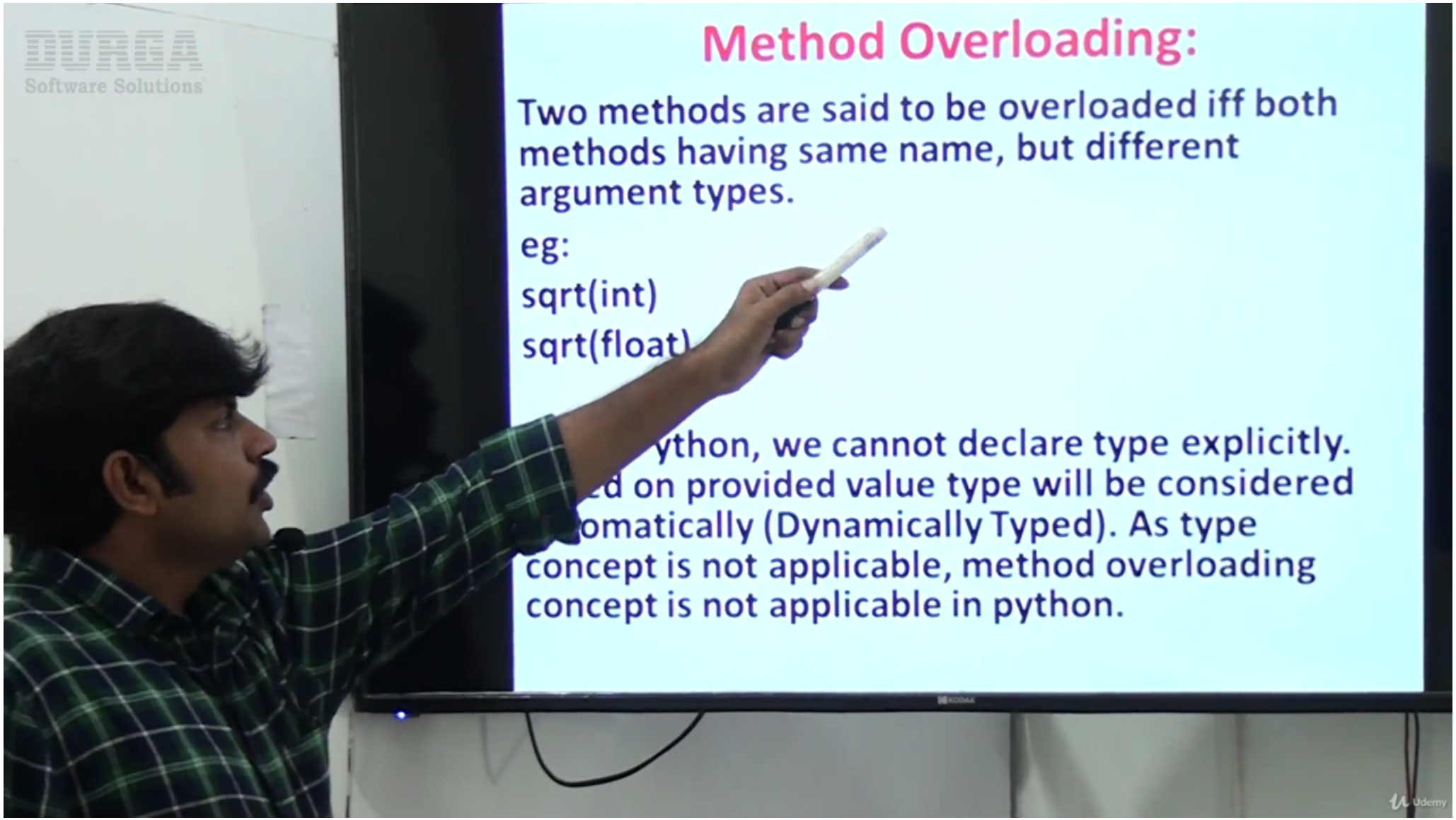
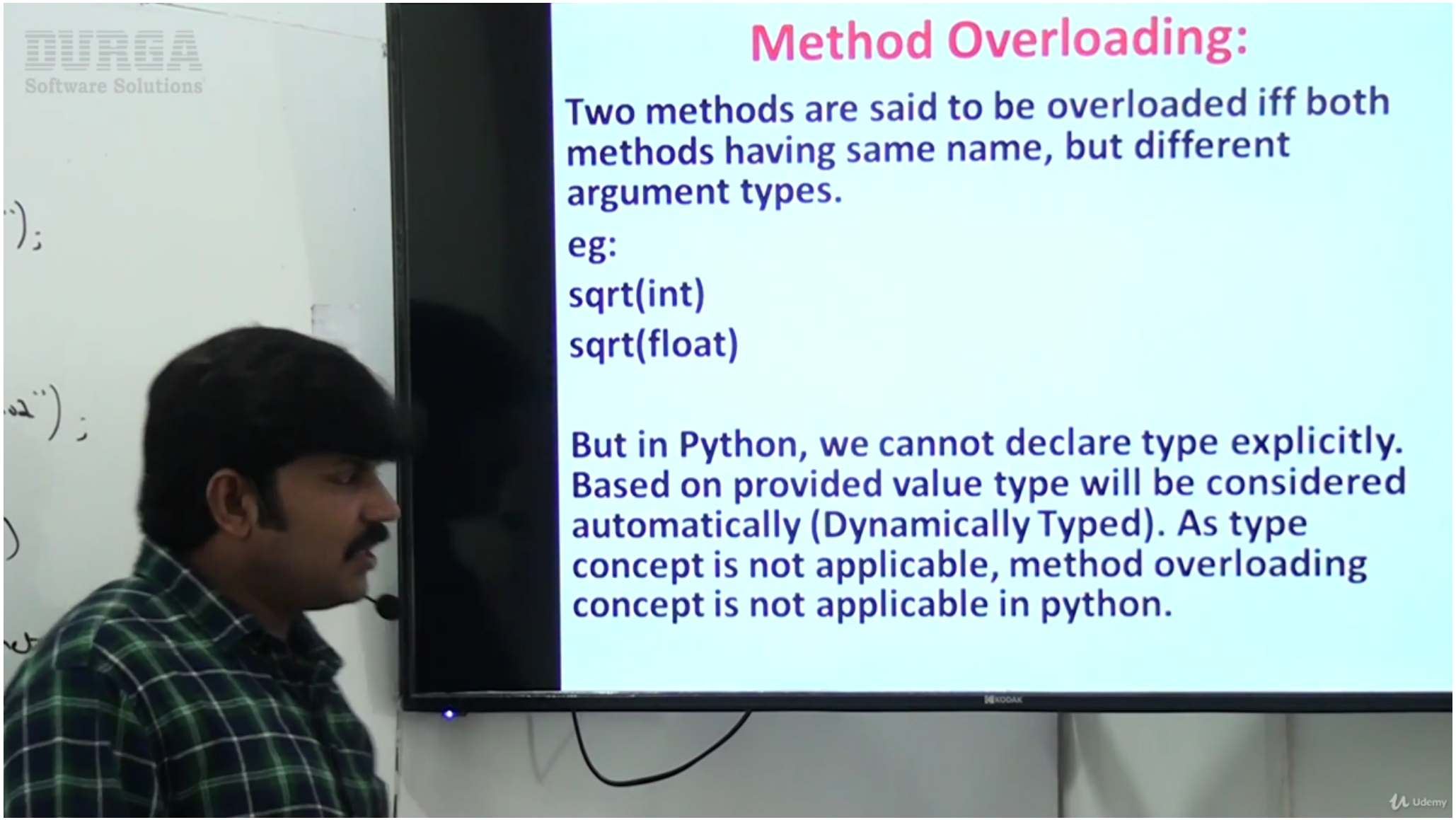
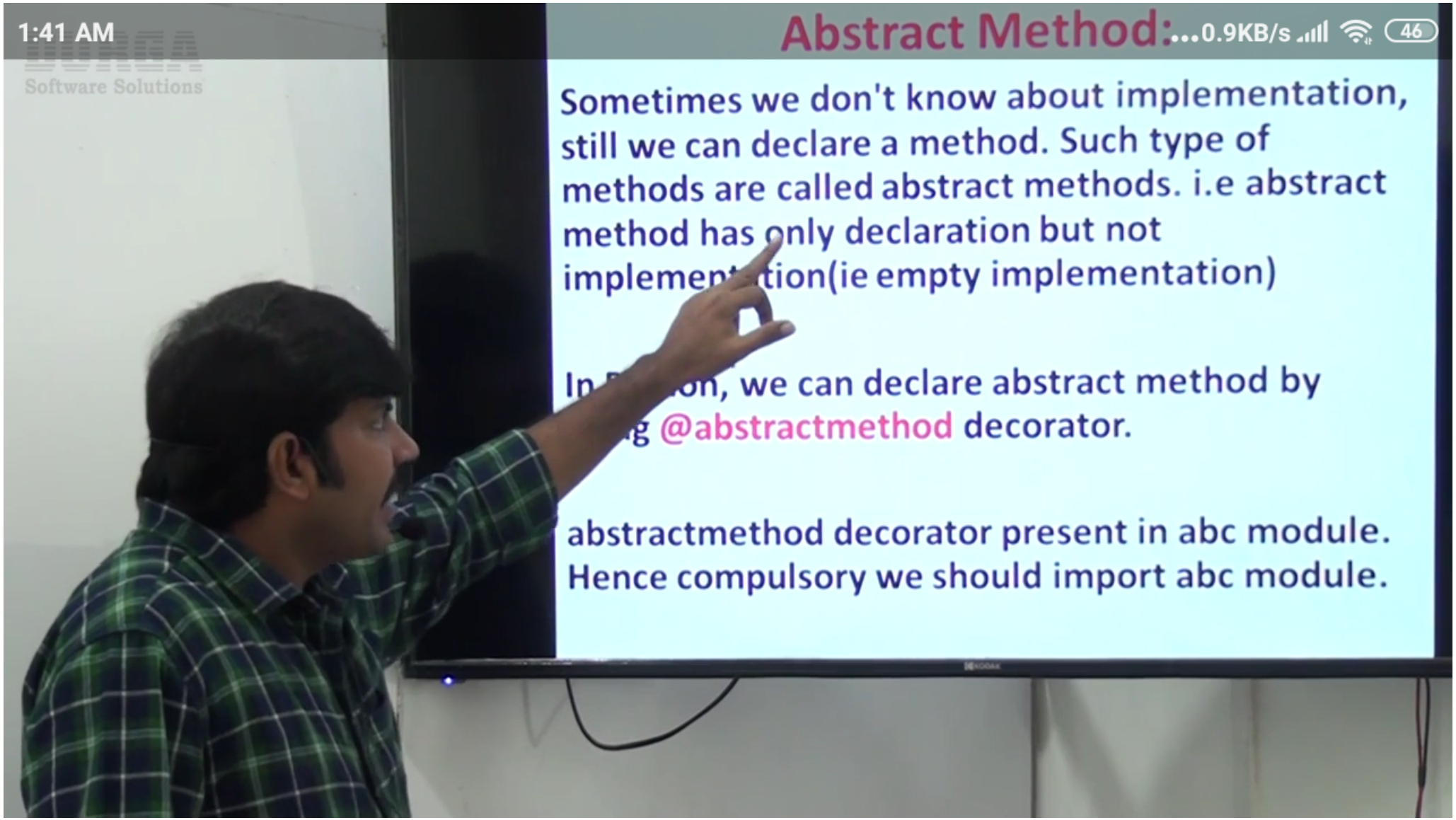
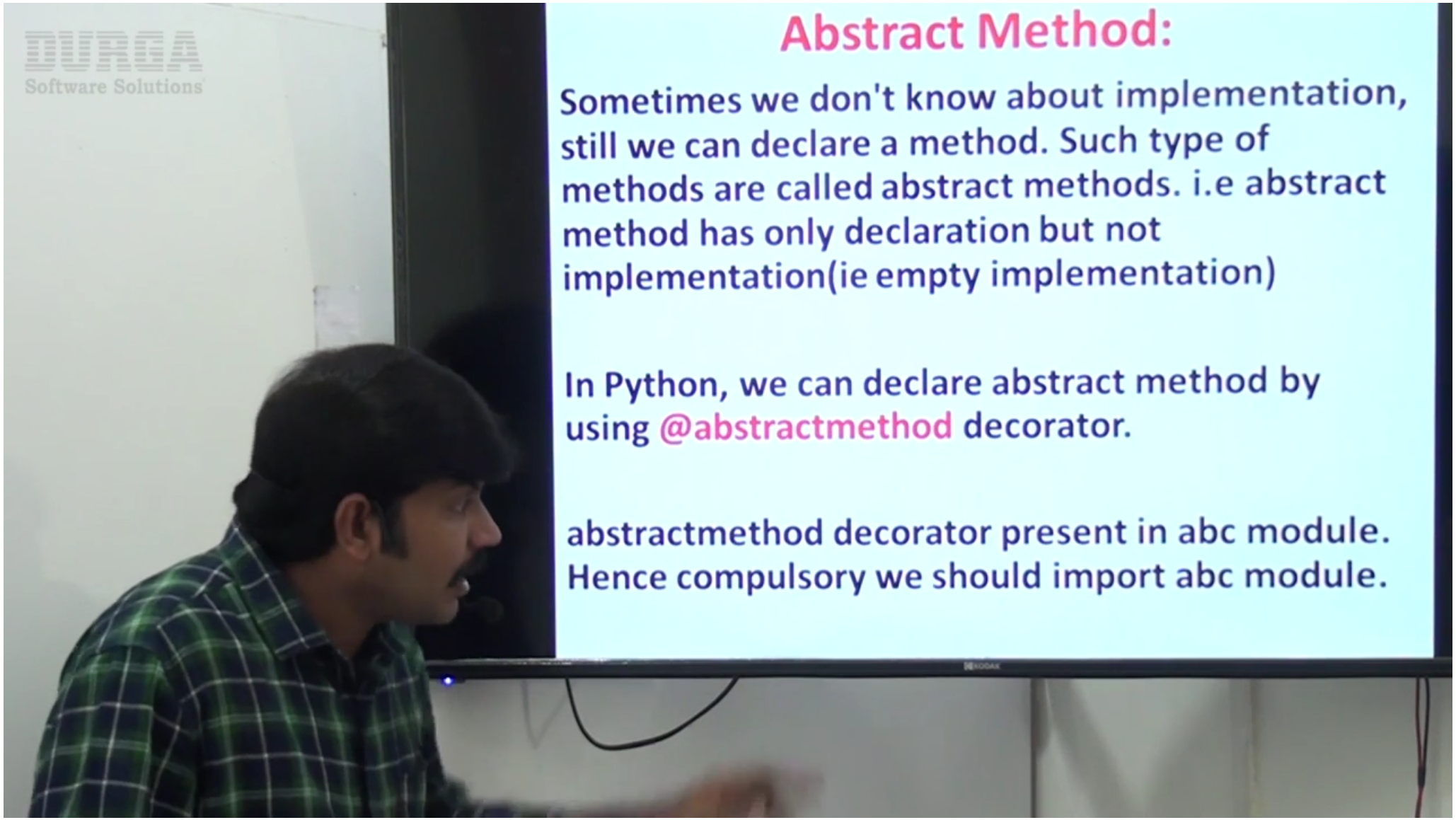
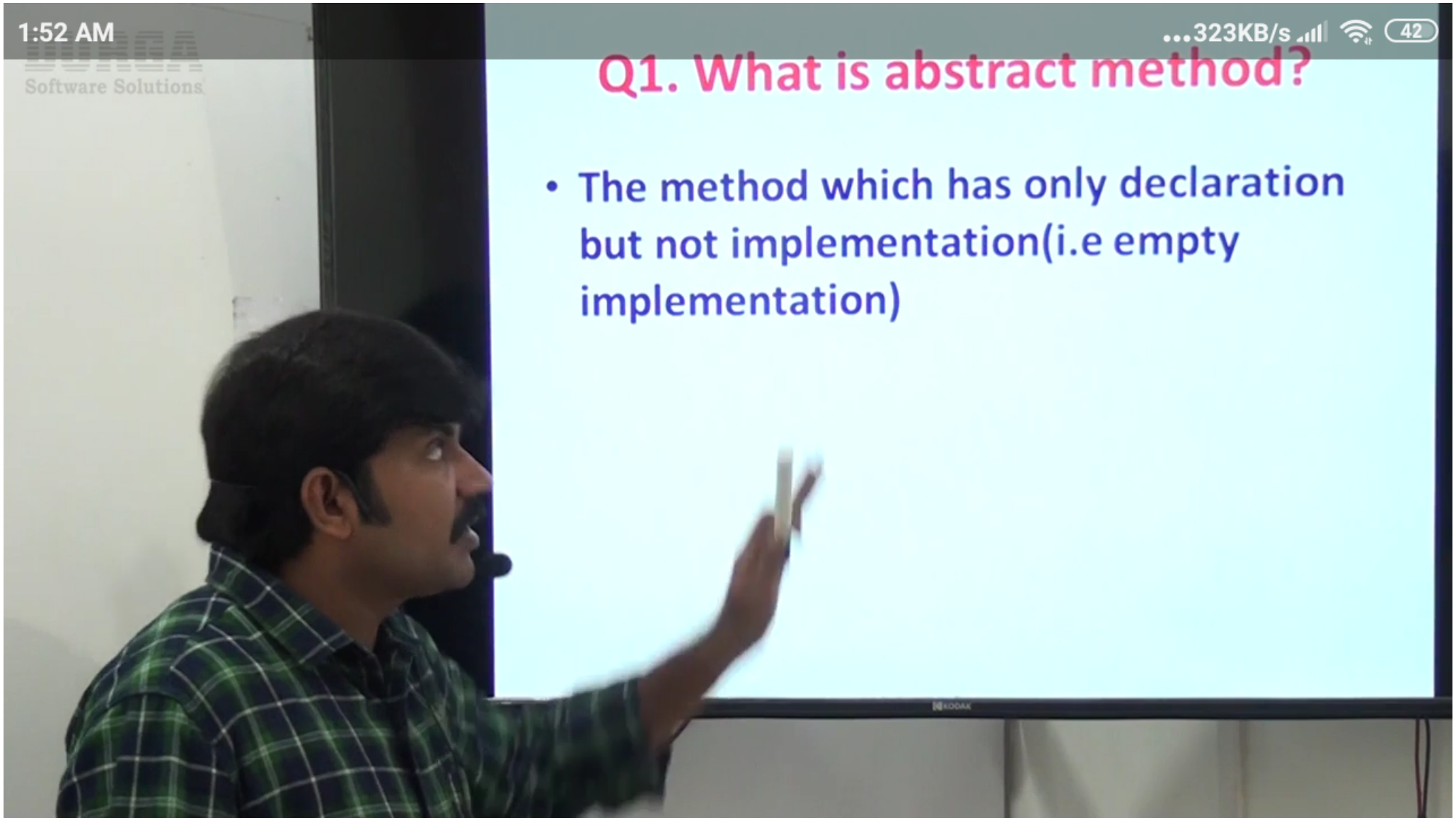
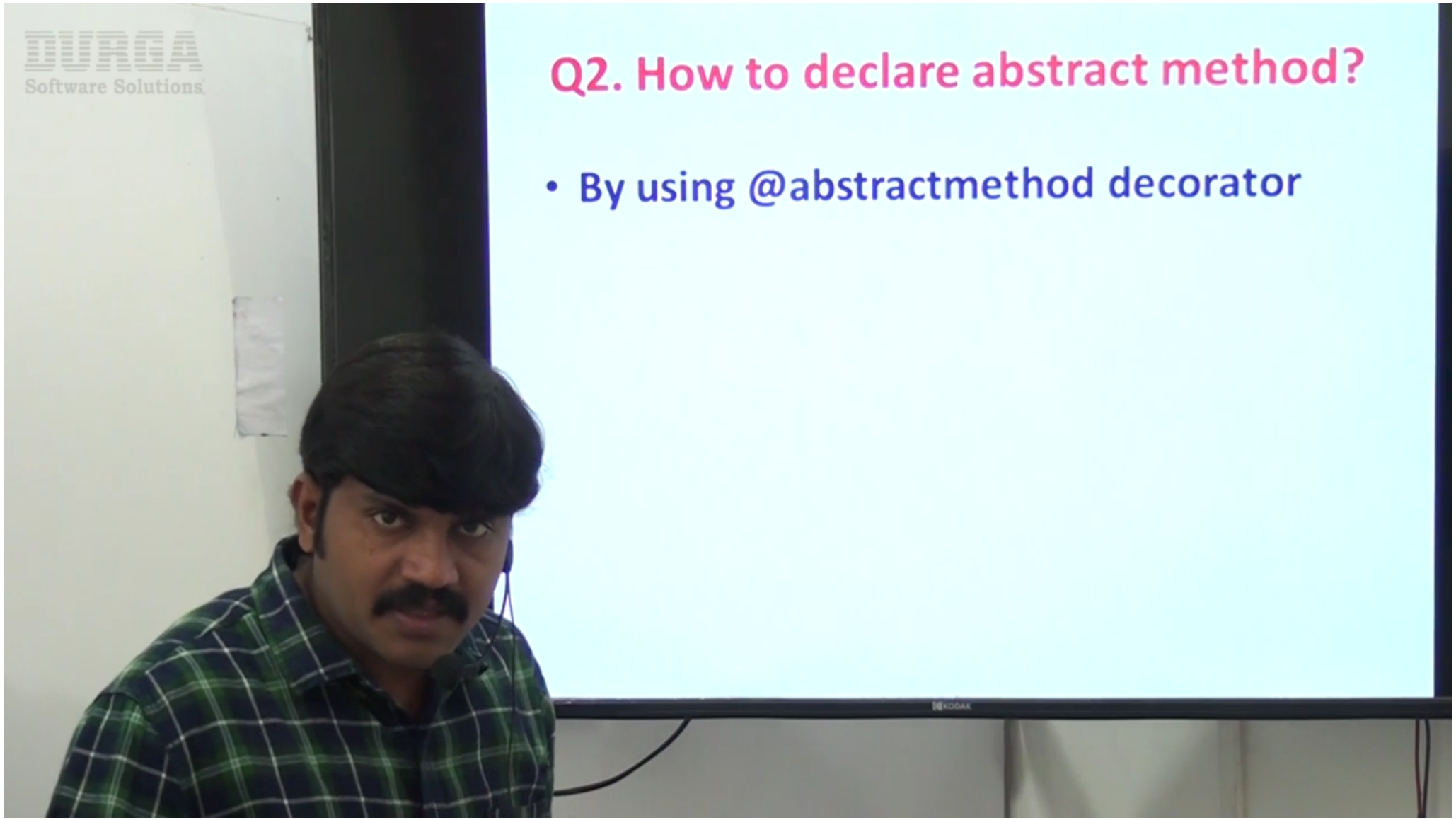

