INDIAN INSTITUTE OF TECHNOLOGY MADRAS

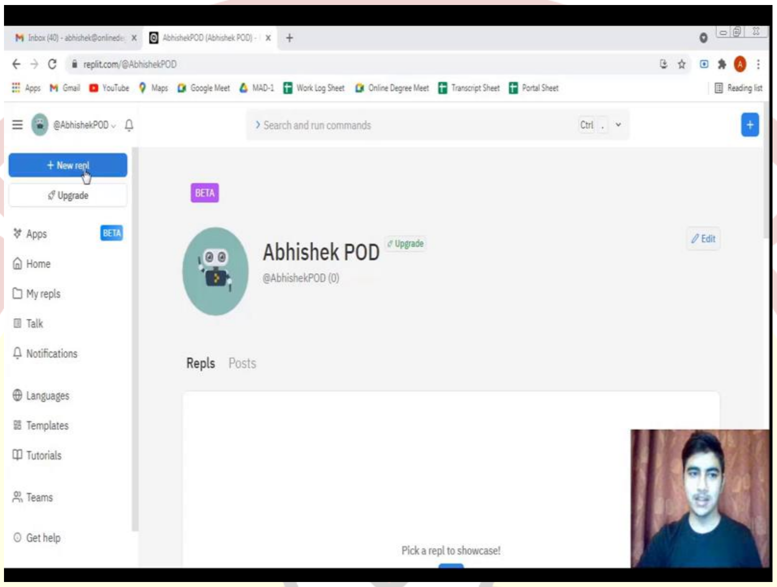
सिद्धिर्भवति कर्मजा

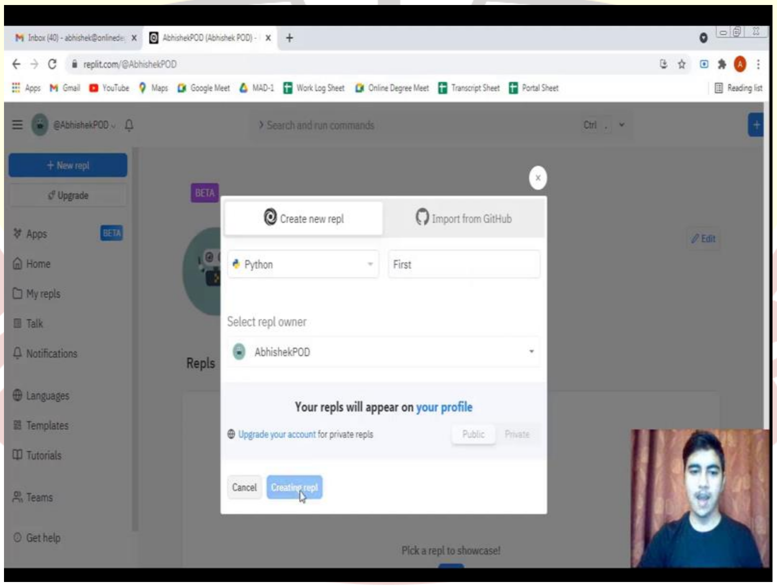
IIT Madras

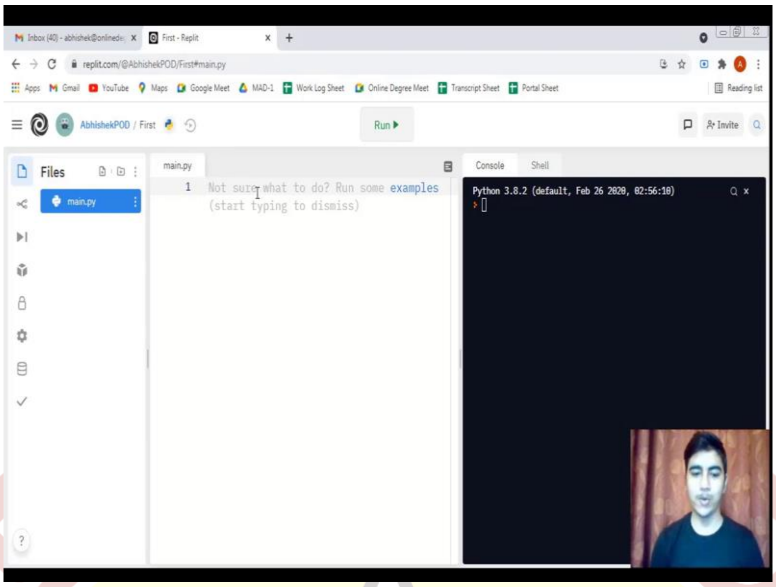
ONLINE DEGREE

Modern Application Development – 1
Professor Nitin Chandrachoodan
Department of Electrical Engineering
Professor Mr. Abhishek
Course Instructor
Indian Institute of Technology, Madras
Command Line Arguments in Python

(Refer Slide Time: 0:20)







```
Not sure what to do? Run some examples
(start typing to dismiss)
```

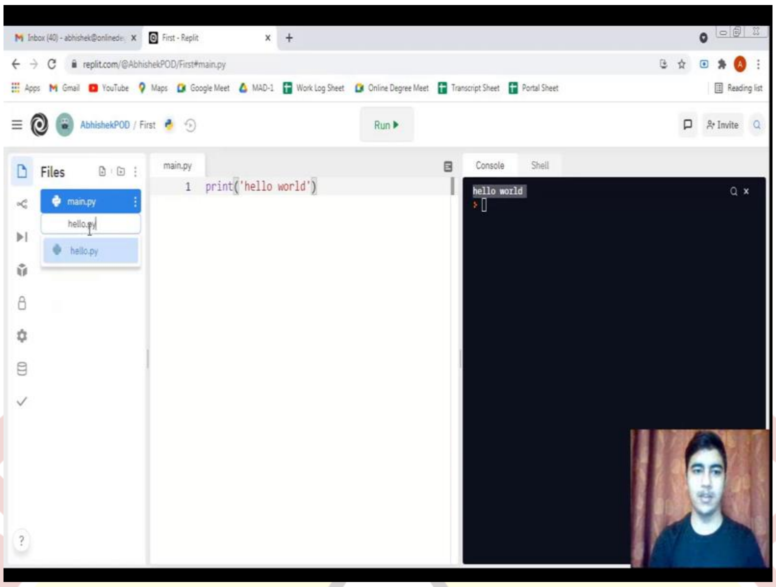
```
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
>
```

Hello everyone. Today we will see how to execute Python scripts via command line interface. We will also see how to pass parameters to those scripts while doing it. So, let us get started. Now, this is my users dashboard Replit screen, and here are the top left, I can see this new repl button.

