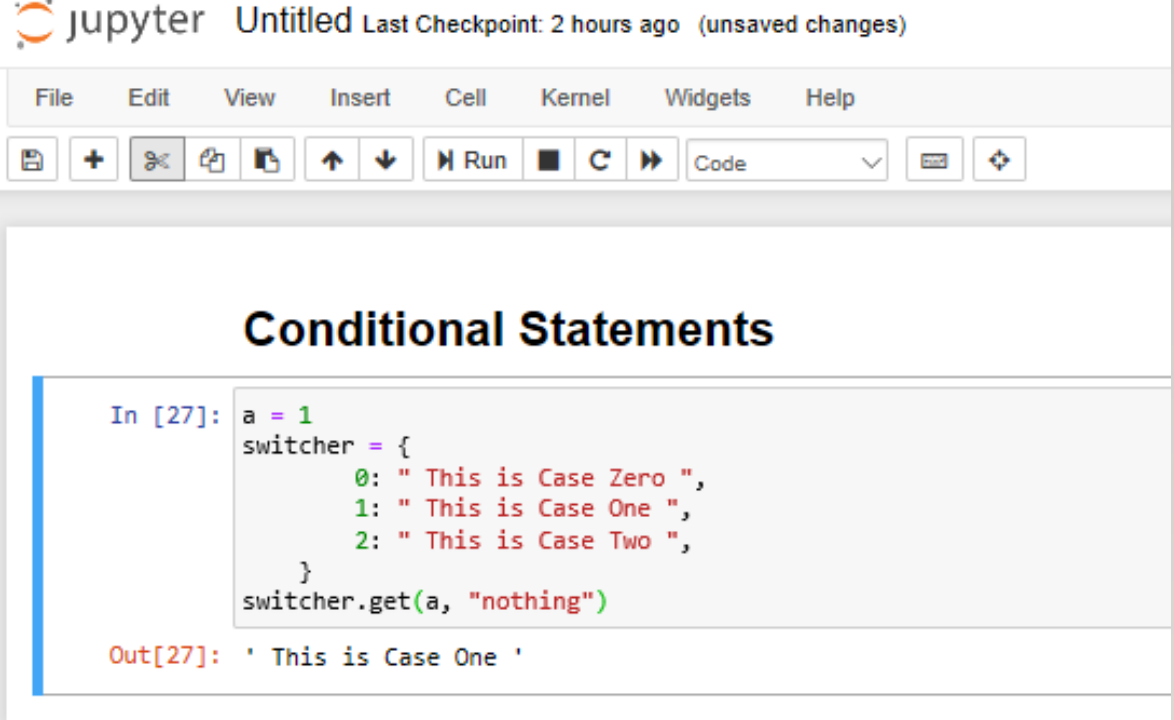
The image shows a Jupyter Notebook interface. The title bar says "jupyter Untitled" and "Last Checkpoint: 2 hours ago (unsaved changes)". The menu bar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar has icons for saving, adding cells, running, and other actions. The code cell contains the following Python code:

```
In [22]: a,b = 8,8
if (a>b):
    print("a is greater than b")
elif (a==b):
    print("a is equal to b")
else:
    print("a is smaller than b")
```

The output of the code cell is "a is equal to b".

SWITCHER(5/5)

- The switcher in python is a multi branched conditional statement.
- Syntax:
 switcher = {
 case 1: expression,
 case 2: expression,
 }
 switcher.get(argument,
 "nothing")



The image shows a Jupyter Notebook window titled "Untitled" with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The notebook contains a code cell with the following Python code:

