

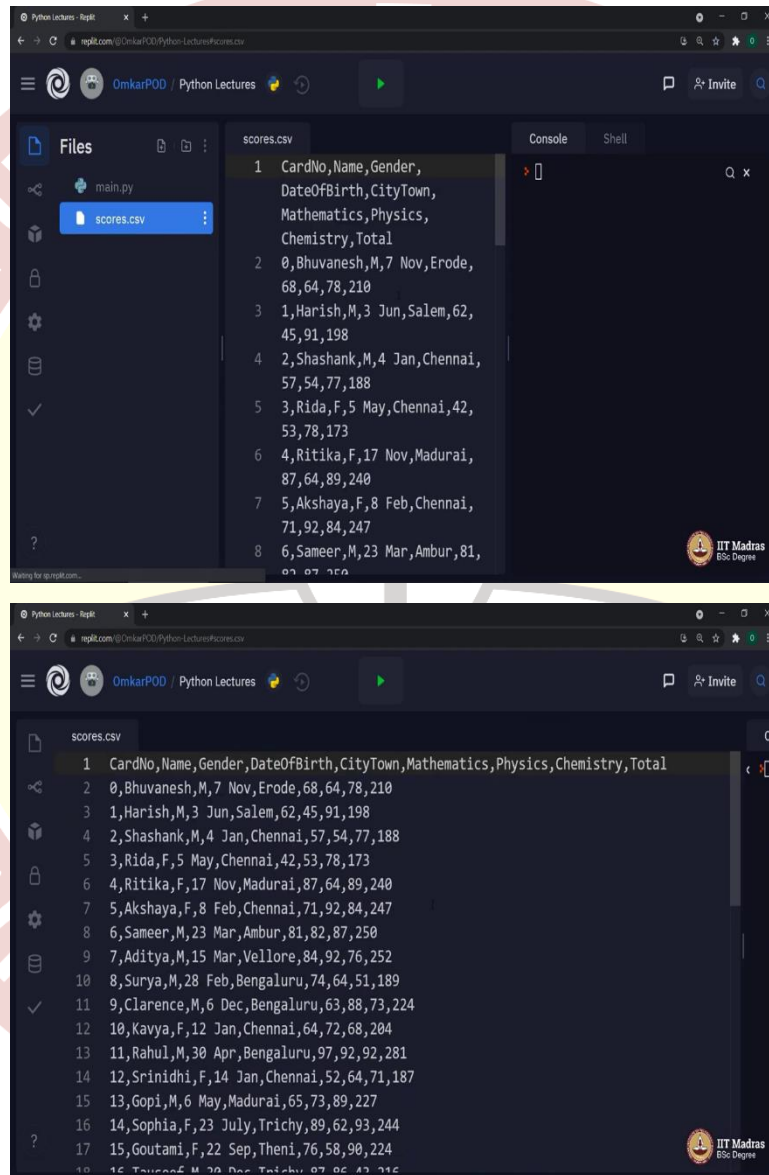


# IIT Madras

ONLINE DEGREE

**Programming in Python**  
**Professor. Sudarshan Iyengar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Ropar**  
**Mr. Omkar Joshi**  
**Course Instructor**  
**Indian Institute of Technology, Madras**  
**Why Pandas**

(Refer Slide Time: 0:16)

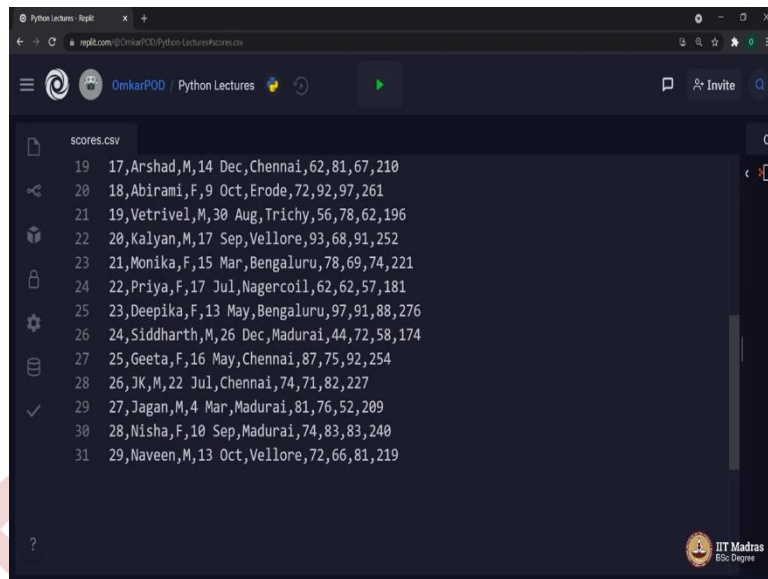


The first screenshot shows the first 8 rows of the CSV file:

	CardNo	Name	Gender	DateOfBirth	CityTown	Mathematics	Physics	Chemistry	Total
0	Bhuvanesh	M	7 Nov	Erode	68	64	78	210	
1	Harish	M	3 Jun	Salem	62	45	91	198	
2	Shashank	M	4 Jan	Chennai	57	54	77	188	
3	Rida	F	5 May	Chennai	42	53	78	173	
4	Ritika	F	17 Nov	Madurai	87	64	89	240	
5	Akshaya	F	8 Feb	Chennai	71	92	84	247	
6	Sameer	M	23 Mar	Ambur	81	82	87	250	

The second screenshot shows the first 16 rows of the CSV file:

	CardNo	Name	Gender	DateOfBirth	CityTown	Mathematics	Physics	Chemistry	Total
0	Bhuvanesh	M	7 Nov	Erode	68	64	78	210	
1	Harish	M	3 Jun	Salem	62	45	91	198	
2	Shashank	M	4 Jan	Chennai	57	54	77	188	
3	Rida	F	5 May	Chennai	42	53	78	173	
4	Ritika	F	17 Nov	Madurai	87	64	89	240	
5	Akshaya	F	8 Feb	Chennai	71	92	84	247	
6	Sameer	M	23 Mar	Ambur	81	82	87	250	
7	Aditya	M	15 Mar	Vellore	84	92	76	252	
8	Surya	M	28 Feb	Bengaluru	74	64	51	189	
9	Clarence	M	6 Dec	Bengaluru	63	88	73	224	
10	Kavya	F	12 Jan	Chennai	64	72	68	204	
11	Rahul	M	30 Apr	Bengaluru	97	92	92	281	
12	Srinidhi	F	14 Jan	Chennai	52	64	71	187	
13	Gopi	M	6 May	Madurai	65	73	89	227	
14	Sophia	F	23 July	Trichy	89	62	93	244	
15	Goutami	F	22 Sep	Theni	76	58	90	224	



The screenshot shows a Jupyter Notebook interface with a file named 'scores.csv' open. The file contains 31 rows of data, each representing a student's scores. The data is as follows:

Index	Card Number	Name	Gender	Date of Birth	City	Score 1	Score 2	Score 3	Total
19	17	Arshad	M	14 Dec	Chennai	62	81	67	210
20	18	Abirami	F	9 Oct	Erode	72	92	97	261
21	19	Vetrivel	M	30 Aug	Trichy	56	78	62	196
22	20	Kalyan	M	17 Sep	Vellore	93	68	91	252
23	21	Monika	F	15 Mar	Bengaluru	78	69	74	221
24	22	Priya	F	17 Jul	Nagercoil	62	62	57	181
25	23	Deepika	F	13 May	Bengaluru	97	91	88	276
26	24	Siddharth	M	26 Dec	Madurai	44	72	58	174
27	25	Geeta	F	16 May	Chennai	87	75	92	254
28	26	JK	M	22 Jul	Chennai	74	71	82	227
29	27	Jagan	M	4 Mar	Madurai	81	76	52	209
30	28	Nisha	F	10 Sep	Madurai	74	83	83	240
31	29	Naveen	M	13 Oct	Vellore	72	66	81	219

Hello, python students. In this lecture, we will discuss about Pandas. Pandas is an external library, which provides high performance data manipulation and analysis tool. We all are aware about the popularity of python in the field of data analytics and data science. And one of the most important reason is Pandas. It makes the python programming language even more powerful.

Therefore, in this lecture, I will try to demonstrate the power of Pandas using some examples. In order to do so, let us consider one input file. I have added one file called scores dot CSV. This file holds the exact data set, which we all know from computational thinking course. If you remember, initially, we started with cards. Later on, we converted those cards into a table, and every column in the table had a specific heading.

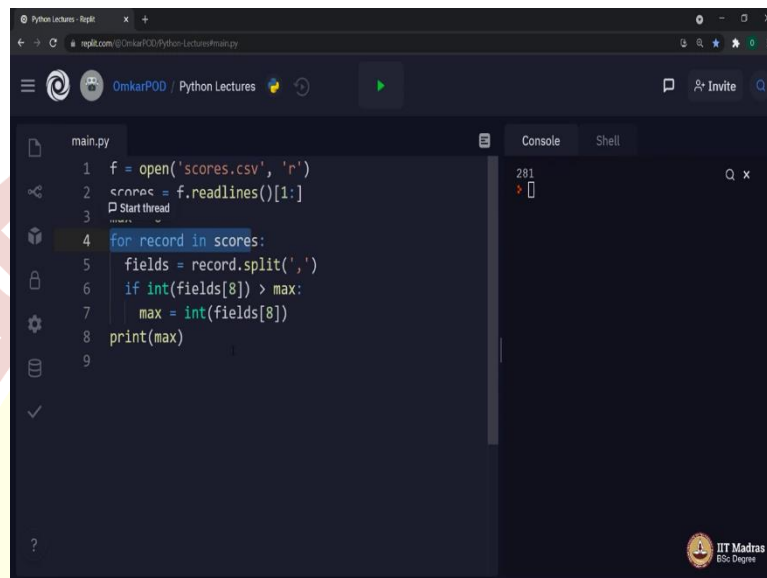
Those headings are given over here in the first row of this particular file, card number, name, gender, date of birth and so on till total. Whereas, from line number two onwards we have our actual data from scores data set. And if you remember, we had 30 such cards, that is where the data is from 0 to 29.

Apart from that, if you have noticed, all the values in this particular data are separated using comma, irrespective of whether we are referring to the column headings or the actual values in the data set every single value is separated using a comma. And that is how data is stored in CSV file. CSV stands for comma separated values. This is one of the most common and popular file type to store most of the data sets.

Now before exploring the usefulness of Pandas, first, let us try to process this file using file handling functions like open and read line. Then only based on the comparison between file

handling approach and Pandas, we will be in the better position to derive a conclusion, which is better. So, let us consider one problem statement, which we have seen earlier many times. What are the total marks of the topper from this scores data set? Let us try to write this Python program using file handling functions.

(Refer Slide Time: 3:28)



```
main.py
1 f = open('scores.csv', 'r')
2 scores = f.readlines()[1:]
3
4 for record in scores:
5     fields = record.split(',')
6     if int(fields[8]) > max:
7         max = int(fields[8])
8 print(max)
9
```

Console

```
281
```

Let us look at this code, F is equal to open scores dot CSV in the read mode. We all know, what this open function is then scores is equal to F dot read lines, as we know, this function will read all the lines from the file at a time and after that, we have to do something like 1 colon.