DURGA
Software Solutions
Method Overloading:
Two methods are said to be overloaded iff both methods having same name, but different argument types.
eg:
sqrt(int)
sqrt(float)
ython, we cannot declare type explicitly.
d on provided value type will be considered
omatically (Dynamically Typed). As type
concept is not applicable, method overloading
concept is not applicable in python.
Udemy

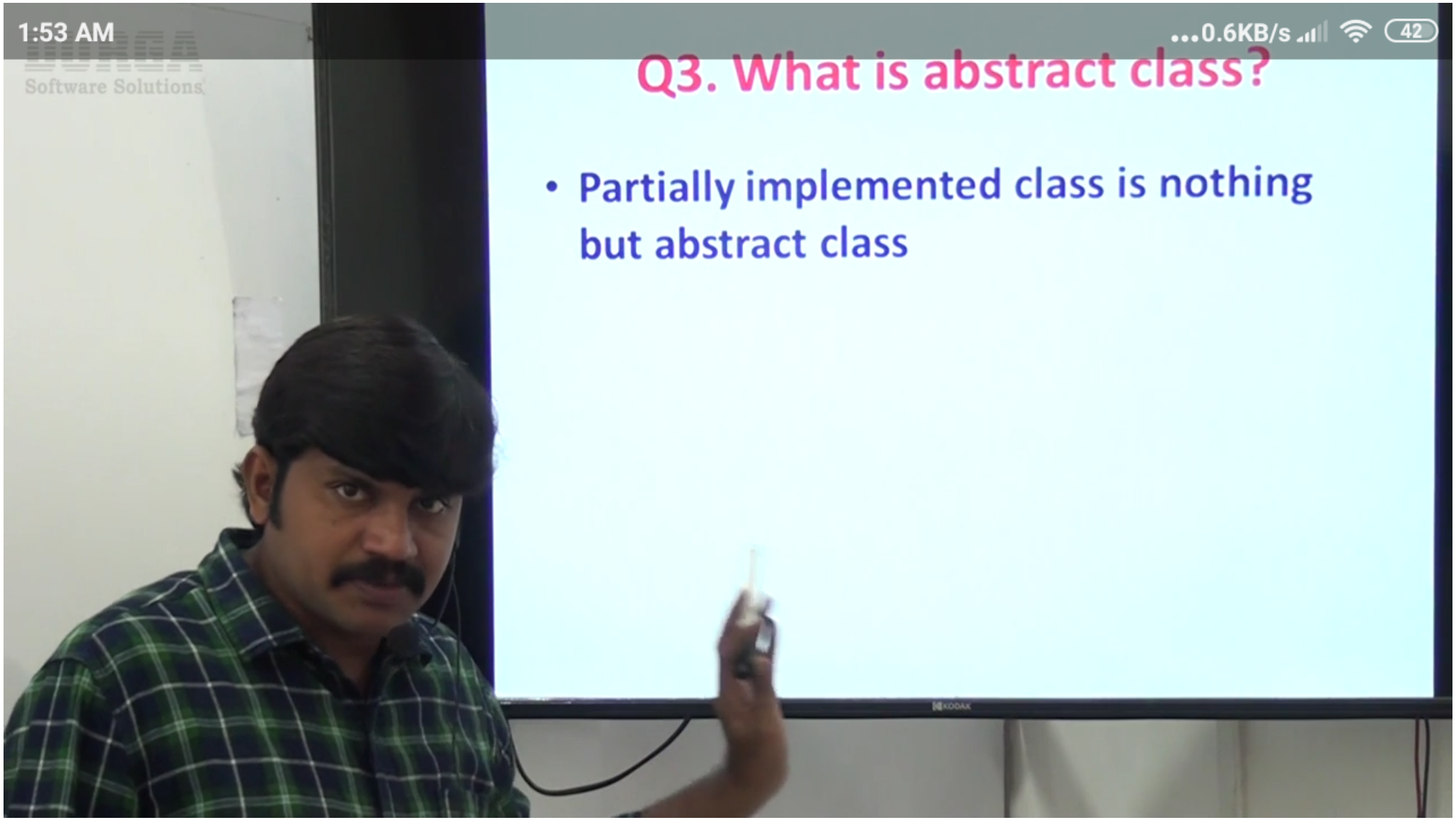

DURGA
Software Solutions
Method Overloading:
Two methods are said to be overloaded iff both methods having same name, but different argument types.
eg:
sqrt(int)
sqrt(float)
But in Python, we cannot declare type explicitly. Based on provided value type will be considered automatically (Dynamically Typed). As type concept is not applicable, method overloading concept is not applicable in python.
KODAK
Udemy

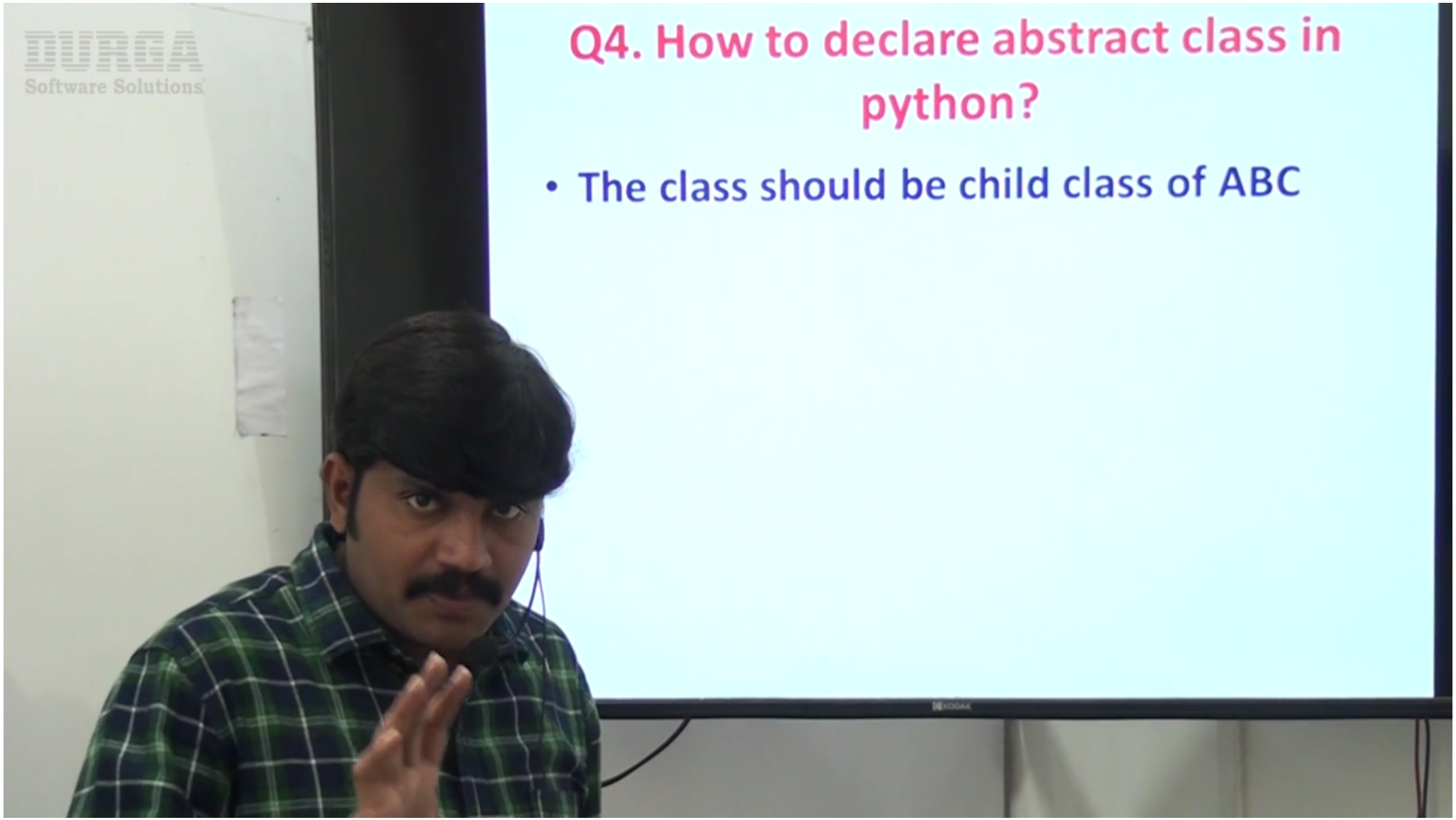

1:41 AM
Software Solutions
Abstract Method:...0.9KB/s
46
Sometimes we don't know about implementation, still we can declare a method. Such type of methods are called abstract methods. i.e abstract method has only declaration but not implementation(ie empty implementation)
In Python, we can declare abstract method by using @abstractmethod decorator.
abstractmethod decorator present in abc module. Hence compulsory we should import abc module.

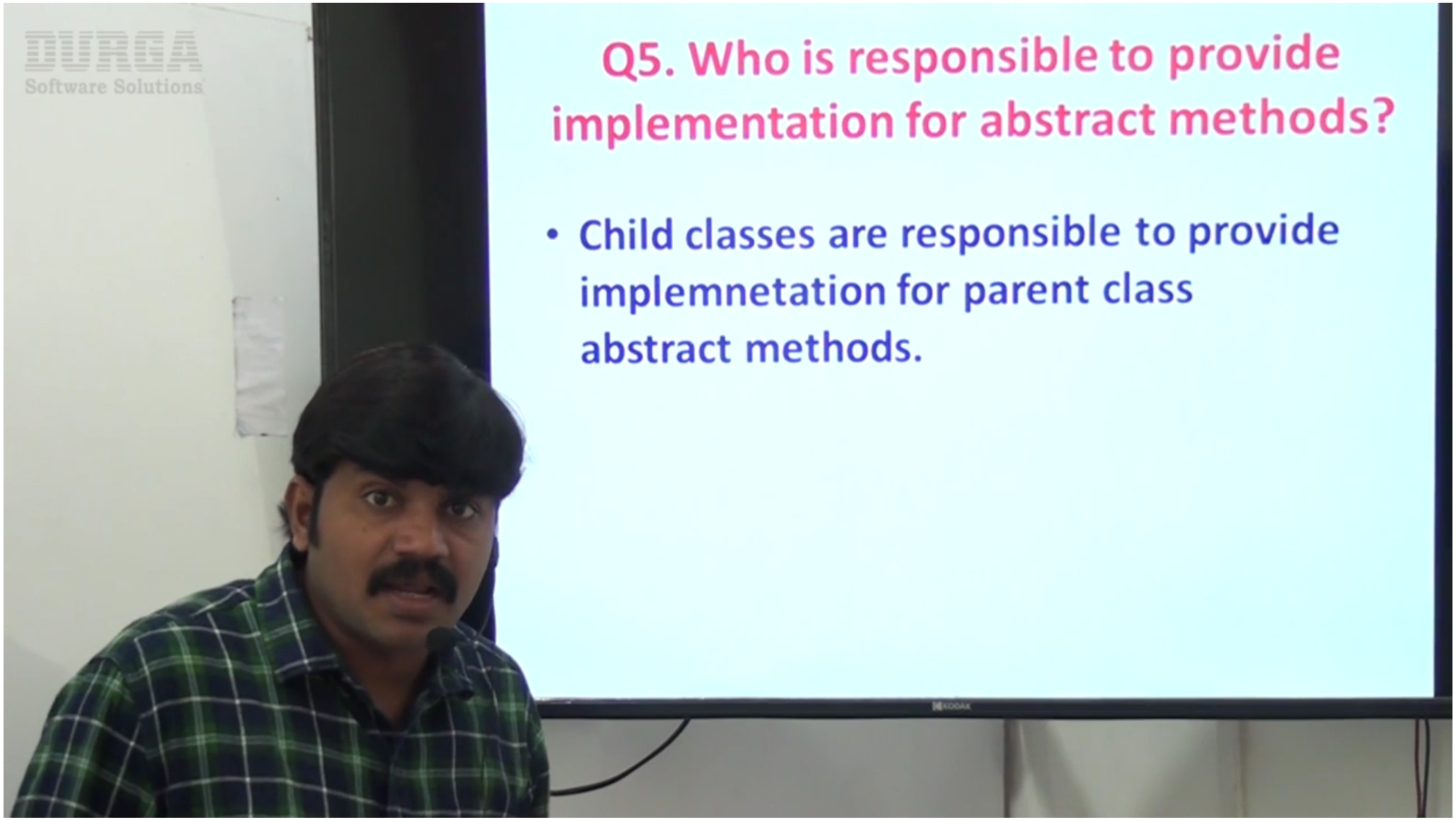

DURGA
Software Solutions
Abstract Method:
Sometimes we don't know about implementation, still we can declare a method. Such type of methods are called abstract methods. i.e abstract method has only declaration but not implementation(ie empty implementation)
In Python, we can declare abstract method by using @abstractmethod decorator.
abstractmethod decorator present in abc module. Hence compulsory we should import abc module.

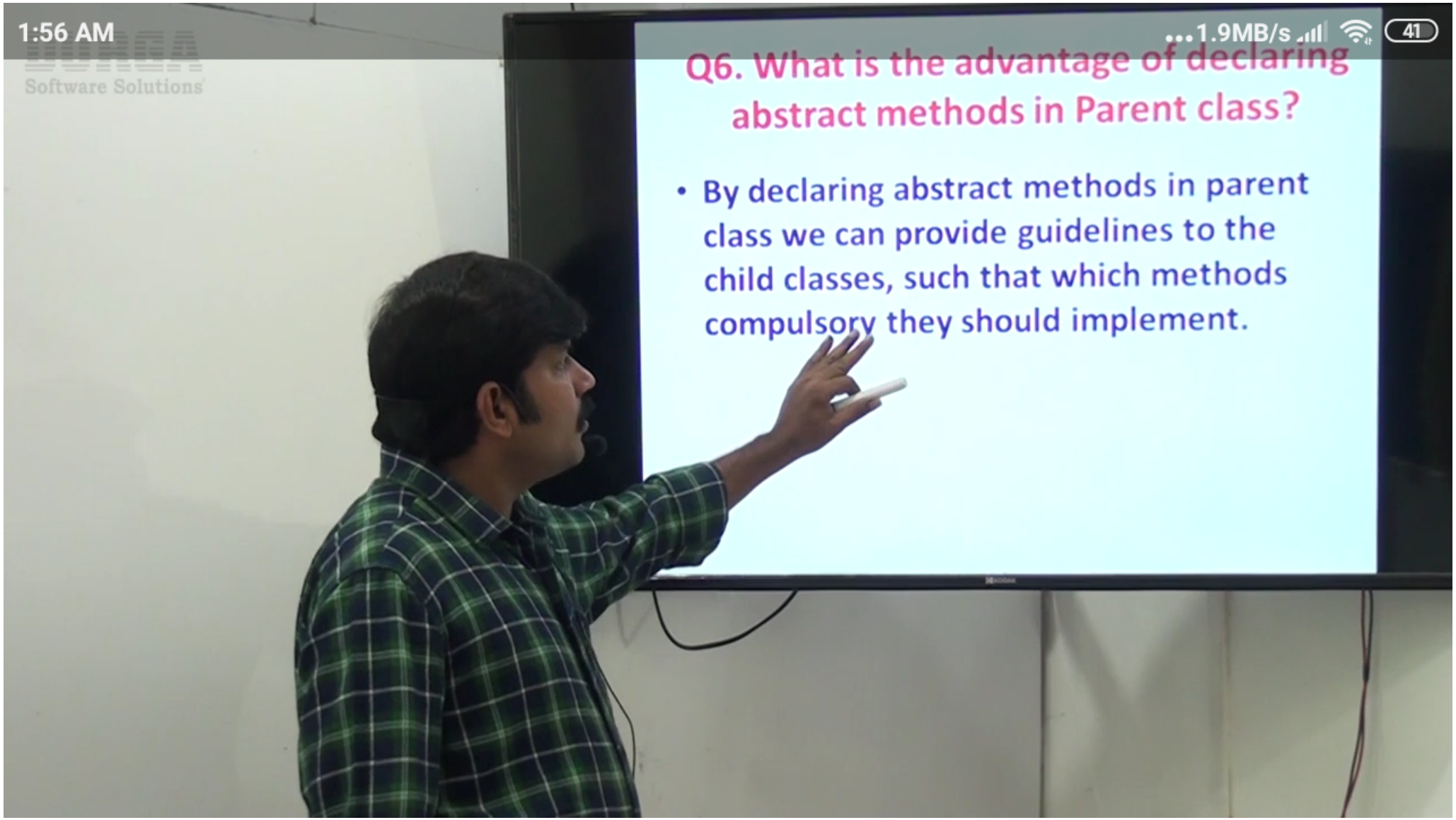

1:52 AM
...323KB/s
42
Software Solutions
Q1. What is abstract method?
• The method which has only declaration but not implementation(i.e empty implementation)

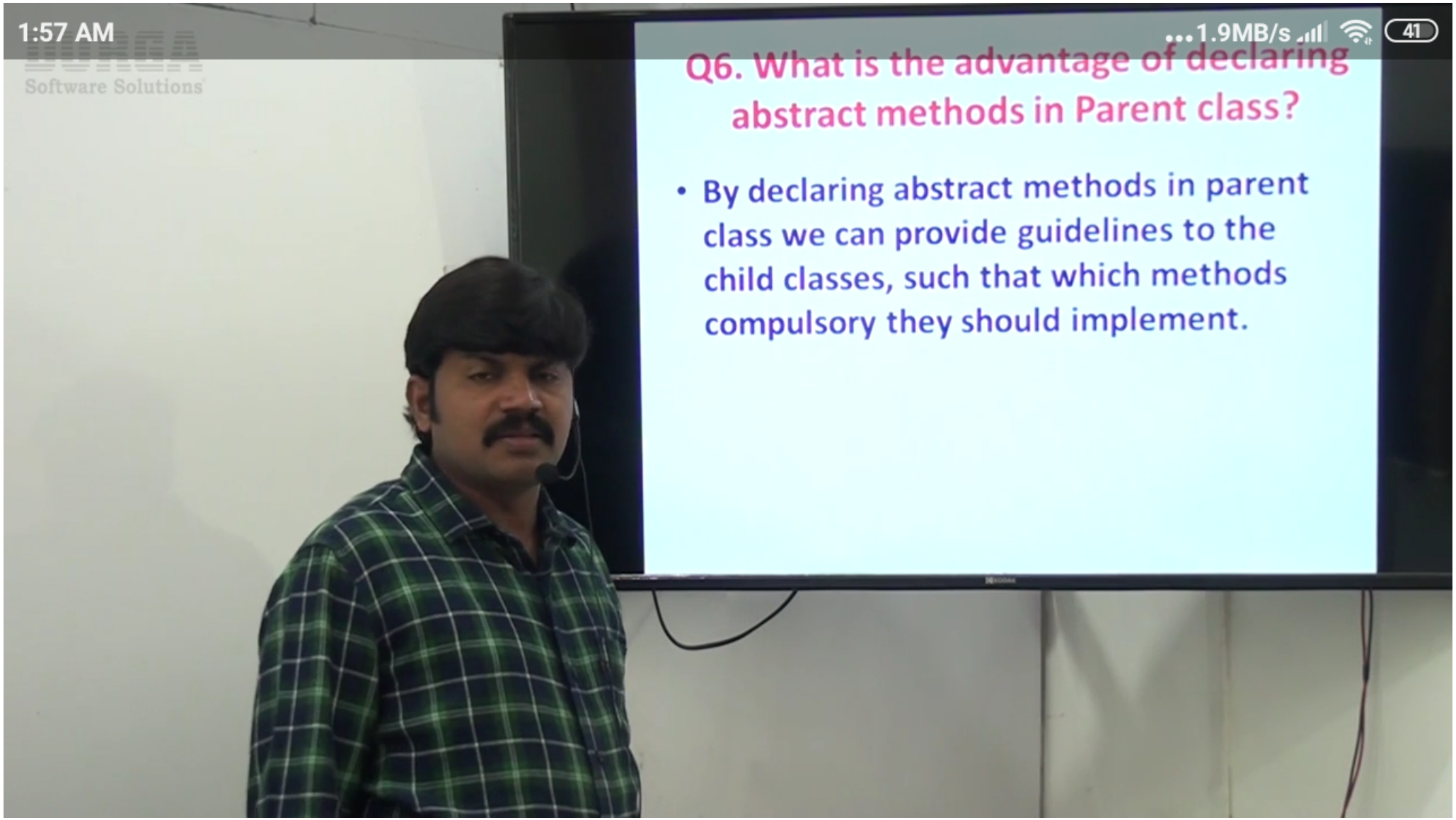

DURGA
Software Solutions
Q2. How to declare abstract method?
• By using @abstractmethod decorator