So, after clicking this button, I would have to select the language that I want to code for. I will select Python here. And of course, I would have to name my project. So, let us name it is first here. After naming it, we can click on this create repl button. And after clicking on this button, we will wait for a small amount of time, so that this type of interface can get loaded for us. Here we can get this connected, connected message. That means the Python interpreter has been connected as well.

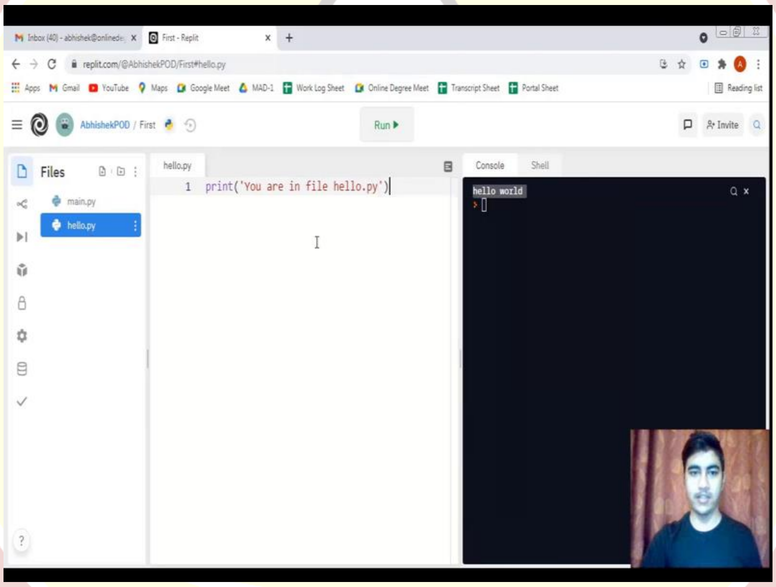
And this whole interface can mainly be divided into 3 sections. One on the left-hand side, this is the layout panel. In the middle, this is going to be your workspace and this black screen, this console, this is basically the place where we will be getting the output of whatever code you will write in the workspace.