This particular thing is nothing, but our slice operation, which we have done earlier using list, and it is required because we have to skip the first line from the file which holds all those field names because we will not find any marks in that particular line. Hence, the processing will start from second line onwards.

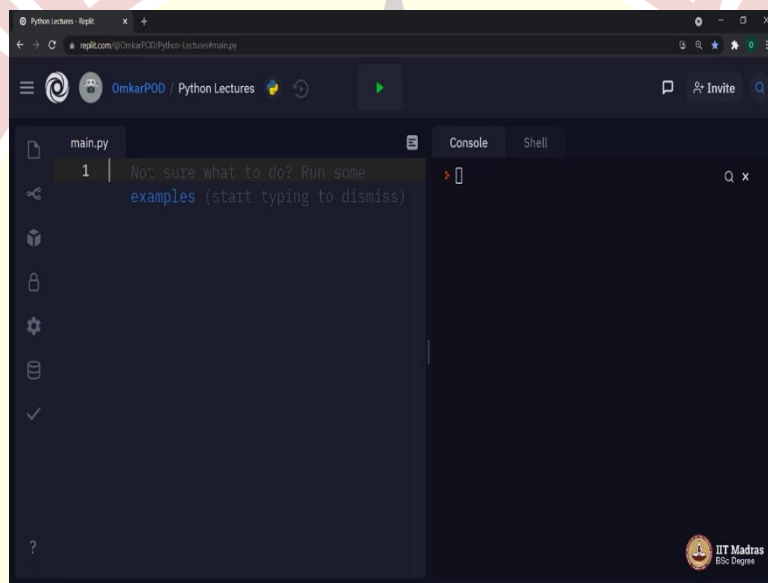
Then max is equal to 0 for every record in scores, which means, the variable record will hold one line from the file at a time. Then we have to split this record using split function over commas in order to get the last value in every single line. Now fields, is a list of all such values. And as there are total 9 fields in the data set, that is why the last field value can be extracted using index 8.

Therefore, int of fields of 8 greater than max. And if that is true, we will update the variable max to int of fields of 8. We will repeat this procedure for every single line in that file. And at the end, we will train the max variable value, which should hold total marks of the topper

student. Let us execute and see. And as expected, he got the result at 281. Now you must be thinking, what is so special about this code.

This is something we have seen with file handling and there is nothing new about it. And if we are supposed to learn Pandas, then why we have not even started with it. And the reason is, look at this particular code. First, it requires 8 lines of code. And on top of that, there are so many different complicated things like this slicing, this splitting then specific index in that particular list a for loop, and so on. Now, we will go to Pandas and try to implement same example in more simplified manner.

(Refer Slide Time: 6:48)



Pandas is also a library hence, we have to import it just like all other libraries, which we have seen earlier like random, calendar, math and so on. But the difference between those libraries and pandas is pandas is an external library, whereas, all those libraries were part of python programming language.

Now what does this mean? This means, Pandas is not part of python language is an external entity, which can be added into python, and then only we can use it. Now you must be thinking how to add Pandas into python? Is it too complicated? Do not worry, there is nothing complicated about it. In fact, you do not even have to do anything additional in order to add Pandas into python. Let us see how it is done.

(Refer Slide Time: 7:56)

The image displays three sequential screenshots of a Replit Python environment, illustrating the process of installing the pandas library using poetry.

**Top Screenshot:** The code editor shows a file named `main.py` with the following content:

```
1 import pandas as pd
2 print('Done!')
3
```

The console output shows the command `--> python3 -m poetry init --no-interaction --name repl_python3_Python-Lectures` being executed. The output indicates that the command will guide the user through creating a `pyproject.toml` config. Below this, a list of package specification formats is provided:

- A single name (requests)
- A name and a constraint (requests ^2.23.0)
- A git url (git+https://github.com/python-poetry/poetry.git)
- A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
- A file path (../my-package/my-package.whl)
- A directory (../my-package/)
- An url (https://example.com/packages/my-package-0.1.0.tar.gz)

The command `--> python3 -m poetry add pandas` is entered in the console.

**Middle Screenshot:** The console output shows the command `--> python3 -m poetry add pandas` being executed. The output indicates that the command will guide the user through creating a `pyproject.toml` config. Below this, a list of package specification formats is provided:

- A single name (requests)
- A name and a constraint (requests ^2.23.0)
- A git url (git+https://github.com/python-poetry/poetry.git)
- A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
- A file path (../my-package/my-package.whl)
- A directory (../my-package/)
- An url (https://example.com/packages/my-package-0.1.0.tar.gz)

The command `--> python3 -m poetry add pandas` is entered in the console. The output shows the command being executed and the version of pandas being installed.

**Bottom Screenshot:** The code editor shows the same `main.py` file. The console output shows the command `--> python3 -m poetry add pandas` being executed. The output indicates that the command will guide the user through creating a `pyproject.toml` config. Below this, a list of package specification formats is provided:

- A single name (requests)
- A name and a constraint (requests ^2.23.0)
- A git url (git+https://github.com/python-poetry/poetry.git)
- A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
- A file path (../my-package/my-package.whl)
- A directory (../my-package/)
- An url (https://example.com/packages/my-package-0.1.0.tar.gz)