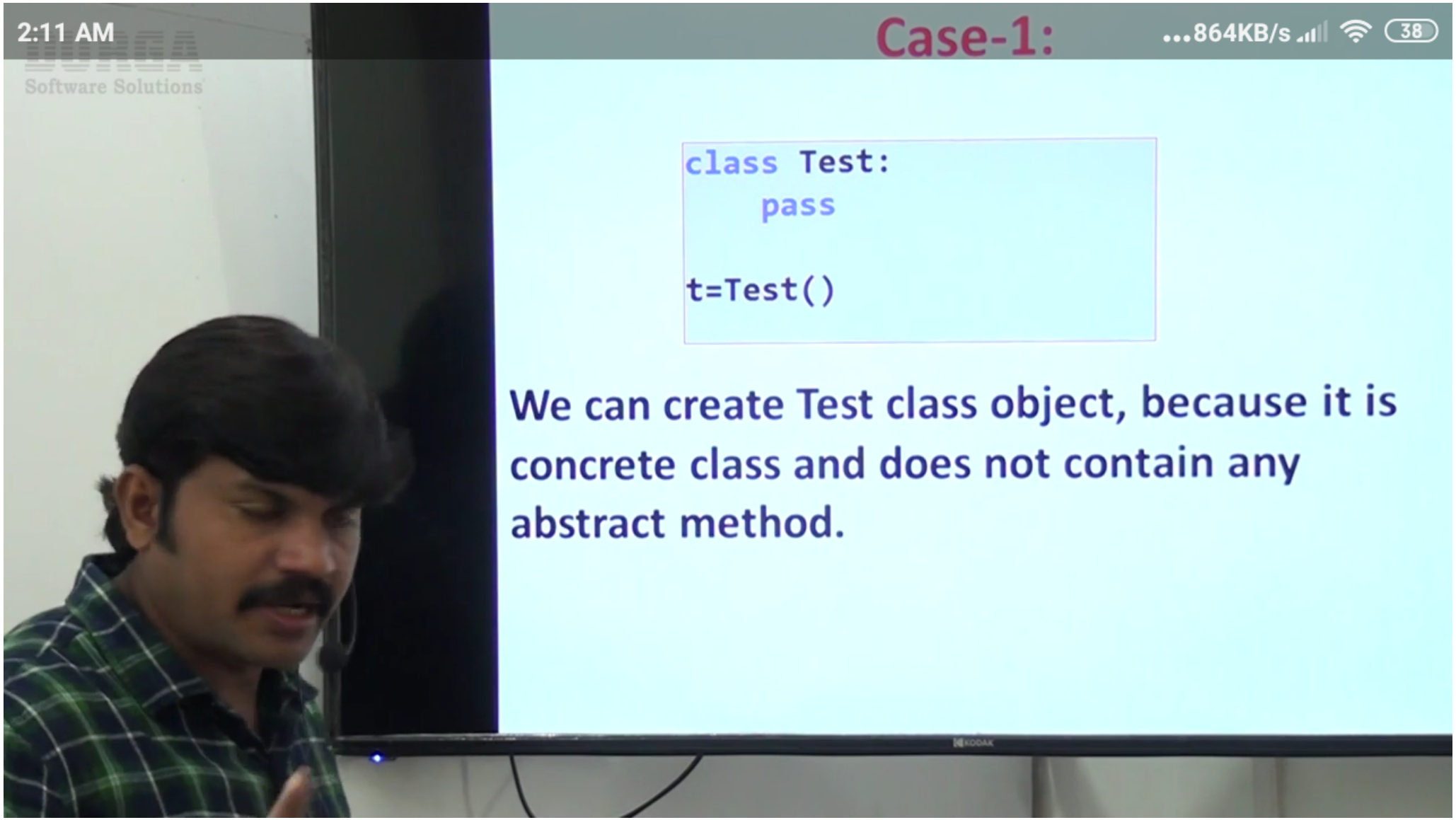

1:53 AM
...0.6KB/s
42
Software Solutions
Q3. What is abstract class?
• Partially implemented class is nothing but abstract class

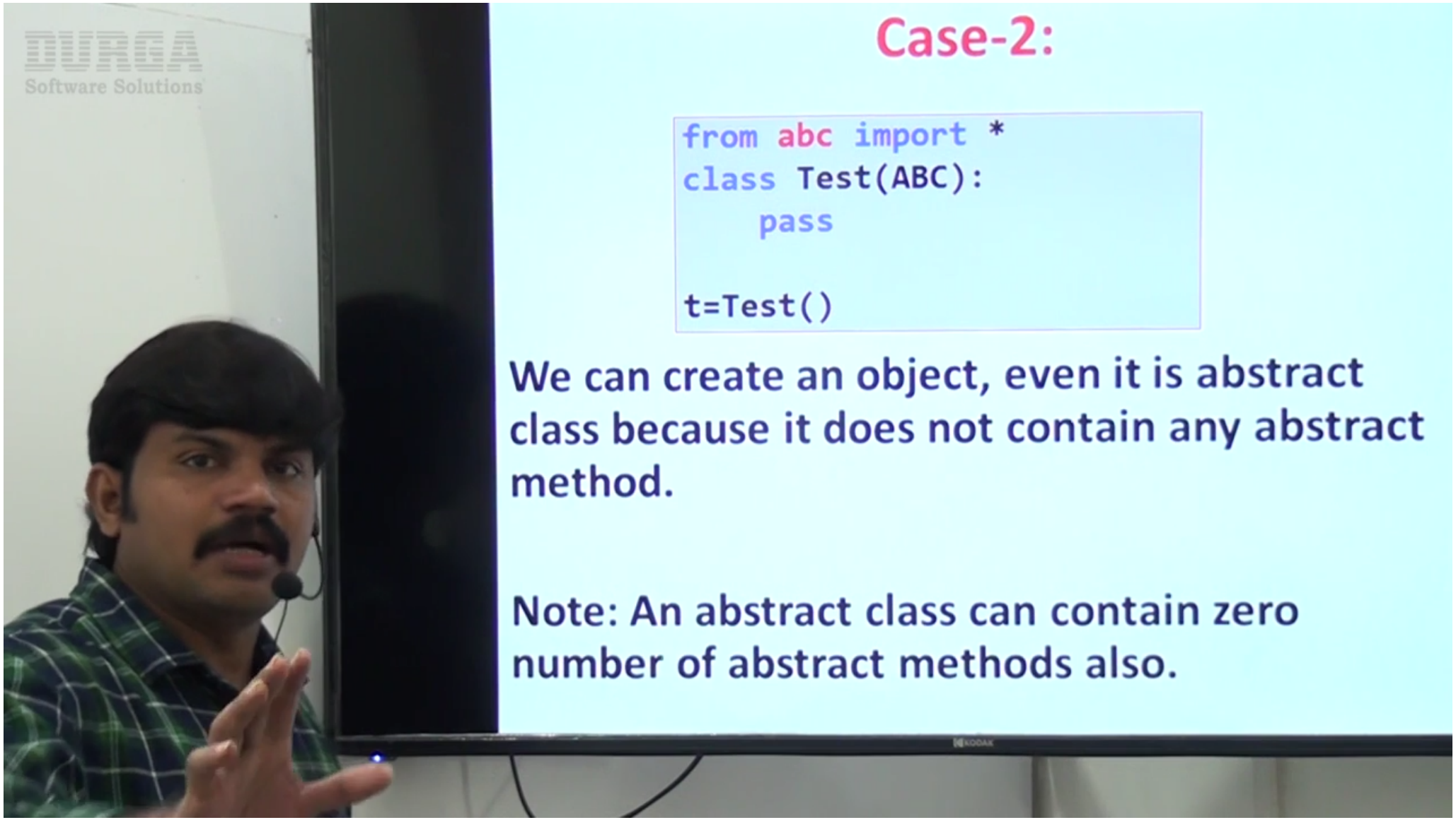

DURGA
Software Solutions
Q4. How to declare abstract class in python?
• The class should be child class of ABC

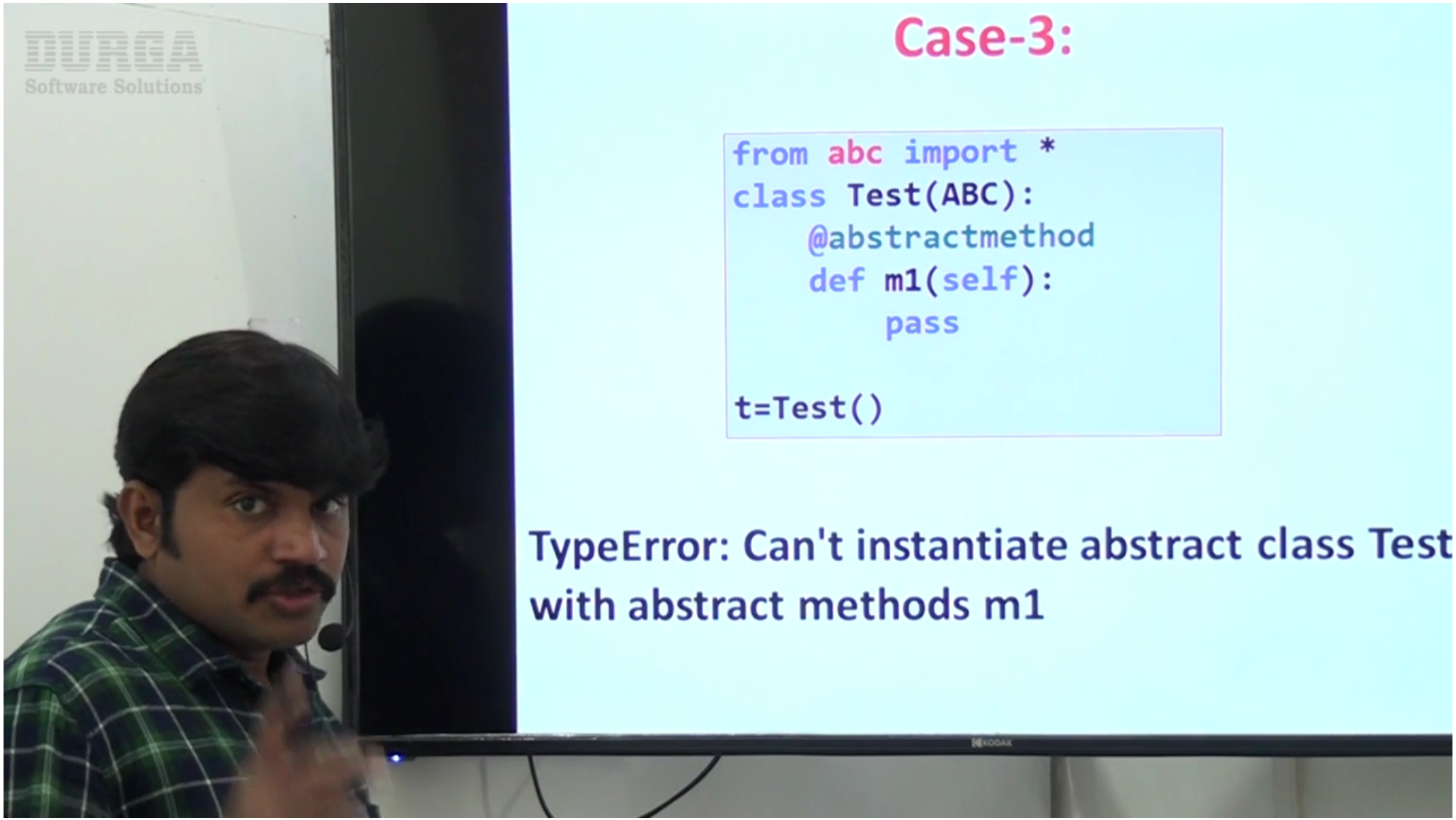

DURGA
Software Solutions
Q5. Who is responsible to provide implementation for abstract methods?
• Child classes are responsible to provide implemnetation for parent class abstract methods.

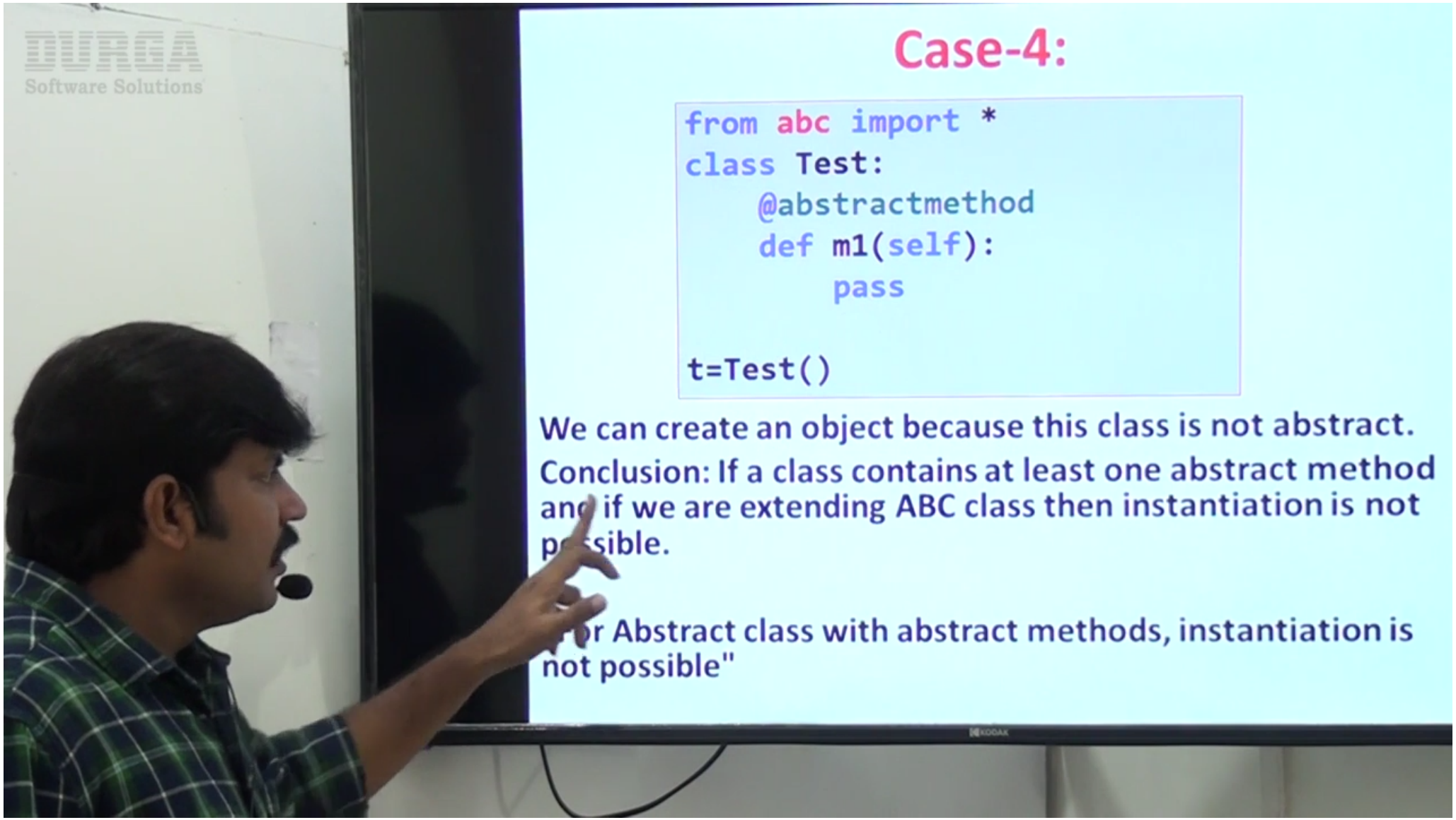

1:56 AM
1.9MB/s
41
Software Solutions
Q6. What is the advantage of declaring abstract methods in Parent class?
• By declaring abstract methods in parent class we can provide guidelines to the child classes, such that which methods compulsory they should implement.

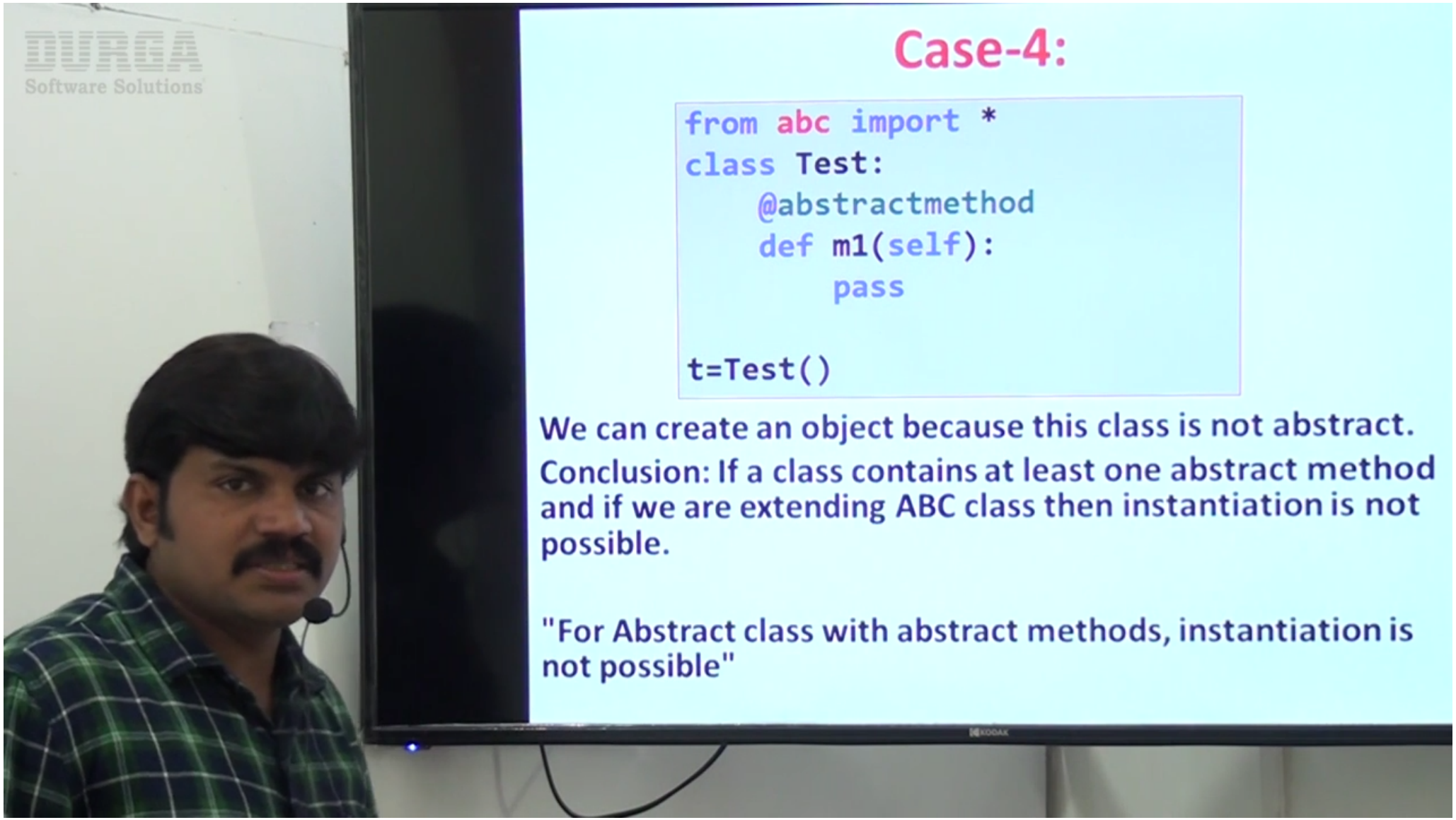

1:57 AM
1.9MB/s
41
Software Solutions
Q6. What is the advantage of declaring abstract methods in Parent class?
• By declaring abstract methods in parent class we can provide guidelines to the child classes, such that which methods compulsory they should implement.

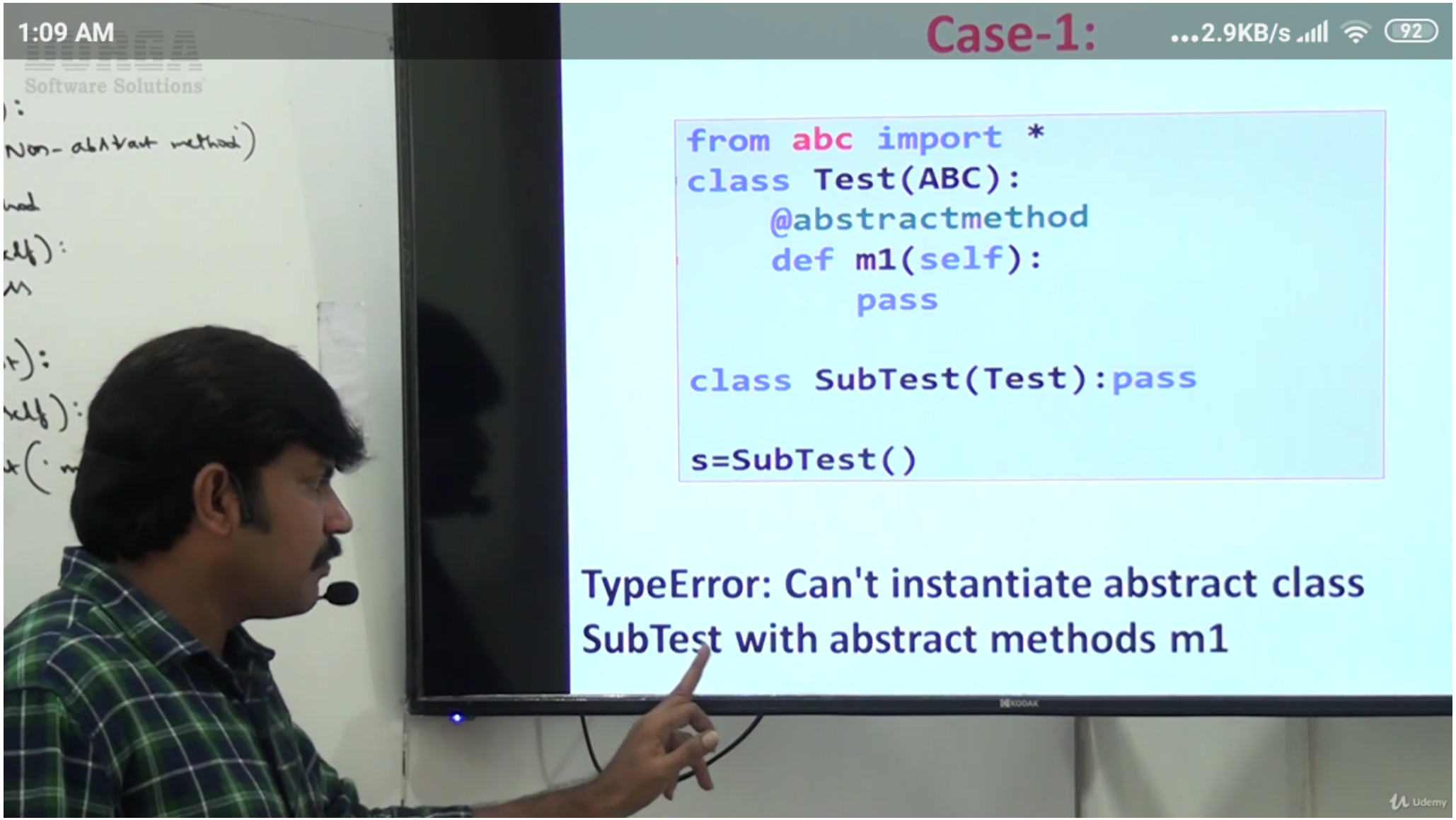

2:11 AM
...864KB/s
38
Software Solutions
Case-1:
class Test:
pass
t=Test()
We can create Test class object, because it is concrete class and does not contain any abstract method.