(Refer Slide Time: 1:32)



```
print('hello world')
```

```
hello world
> 
```



```
print('You are in file hello.py')
```

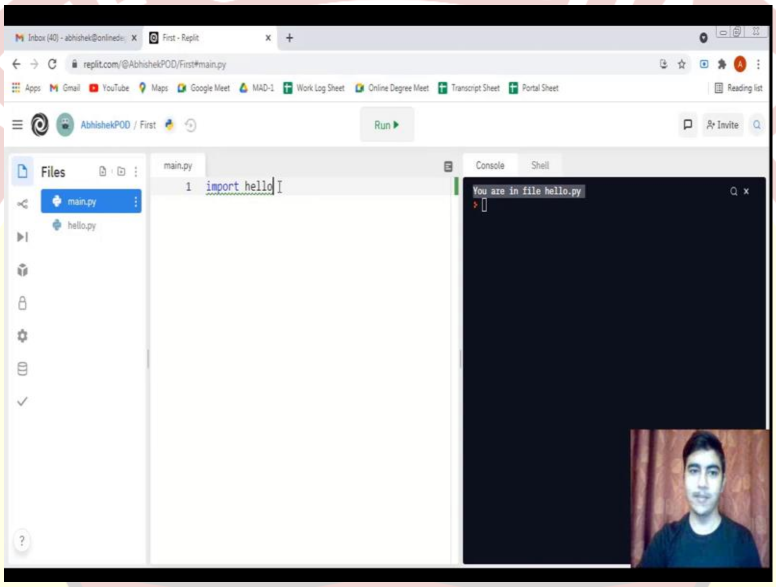
```
hello world
> 
```

Now since this is a Python project, so let us write a one line code here. Let us say print hello world, that should be enough. And we can execute this code by using this run button. So, I would click on this run button. And I am getting this hello world, that is fine. Now, as we know that we can add as many as files in a project, and here repl x as a project. So, we can also add as many as files as we want.

And we are being given these two buttons, add File add folder in order to create a directory like structure, we can also upload some files or folders if we want. For now, I am not going too much into those things, I will simply add this file here. Let us say the name can be hello.py. And since this is a py file, that means I can also write some Python code here.

So, let me write print you are in file hello.py. I have written this small line here. But as we know that clicking this run button would not execute this thing. Why? Because this run button will only execute this main.py file that would show hello world on the Output screen. See, I would click on Run, I am getting this hello world on the output screen. Instead of getting this you are in file hello.py. Now, as we are able to add some external files into our project. So, there should be some way that we can execute those files separately. And yes, there is. So, let us discuss about some of those ways.

(Refer Slide Time: 3:14)

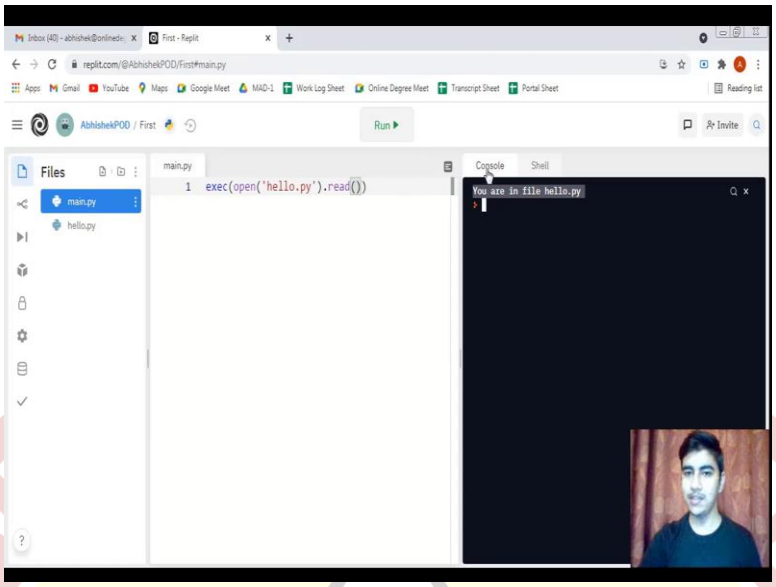


```
import hello
```

```
You are in file hello.py
›
```

