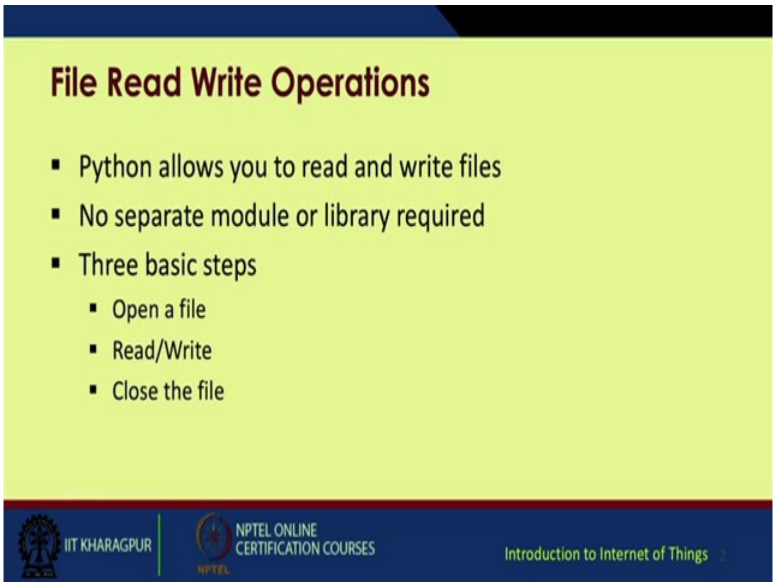
Introduction to Internet of Things

Prof. Sudip Misra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 27
Introduction to Python Programming- II

Hello, in this lecture in Continuation to Python Programming, I will be covering a few interesting applications, you can develop using normal python programming.

(Refer Slide Time: 00:34)



File Read Write Operations

- Python allows you to read and write files
- No separate module or library required
- Three basic steps
  - Open a file
  - Read/Write
  - Close the file

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet of Things

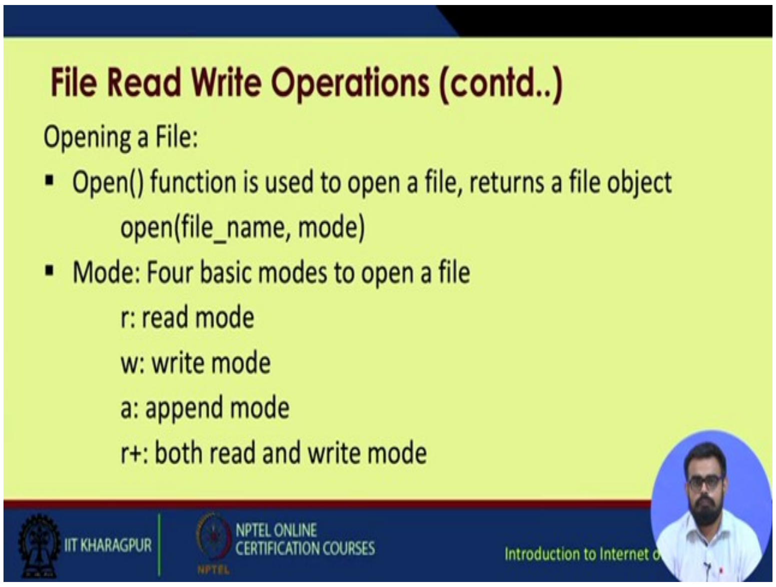
So, to start off with, we will be covering a few basic operations starting from file, read/write, operations, we will be dealing with normal text files as well as csv files. Csv are basically comma separated value files. So, each value in the file is separated by a comma.

Next we will move on to your developing a network socket using python scripting. So, we will develop two parts. First, we will develop a udp host or a server and a udp client and in the third part, we will be dealing with some basic image, read, write operations using python PIL library. So, to start off with the file read write operations as necessary in normal IoT based applications, imagine your sensors or lots and lots of sensors are uploading the data on a particular database or a file system.

So, the data is separated, separately stored or data is collectively stored using various time stamps or signatures or identifiers. It may be in a text file for small scale systems, it may be in a csv file, but larger scale systems and then, again for much larger implementations of IoT, people normally prefer using data bases like MySQL or Oracle ecetera. So, python allows us the functionality to read and write files.

Generally no external module or library is required. It is available with the basic distribution of python itself and while reading or writing a file, three basic steps have to be followed. First and foremost is opening a file itself. Next is reading or writing operation and third one is closing the file. So, keeping these three in mind, we will dive into a bit more details about these.

(Refer Slide Time: 02:31)



File Read Write Operations (contd..)

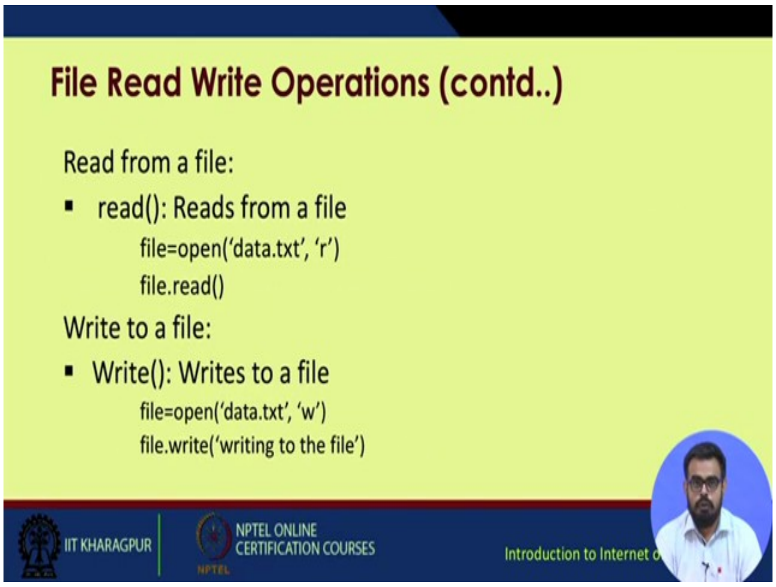
Opening a File:

- Open() function is used to open a file, returns a file object
  open(file_name, mode)
- Mode: Four basic modes to open a file
  r: read mode
  w: write mode
  a: append mode
  r+: both read and write mode

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet o

So, the open function is used to open a file and return a file object. That means, whenever you give the syntax, open first argument is filename and the second argument is mode. There are the following four modes, r for read or w for write, a for append and r plus for both read as well as write modes. Through reading from a file as you can see the syntax is pretty simple.

(Refer Slide Time: 03:01)



File Read Write Operations (contd..)

Read from a file:

- read(): Reads from a file

```
file=open('data.txt', 'r')
file.read()
```

Write to a file:

- Write(): Writes to a file

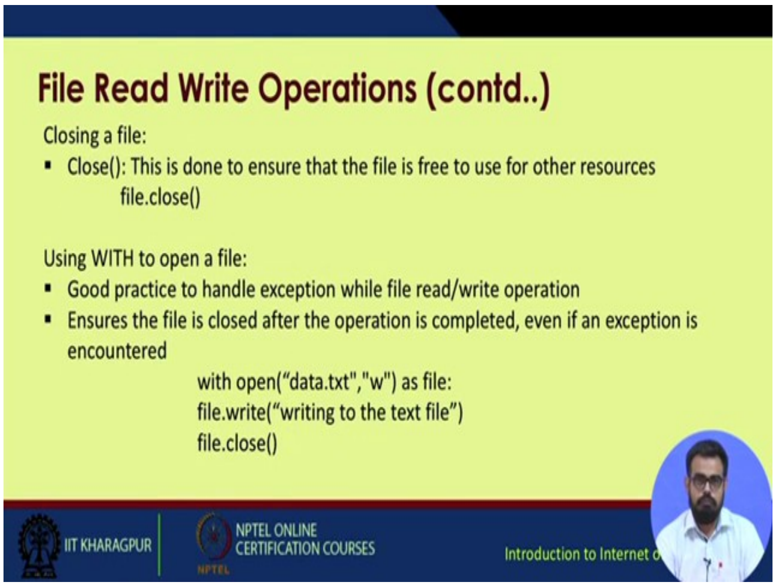
```
file=open('data.txt', 'w')
file.write('writing to the file')
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet o

So, imagine you have a file text, file named data data.txt. So, we are going to attempt to open it using this line file equal to open('data.txt' , 'r') point to notice both the filename as well as the mode are within quotations because these are taken as strings and then, in the next line we have file dot read. So, this file equal to open function, the file is actually a variable. You can assign any other name to it. Next is the writing to a file part.

So, whenever we are writing to a file, again we have to open the file. Only the mode changes from r to w, then file.write and within arguments, you just put in the string here about to write to the file you can. Obviously, put various variables you can store your incoming sensor data and some variable and iteratively write those variables using the file.write function.

(Refer Slide Time: 04:14)



File Read Write Operations (contd..)

Closing a file:

- Close(): This is done to ensure that the file is free to use for other resources
  file.close()

Using WITH to open a file:

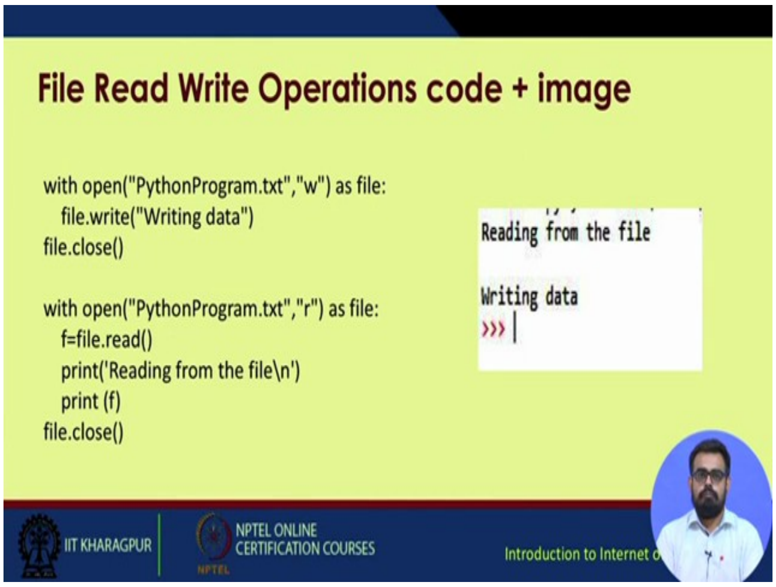
- Good practice to handle exception while file read/write operation
- Ensures the file is closed after the operation is completed, even if an exception is encountered