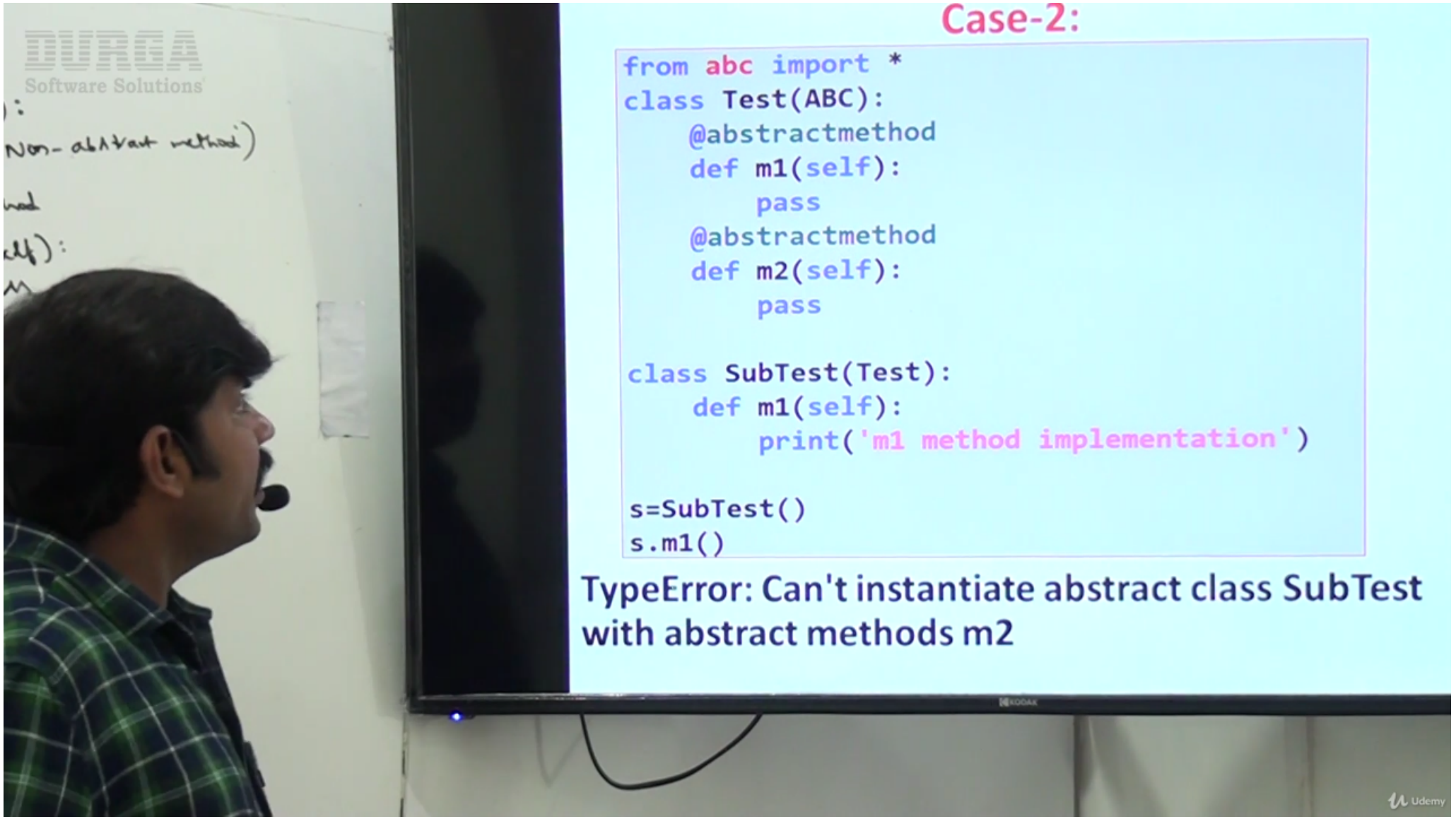

DURGA
Software Solutions
Case-2:
from abc import *
class Test(ABC):
pass
t=Test()
We can create an object, even it is abstract class because it does not contain any abstract method.
Note: An abstract class can contain zero number of abstract methods also.

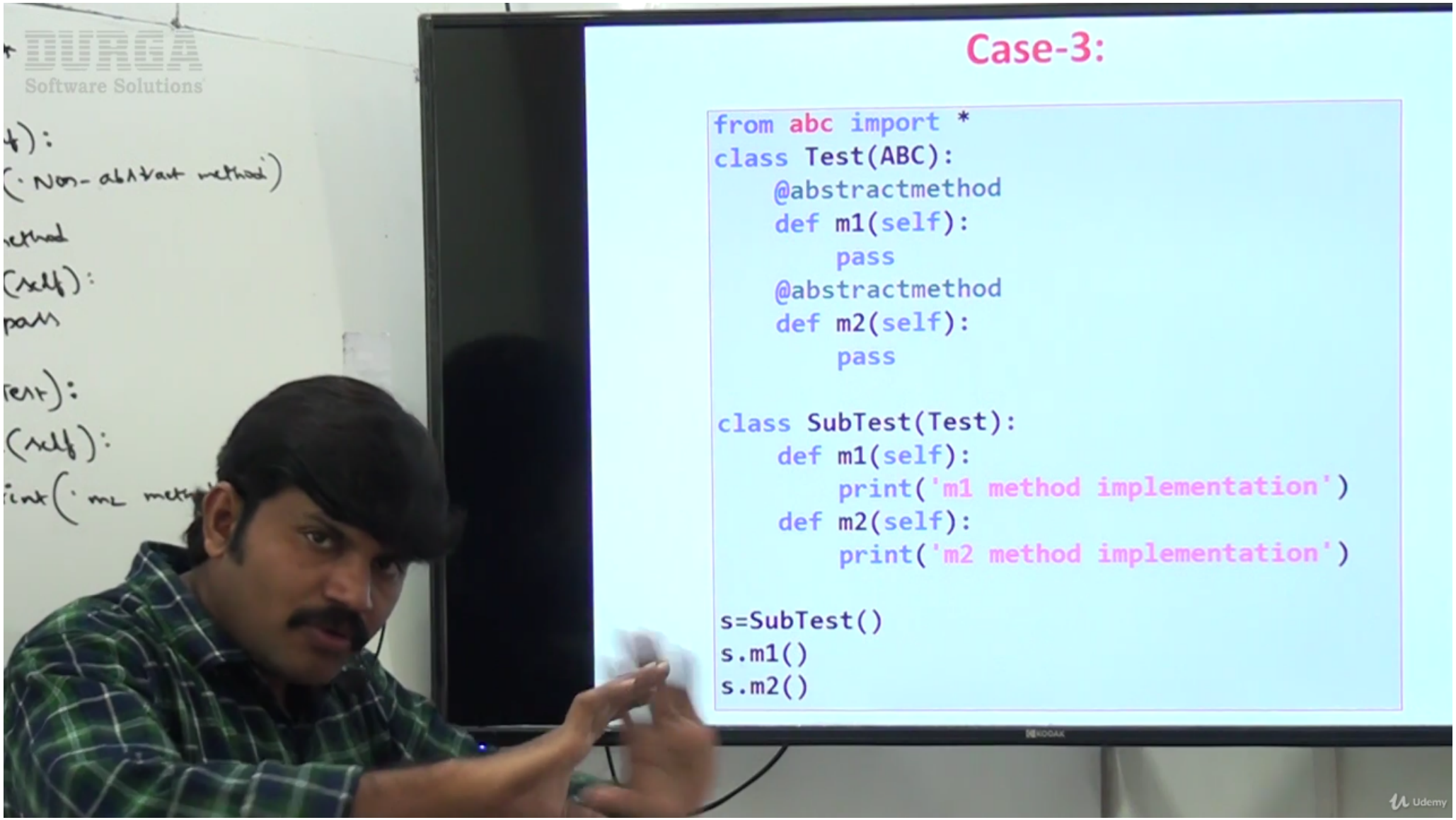

DURGA
Software Solutions
Case-3:
from abc import *
class Test(ABC):
@abstractmethod
def m1(self):
pass
t=Test()
TypeError: Can't instantiate abstract class Test
with abstract methods m1

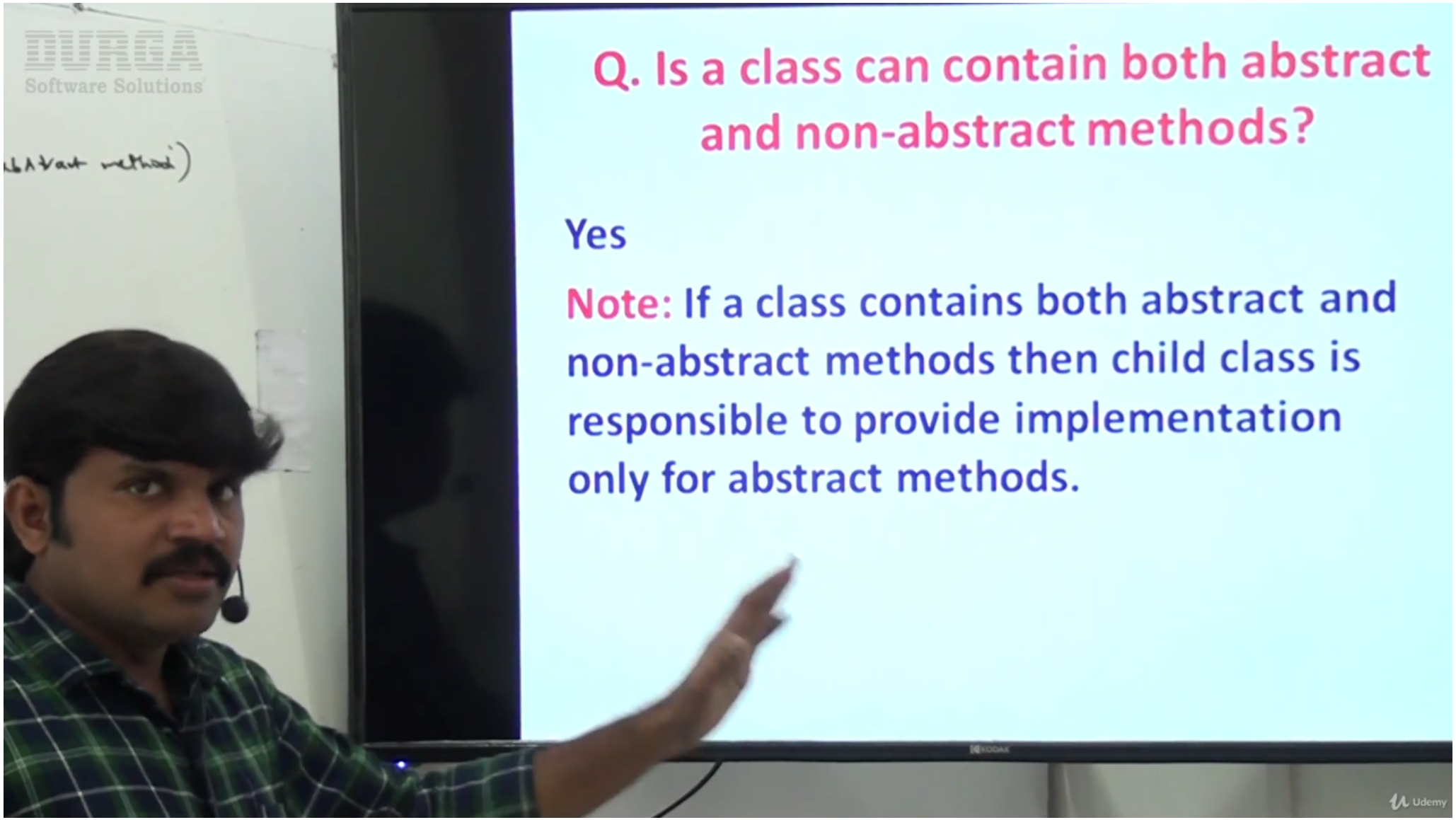

DURGA
Software Solutions
Case-4:
from abc import *
class Test:
@abstractmethod
def m1(self):
pass
t=Test()
We can create an object because this class is not abstract.
Conclusion: If a class contains at least one abstract method and if we are extending ABC class then instantiation is not possible.
Abstract class with abstract methods, instantiation is not possible"

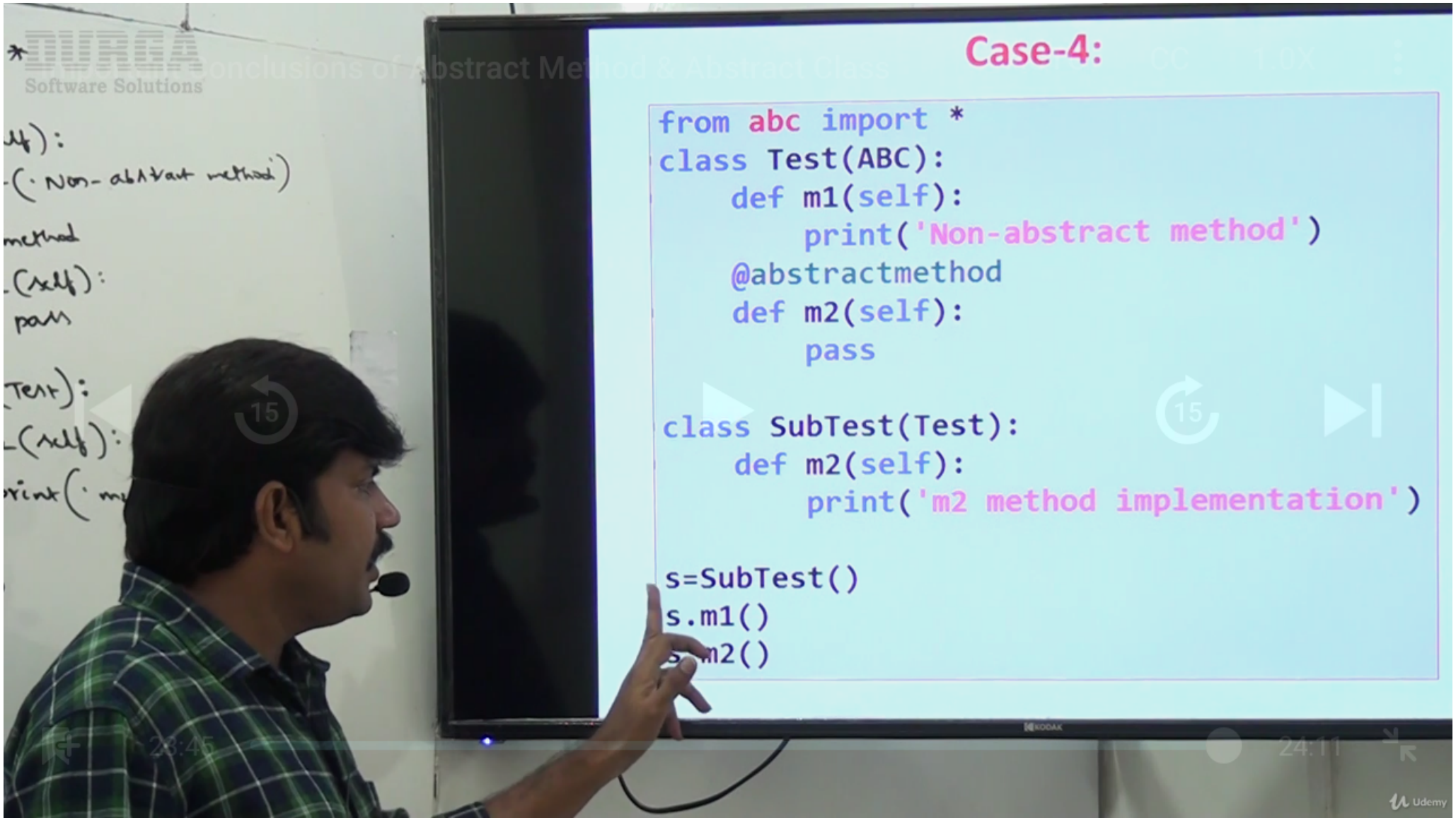

DURGA
Software Solutions
Case-4:
from abc import *
class Test:
@abstractmethod
def m1(self):
pass
t=Test()
We can create an object because this class is not abstract.
Conclusion: If a class contains at least one abstract method and if we are extending ABC class then instantiation is not possible.
"For Abstract class with abstract methods, instantiation is not possible"

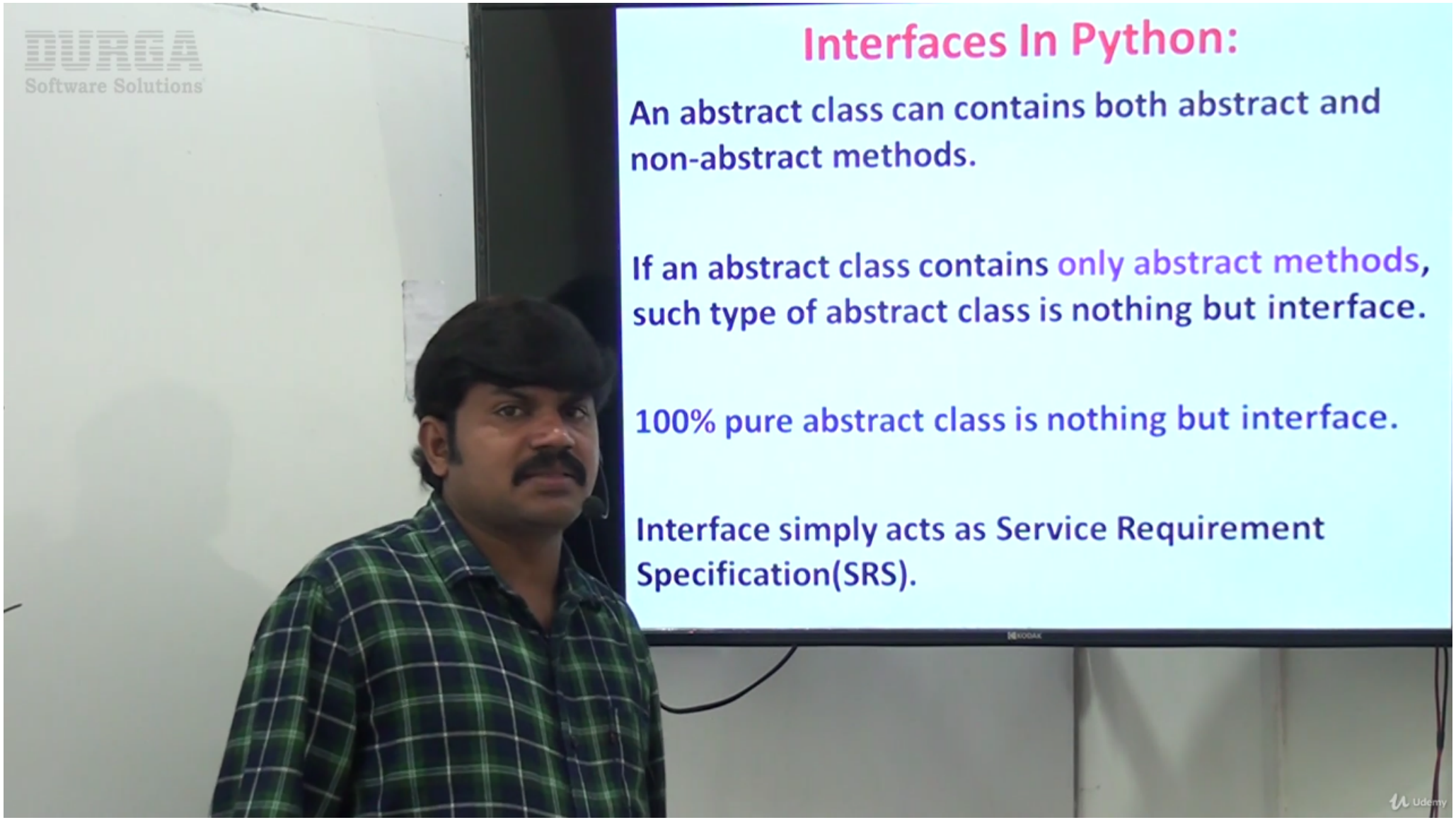

1:09 AM
Case-1:
...2.9KB/s
92
Software Solutions
from abc import *
class Test(ABC):
@abstractmethod
def m1(self):
pass
class SubTest(Test):pass
s=SubTest()
TypeError: Can't instantiate abstract class
SubTest with abstract methods m1
Udemy

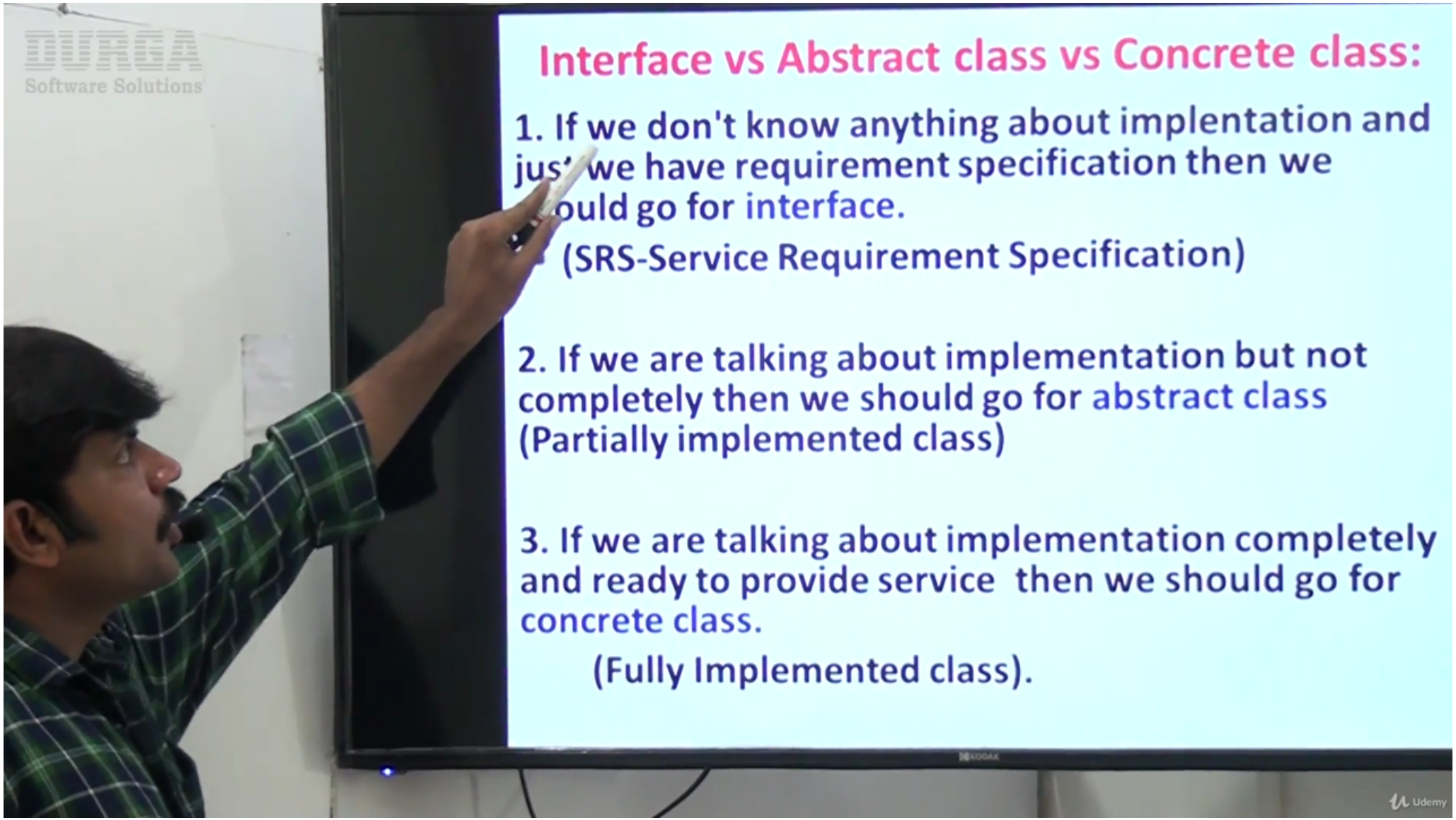

DURGA
Software Solutions
Case-2:
from abc import *
class Test(ABC):
@abstractmethod
def m1(self):
pass
@abstractmethod
def m2(self):
pass
class SubTest(Test):
def m1(self):
print('m1 method implementation')
s=SubTest()
s.m1()
TypeError: Can't instantiate abstract class SubTest with abstract methods m2
Udemy