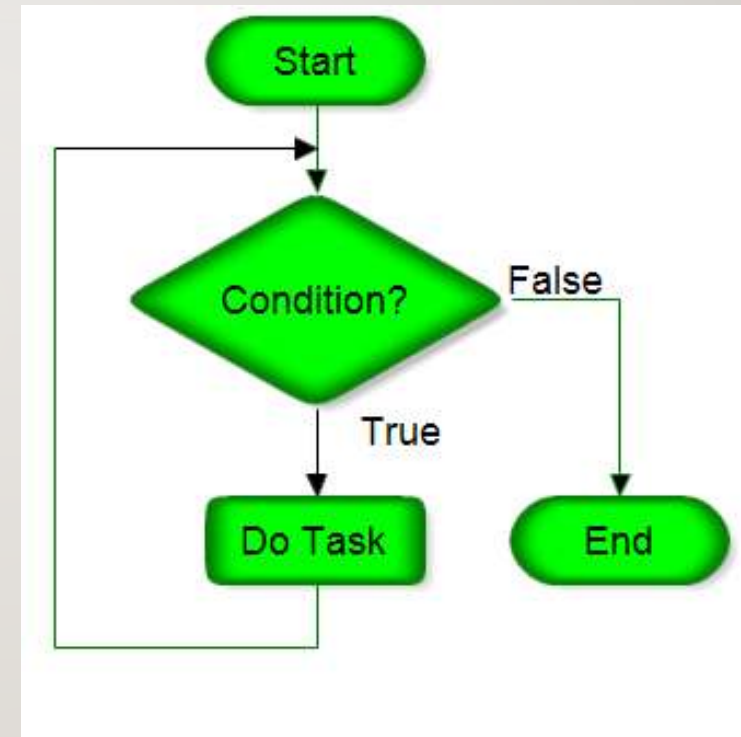
```
In [27]: a = 1  
switcher = {  
    0: " This is Case Zero ",  
    1: " This is Case One ",  
    2: " This is Case Two ",  
}  
switcher.get(a, "nothing")
```

The output of the code cell is:

```
Out[27]: ' This is Case One '
```

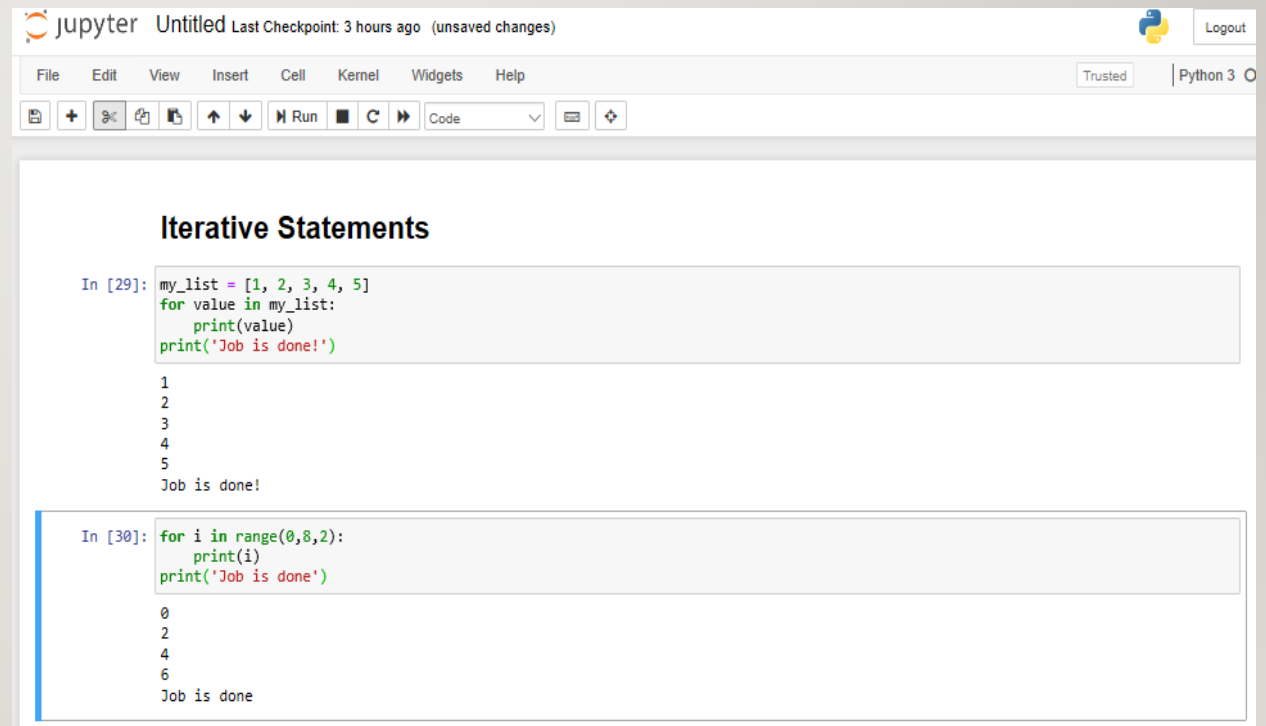

ITERATIVE STATEMENTS(I/3)

- Iterative statements are also known as the looping statements or repetitive statements.
- They are used to repeat a part of code repeatedly as long as the given condition is true.