The very first ways we can use the import statement, we can say import. And we have to mention the module name that we want to import here, I would say hello, hello import means or import hello means I am going to import whatever the code that is returned in this hello.py file into my main.py file. That means the code from this file could be collected and will be pasted in this main.py file. After that, I can click on this run button. And I will get this output that you are in file hello.py. That is exactly what we had expected of it.

(Refer Slide Time: 3:53)



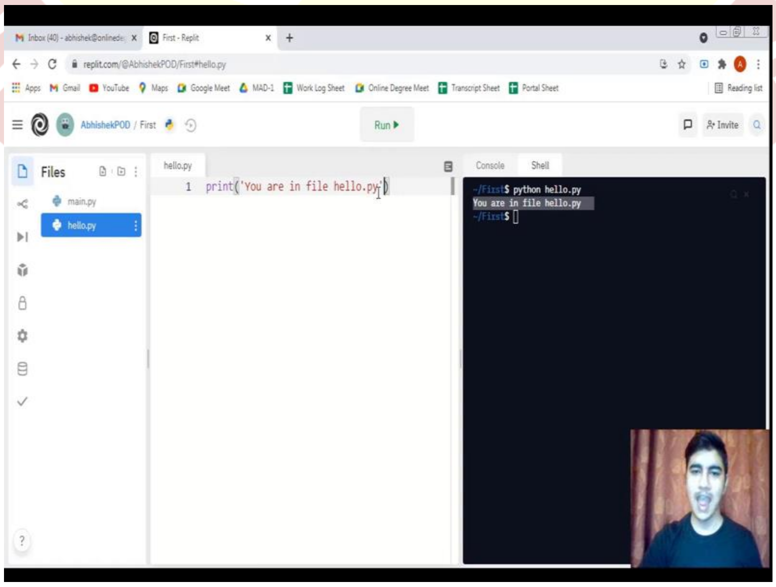
```
exec(open('hello.py').read())
```

```
You are in file hello.py
>
```

So, this is one of the way the other ways we can use the exec function. Now, this exec function will basically accept the Python code. But we somehow need to extract the Python code from this file. And we need to put it here. What can we do here is we can use open the file name is hello.py. And we can use the read method in order to extract the content from this hello.py file and can put it here.

Now, if I will click on Run button, I would again get the same output that means I am able to execute this file separately. We have seen two ways. Now, there is one more way that we can execute this file and even without run button. Yes, you heard it right. We can execute this file without using this run button. How?

(Refer Slide Time: 4:46)



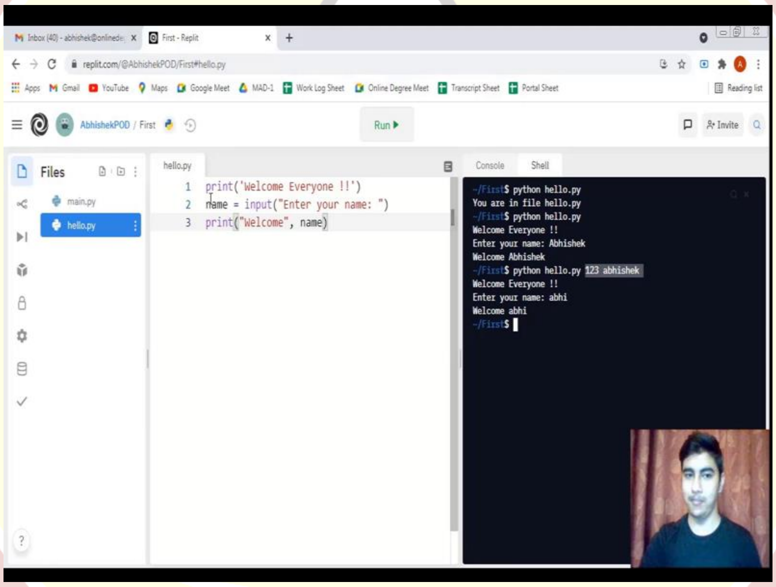
```
print('You are in file hello.py')
```

```
~/First$ python hello.py
You are in file hello.py
~/First$
```

We were using this console tab till now, what we will be doing is we will switch to the shelter shell interface here. Now, this shell acts as a terminal for us. And we know that there are some commands through which we can execute our Python scripts. So, to execute a Python script, the command is we first have to write the keyword python here, give a space. Now, we have to provide the file name that we want to be executed here.