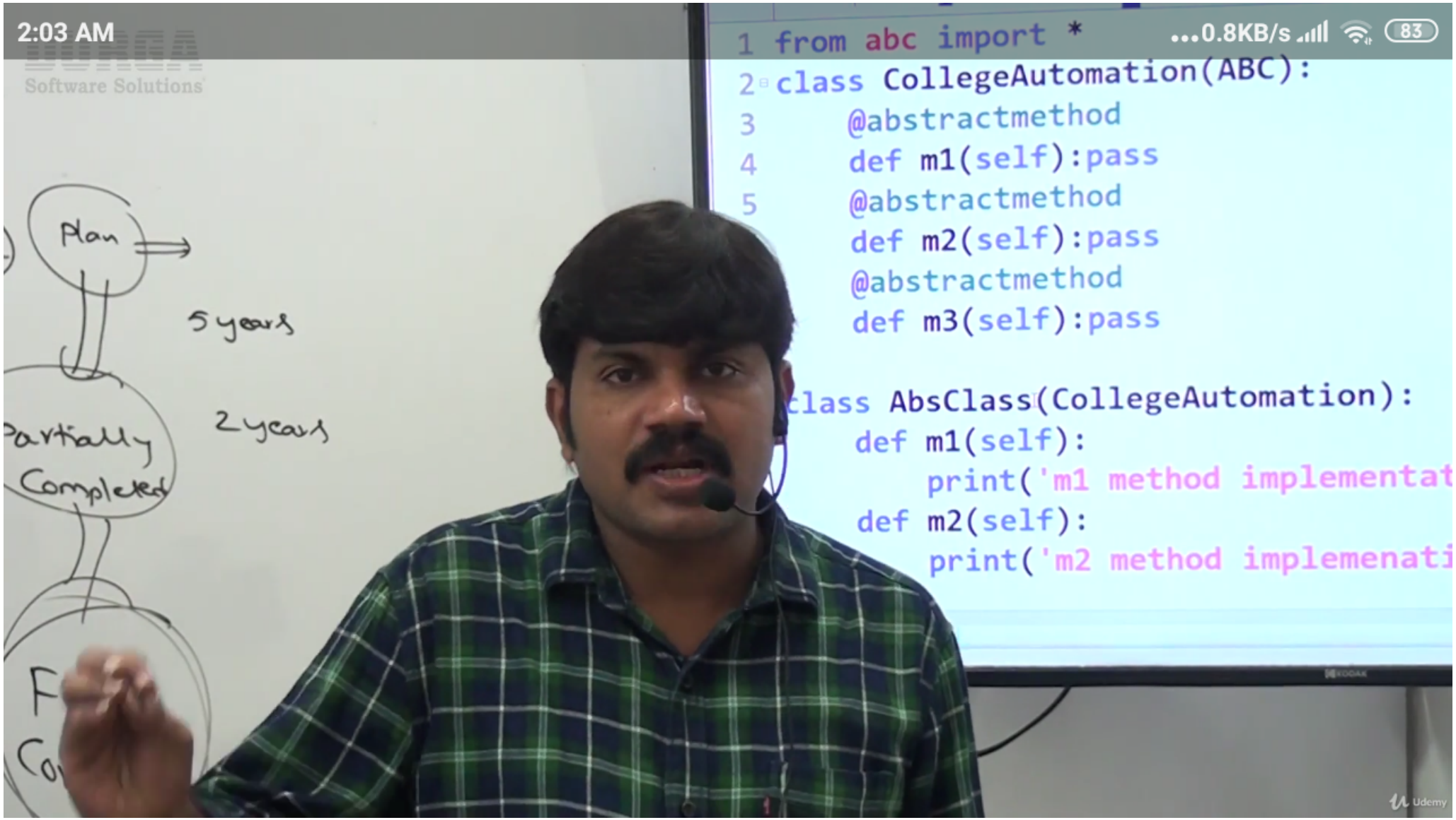

DURGA
Software Solutions
Case-3:
from abc import *
class Test(ABC):
@abstractmethod
def m1(self):
pass
@abstractmethod
def m2(self):
pass
class SubTest(Test):
def m1(self):
print('m1 method implementation')
def m2(self):
print('m2 method implementation')
s=SubTest()
s.m1()
s.m2()
Udemy

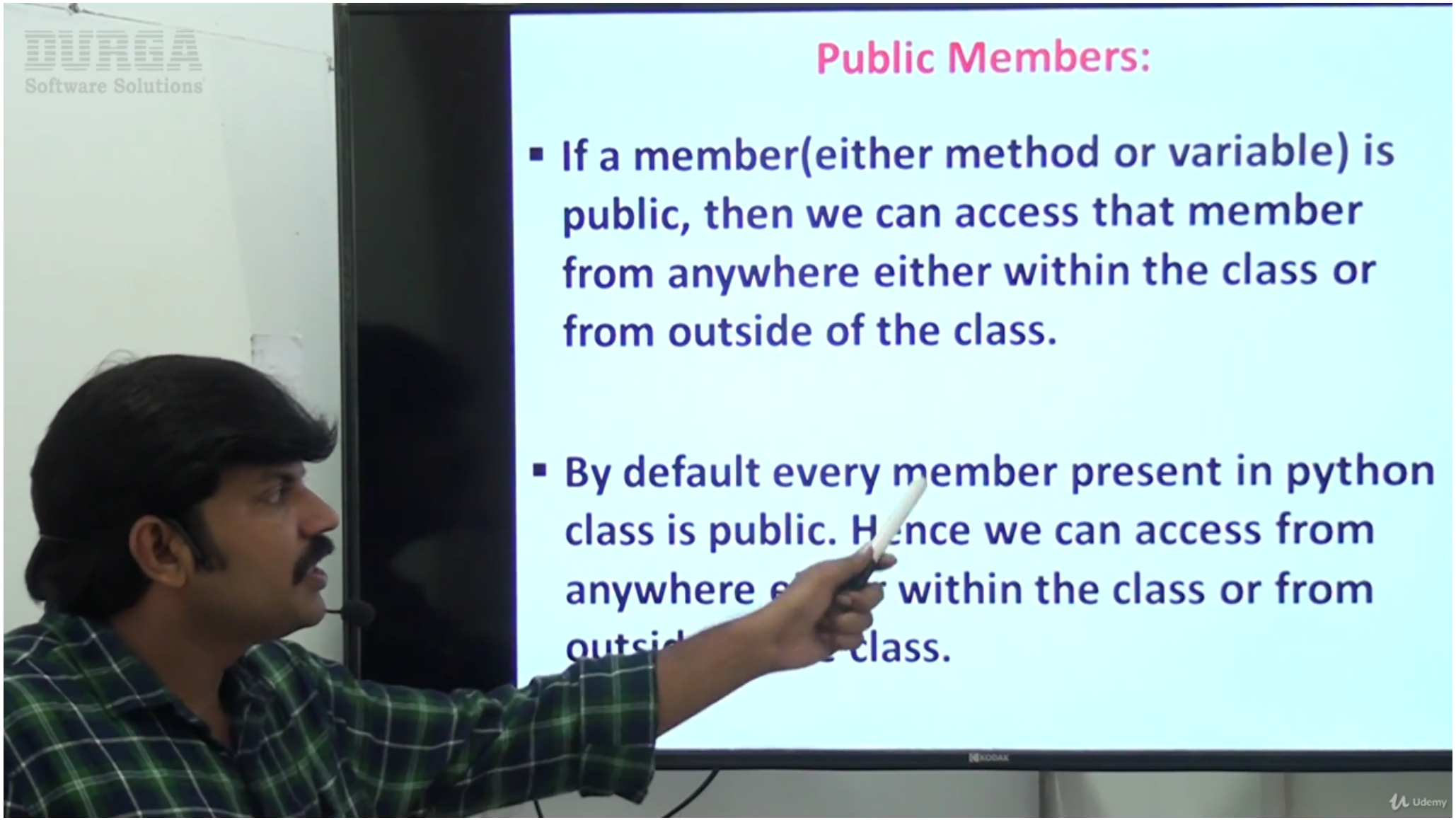

DURGA
Software Solutions
Q. Is a class can contain both abstract and non-abstract methods?
Yes
Note: If a class contains both abstract and non-abstract methods then child class is responsible to provide implementation only for abstract methods.
Udemy

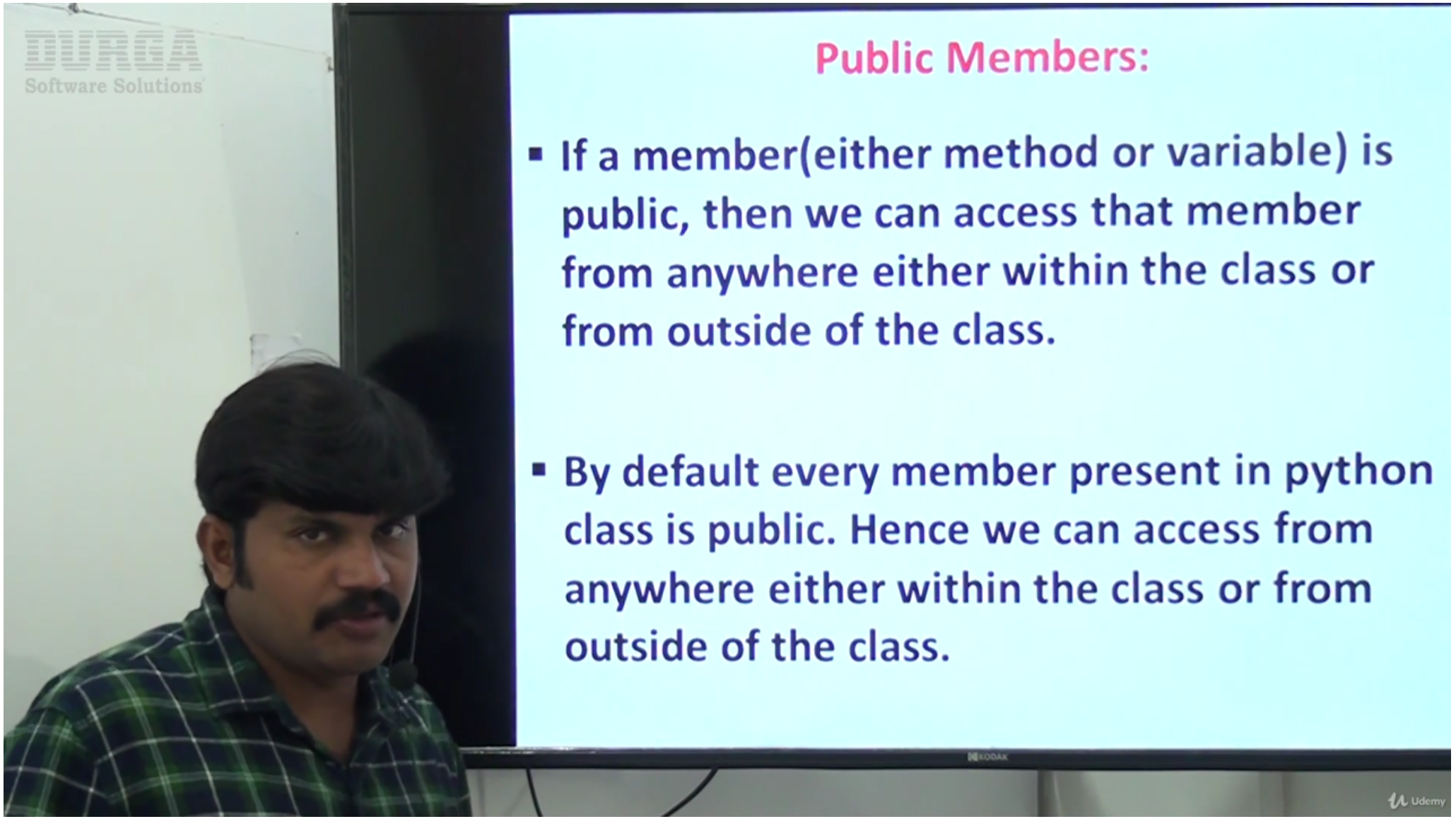

Case-4:
from abc import *
class Test(ABC):
def m1(self):
print('Non-abstract method')
@abstractmethod
def m2(self):
pass
class SubTest(Test):
def m2(self):
print('m2 method implementation')
s=SubTest()
s.m1()
s.m2()

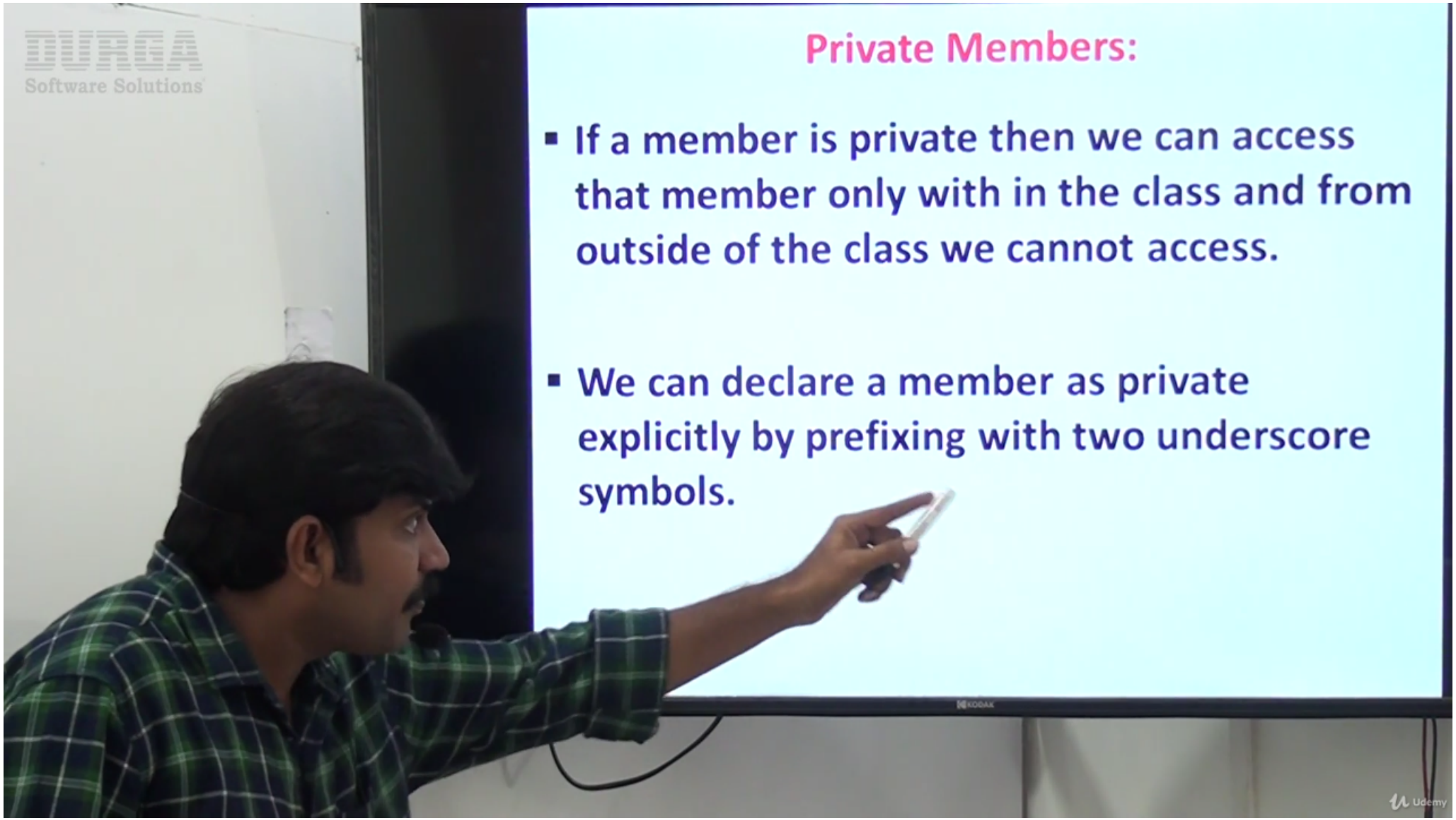

DURGA
Software Solutions
Interfaces In Python:
An abstract class can contains both abstract and non-abstract methods.
If an abstract class contains only abstract methods, such type of abstract class is nothing but interface.
100% pure abstract class is nothing but interface.
Interface simply acts as Service Requirement Specification(SRS).
Udemy


DURGA
Software Solutions
Interface vs Abstract class vs Concrete class:
1. If we don't know anything about implentation and just we have requirement specification then we should go for interface.
(SRS-Service Requirement Specification)
2. If we are talking about implementation but not completely then we should go for abstract class (Partially implemented class)
3. If we are talking about implementation completely and ready to provide service then we should go for concrete class.
(Fully Implemented class).
Udemy

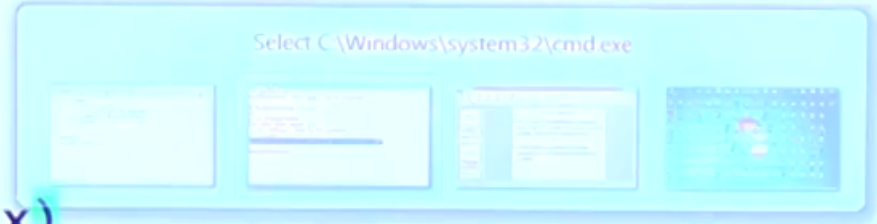

2:03 AM
...0.8KB/s
83
Software Solutions
Plan
5 years
2 years
Partially
Completed
from abc import *
class CollegeAutomation(ABC):
@abstractmethod
def m1(self):pass
@abstractmethod
def m2(self):pass
@abstractmethod
def m3(self):pass
class AbsClass(CollegeAutomation):
def m1(self):
print('m1 method implementat
def m2(self):
print('m2 method implemenati
Udemy


DURGA
Software Solutions
Public Members:
If a member(either method or variable) is public, then we can access that member from anywhere either within the class or from outside of the class.
By default every member present in python class is public. Hence we can access from anywhere either within the class or from outside class.
KODAK
Udemy

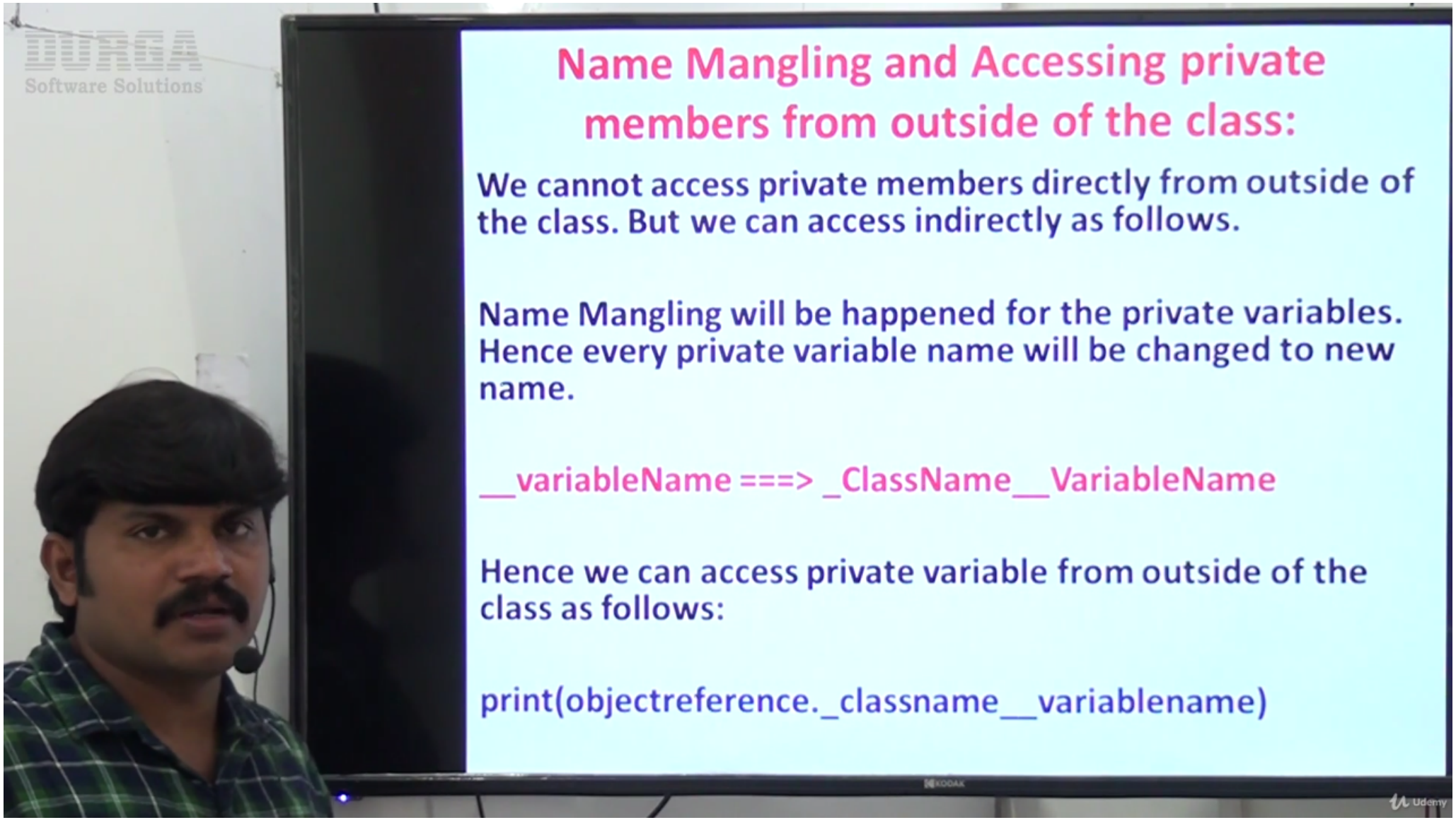

DURGA
Software Solutions
Public Members:
If a member(either method or variable) is public, then we can access that member from anywhere either within the class or from outside of the class.
By default every member present in python class is public. Hence we can access from anywhere either within the class or from outside of the class.
KODAK
Udemy