FOR() LOOP(2/3)

- The for() loop in python is generally used to iterate through a sequence like a list, a tuple, a set, a dictionary, or a string.
- Syntax:
 for <test condition> in
 <sequence>:
 statement
 statement
 statement



The image shows a Jupyter Notebook interface with two code cells. The first cell, labeled 'In [29]:', contains a for loop that iterates over a list named 'my_list' with values [1, 2, 3, 4, 5]. The output shows the numbers 1 through 5, followed by 'Job is done!'. The second cell, labeled 'In [30]:', contains a for loop that iterates over a range from 0 to 8 with a step of 2. The output shows the numbers 0, 2, 4, 6, followed by 'Job is done!'.

```
jupyter Untitled Last Checkpoint: 3 hours ago (unsaved changes) Python 3.0
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.0

Iterative Statements

```
In [29]: my_list = [1, 2, 3, 4, 5]
for value in my_list:
    print(value)
print('Job is done!')
```

```
1
2
3
4
5
Job is done!
```

```
In [30]: for i in range(0,8,2):
        print(i)
        print('Job is done')
```

```
0
2
4
6
Job is done
```


WHILE() LOOP

- While() loop is an entry control loop which means that the condition is checked before entering in loop and running the statements.
- Syntax:
 while text condition:
 statement
 statement

A screenshot of a Jupyter Notebook interface. The top bar shows 'jupyter Untitled' with a 'Last Checkpoint: 3 hours ago (unsaved changes)' message and a 'Logout' button. Below the top bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A toolbar contains icons for file operations, running, and other notebook functions. The main area is titled 'Iterative Statements' and contains a code cell with the following Python code:

```
In [31]: count = int(input('How many times you want to say "Hello": '))
i = 1
while i <= count:
    print('Hello')
    i += 1
print('Job is done! Thank you!!')
```

The output of the code is displayed below the code cell:

```
How many times you want to say "Hello": 4
Hello
Hello
Hello
Hello
Job is done! Thank you!!
```

FUNCTIONS IN PYTHON(I/3)

- Functions are the blocks of code which are executed only when they are called.
- To create a function we use 'def' keyword.

```
def funct():  
    print('Gurvansh')
```

- To call the function we writes its name followed by parenthesis.

```
funct()
```

FUNCTIONS IN PYTHON(I/3)

- Functions are the blocks of code which are executed only when they are called.
- To create a function we use 'def' keyword.

```
def funct():  
    print('Gurvansh')
```

- To call the function we writes its name followed by parenthesis.

```
funct()
```

FUNCTIONS IN PYTHON(I/3)

- Functions are the blocks of code which are executed only when they are called.
- To create a function we use 'def' keyword.

```
def funct():  
    print('Gurvansh')
```

- To call the function we writes its name followed by parenthesis.