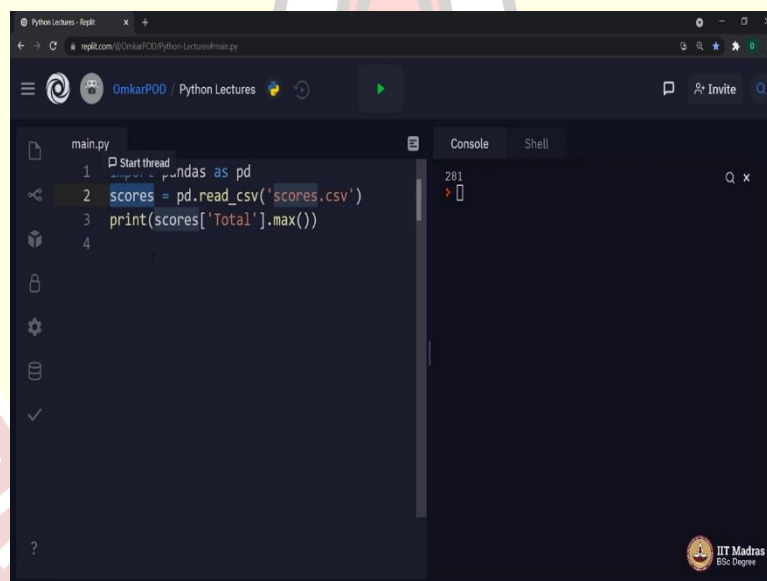
The command `--> python3 -m poetry add pandas` is entered in the console. The output shows the command being executed and the version of pandas being installed. The console output shows the command being executed and the version of pandas being installed.

Look at this particular code, import pandas as PD. We all know, this is one of the ways to import a library and then is simply printing one statement, which will indicate that the process is completed. Before I execute this particular code, let me tell you make sure that you are observing the console on the right-hand side of your screen.

Let me execute it. You must have observed python implemented multiple steps in order to install this pandas inside your repl.it and make it available to you. Usually, these installation steps take 10 to 20 seconds depending on your internet speed. Once that is done, now we are all set to use pandas. And as I said, you do not have to worry at all about how to install pandas in Replit.

As you must have seen it is a very straightforward and simple process. Now as the pandas are ready, now let us jump to the program, which we are supposed to execute to find the total marks of the topic students.

(Refer Slide Time: 9:54)

A screenshot of a Replit Python environment. The code editor on the left shows a file named 'main.py' with the following code:

```
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 print(scores['Total'].max())
4
```

The right-hand side of the interface shows the 'Console' tab with the output '281'. The Replit logo and 'OmkarPOD / Python Lectures' are visible at the top. A small IIT Madras logo is in the bottom right corner of the code editor area.

Observe this code. Import pandas as pd. There is nothing new about it, then scores is equal to pd dot read underscore CSV and the name of the file scores dot CSV. This particular function read underscore CSV is used to read a CSV file and store the values into a variable scores. After this, you have written something like scores of total dot max. At this point of time, all these things are new and may even look odd to all of you. But do not worry, we will look into all these details and all these features of pandas in detail in the next lecture.

The goal of this lecture is to demonstrate the power of pandas. Hence, we will not go into details of all these things. We will discuss about all these details and various different

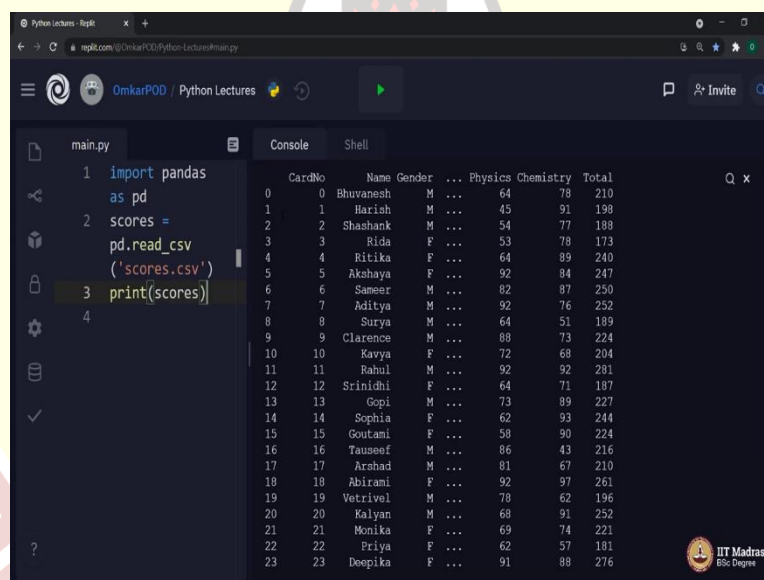


features of pandas in next lecture or so. As of now, do not go in any of these details, which you are currently seeing in these 3 lines of python program. Just observe the output and see how it is easy using pandas to execute some python programs, which were very difficult before this.

Let us execute. As expected, we got output 281. Now if you remember, the code which we wrote earlier using file handling functions it was 8 lines code. Now, we are able to same in 3 lines. And if you observe this very simple and straightforward code, first line for import, second line for read and third line for print, that is it.

Now you must be thinking, is this the only thing we can do using pandas or are there any similar things, which we can do so quickly and so easily. So, let us try a few more things. First, let us see what this course is all about. What exactly it stores.