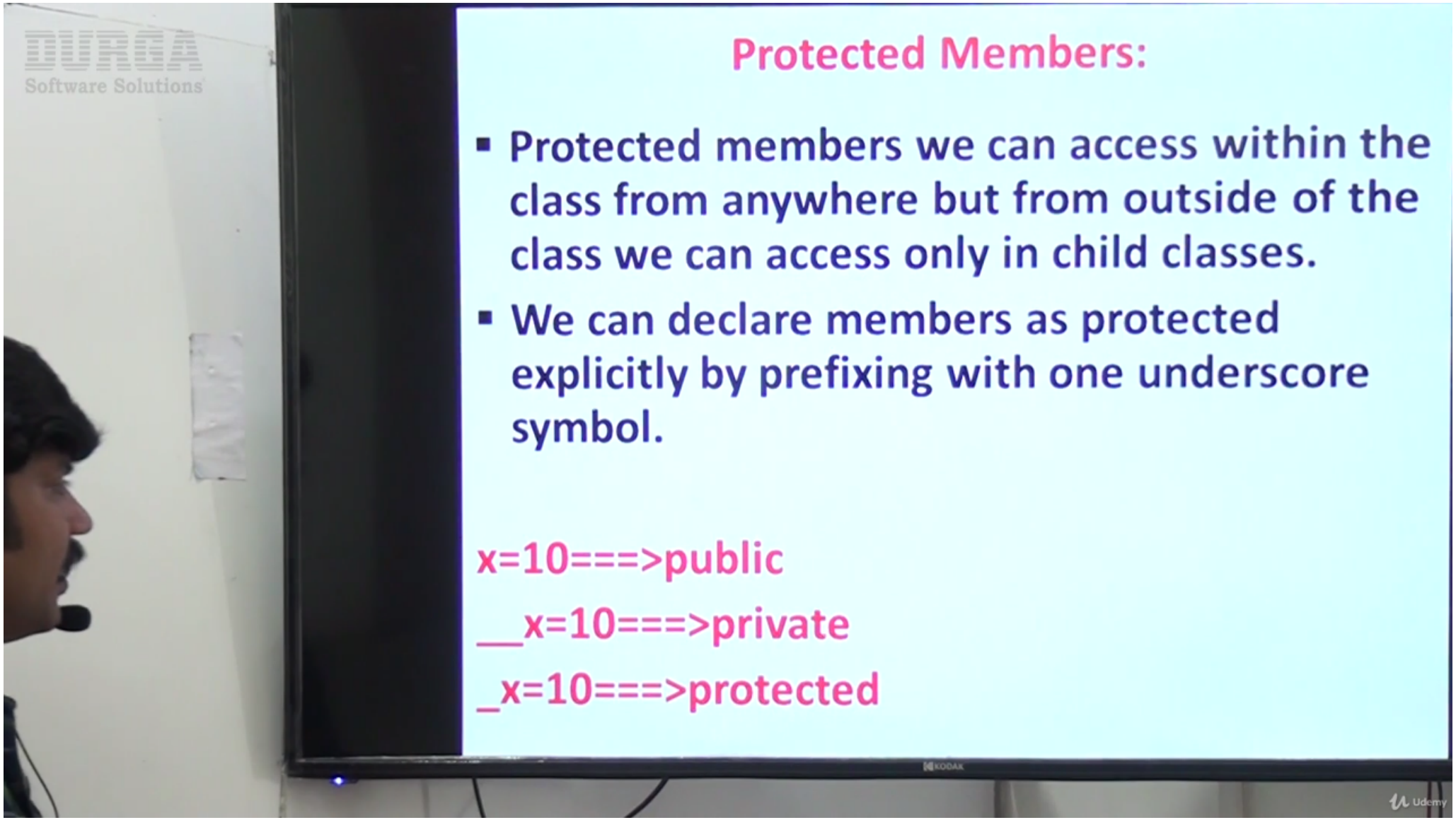

DURGA
Software Solutions
Private Members:
If a member is private then we can access that member only with in the class and from outside of the class we cannot access.
We can declare a member as private explicitly by prefixing with two underscore symbols.
Udemy

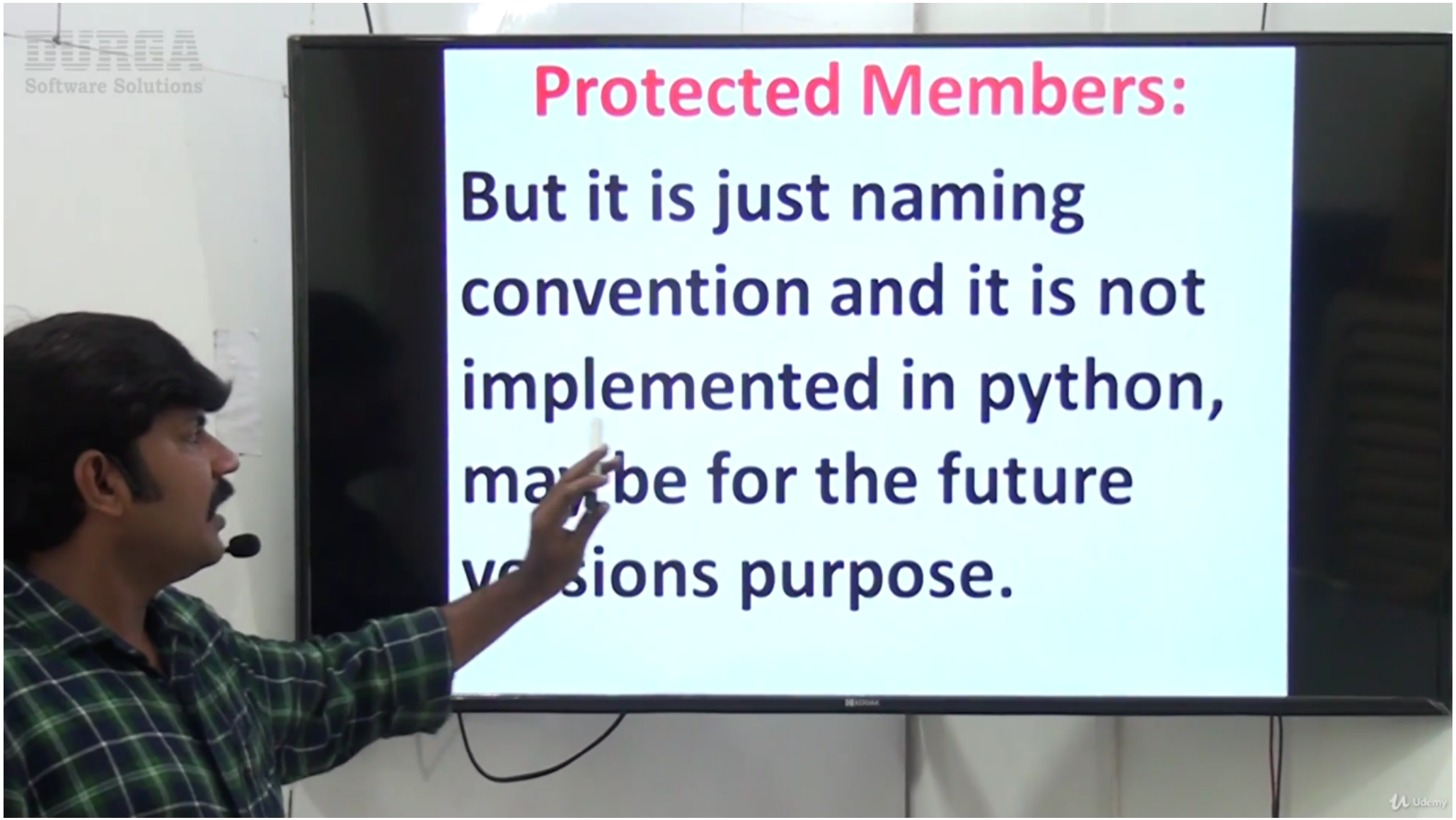
```
class Test:
    def __init__(self):
        self.__x=10 #private variable
    def __m1(self): #private method
        print('It is private method')


t=Test()
print(t._Test__x)
```

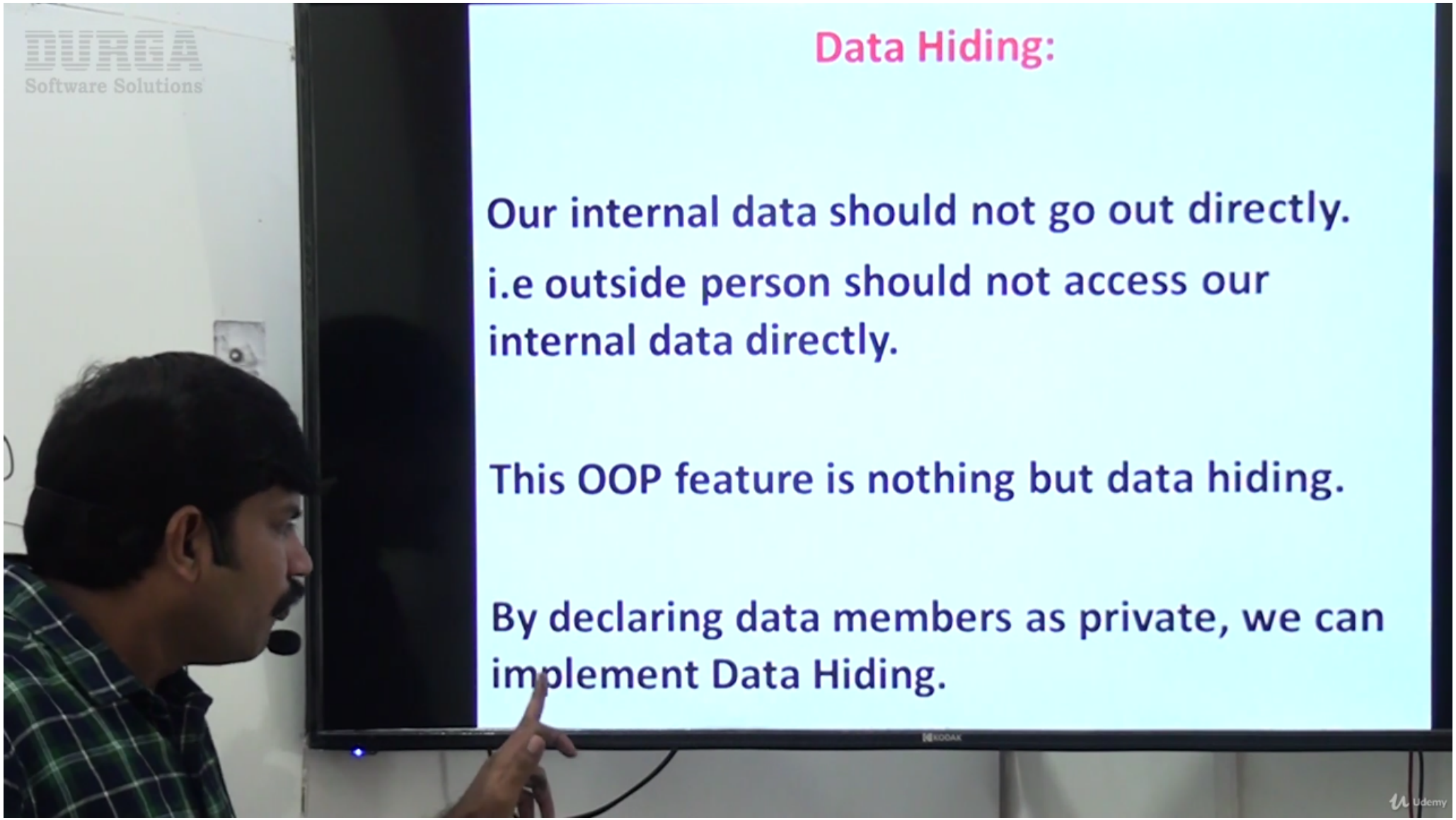


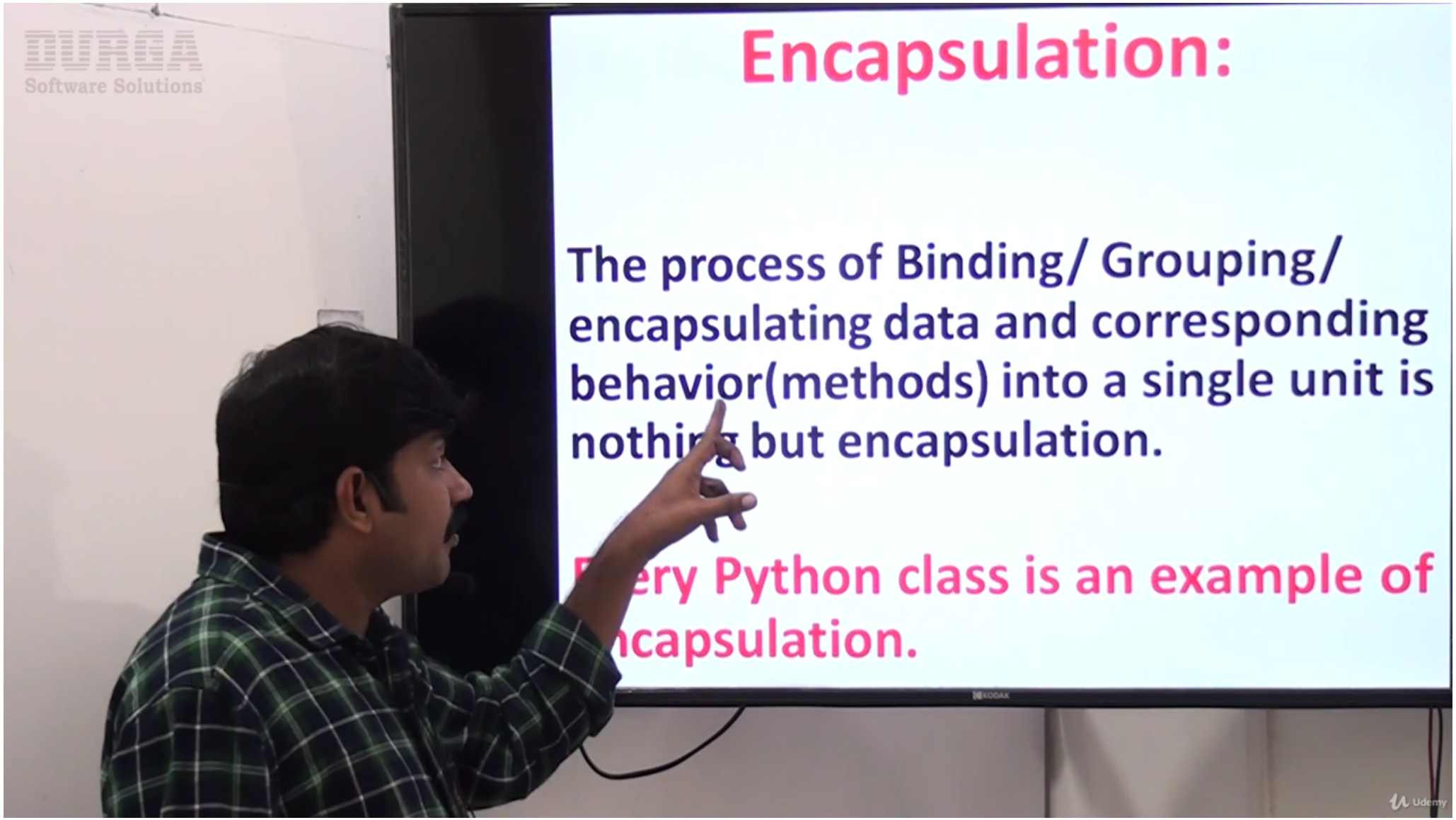
inner_classes_runn
test.py
For Help, press F1
ln 9 col 18 11 00 PC ANSI

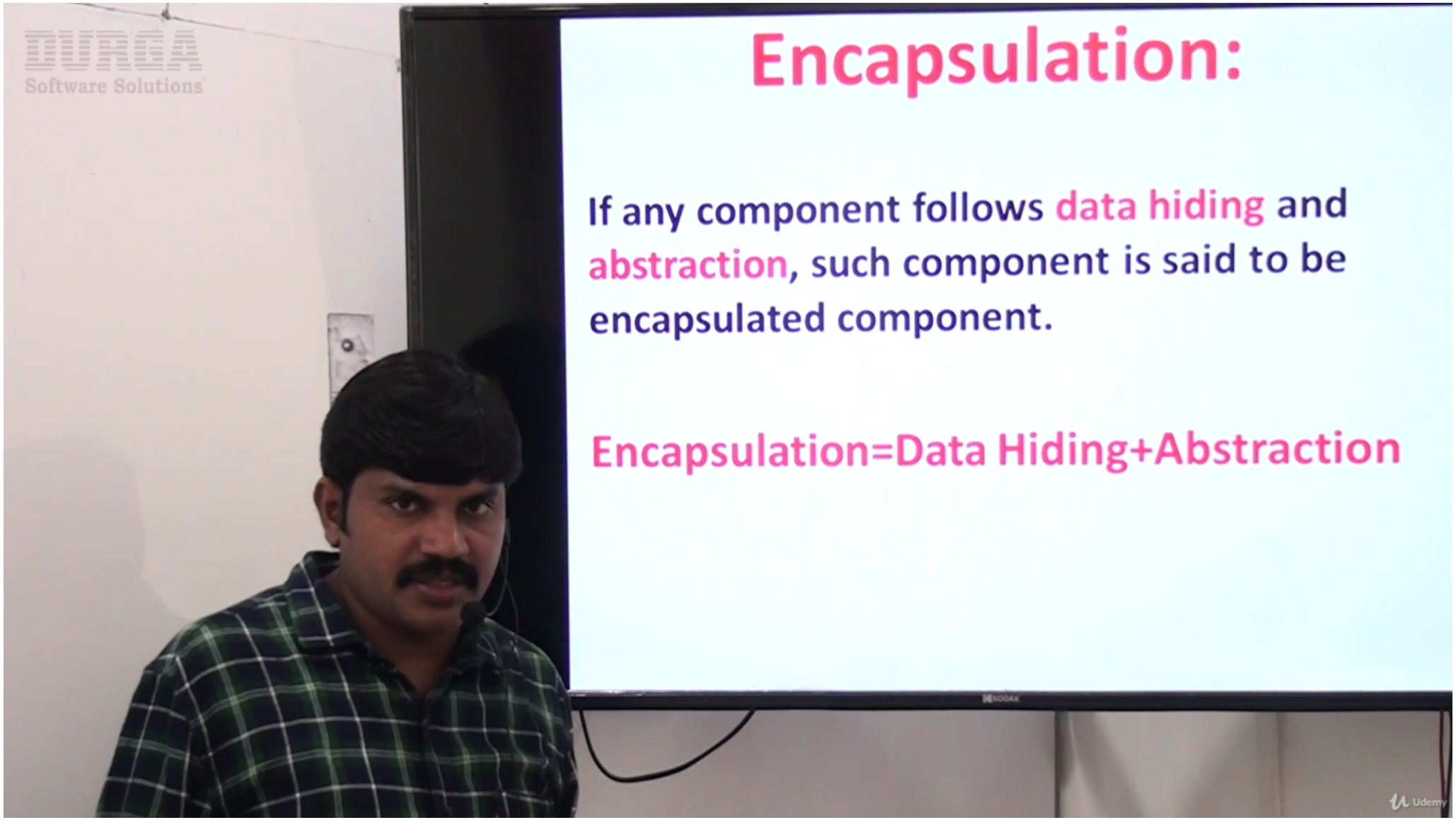

DURGA
Software Solutions
Name Mangling and Accessing private members from outside of the class:
We cannot access private members directly from outside of the class. But we can access indirectly as follows.
Name Mangling will be happened for the private variables. Hence every private variable name will be changed to new name.
__variableName ===> _ClassName__VariableName
Hence we can access private variable from outside of the class as follows:
print(objectreference._classname__variablename)
KODAK
Udemy