```
with open("data.txt","w") as file:
file.write("writing to the text file")
file.close()
```

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Introduction to Internet o

Now, at the end, once a file access has been done, you must ensure that your file handle is closed by using this function file dot close and whenever since in normal normal scenarios specially IoT based scenarios, there is a significant need to iteratively access the same file over and over again. So, using this script with open data.txt in write mode as file in file.write, we write the string which has to be written to the text file and then, we release the file handle by calling file.close().

(Refer Slide Time: 05:02)



File Read Write Operations code + image

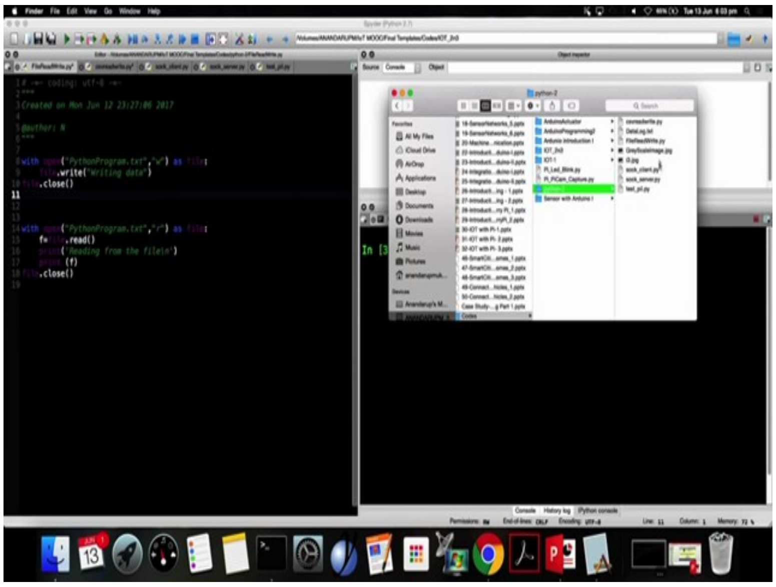
```
with open("PythonProgram.txt","w") as file:
  file.write("Writing data")
file.close()

with open("PythonProgram.txt","r") as file:
  f=file.read()
  print('Reading from the file\n')
  print (f)
file.close()
```

```
Reading from the file

Writing data
>>> |
```

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Introduction to Internet o

So, it would be better if we give a bit of hands on to this thing. So, as you can see I have already kept the program ready.

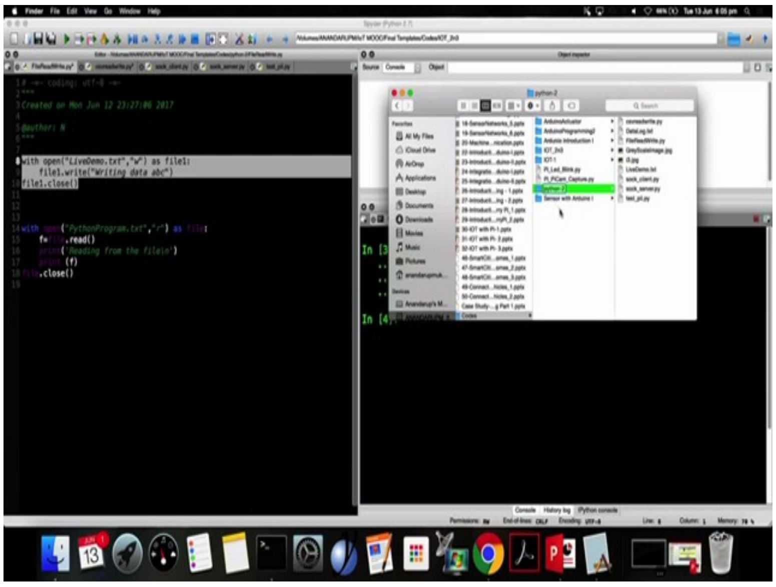
(Refer Slide Time: 05:29)



```python
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 12 23:27:06 2017

@author: N
"""

with open("PythonProgram.txt","w") as file:
    file.write("Writing data")
file.close()



with open("PythonProgram.txt","r") as file:
    f=file.read()
    print('Reading from the file\n')
    print (f)
file.close()
```

So, we just have to call a file, any file name we want to put over here. We have python program.txt. We can change it to something else also. So, my files and folders basically recite in this directory. So, as you can see there is no python program.txt file in this list. So, maybe we can just change the name to live demo.txt.

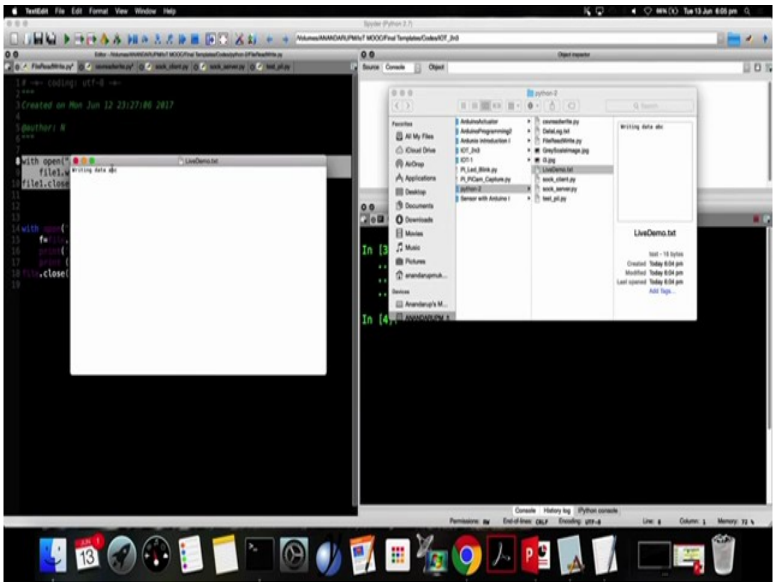
(Refer Slide Time: 06:26)



```python
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 12 23:27:06 2017

@author: N
"""

with open("LiveDemo.txt","w") as file1:
    file1.write("Writing data abc")
file1.close()



with open("PythonProgram.txt","r") as file:
    f=file.read()
    print('Reading from the file\n')
    print (f)
file.close()
```

Then, in the next line for speed, iteratively we have to write something on it. So, with open live demo.txt as w or in write mode as file, let us call it by another name file1. Then, file1.write, we are writing this writing data abc and then, we release the file handle by calling file1.close(). So, once we run this thing, we have executed this thing, these three lines of code. Now again we will check the directory.

(Refer Slide Time: 07:40)

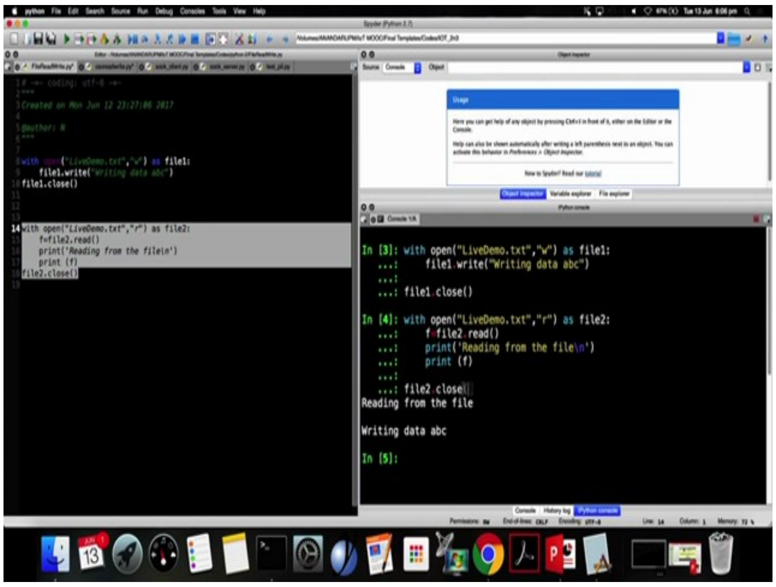


```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 12 23:27:06 2017

@author: N
"""

with open("
    file1.w
file1.close



with open('
    f=file.
    print('
    print (
file.close(
```

```
Writing data abc
```

Here we have the file named as live demo.txt. We open it as you can see the intended data has been written to the text file. So, this was part one. Half of the job it is reading that is writing to the file. Next part comes that is reading from that file.

(Refer Slide Time: 07:56)



```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 12 23:27:06 2017

@author: N
"""

with open("LiveDemo.txt","w") as file1:
    file1.write("Writing data abc")
file1.close()



with open("LiveDemo.txt","r") as file2:
    f=file2.read()
    print('Reading from the file\n')
    print (f)
file2.close()
```

```
In [3]: with open("LiveDemo.txt","w") as file1:
   ...:     file1.write("Writing data abc")
   ...:
   ...: file1.close()

In [4]: with open("LiveDemo.txt","r") as file2:
   ...:     f=file2.read()
   ...:     print('Reading from the file\n')
   ...:     print (f)
   ...:
   ...: file2.close()
Reading from the file