(Refer Slide Time: 12:36)



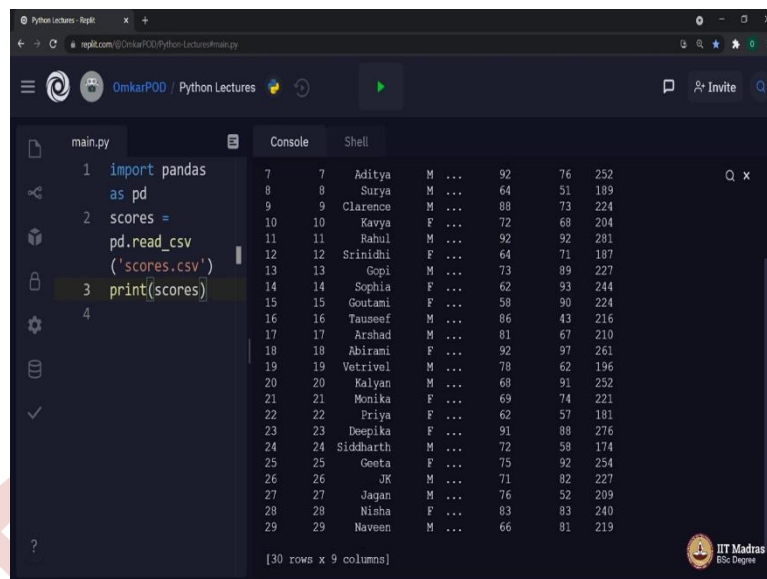
The screenshot shows a Jupyter Notebook interface with a file explorer on the left, a code editor in the center, and a console output on the right. The code in the editor is as follows:

```
1 import pandas
  as pd
2 scores =
  pd.read_csv
  ('scores.csv')
3 print(scores)
```

The console output displays a table of student scores:

CardNo	Name	Gender	...	Physics	Chemistry	Total
0	Bhuvanesh	M	...	64	78	210
1	Harish	M	...	45	91	198
2	Shashank	M	...	54	77	188
3	Rida	F	...	53	78	173
4	Ritika	F	...	64	89	240
5	Akshaya	F	...	92	84	247
6	Sameer	M	...	82	87	250
7	Aditya	M	...	92	76	252
8	Surya	M	...	64	51	189
9	Clarence	M	...	88	73	224
10	Kavya	F	...	72	68	204
11	Rahul	M	...	92	92	281
12	Srinidhi	F	...	64	71	187
13	Gopi	M	...	73	89	227
14	Sophia	F	...	62	93	244
15	Goutami	F	...	58	90	224
16	Tauseef	M	...	86	43	216
17	Arshad	M	...	81	67	210
18	Abirami	F	...	92	97	261
19	Vetriivel	M	...	78	62	196
20	Kalyan	M	...	68	91	252
21	Monika	F	...	69	74	221
22	Priya	F	...	62	57	181
23	Deepika	F	...	91	88	276





The screenshot shows a Jupyter Notebook with a file explorer on the left containing 'main.py'. The code in 'main.py' is:

```
1 import pandas
2 as pd
3 scores =
4 pd.read_csv
5 ('scores.csv')
6 print(scores)
```

The console output displays a DataFrame with 30 rows and 9 columns. The first column is an index from 0 to 29. The subsequent columns are: 'name', 'gender', 'town', 'city', 'maths', 'physics', 'chemistry', and 'total'. Ellipses (...) indicate that some columns are hidden due to display restrictions.

	name	gender	town	city	maths	physics	chemistry	total
0	Aditya	M	...	...	92	76	252	
1	Surya	M	...	...	64	51	189	
2	Clarence	M	...	...	88	73	224	
3	Kavya	F	...	...	72	68	204	
4	Rahul	M	...	...	92	92	281	
5	Srinidhi	F	...	...	64	71	187	
6	Gopi	M	...	...	73	89	227	
7	Sophia	F	...	...	62	93	244	
8	Goutami	F	...	...	58	90	224	
9	Tauseef	M	...	...	86	43	216	
10	Arshad	M	...	...	81	67	210	
11	Abirami	F	...	...	92	97	261	
12	Vetrivel	M	...	...	78	62	196	
13	Kalyan	M	...	...	68	91	252	
14	Monika	F	...	...	69	74	221	
15	Priya	F	...	...	62	57	181	
16	Deepika	F	...	...	91	88	276	
17	Siddharth	M	...	...	72	58	174	
18	Geeta	F	...	...	75	92	254	
19	JK	M	...	...	71	82	227	
20	Jagan	M	...	...	76	52	209	
21	Nisha	F	...	...	83	83	240	
22	Naveen	M	...	...	66	81	219	
23								
24								
25								
26								
27								
28								
29								

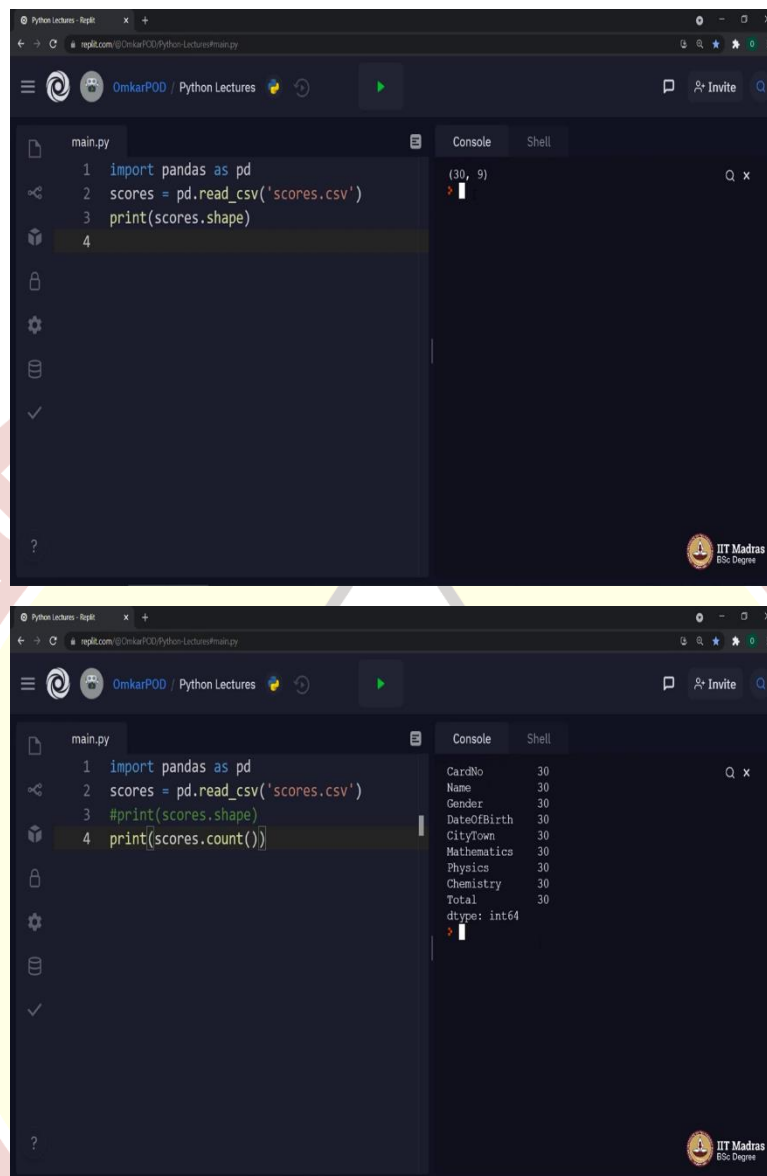
[30 rows x 9 columns]