So, the file name is hello.py, I can also write main.py file, but I am writing hello.py here, you can write any file name that you want to be executed. After that, I will hit an enter key. And after hitting an enter key, we can see that we are getting an output of that file. That is exactly the same that what we had expected. Now, since we have written just a single line code in this hello.py file, let us try to add some more lines of code.

(Refer Slide Time: 5:43)



```python
print('Welcome Everyone !!')
name = input("Enter your name: ")
print("Welcome", name)
```

```
~/First$ python hello.py
You are in file hello.py
~/First$ python hello.py
Welcome Everyone !!
Enter your name: Abhishek
Welcome Abhishek
~/First$ python hello.py 123 abhishek
Welcome Everyone !!
Enter your name: abhi
Welcome abhi
~/First$
```

I would say, maybe Welcome everyone something like that. Or maybe we can use the input function? Yes, let us use the input function in order to accept the name from the user. So, we would say input, enter your name. And after that, we would simply show the name that has been given by the user along with a message; welcome. So, welcome, name.

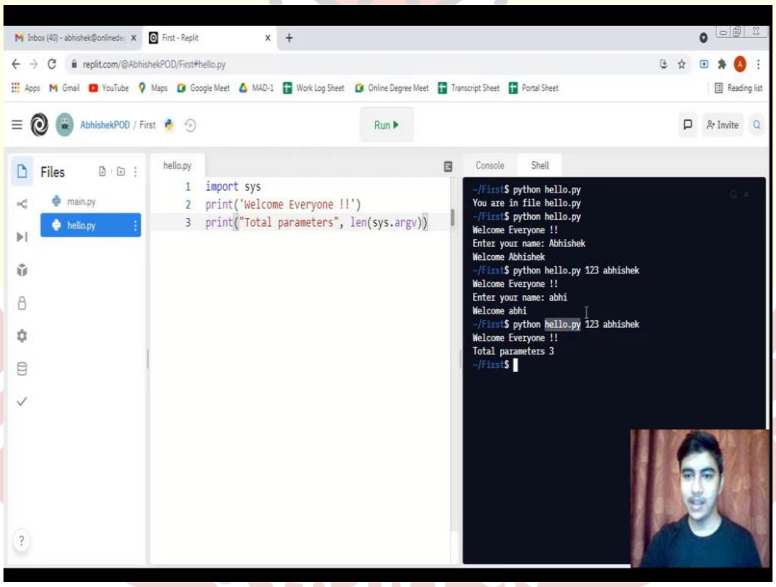
So, after this, I would, again call this file, I would again execute this Python file in the command line interface python hello.py, I would again hit enter. This time it is showing welcome everyone. And here it is asking enter your name. That means it is asking me to enter some name here. So, let me enter this thing. And let us see if I will hit an enter key what will happen, it would say welcome Abhishek that means whatever the name whatever the input has been provided, that has been shown here again, along with this message, welcome.

So, we have seen how can we execute Python scripts via command line interface? Now, in order to know that how we can pass command line parameters, or how we can use those command line parameters in our program, we would first like to check that is it really possible to pass some command line parameters to a Python script? Let us do that.

I would say python hello.py. Now, since we have to pass some parameters, we have to separate them with a space. So, let us say 123. Let me create one more parameter. Let us say Abhishek and let me hit enter. Let us see if it is going to impact on our program or not. Everything is going pretty much well, as of now, I would again, enter something here.

And, so this does not make any impact on our program. Why? Because we are not doing anything with those command line parameters, that is it. Our program is behaving normally. Now, the thing is that this thing is confirmed that we are able to pass some command line parameters to a Python script. Now, if we can pass these things, so there should be some way that we can access these values in our program. And yes, there is a way or what is that way? Let us see.