Writing data abc

In [5]:
```

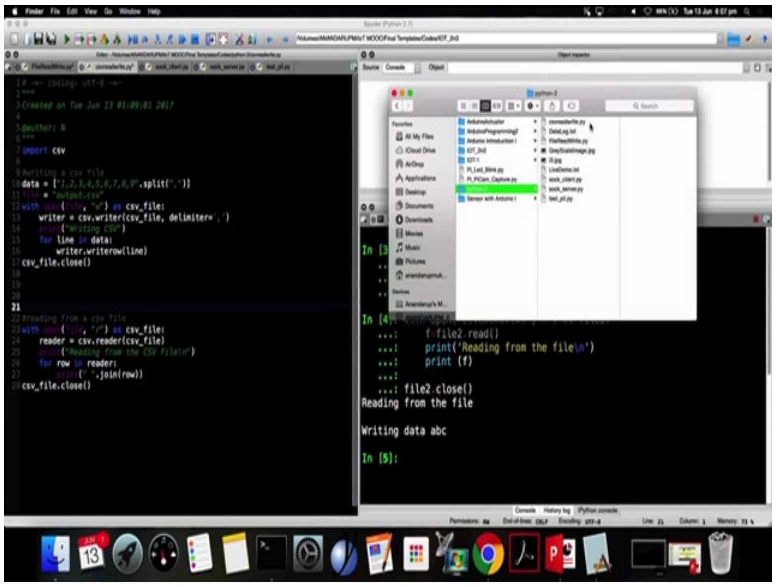
So, now I have a filename live demo.txt. So, in the same manner as before I will open the same text file, but in read mode. So, I will change the mode to r. Initial it was w, now changing it to r.

So, I am opening the file mode in read opening, the file in reading mode assigning the variable let us call file 1 or maybe just for the sake of it file2, f equal to file2.read. Then, I print reading from the file and then, again I print whatever has been read and again I release the file handle. I release this file 2 by calling file2.close(). So, once again I will execute these lines.

See once this execution is complete, the first line has been printed reading from the file. This was a first print statement and then, if you recall we had earlier written data to the text file like writing data abc. So, this has been extracted from the text file and put up and printed on the console. So, this was part one, reading and writing from normal text files. So, there can be multiple operations, we can consider while doing file read write operations.

So, not to go into that much details, you can easily avail various resources online to check those.

(Refer Slide Time: 09:50)



```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 01:09:01 2017

@author: N
"""
import csv

#writing a csv file
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file = "output.csv"
with open(file, "w") as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    print("Writing CSV")
    for line in data:
        writer.writerow(line)
csv_file.close()




#reading from a csv file
with open(file, "r") as csv_file:
    reader = csv.reader(csv_file)
    print("Reading from the CSV File\n")
    for row in reader:
        print(" ".join(row))
csv_file.close()
```

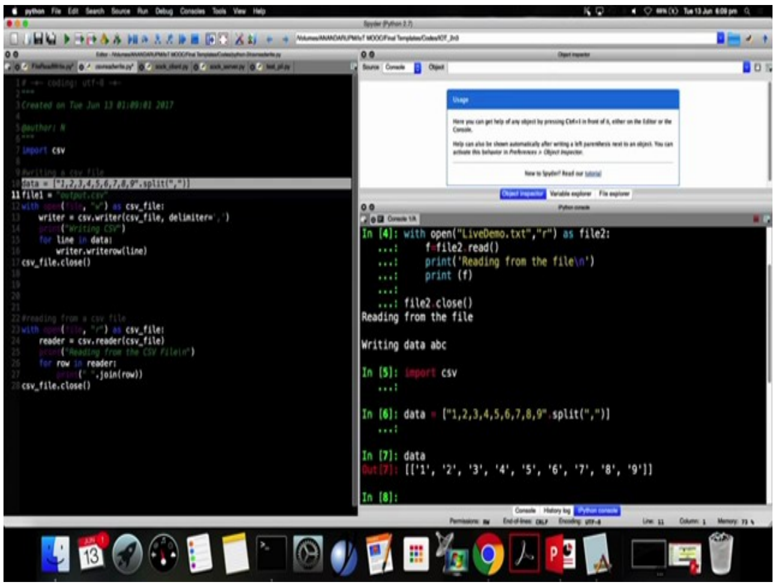
```
   ...:     f=file2.read()
   ...:     print('Reading from the file\n')
   ...:     print (f)
   ...:
   ...: file2.close()
Reading from the file

Writing data abc

In [5]:
```

Another common thing being used is this csv file format or comma separated values. You can again check I do not have any csv file in my directory list. So, I am using this script csvreadwrite.py. So, once this has been executed, it will generate a file with the extension dot csv.

For this I have to import an external library or csv library. Once this library has been imported, I define the data. I want to write to the file. First I will write to the file, then once the file has been created and the data written to it, I will read from that same file. So, one point to remember is whenever I am trying to open a file in write mode, if the file does not already exist, the system will automatically create that file, that same file.

So, I am giving the file name as output.csv. Let us name the variable as file1. The data is within since we are putting up a string data. So, we are giving 1 2 3 4 upto 9 and we split it using the comma character, you can use other characters also like semicolon, colon so on, but for the sake of demonstration, I am just using a comma. Let us see what happens when I execute this line. So, this script has been executed and I will check what is stored in this.

(Refer Slide Time: 11:37)

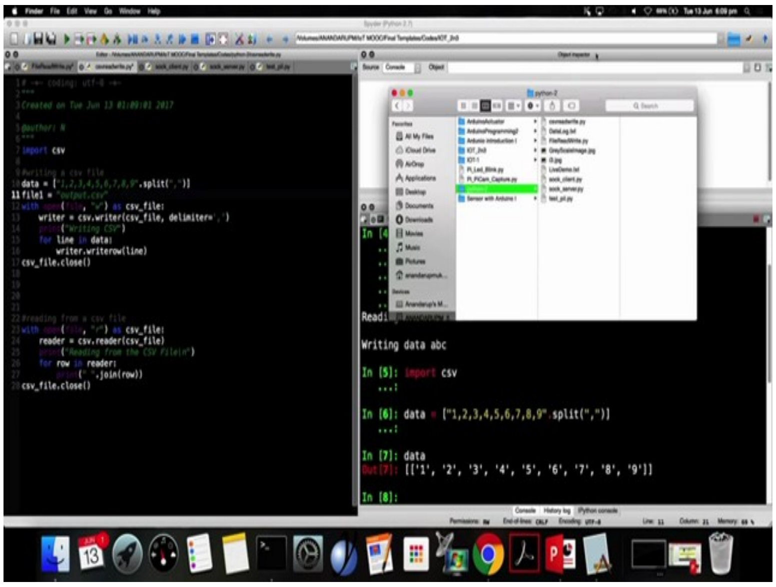


```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 01:09:01 2017

@author: N
"""
import csv

#writing a csv file
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file1 = "output.csv"
with open(file, "w") as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    print("Writing CSV")
    for line in data:
        writer.writerow(line)
csv_file.close()




#reading from a csv file
with open(file, "r") as csv_file:
    reader = csv.reader(csv_file)
    print("Reading from the CSV File\n")
    for row in reader:
        print(" ".join(row))
csv_file.close()
```

```
In [4]: with open("LiveDemo.txt","r") as file2:
   ...:     f=file2.read()
   ...:     print('Reading from the file\n')
   ...:     print (f)
   ...:
   ...: file2.close()
Reading from the file

Writing data abc

In [5]: import csv
   ...:

In [6]: data = ["1,2,3,4,5,6,7,8,9".split(",")]
   ...:

In [7]: data
Out[7]: [['1', '2', '3', '4', '5', '6', '7', '8', '9']]

In [8]:
```

So, you can see individually you have various strings separated by comma. The first string is 1, second string is 2, and so on upto 9. So, whatever input you had given, it has been individually separated into substrings. Then, next I open a I create a variable file1 and try to open output.csv. So, in this directory, we have nothing called an output.csv file.

(Refer Slide Time: 12:16)

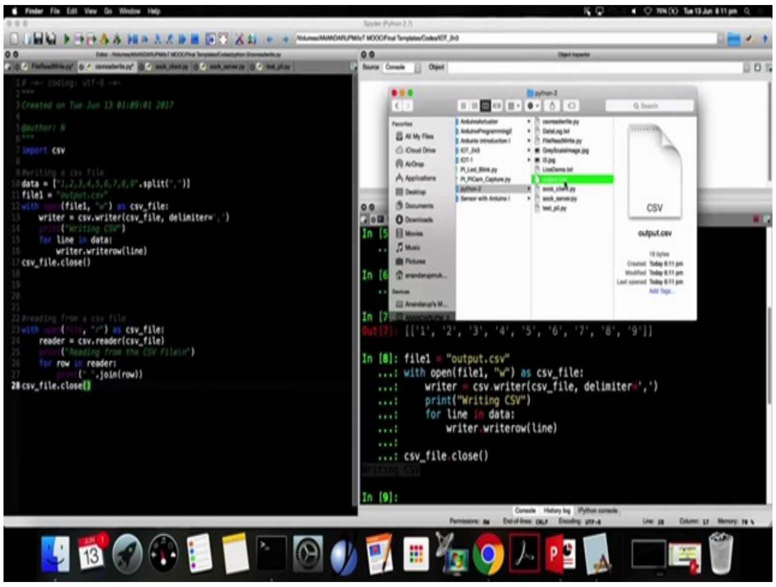


```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 01:09:01 2017

@author: N
"""
import csv

#writing a csv file
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file1 = "output.csv"
with open(file, "w") as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    print("Writing CSV")
    for line in data:
        writer.writerow(line)
csv_file.close()




#reading from a csv file
with open(file, "r") as csv_file:
    reader = csv.reader(csv_file)
    print("Reading from the CSV File\n")
    for row in reader:
        print(" ".join(row))
csv_file.close()
```

```
Writing data abc

In [5]: import csv
   ...:

In [6]: data = ["1,2,3,4,5,6,7,8,9".split(",")]
   ...:

In [7]: data
Out[7]: [['1', '2', '3', '4', '5', '6', '7', '8', '9']]

In [8]:
```

So, we are assigning this output.csv string to variable file1. Now, again using with open file1 in write mode as csv file, we call another variable writer equal to csv.writer .

(Refer Slide Time: 12:35)



```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 01:09:01 2017

@author: N
"""
import csv

#writing a csv file
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file1 = "output.csv"
with open(file1, "w") as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    print("Writing CSV")
    for line in data:
        writer.writerow(line)
csv_file.close()




#reading from a csv file
with open(file, "r") as csv_file:
    reader = csv.reader(csv_file)
    print("Reading from the CSV File\n")
    for row in reader:
        print(" ".join(row))
csv_file.close()
```