Let us remove all this and print only scores. As you can see, it prints the entire scores data set in the form of a table. Card number from 0 to 29, name, gender, physics marks, chemistry marks and total. If you can see there are some dots in between that means some columns are hidden over here. That is because there is some restriction on how many columns can be displayed at the same time. Even though the columns are not displayed, that does not mean they are removed or anything. They are there only they are not been shown in this particular output.

Apart from that, if you can observe, there are some additional numbers added at the beginning of each line starting from 0 to 29. This is done by Pandas. Pandas always adds a index number for every single line, which you are reading from the file. In our case, we already had card numbers, but still pandas will always add this index numbers, which will be very helpful if you do not have such numbers in your existing file, and at the bottom it says total 30 rows and 9 columns.

Based on this we can easily tell that there are 30 students and 9 different details of every students are there. Like card number name, gender, town, city, date of birth, mathematics marks, physics marks, chemistry marks, and total. Now let us go back to the code and try to execute see more such interesting things.

(Refer Slide Time: 14:42)



The image shows two screenshots of a Jupyter Notebook interface. The top screenshot shows the first three lines of code: `import pandas as pd`, `scores = pd.read_csv('scores.csv')`, and `print(scores.shape)`. The output in the console is `(30, 9)`. The bottom screenshot shows the same code with a fourth line added: `print(scores.count())`. The output in the console is a series of 30s for each column, indicating that every cell in the CSV file contains a non-null value.

```
main.py
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 print(scores.shape)
4
```

Console

```
(30, 9)
```

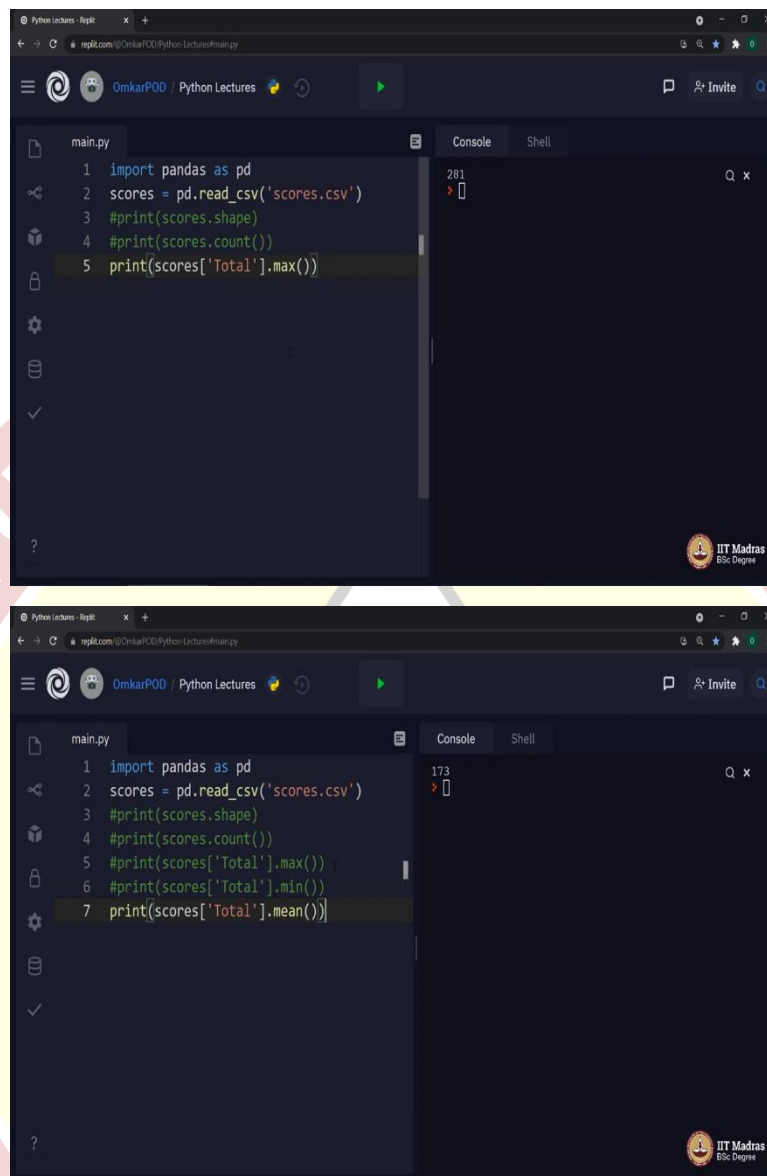
```
main.py
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 #print(scores.shape)
4 print(scores.count())
```

Console

```
CardNo      30
Name        30
Gender      30
DateOfBirth 30
CityTown    30
Mathematics 30
Physics     30
Chemistry   30
Total       30
dtype: int64
```

Scores dot shape, 30 comma 9. As you can see, it is enclosed in a round bracket, which means, it is tuple with first value as number of rows and second value as number of columns, and this is one more place where python uses tuples for its internal purpose. Scores dot count. This particular function is displaying how many values are there in each individual column. And as we know, we have values in every column, therefore, for every heading it says 30.