(Refer Slide Time: 8:18)



```python
import sys
print('Welcome Everyone !!')
print("Total parameters", len(sys.argv))
```

```
~/First$ python hello.py
You are in file hello.py
~/First$ python hello.py
Welcome Everyone !!
Enter your name: Abhishek
Welcome Abhishek
~/First$ python hello.py 123 abhishek
Welcome Everyone !!
Enter your name: abhi
Welcome abhi
~/First$ python hello.py 123 abhishek
Welcome Everyone !!
Total parameters 3
~/First$
```

For that we will be importing a module, the name is sys module, this is a standard Python module. You do not need to we can install it. I would remove this input statement. And here let us do one thing. Let us try to show the number of parameters given to a Python script. Let us try to first show that value. So, I would say total parameters. And here I would say len(sys.argv).

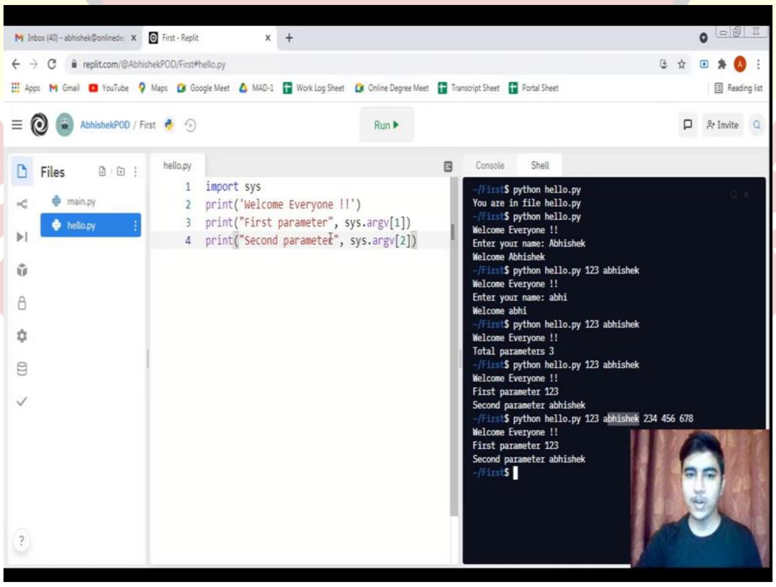
Now, you might be wondering what this sys.argv is? So, this sys.argv is a special kind of list in Python that has all of the values that have been passed as command line parameter to the Python script, that means this list is a collection of all of the command line parameters passed values to a particular Python script. So, what we are doing here is we are simply showing the number of parameters that have been passed to that Python script. So, let us see.

I would call this again python hello.py and I am passing two parameters 123 and Abhishek. Both are separated with a space here please note. After that, I would hit the enter key, and what I am getting here is the total parameters are 3, but I have just passed 2 here. Why is it saying 3?

The answer is that this sys.argv will always have the name of the Python module as its first element, that means the name of the Python file that is being executed by a command line interface will always be the first element in this argv list, that means they would be stored at zero th index in this list, as we know that the first element in a list always stored at zero th index.

So, that is why it is saying 3 that means one is this hello.py that is originally stored at argv zero th index, this 123 would be stored at one th index, this Abhishek would be stored at second index. Now, let us try to do that if we can access all of the values separately or not.

(Refer Slide Time: 10:38)



```python
import sys
print('Welcome Everyone !!')
print("First parameter", sys.argv[1])
print("Second parameter", sys.argv[2])
```

```
~/First$ python hello.py
You are in file hello.py
~/First$ python hello.py
Welcome Everyone !!
Enter your name: Abhishek
Welcome Abhishek
~/First$ python hello.py 123 abhishek
Welcome Everyone !!
Enter your name: abhi
Welcome abhi
~/First$ python hello.py 123 abhishek
Welcome Everyone !!
Total parameters 3
~/First$ python hello.py 123 abhishek
Welcome Everyone !!
First parameter 123
Second parameter abhishek
~/First$ python hello.py 123 abhishek 234 456 678
Welcome Everyone !!
First parameter 123
Second parameter abhishek
~/First$
```

Say first parameter, first parameter, I would remove this len, now what will I say is I will say directly 1, I am not considering the zero th index, because I know that the zero th index will always have the name of the Python file. So, what means will directly point out to 123, if we will talk about this example. In the same way, I would copy this statement. And I would paste it in here, I would say that this is my second parameter, second parameter sys.argv[2].