```
Out[7]: [['1', '2', '3', '4', '5', '6', '7', '8', '9']]

In [8]: file1 = "output.csv"
   ...: with open(file1, "w") as csv_file:
   ...:     writer = csv.writer(csv_file, delimiter=',')
   ...:     print("Writing CSV")
   ...:     for line in data:
   ...:         writer.writerow(line)
   ...:
   ...: csv_file.close()
Writing CSV

In [9]:
```

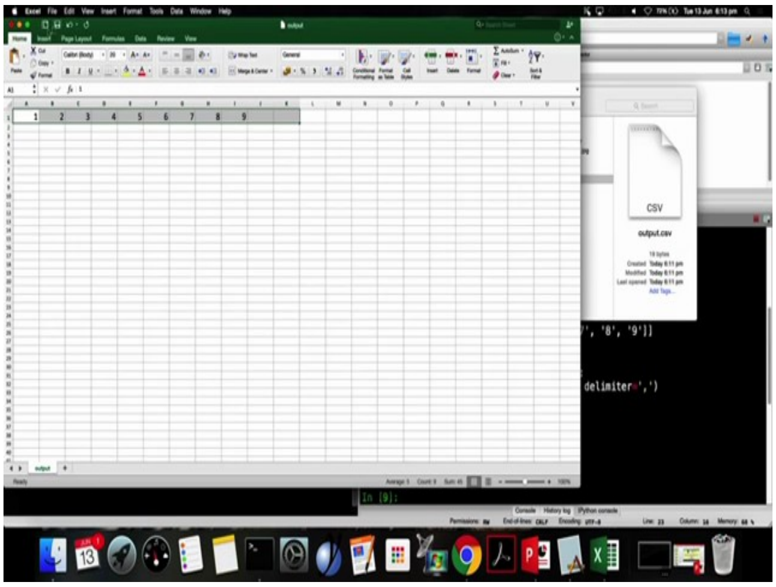
In csv file delimiter is comma. Suppose you had given a semicolon as delimiter, you could have replaced this comma by a semicolon and we print a normal statement to check whether it is working fine or not and then, whatever data is being written into the file.

So, we call the writer again and each row report in a separate value like 1 in the first row, 2 on the second row, 3 on the third row and so on. Then, eventually we call the close function file.close(). Now, let us see what happens when we run this collectively. So, my print statement writing csv has been executed. Let us check. So, the file name was output.csv . Yes we have a file called output.csv over here.

So, by default it is opening with microsoft excel. So, for my system, the default csv file viewer is microsoft excel. So, it is directly opening through excel. This will take some time. Let it load. In the meanwhile, we will look into this writing from a csv file part. Now, again we go on with open in a name variable file1 in read mode as csv file and reader equal to csv reader csv file.

We print that it is reading from a file that is we will just give an indicator to the user that this script is executing and then, for each row being read by the reader, the value being read will be printed one at a time and eventually, we will release the handle by calling the filename.close() function.

(Refer Slide Time: 15:34)

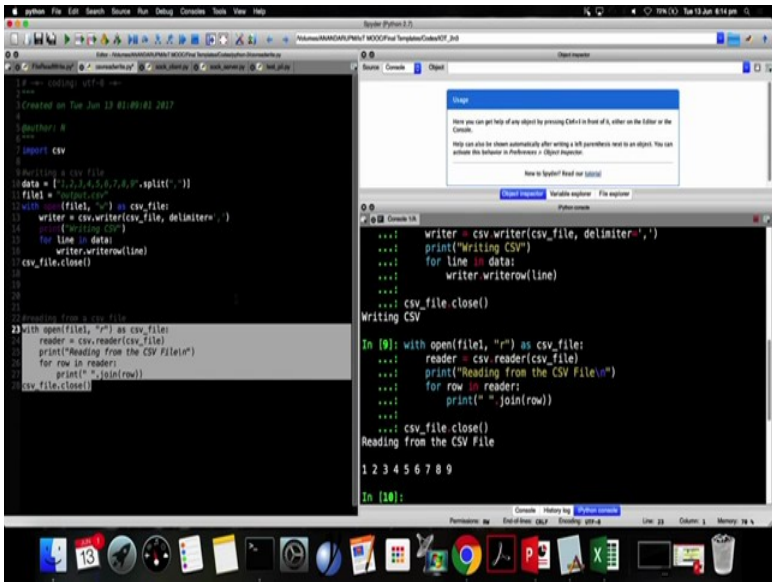


A	B	C	D	E	F	G	H	I
1	2	3	4	5	6	7	8	9

```
'8', '9']]
delimiter=',')
In [9]:
```

So, I have finally managed to open my excel file. As you can see the values 1 2 3 4 have been written in separate columns. So, instead of these static values, we can acquire data from various sensors and we can keep on updating this csv file in a iterative manner. So, this will give me a database, a rudimentary database, ok.

(Refer Slide Time: 16:28)



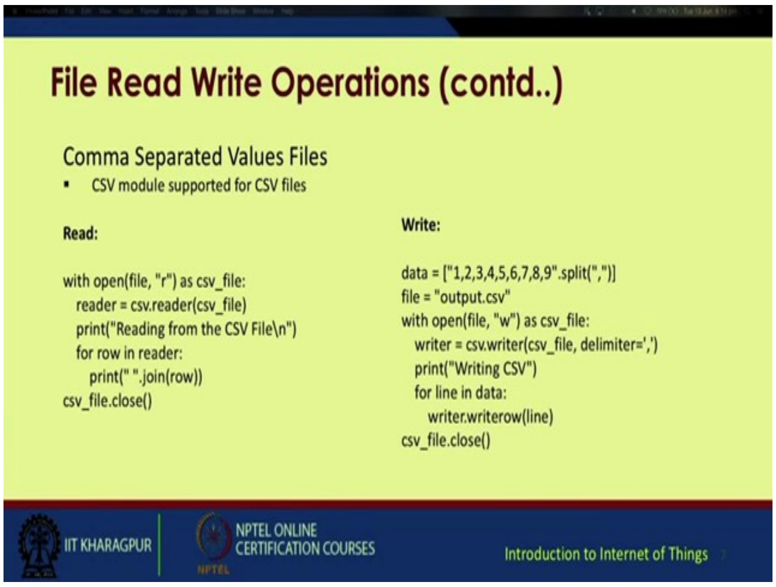
```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 01:09:01 2017

@author: N
"""
import csv

#Writing a csv file
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file1 = "output.csv"
with open(file1, "w") as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    print("Writing CSV")
    for line in data:
        writer.writerow(line)
csv_file.close()




#reading from a csv file
with open(file1, "r") as csv_file:
    reader = csv.reader(csv_file)
    print("Reading from the CSV File\n")
    for row in reader:
        print(" ".join(row))
csv_file.close()
```

```
   ...:     writer = csv.writer(csv_file, delimiter=',')
   ...:     print("Writing CSV")
   ...:     for line in data:
   ...:         writer.writerow(line)
   ...:
   ...: csv_file.close()
Writing CSV

In [9]: with open(file1, "r") as csv_file:
   ...:     reader = csv.reader(csv_file)
   ...:     print("Reading from the CSV File\n")
   ...:     for row in reader:
   ...:         print(" ".join(row))
   ...:
   ...: csv_file.close()
Reading from the CSV File

1 2 3 4 5 6 7 8 9

In [10]:
```

Now, I know my output dot csv file is in place. I will try reading from that file. Now, the first after the script has finished executing, the first print statement was reading from the csv file and it has read the values individually like 1 to 9. So, I hope this was quite easy. We will move on to the next part.

So, we have already completed these. Now, the next part is a, but interesting since it will deal with various images.

(Refer Slide Time: 17:05)



File Read Write Operations (contd..)

Comma Separated Values Files

- CSV module supported for CSV files

Read:

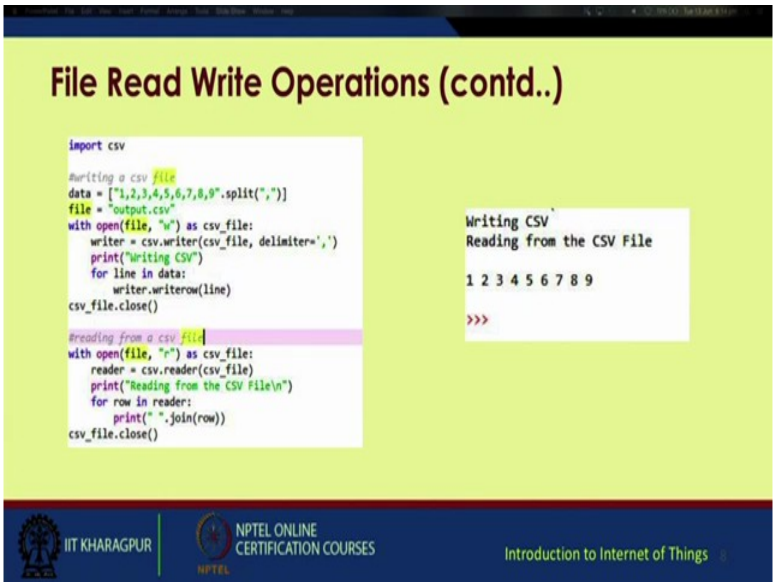
```
with open(file, "r") as csv_file:
   reader = csv.reader(csv_file)
   print("Reading from the CSV File\n")
   for row in reader:
      print(" ".join(row))
csv_file.close()
```

Write:

```
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file = "output.csv"
with open(file, "w") as csv_file:
   writer = csv.writer(csv_file, delimiter=',')
   print("Writing CSV")
   for line in data:
      writer.writerow(line)
csv_file.close()
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

(Refer Slide Time: 17:07)