```
funct()
```

CONTD.(2/3)

- We can also pass data into the functions known as arguments.

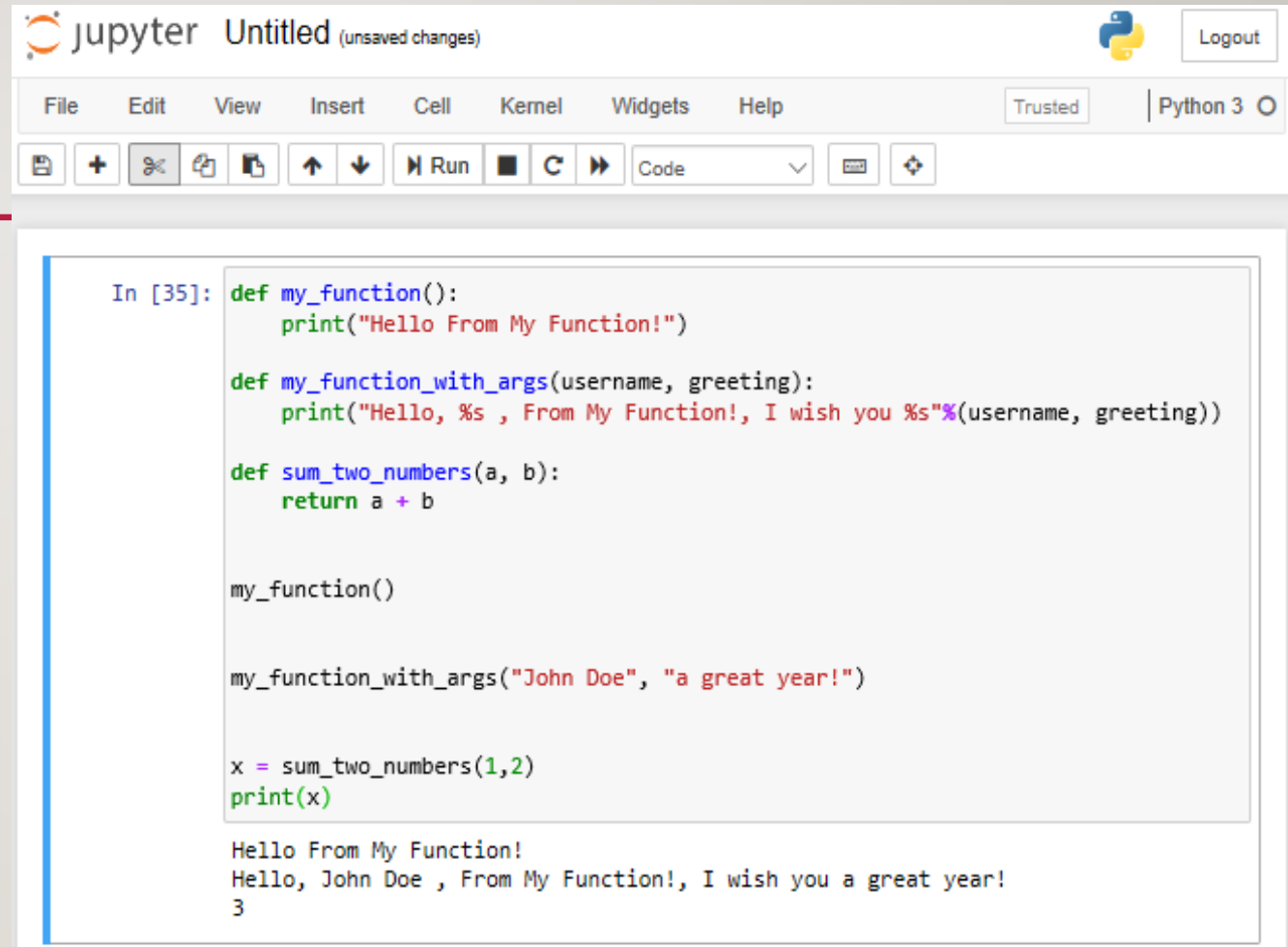
```
def my_function(fname):  
    print(fname + " Refsnes")
```

```
my_function("Emil")
```

Output:

Emil Refsnes

CONTD.(3/3)



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a code editor. The code defines three functions: `my_function`, `my_function_with_args`, and `sum_two_numbers`. It then calls `my_function`, `my_function_with_args` with arguments "John Doe" and "a great year!", and `sum_two_numbers` with arguments 1 and 2, printing the results.

```
In [35]: def my_function():
          print("Hello From My Function!")

          def my_function_with_args(username, greeting):
              print("Hello, %s , From My Function!, I wish you %s"%(username, greeting))

          def sum_two_numbers(a, b):
              return a + b

          my_function()

          my_function_with_args("John Doe", "a great year!")

          x = sum_two_numbers(1,2)
          print(x)

          Hello From My Function!
          Hello, John Doe , From My Function!, I wish you a great year!
          3
```

THANK YOU