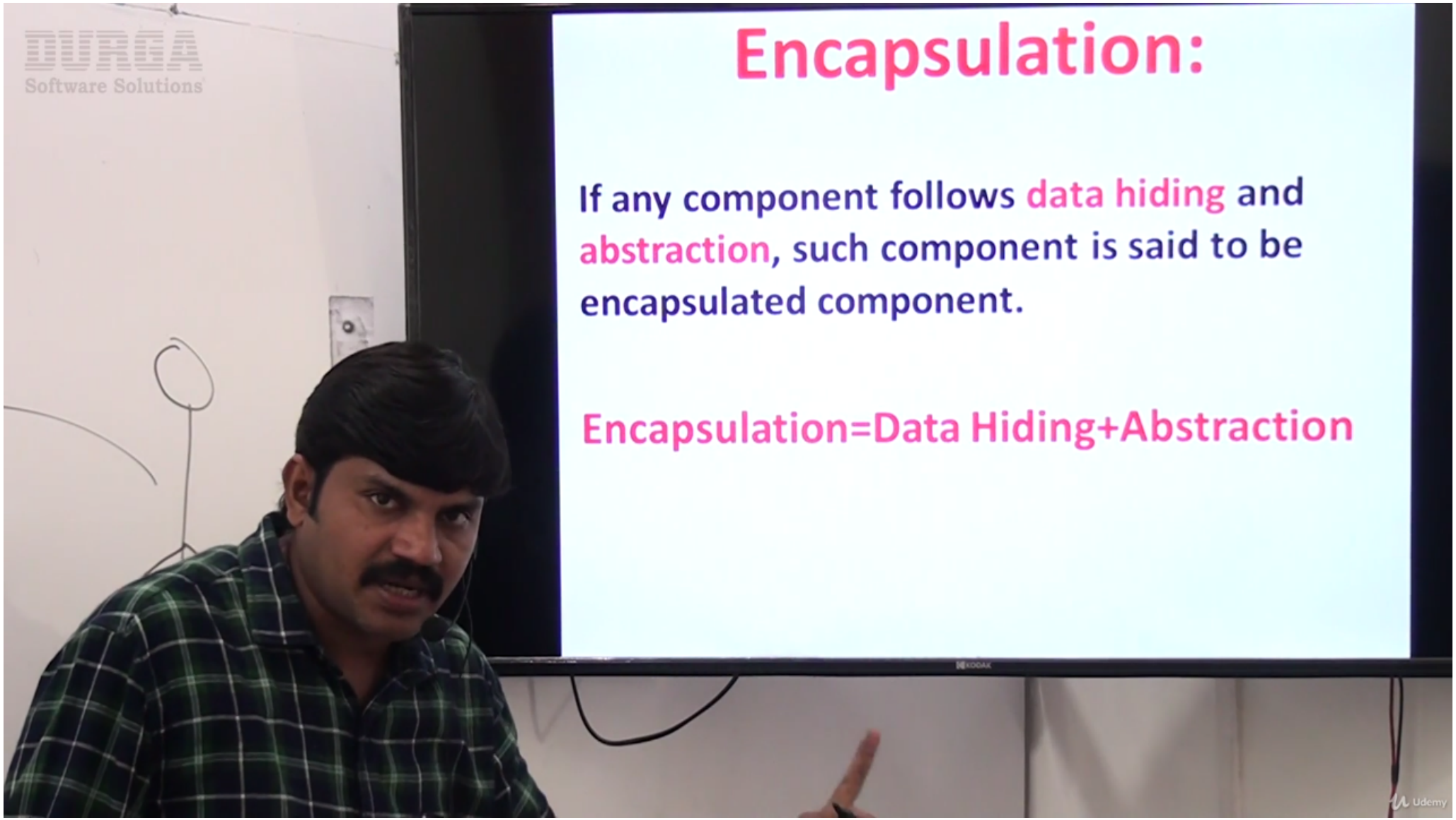

DURGA
Software Solutions
Protected Members:
Protected members we can access within the class from anywhere but from outside of the class we can access only in child classes.
We can declare members as protected explicitly by prefixing with one underscore symbol.
x=10===>public
__x=10===>private
_x=10===>protected
KODAK
Udemy

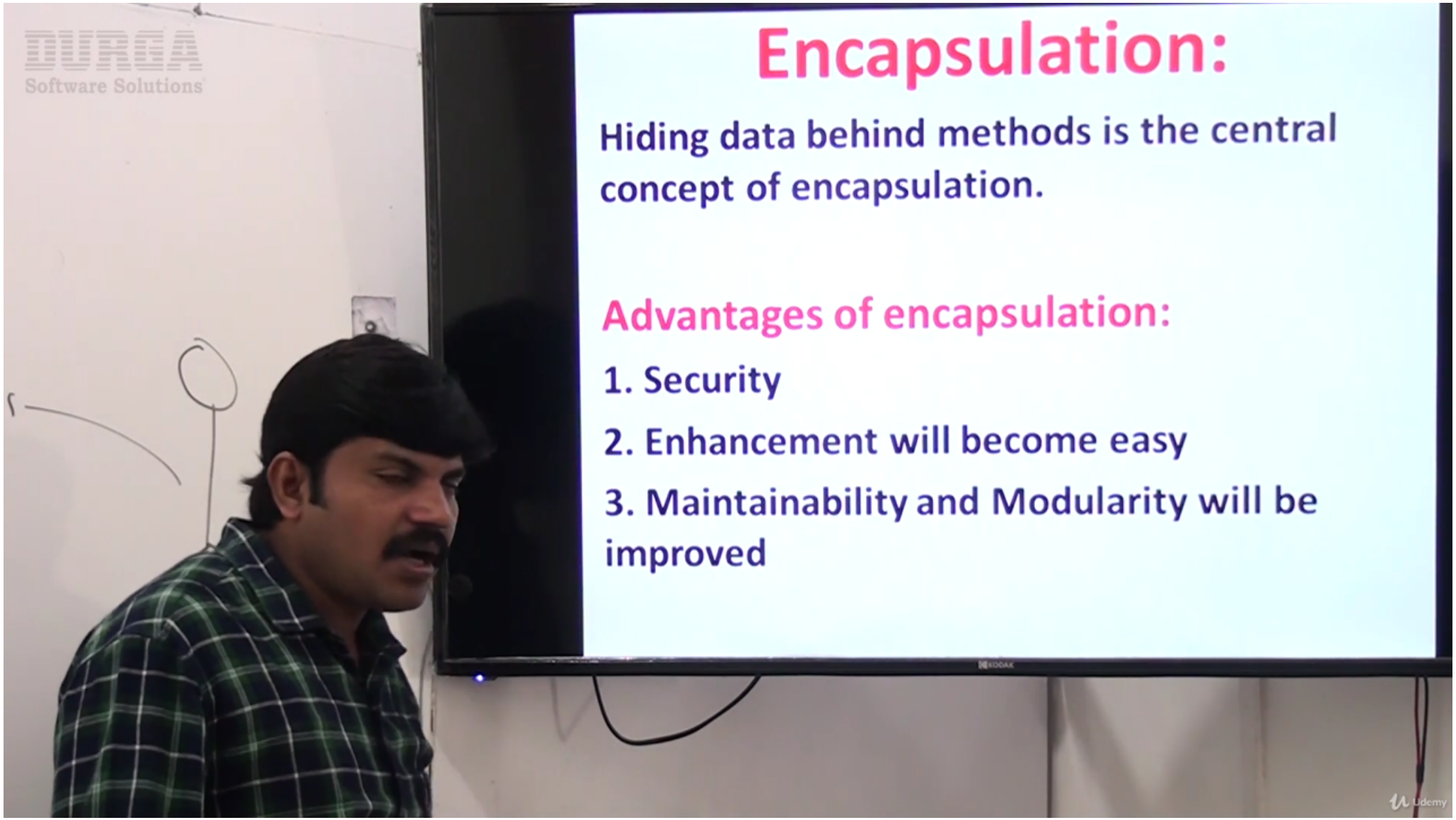

DURGA
Software Solutions
Protected Members:
But it is just naming convention and it is not implemented in python, maybe for the future versions purpose.
Udemy

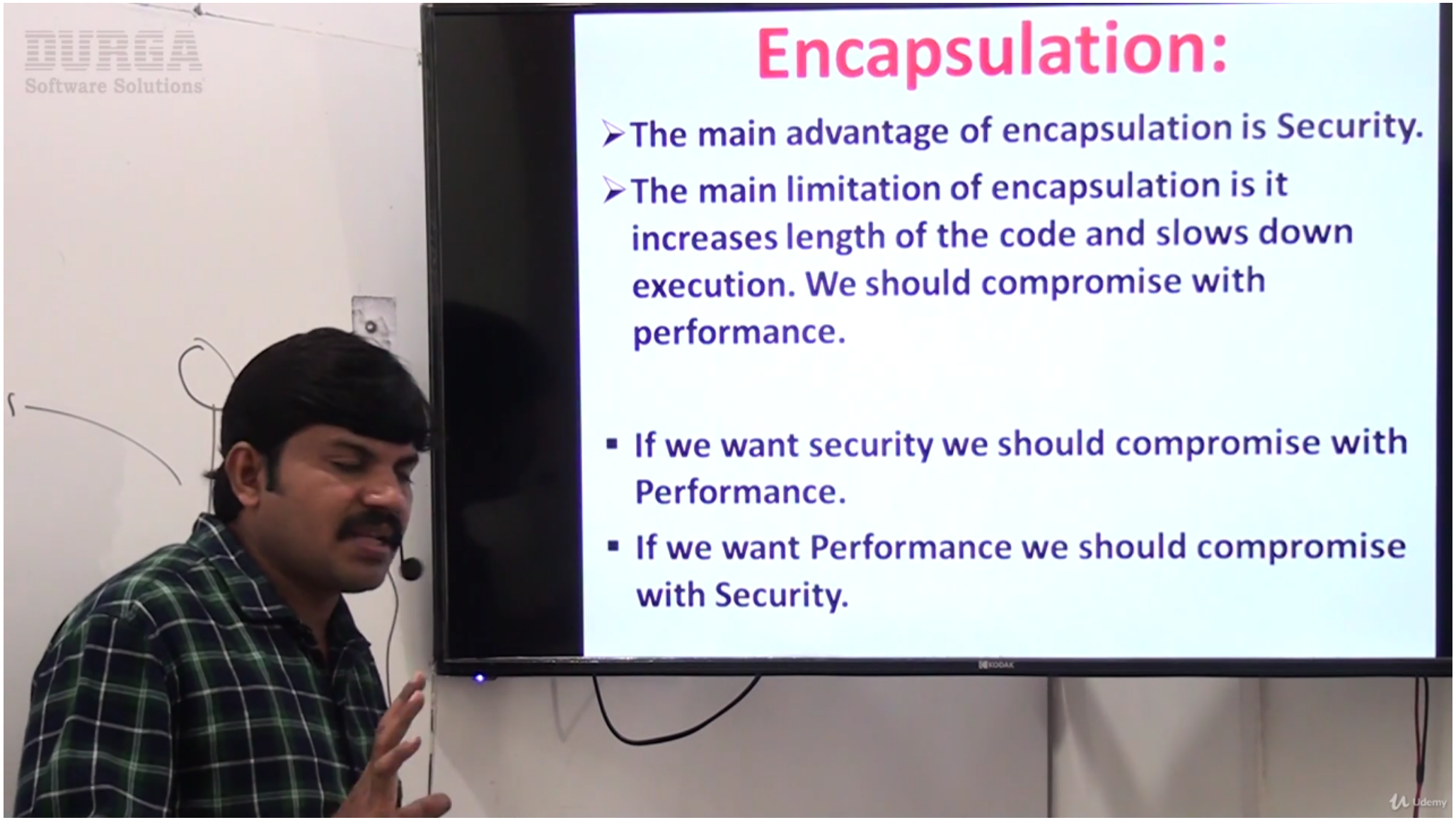

DURGA
Software Solutions
Data Hiding:
Our internal data should not go out directly.
i.e outside person should not access our internal data directly.
This OOP feature is nothing but data hiding.
By declaring data members as private, we can implement Data Hiding.
Udemy

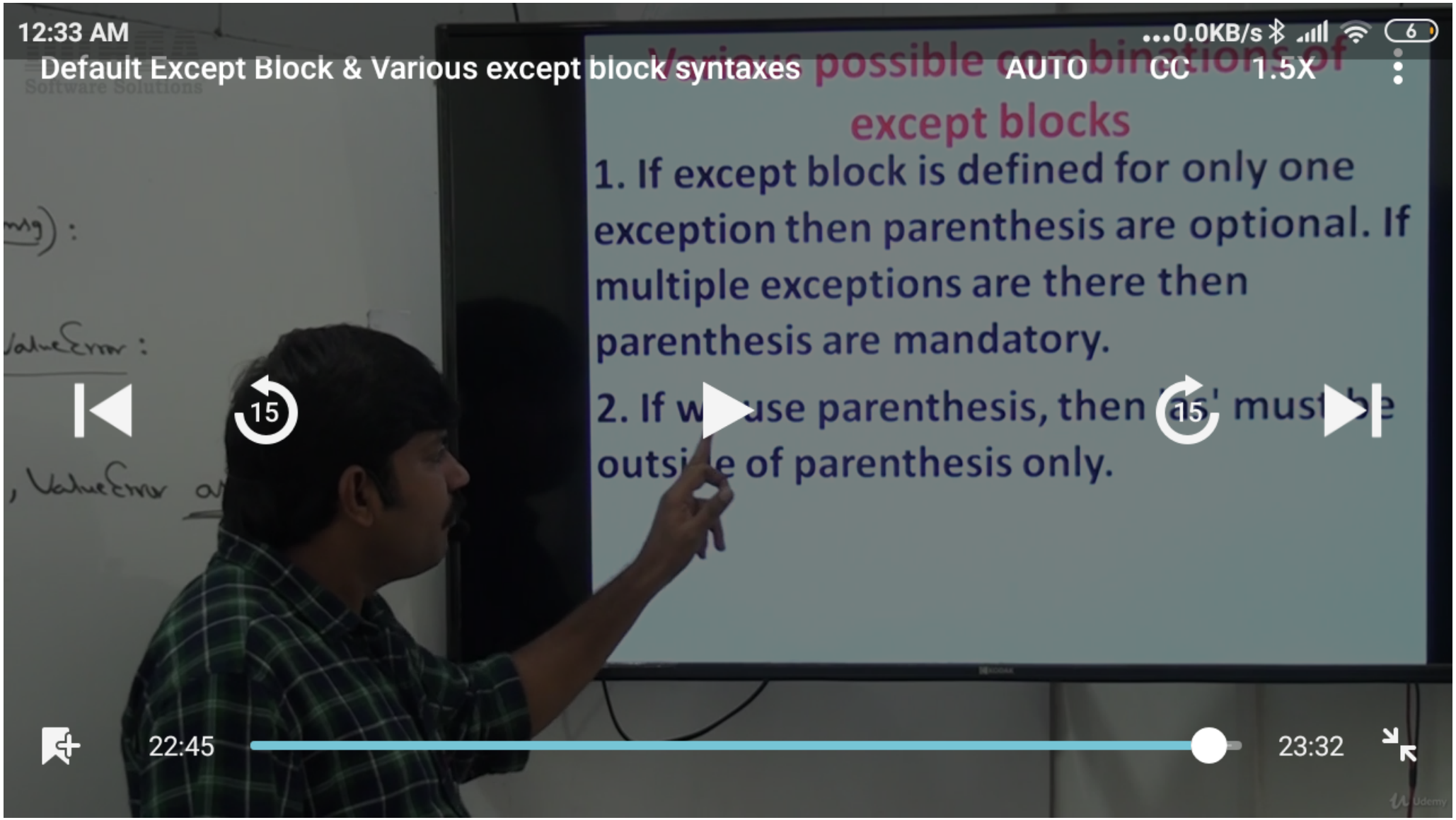

DURGA
Software Solutions
Encapsulation:
The process of Binding/ Grouping/ encapsulating data and corresponding behavior(methods) into a single unit is nothing but encapsulation.
Every Python class is an example of encapsulation.
KODAK
Udemy

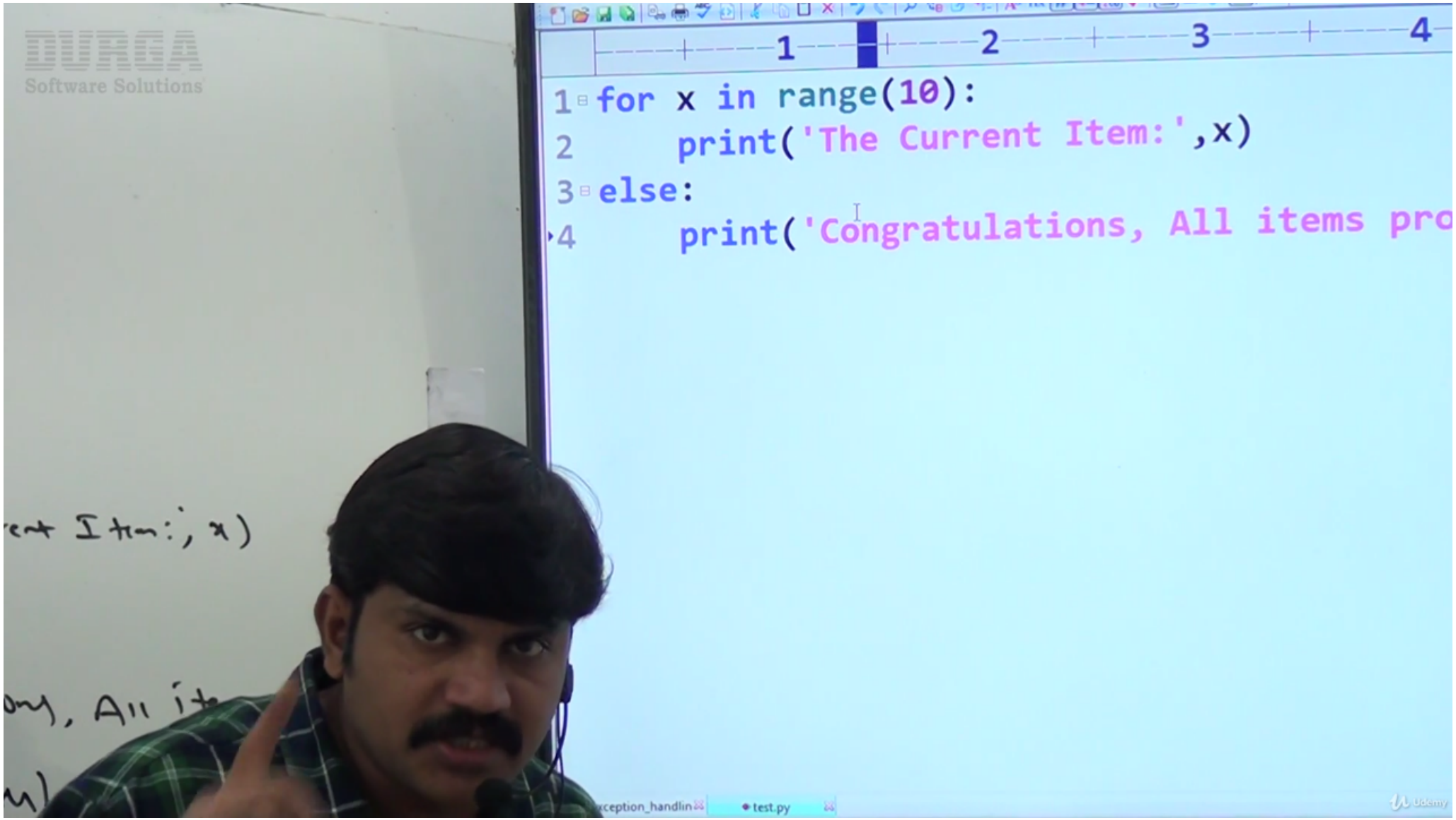

DURGA
Software Solutions
Encapsulation:
If any component follows data hiding and abstraction, such component is said to be encapsulated component.
Encapsulation=Data Hiding+Abstraction
Udemy

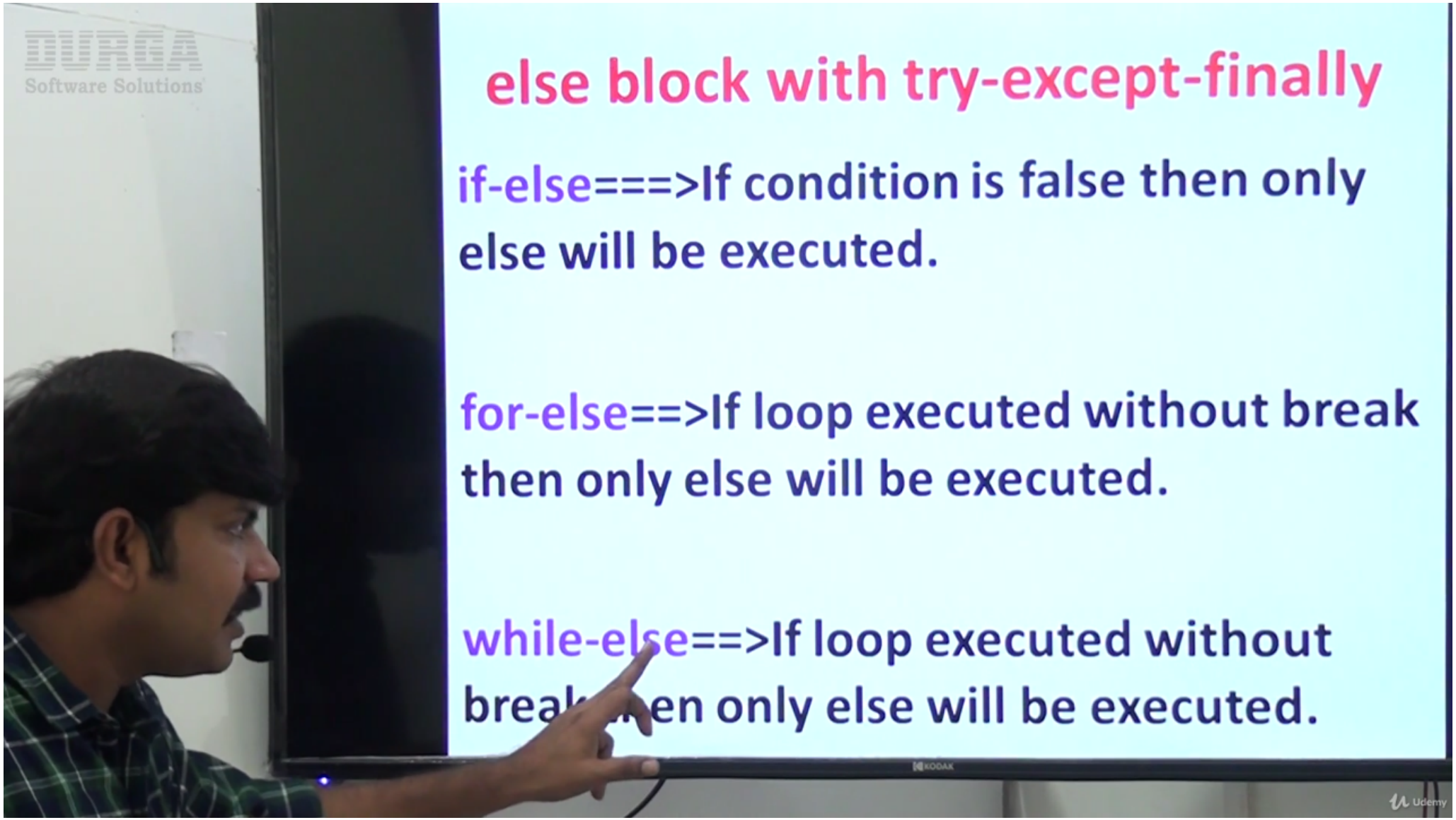

DURGA
Software Solutions
Encapsulation:
If any component follows data hiding and abstraction, such component is said to be encapsulated component.
Encapsulation=Data Hiding+Abstraction
Udemy