File Read Write Operations (contd..)

```
import csv

#writing a csv file
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file = "output.csv"
with open(file, "w") as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    print("Writing CSV")
    for line in data:
        writer.writerow(line)
csv_file.close()

#reading from a csv file
with open(file, "r") as csv_file:
    reader = csv.reader(csv_file)
    print("Reading from the CSV File\n")
    for row in reader:
        print(" ".join(row))
csv_file.close()
```

```
Writing CSV
Reading from the CSV File

1 2 3 4 5 6 7 8 9

>>>
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

(Refer Slide Time: 17:10)

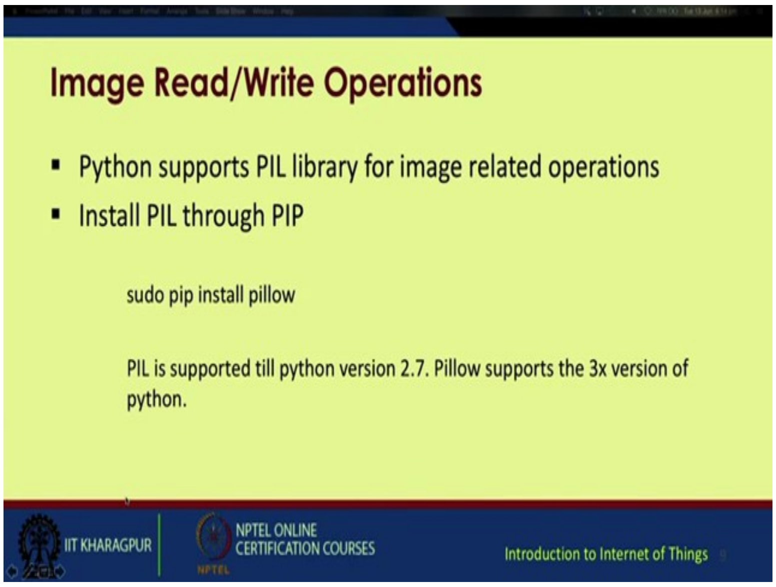


Image Read/Write Operations

- Python supports PIL library for image related operations
- Install PIL through PIP

```
sudo pip install pillow
```

PIL is supported till python version 2.7. Pillow supports the 3x version of python.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

So, we will functionally work with image read write operations. Now, python supports this library called PIL or python imaging library. This is generally used for image related operations. So, if by default your PIL is not on the system, we use PIL or we acquire PIL using sudo pip install pillow.

So, normally PIL is supported in the python version 2.7 till 2.7, otherwise for higher versions PIL needs to be installed to get this working.

(Refer Slide Time: 18:02)

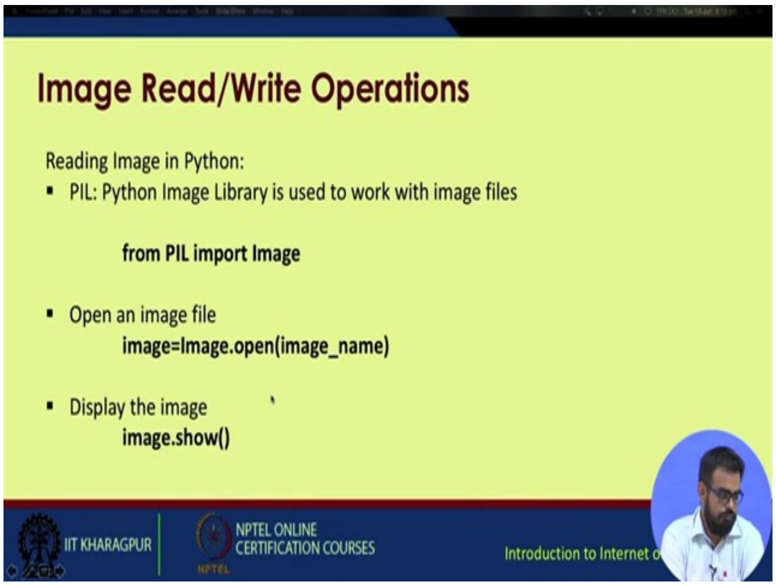


Image Read/Write Operations

Reading Image in Python:

- PIL: Python Image Library is used to work with image files

```
from PIL import Image
```

- Open an image file

```
image=Image.open(image_name)
```

- Display the image

```
image.show()
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet o

So, for image read write operations, first of all we have to call this PIL library either we import PIL directly or if we have any particular function in mind like the function image, we just import that particular function.

So, for that we have from PIL import image, then opening that image file, image equal to image.open name of the image and for displaying the image, we just write image.show()

(Refer Slide Time: 18:36)

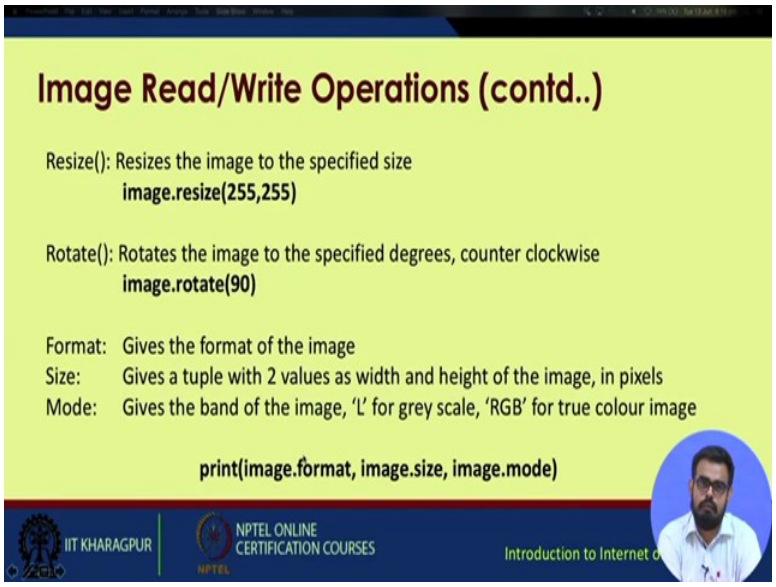


Image Read/Write Operations (contd..)

Resize(): Resizes the image to the specified size
image.resize(255,255)

Rotate(): Rotates the image to the specified degrees, counter clockwise
image.rotate(90)

Format:	Gives the format of the image
Size:	Gives a tuple with 2 values as width and height of the image, in pixels
Mode:	Gives the band of the image, 'L' for grey scale, 'RGB' for true colour image

print(image.format, image.size, image.mode)

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet o

So, you have multiple functions related to images. You want to resize your image, you call the resize function and within arguments you give the number of rows and columns you want to rotate your image.

(Refer Slide Time: 18:59)

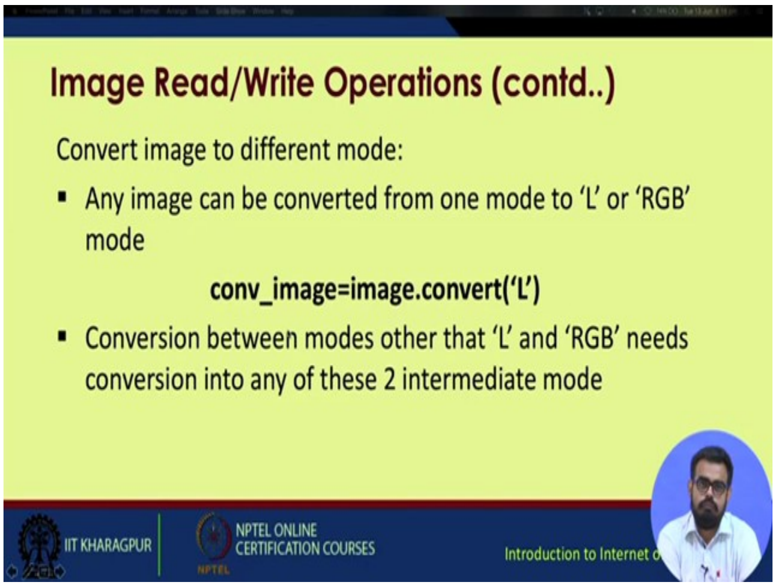


Image Read/Write Operations (contd..)

Convert image to different mode:

- Any image can be converted from one mode to 'L' or 'RGB' mode

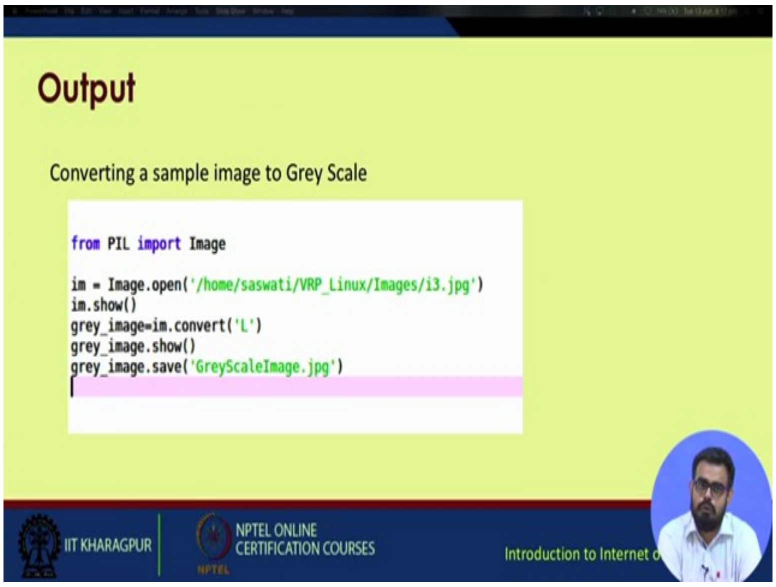
  conv_image=image.convert('L')

- Conversion between modes other that 'L' and 'RGB' needs conversion into any of these 2 intermediate mode

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet o

You call rotate and within arguments, the degrees you want to rotate your image, also you can convert between various color maps. So, for example you can convert your normal images acquired from standard cameras. We call them as RGB images because of the presence of three channels red, green and blue. We can convert this into gray scale also. So, the conversion function is pretty simple, image.convert and within the arguments within quotes, you write that mode. "L" stands for pay scale and RGB stands for the normal RGB mode. So, you can alternatively convert between these two.