(Refer Slide Time: 15:32)



The image displays two screenshots of a Jupyter Notebook interface, likely from a video lecture. The interface shows a code editor on the left and a console output on the right. The code is written in Python and uses the pandas library to read a CSV file named 'scores.csv' and perform operations on the 'Total' column.

**Top Screenshot:**

```
main.py
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 #print(scores.shape)
4 #print(scores.count())
5 print(scores['Total'].max())
```

The console output shows the result of the `max()` function: 281.

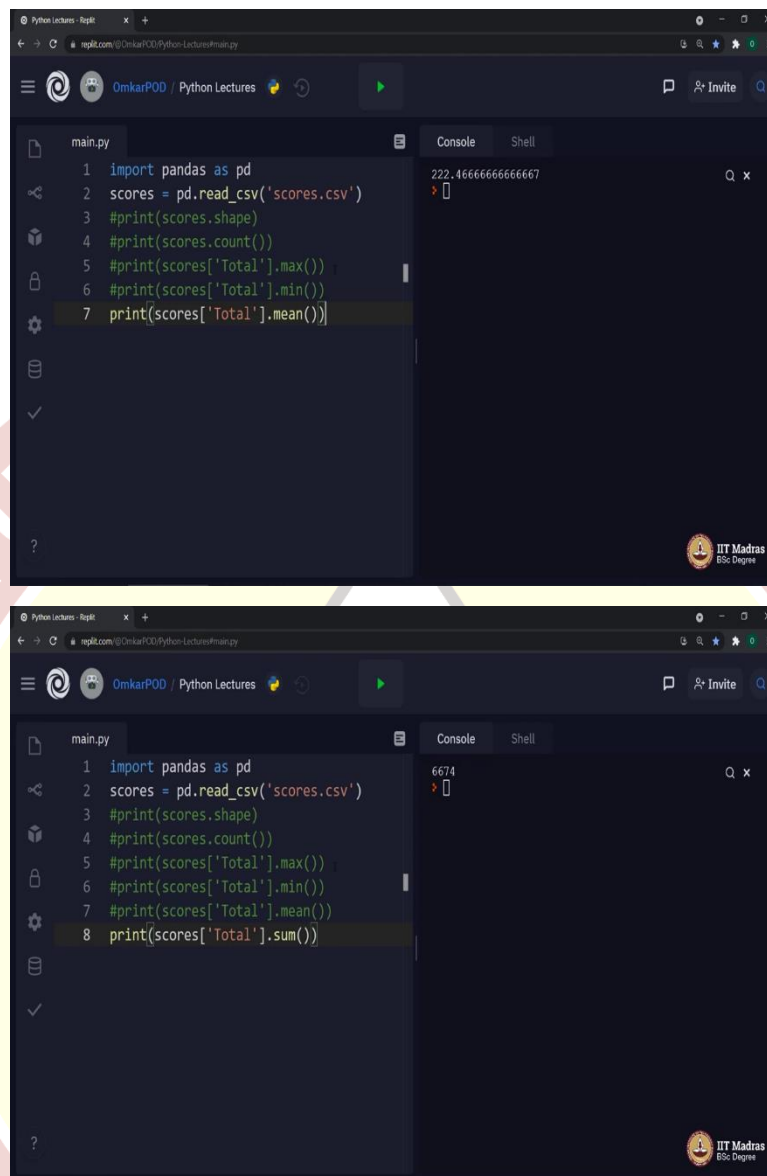
**Bottom Screenshot:**

```
main.py
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 #print(scores.shape)
4 #print(scores.count())
5 #print(scores['Total'].max())
6 #print(scores['Total'].min())
7 print(scores['Total'].mean())
```

The console output shows the result of the `min()` function: 173.

This is max function, which we have already executed. But let us execute it again because I want to show you, it is not just about max, there are so many other things, which you can do in a same manner. Like this mean function, which will give us the minimum value in the total column, which is 173.

(Refer Slide Time: 16:00)



The image displays two screenshots of a Jupyter Notebook interface, likely from a video lecture. The interface shows a code editor on the left and a console on the right. The code in the first screenshot uses the `mean()` function to calculate the average of the 'Total' column. The console output shows a large decimal value. The second screenshot shows the code being modified to use the `sum()` function to calculate the total of the 'Total' column. The console output shows a single integer value.

```
main.py
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 #print(scores.shape)
4 #print(scores.count())
5 #print(scores['Total'].max())
6 #print(scores['Total'].min())
7 print(scores['Total'].mean())

Console
222.46666666666667
```

```
main.py
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 #print(scores.shape)
4 #print(scores.count())
5 #print(scores['Total'].max())
6 #print(scores['Total'].min())
7 #print(scores['Total'].mean())
8 print(scores['Total'].sum())

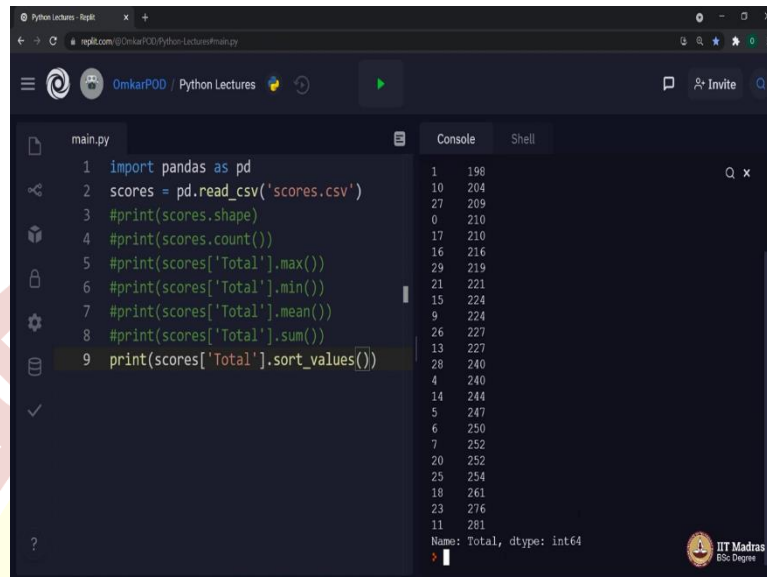
Console
6674
```