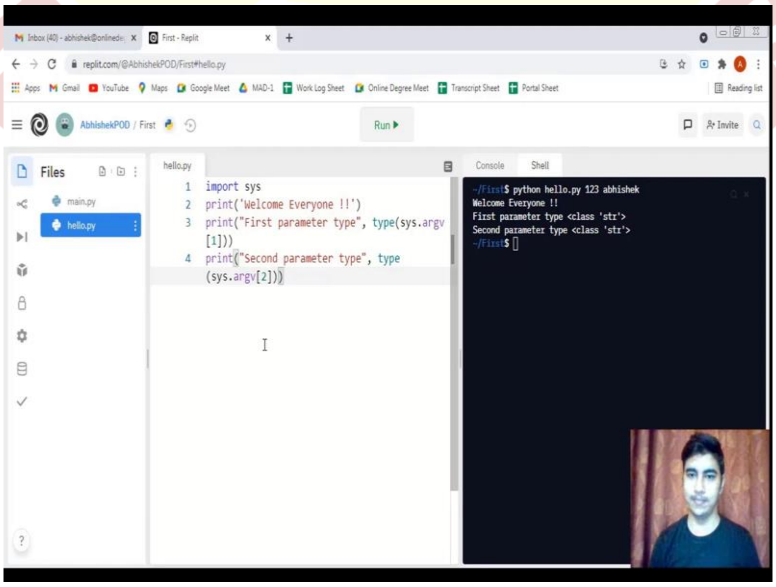
Now, let us see what is going to happen I would again execute this Python script in the command line interface itself. So again, Python hello.py 123 Abhishek. I am passing two parameters here. But originally, this argv list will consist of 3 because this hello.py will also be considered an element. But for now, we are just leaving out that zero th index element we are just considering after it.

So, I will hit enter key and I can see that the first parameter is 123. The second parameter is Abhishek. I am able to access these values in my programs. So, that is great thing. Now, let us see that if we are only using two values in a Python program, but we are trying to pass more values, let us see what is going to happen.

I am maybe trying to pass some number of parameters, remember, all of them will be separated by a space, I would hit enter again, this would not make any impact on our program. Because we can pass as many as number of arguments, parameters to our Python script. If we are not using them in a Python program, that maybe kind of useless thing. But still, we can pass as many as number of parameters as we want. So, this can be done as well.

But the values that we are passing here, the question is, what is the data type of those values? If I am passing 123, it is a number we know it is a number. But how Python is accepting that via command line? This Abhishek is a collection of characters. So, there is, this should originally be a string, but how Python is accepting that. Let us see.

(Refer Slide Time: 13:01)



```python
import sys
print('Welcome Everyone !!')
print("First parameter type", type(sys.argv[1]))
print("Second parameter type", type(sys.argv[2]))
```

```
~/First$ python hello.py 123 abhishek
Welcome Everyone !!
First parameter type <class 'str'>
Second parameter type <class 'str'>
~/First$
```

So, what we will do is we will say we will actually print that type of both of the attribute. Type, and here also, I will use the type function. In the second case, as well, I will use the type function. So, maybe let me clear this console, I would run it again.

And I will just pass two values, that should be fine. So, here we can see that it will be of class string only. Last string only, it does not matter if it is a number or if it is of any kind of different value in your view, the Python will always be considering them as a string value only. Now, you can try to recall the input function, the input function always returns a string type value unless it is typecast it into a different data type.

Now, here also, this is accepting all of the values as string values. But if you want, you can always typecast them, you can always convert them into any desired data type that you want as per your requirements. This is very much possible. So, this can be done. I wait. So, we have seen how can we execute Python scripts via command line interface and if we are doing so, how can we pass parameters to that script as well.

And the main important thing here is how to accept how to access those values that have been passed as command line parameters to that Python script in a Python program. How can we do that, we can simply use the sys.argv list in order to access that. Alright, so I hope you all enjoyed this video. Hope you have a great learning experience. Thank you.