(Refer Slide Time: 19:40)



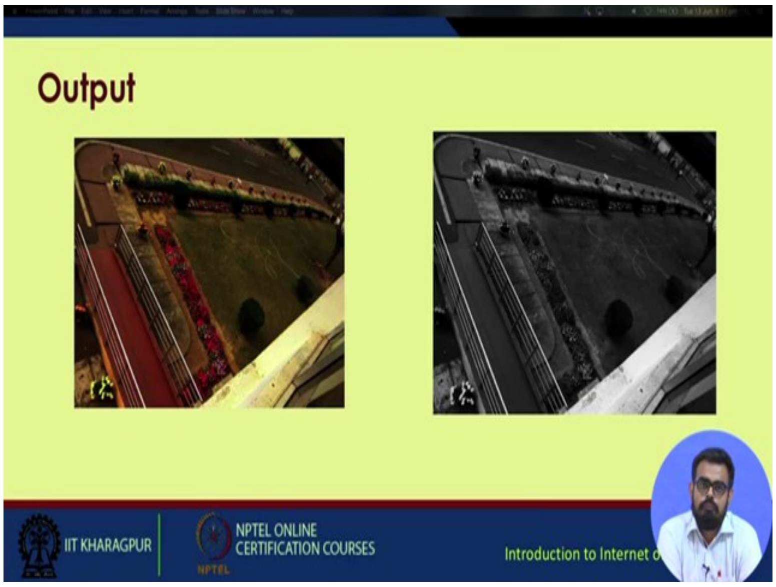
Output

Converting a sample image to Grey Scale

```
from PIL import Image

im = Image.open('/home/saswati/VRP_Linux/Images/i3.jpg')
im.show()
grey_image=im.convert('L')
grey_image.show()
grey_image.save('GreyScaleImage.jpg')
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet o

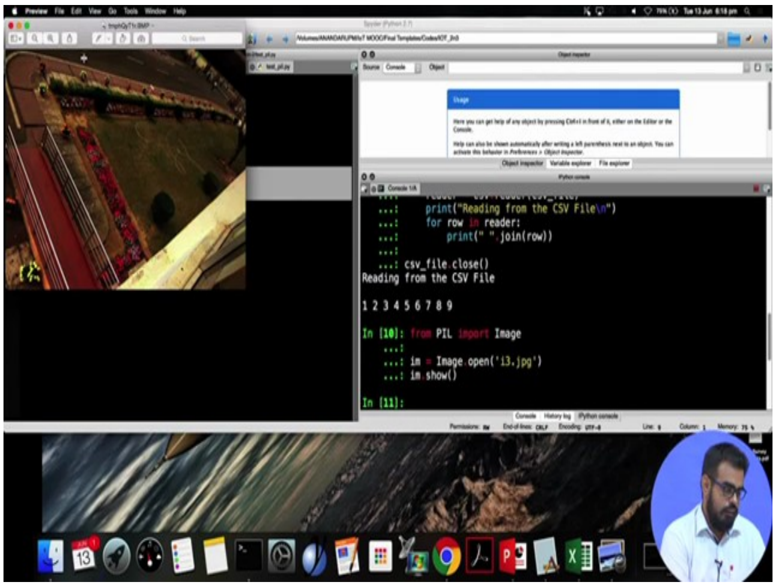
(Refer Slide Time: 19:45)


Output

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Introduction to Internet o

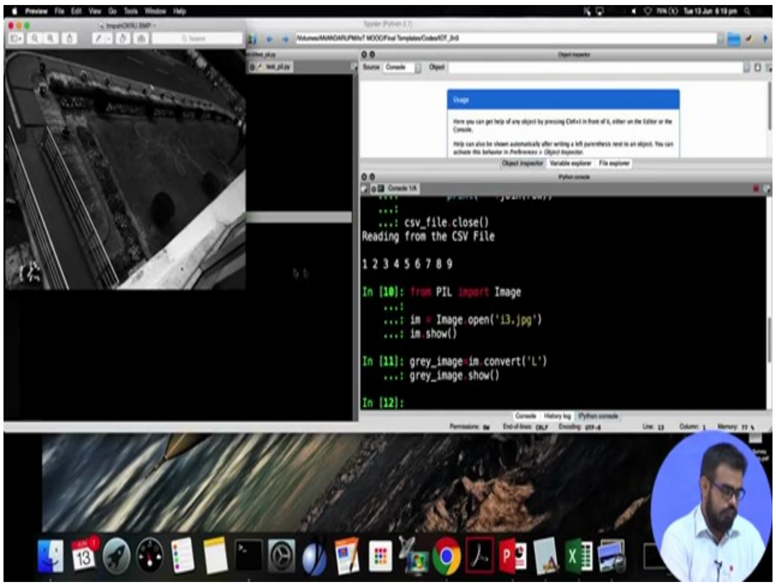
So, this is just for the sake of demonstration. See for example, once we execute, this script will come to that in the hands on part. Once we execute the script, it will give you the following output. The left hand side, this image was taken as the input image and after conversion using image.convert function, it has converted to your grey scale. So, we will check out this function.

(Refer Slide Time: 20:24)


```
   ...:     print("Reading from the CSV File\n")
   ...:     for row in reader:
   ...:         print(" ".join(row))
   ...:
   ...: csv_file.close()
Reading from the CSV File

1 2 3 4 5 6 7 8 9

In [10]: from PIL import Image
    ...:
    ...: im = Image.open('i3.jpg')
    ...: im.show()

In [11]:
```

So, now I have from PIL import image and assigning a variable to the function image.open. So, I have a input image, I3.jpeg in my folders, same folder as the script, then I execute this im dot show. I will just execute these three lines you see. Once the im dot show has been executed, the default image viewer application of your system opens up and this was this is the input image null convert it to grey scale.

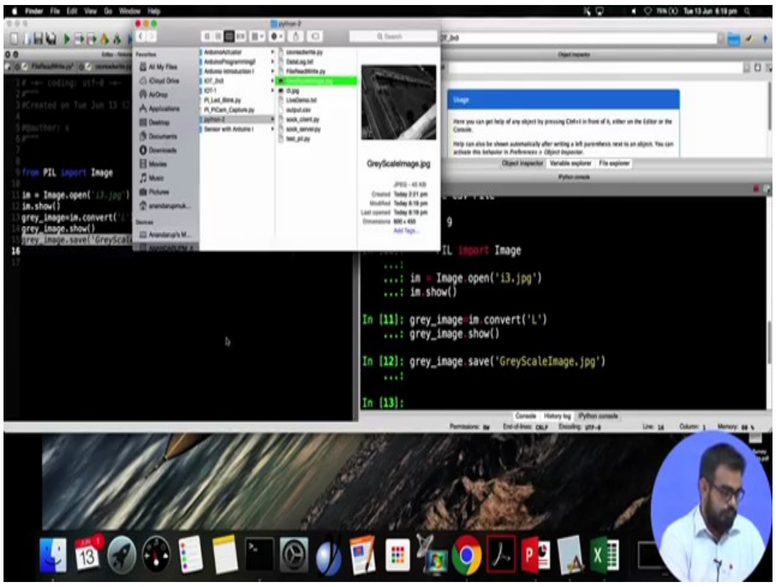
(Refer Slide Time: 21:31)



```
   ...:
   ...: csv_file.close()
Reading from the CSV File

1 2 3 4 5 6 7 8 9

In [10]: from PIL import Image
    ...:
    ...: im = Image.open('i3.jpg')
    ...: im.show()

In [11]: grey_image=im.convert('L')
    ...: grey_image.show()

In [12]:
```

So, im dot convert and within quotes l and again, I will ask the system to show the image. So, as you can see again on the default image doing application, the image has been converted to grey scale from RGB. Now, I need to save this image. So, since the converted color map of the grey scale image is stored in a variable grey_image, I need to save this grey_image.save and the name I want to save it as.

(Refer Slide Time: 22:07)

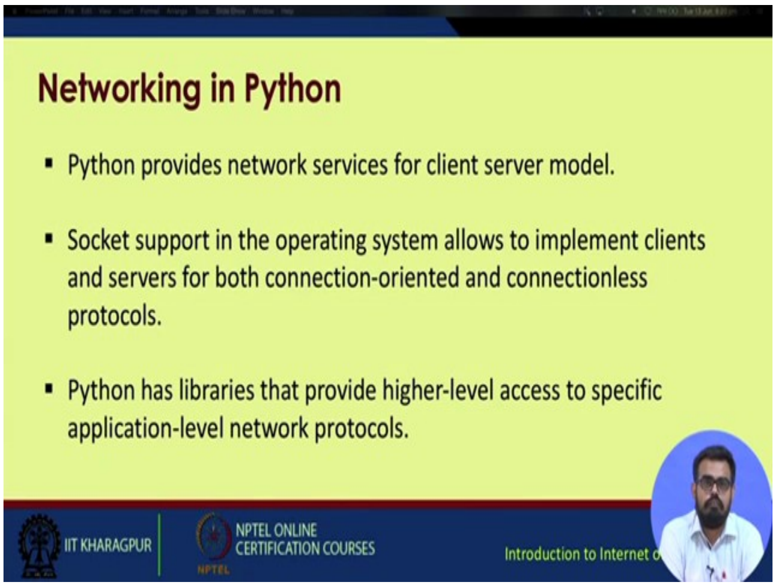


```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 1
@author: a
"""

from PIL import Image

im = Image.open('i3.jpg')
im.show()
grey_image=im.convert('L
grey_image.show()
grey_image.save('GreyScal
```

```
            9
...IL import Image
   ...:
   ...: im = Image.open('i3.jpg')
   ...: im.show()

In [11]: grey_image=im.convert('L')
    ...: grey_image.show()

In [12]: grey_image.save('GreyScaleImage.jpg')
    ...:

In [13]:
```

So, once save function has executed, I get the required file in my directory. These are just some of the very basic functions I can work with.

(Refer Slide Time: 22:29)



Networking in Python

- Python provides network services for client server model.
- Socket support in the operating system allows to implement clients and servers for both connection-oriented and connectionless protocols.
- Python has libraries that provide higher-level access to specific application-level network protocols.

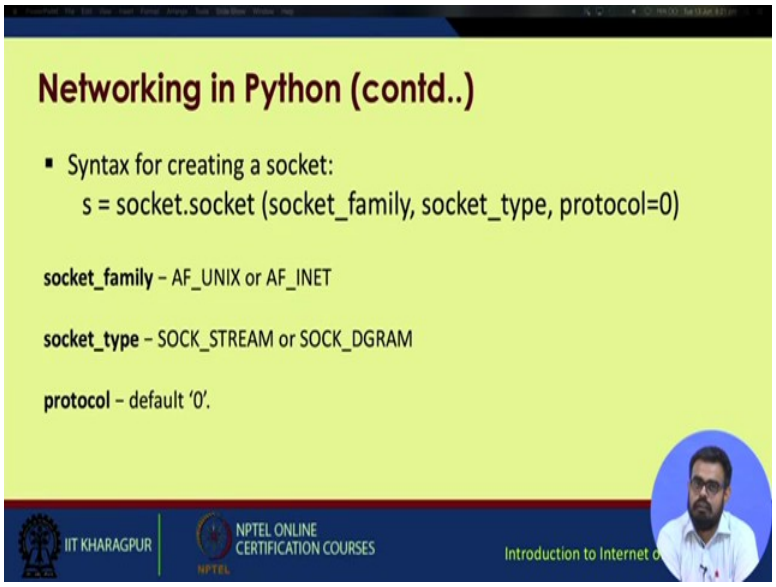
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet o

Next we will move on to basic networking in python. So, for this we will create udp sockets. So, we will try to simulate two systems like you will have a client and then, you will have a server. So, server will expect data from the client. Later on you can actually now itself you can actually imagine your client to be deployed IoT devices which read various sensorial data and transmit it over the network to the server. It works in a same principle.

So, for starters we will just see how this networking part happens. So, maybe your server is located far apart, but connected to the network using this same code, same principle. With a bit of modifications, you can get your field deployed IoT devices to send data to a remotely located server. So, python provides this network services using the client server model and then, this socket support in operating system, it allows the clients to implement servers.

It allows the implementation of clients and servers for both TCP IP that is connection oriented as well as UDP protocols which are mainly connectionless protocol and python has additionally libraries which allow higher access to specific application level network protocols.

(Refer Slide Time: 23:57)



Networking in Python (contd..)

- Syntax for creating a socket:
  s = socket.socket (socket_family, socket_type, protocol=0)

socket_family – AF_UNIX or AF_INET

socket_type – SOCK_STREAM or SOCK_DGRAM

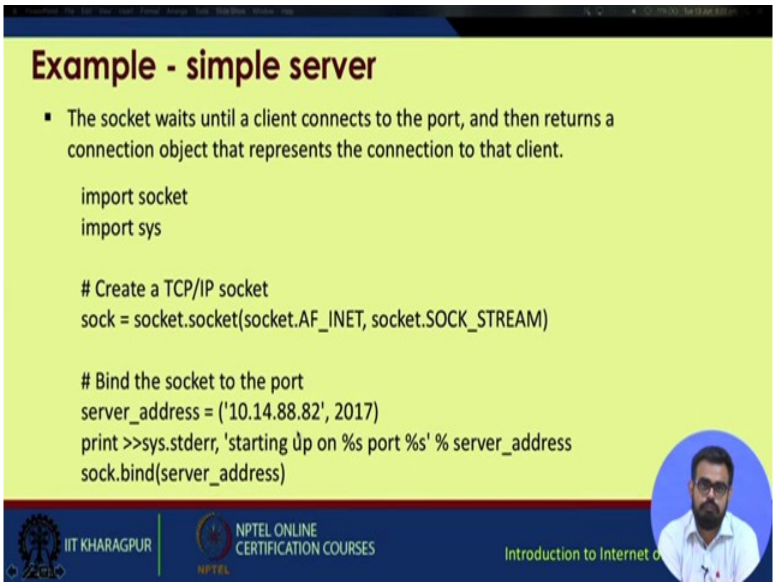
protocol – default '0'.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet o

So, general syntax is locating the socket a variable which is equal to socket dot socket then socket_family, socket_type, protocol. By default the protocol is 0. So, within socket family you have AF_UNIX or AF_INET.

So, generally UNIX option is only available for UNIX base systems, where as INET stand for all internet protocols. Socket type is SOCK_STREAM or sock datagram and protocol by default is 0.

(Refer Slide Time: 24:35)



Example - simple server

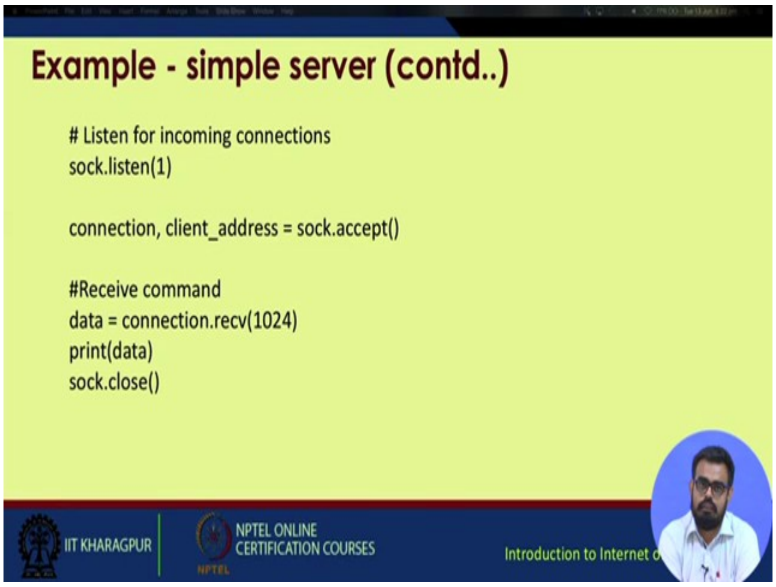
- The socket waits until a client connects to the port, and then returns a connection object that represents the connection to that client.

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
server_address = ('10.14.88.82', 2017)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)
```

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Introduction to Internet o

So, we have created a TCP/IP socket, we give the server address as this is a particular IP address, then we give a particular port to which server your client will bind and just some print statements and eventually, the main functions sock dot bind and server address, so that this will be listening for connections and then, we have sock.listen this initializes the listening part that the server is expecting connections from client.

(Refer Slide Time: 25:04)



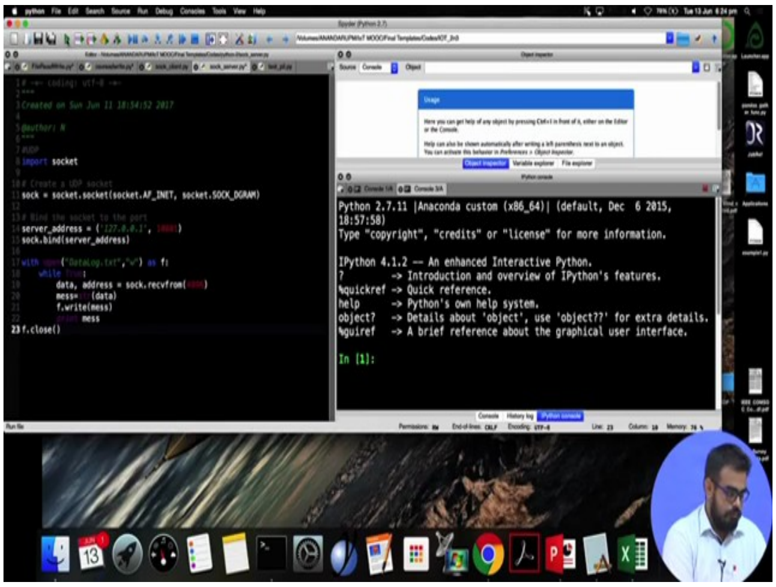
Example - simple server (contd..)

```
# Listen for incoming connections
sock.listen(1)

connection, client_address = sock.accept()

#Receive command
data = connection.recv(1024)
print(data)
sock.close()
```

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Introduction to Internet o

So, it would be better if I give a demonstration of this thing. So, I will need two consoles for this. From 1, I will run the server and from another, I will run the client.

(Refer Slide Time: 25:30)



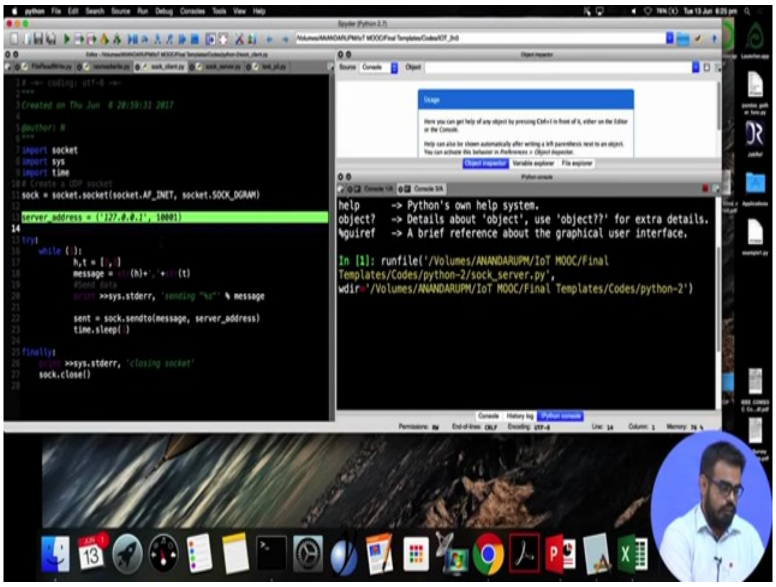
```python
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 11 18:54:52 2017

@author: R
"""
#UDP
import socket

# Create a UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind the socket to the port
server_address = ('127.0.0.1', 10001)
sock.bind(server_address)

with open("DataLog.txt","w") as f:
    while True:
        data, address = sock.recvfrom(4096)
        mess=str(data)
        f.write(mess)
        print mess
f.close()
```

```
Python 2.7.11 |Anaconda custom (x86_64)| (default, Dec  6 2015,
18:57:58)
Type "copyright", "credits" or "license" for more information.

IPython 4.1.2 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

So, this is my socket server. For the server address, I have given a normal loopback address that is 127.0.0.1

I given a random port 10000 and one and we intend to write the data which is being generated into a text file. So, data log.txt in write mode and this data socket will be expecting the data which it receives from the client in junks of 4096 bytes and whenever this data is received, it will convert the data to string and write the data to the file and eventually once the transaction has finished, it will close the file.

(Refer Slide Time: 26:51)



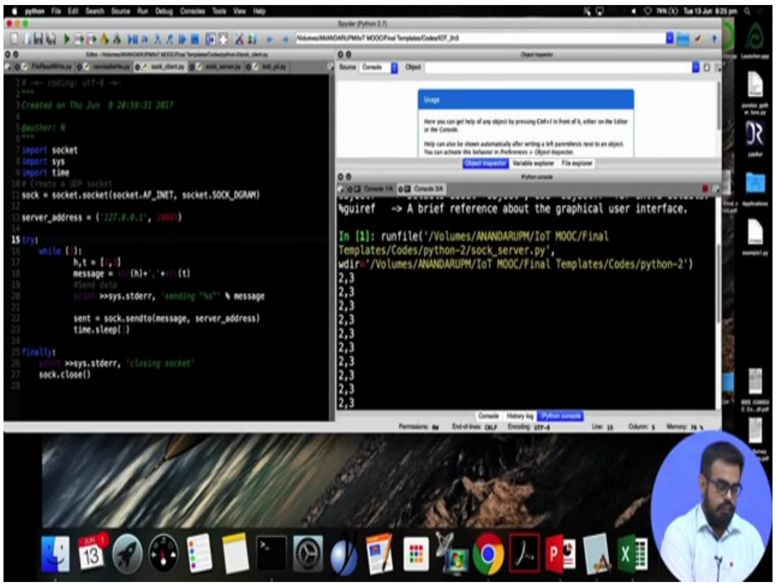
```python
# -*- coding: utf-8 -*-
"""
Created on Thu Jun  8 20:59:31 2017