Now let us look at this particular function. This is mathematical mean, which is nothing but average of the column total. Once again, without putting much efforts we are getting average total marks from the scores data set and that too in only three lines, sum, sum of all the values in the total column and we are getting whatever the answer is. Now you must be thinking, is that it or is there anything even more complex, which pandas can do? The answer is, this is just the beginning. Pandas can do so many more things.

And for one more example, let us look at this next line and the function is sort underscore values, which means, we are asking computer to sort all the values in this particular total column. If you remember from earlier, sorting some values required a lot of efforts, especially if we are supposed to write it using some loops, then it even requests nested loops,

which itself is very complicated. But using pandas, we can do it once again, in just three lines.

(Refer Slide Time: 17:33)



The screenshot shows a Jupyter Notebook interface with a file named 'main.py'. The code in the notebook is as follows:

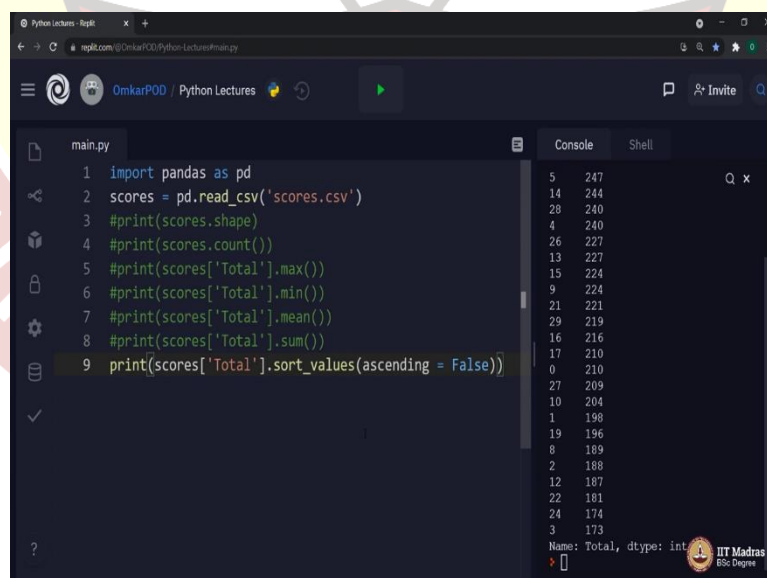
```
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 #print(scores.shape)
4 #print(scores.count())
5 #print(scores['Total'].max())
6 #print(scores['Total'].min())
7 #print(scores['Total'].mean())
8 #print(scores['Total'].sum())
9 print(scores['Total'].sort_values())
```

The console output shows the sorted values of the 'Total' column in ascending order:

```
1    198
10   204
27   209
0    210
17   210
16   216
29   219
21   221
15   224
9    224
26   227
13   227
28   240
4    240
14   244
5    247
6    250
7    252
20   252
25   254
18   261
23   276
11   281
Name: Total, dtype: int64
```

And we have our output from 173 to 281. It is sorting all these values in ascending order, but that is not it, we can also sort these values in descending order just by adding one more parameter to this particular sort underscore values function. Let us see what that parameter is.

(Refer Slide Time: 18:05)



The screenshot shows a Jupyter Notebook interface with a file named 'main.py'. The code in the notebook is as follows:

```
1 import pandas as pd
2 scores = pd.read_csv('scores.csv')
3 #print(scores.shape)
4 #print(scores.count())
5 #print(scores['Total'].max())
6 #print(scores['Total'].min())
7 #print(scores['Total'].mean())
8 #print(scores['Total'].sum())
9 print(scores['Total'].sort_values(ascending = False))
```

The console output shows the sorted values of the 'Total' column in descending order:

```
5    247
14   244
28   240
4    240
26   227
13   227
15   224
9    224
21   221
29   219
16   216
17   210
0    210
27   209
10   204
1    198
19   196
8    189
2    188
12   187
22   181
24   174
3    173
Name: Total, dtype: int64
```

Ascending is equal to false. Let us execute. Now we have values from 281 to 173. As you can see even a complicated code like sorting also can be executed using Pandas without any hassle. Now, just imagine you do not have Pandas and you are supposed to sort the total marks using plain file handling operations. Can you imagine how complex that code will be?

Maybe you should try to write that code, then only you will understand the complexity involved in that particular program.

Now, I hope you all are convinced that pandas is a very powerful tool and using pandas, we can execute so many different complicated codes within very few lines of Python program and that too in a very simpler manner. As I said earlier, the idea behind this lecture was to tell you why we are studying pandas. More details about pandas will be covered in next lecture also. Thank you for watching this lecture. Happy learning