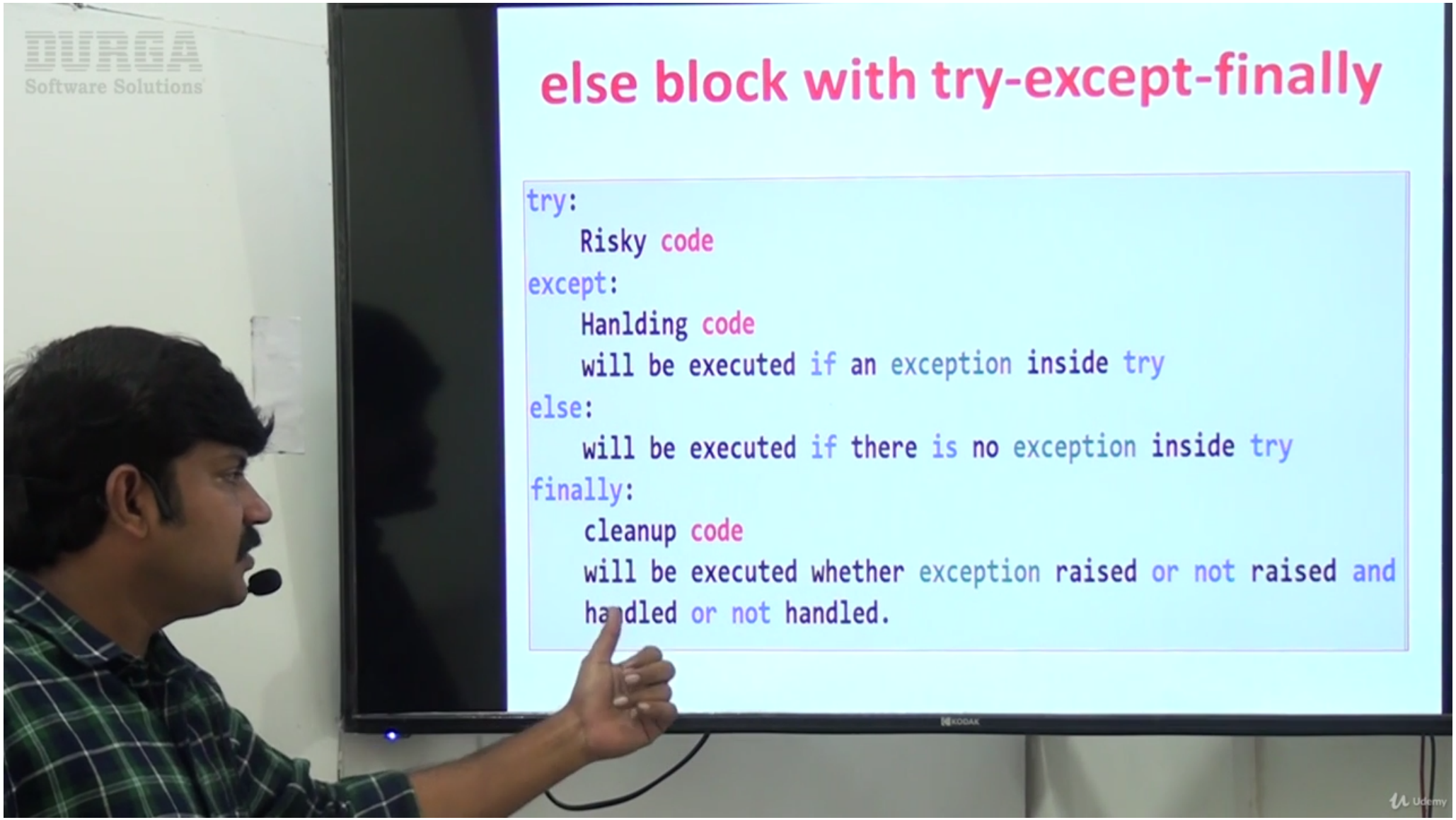

DURGA
Software Solutions
Encapsulation:
Hiding data behind methods is the central concept of encapsulation.
Advantages of encapsulation:
1. Security
2. Enhancement will become easy
3. Maintainability and Modularity will be improved
KODAK
Udemy

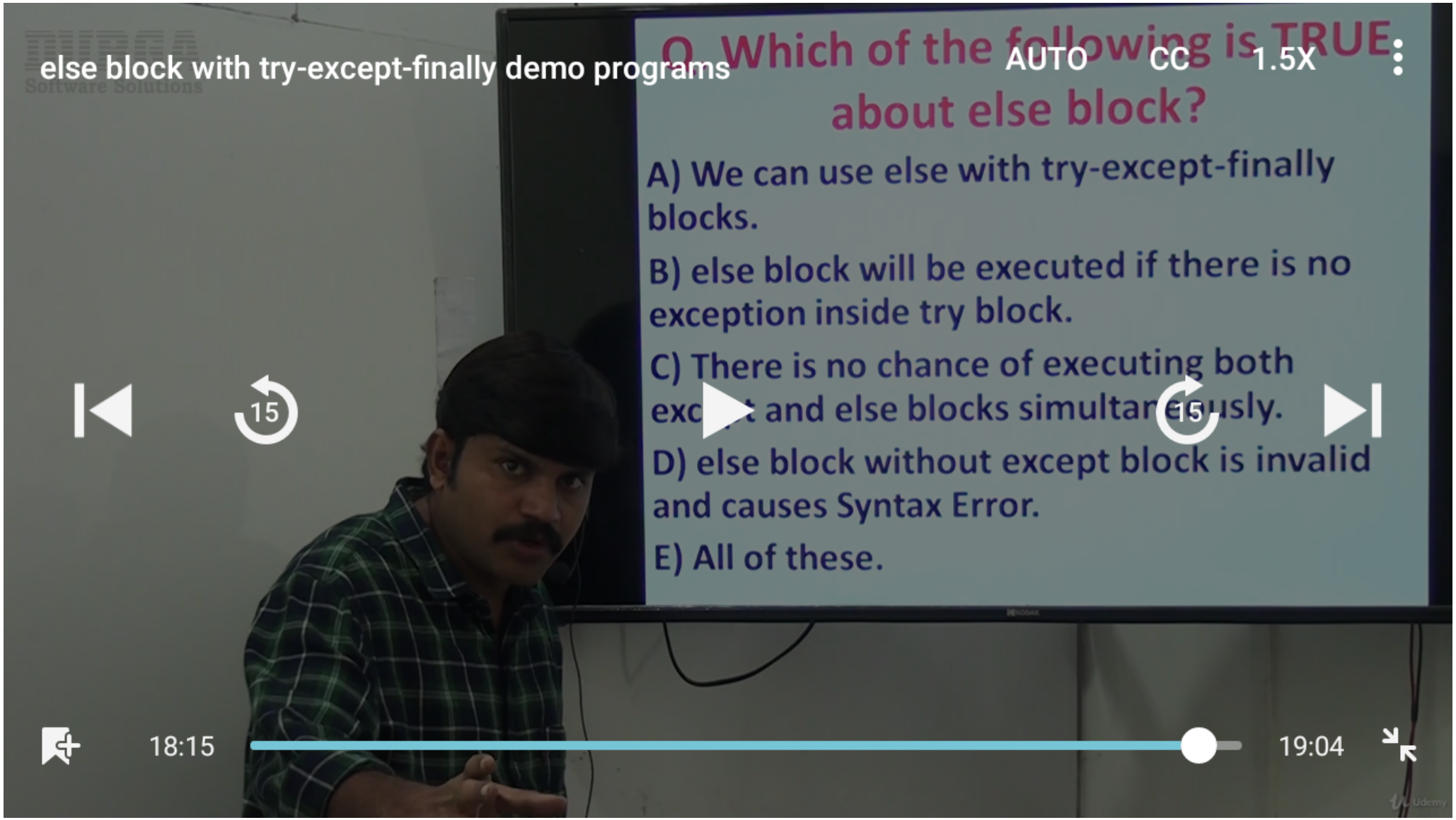

DURGA
Software Solutions
Encapsulation:
The main advantage of encapsulation is Security.
The main limitation of encapsulation is it increases length of the code and slows down execution. We should compromise with performance.
If we want security we should compromise with Performance.
If we want Performance we should compromise with Security.
KODAK
Udemy

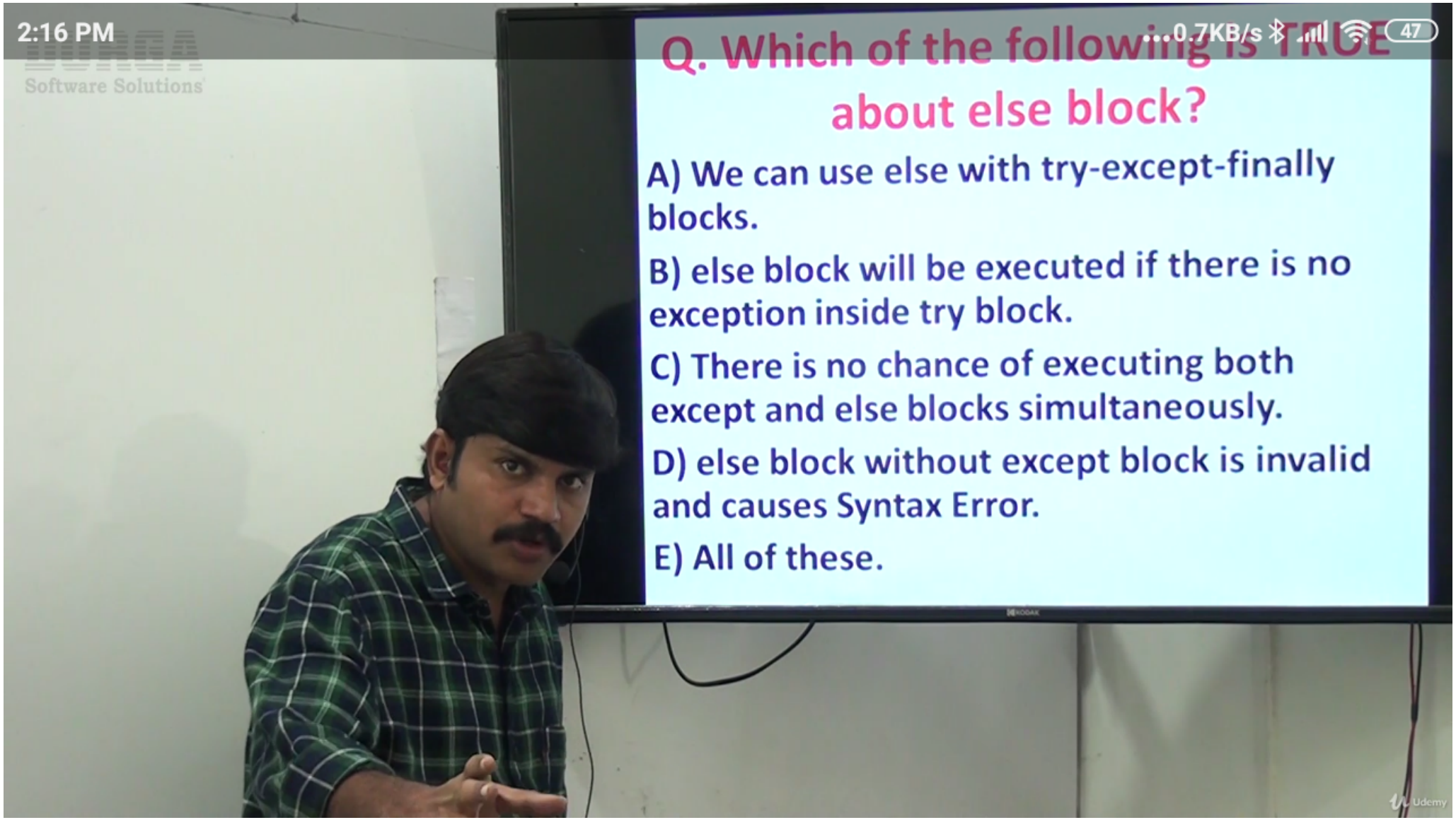

12:33 AM
0.0KB/s
Default Except Block & Various except block syntaxes
AUTO
CC
1.5X
Various possible combinations of except blocks
1. If except block is defined for only one exception then parenthesis are optional. If multiple exceptions are there then parenthesis are mandatory.
2. If we use parenthesis, then 'as' must be outside of parenthesis only.
15
15
22:45
23:32

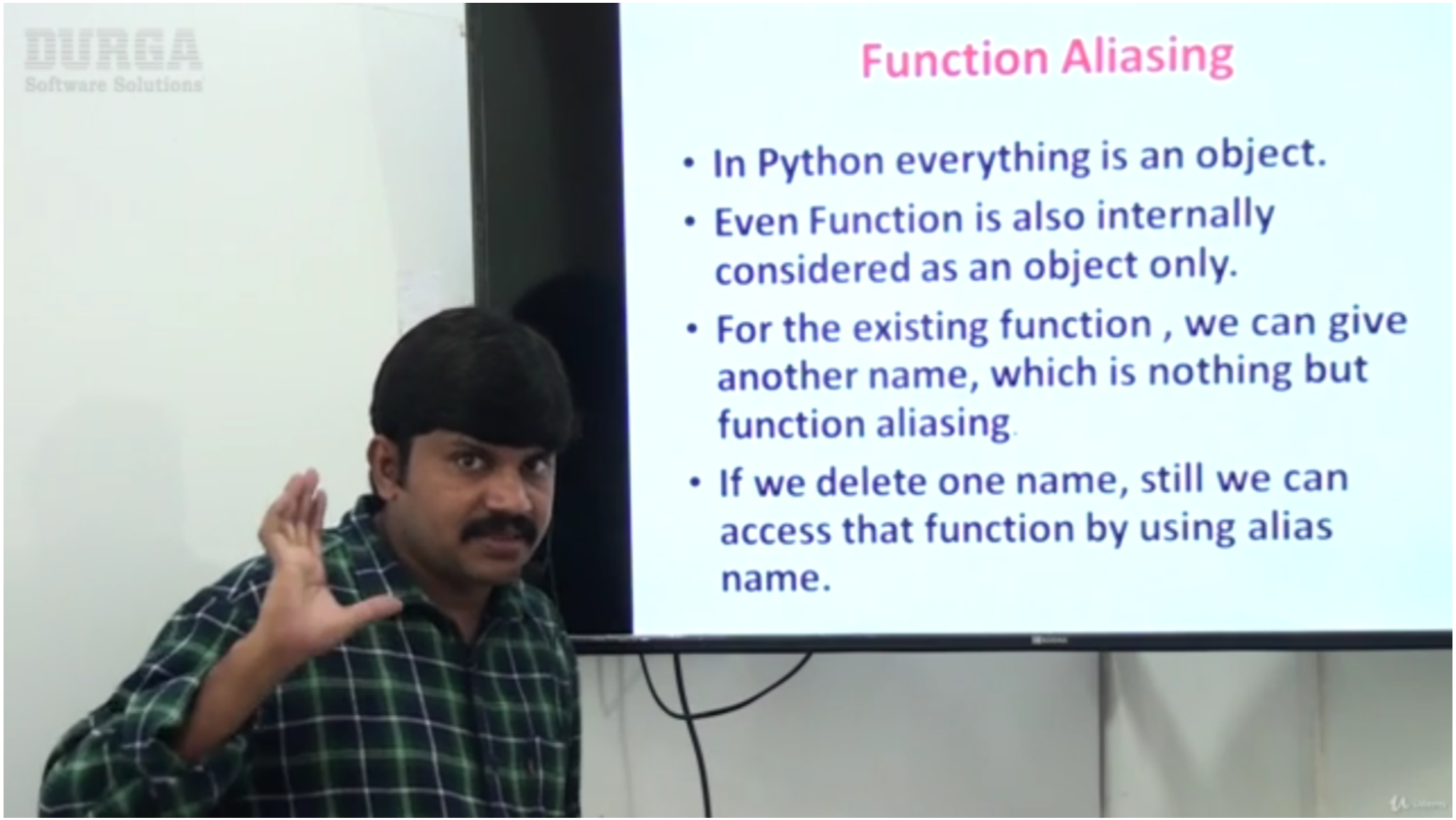

DURGA
Software Solutions
1 for x in range(10):
2 print('The Current Item:',x)
3 else:
4 print('Congratulations, All items pro
test.py

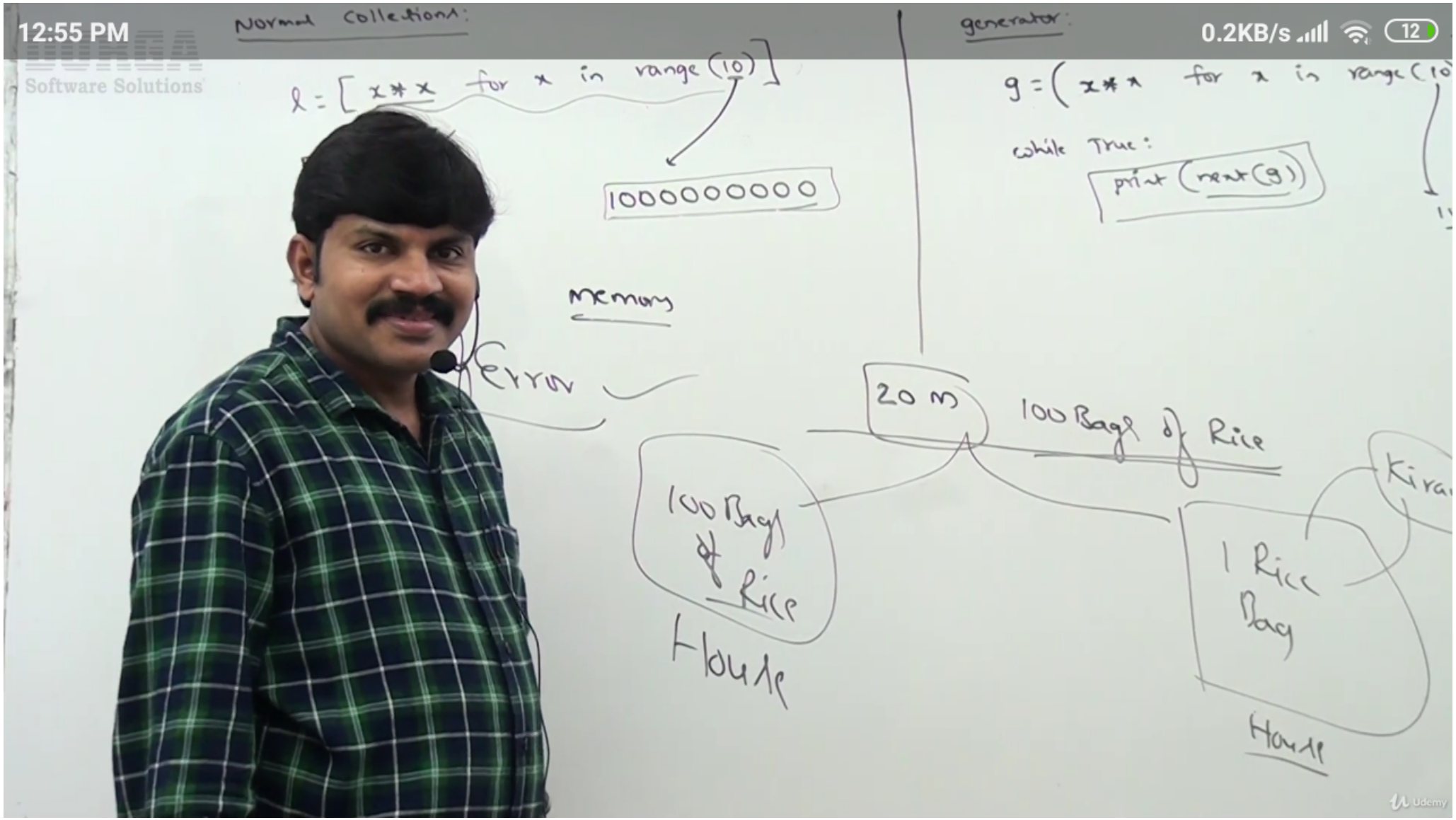

DURGA
Software Solutions
else block with try-except-finally
if-else===>If condition is false then only else will be executed.
for-else==>If loop executed without break then only else will be executed.
while-else==>If loop executed without
en only else will be executed.
KODAK
Udemy


DURGA
Software Solutions
else block with try-except-finally
try:
Risky code
except:
Hanlding code
will be executed if an exception inside try
else:
will be executed if there is no exception inside try
finally:
cleanup code
will be executed whether exception raised or not raised and
handled or not handled.
KODAK
Udemy


else block with try-except-finally demo programs
AUTO
CC
1.5X
Q. Which of the following is TRUE about else block?
A) We can use else with try-except-finally blocks.
B) else block will be executed if there is no exception inside try block.
C) There is no chance of executing both except and else blocks simultaneously.
D) else block without except block is invalid and causes Syntax Error.
E) All of these.
15
15
18:15
19:04


2:16 PM
0.7KB/s
47
Software Solutions
Q. Which of the following is TRUE about else block?
A) We can use else with try-except-finally blocks.
B) else block will be executed if there is no exception inside try block.
C) There is no chance of executing both except and else blocks simultaneously.
D) else block without except block is invalid and causes Syntax Error.
E) All of these.
Udemy


DURGA
Software Solutions
Function Aliasing
• In Python everything is an object.
• Even Function is also internally considered as an object only.
• For the existing function , we can give another name, which is nothing but function aliasing.
• If we delete one name, still we can access that function by using alias name.


12:55 PM
0.2KB/s
Software Solutions
Normal Collections:
l = [x * x for x in range (10)]
1000000000
Memory
Error
generator:
g = (x * x for x in range (10
while True:
print (next(g))
20 m
100 Bags of Rice
100 Bags of Rice
Hourly
1 Rice Bag
Hourly
Udemy