@author: R
"""
import socket
import sys
import time
# Create a UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('127.0.0.1', 10001)

try:
    while (1):
            h,t = [1,1]
            message = str(h)+','+str(t)
            #Send data
            print >>sys.stderr, 'sending "%s"' % message

            sent = sock.sendto(message, server_address)
            time.sleep(1)

finally:
    print >>sys.stderr, 'closing socket'
    sock.close()
```

```
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]: runfile('/Volumes/ANANDARUPM/IoT MOOC/Final
Templates/Codes/python-2/sock_server.py',
wdir='/Volumes/ANANDARUPM/IoT MOOC/Final Templates/Codes/python-2')
```

So, till this part the socket operations take place and within this the data read write operation take place, similarly for start I will get the server running. So, it does not show anything because it is expecting data from a client. On the client side, I give the same loopback address as the server, then the same port number. It is very important to have the same port numbers.

Now, with try and finally we just iteratively loop again and again and send two values h and t. I have fixed it to 2 and 3 and this will keep on sending these values over the socket. You see this sent equal to this sock dot sent to message, the message that is 2, 3 to the server address that was this loopback address.

(Refer Slide Time: 27:42)



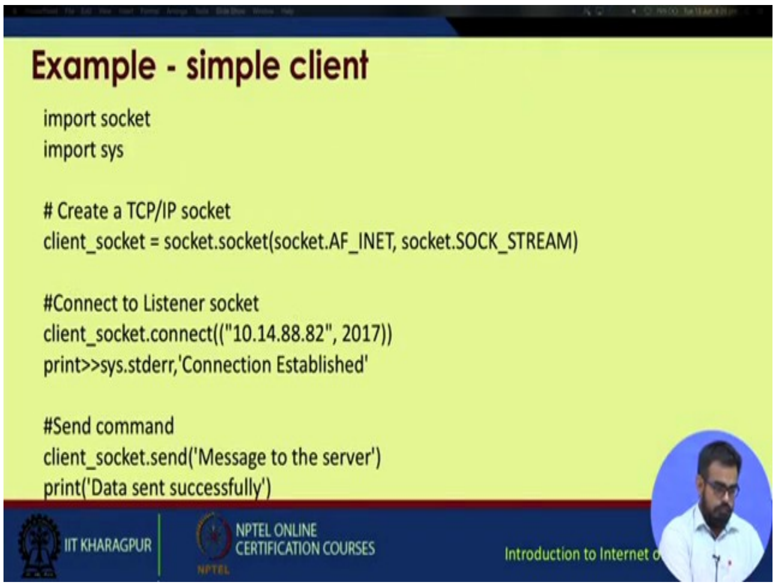
```python
# -*- coding: utf-8 -*-
"""
Created on Thu Jun  8 20:59:31 2017

@author: N
"""
import socket
import sys
import time
# Create a UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('127.0.0.1', [illegible])

try:
    while (1):
            h,t = [2,3]
            message = str(h)+','+str(t)
            #Send data
            print >>sys.stderr, 'sending "%s"' % message

            sent = sock.sendto(message, server_address)
            time.sleep(1)

finally:
    print >>sys.stderr, 'closing socket'
    sock.close()
```

```
%guiref   -> A brief reference about the graphical user interface.

In [1]: runfile('/Volumes/ANANDARUPM/IoT MOOC/Final Templates/Codes/python-2/sock_server.py', wdir='/Volumes/ANANDARUPM/IoT MOOC/Final Templates/Codes/python-2')
2,3
2,3
2,3
2,3
2,3
2,3
2,3
2,3
2,3
2,3
```

So, in the second console, I will run the client. You see the client is sending 2 3 2 3. It will keep on sending this until I terminate the execution and at the server side, you are receiving this 2 and 3.

So, this thing you can implement on two different machines connected to the network. It may be in the same room, it may be in different cities, it may be in different countries, but your server, your IP and port needs to be the same.

(Refer Slide Time: 28:19)

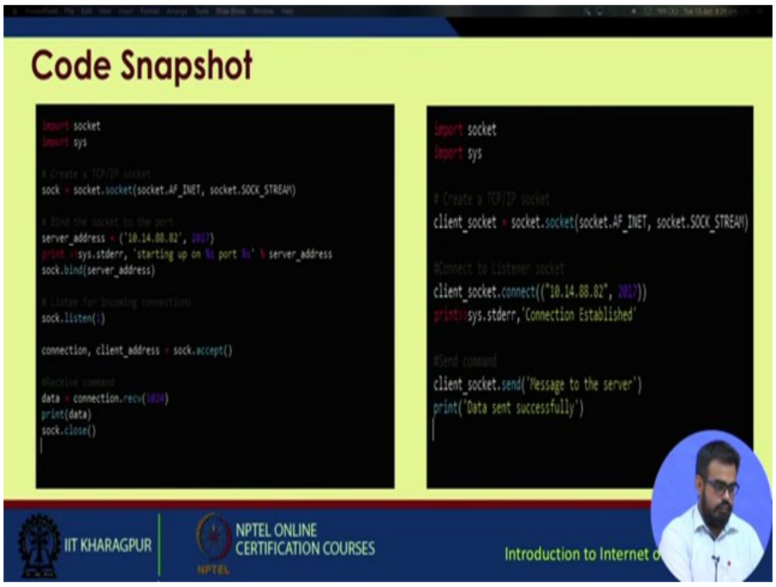


Example - simple client

```
import socket
import sys

# Create a TCP/IP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Connect to Listener socket
client_socket.connect(("10.14.88.82", 2017))
print>>sys.stderr,'Connection Established'

#Send command
client_socket.send('Message to the server')
print('Data sent successfully')
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet o

(Refer Slide Time: 28:22)



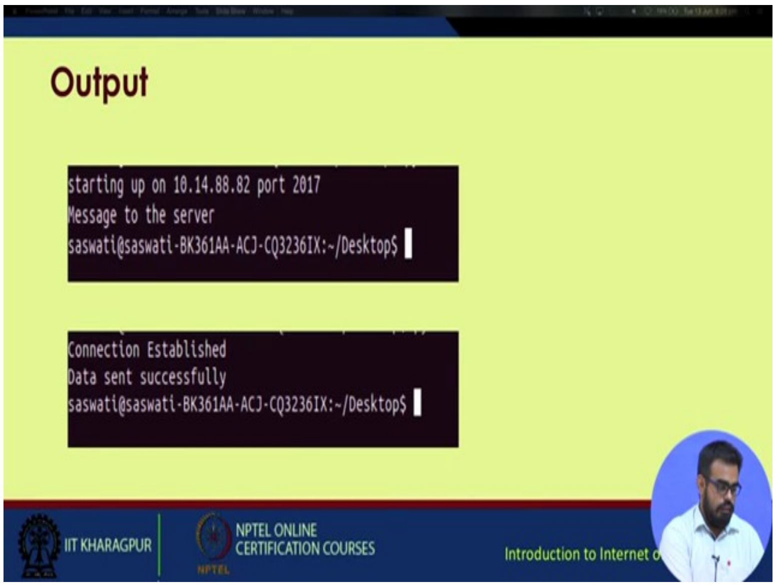
Code Snapshot

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
server_address = ('10.14.88.82', 2017)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)

# Listen for incoming connections
sock.listen(1)

connection, client_address = sock.accept()

#Receive command
data = connection.recv(1024)
print(data)
sock.close()
```

```
import socket
import sys

# Create a TCP/IP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Connect to Listener socket
client_socket.connect(("10.14.88.82", 2017))
print>>sys.stderr,'Connection Established'

#Send command
client_socket.send('Message to the server')
print('Data sent successfully')
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet o

(Refer Slide Time: 28:25)



Output

```
starting up on 10.14.88.82 port 2017
Message to the server
saswati@saswati-BK361AA-ACJ-CQ3236IX:~/Desktop$
```

```
Connection Established
Data sent successfully
saswati@saswati-BK361AA-ACJ-CQ3236IX:~/Desktop$
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Introduction to Internet o

So, we have covered this client. You have seen the code for stand and outputs you have seen. So, I hope you can now use this knowledge to generate some interesting applications on python.

Thank you.