Presentation 3 / 111

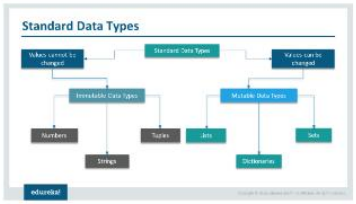
# COURSE OUTLINE

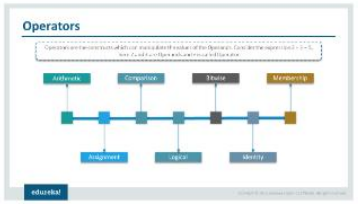

## MODULE 03

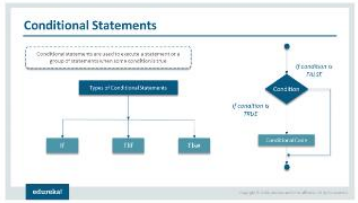
- Introduction to Big Data Hadoop and Spark
- Introduction to Python for Apache Spark
- **Functions, OOPs, Modules in Python**
- Deep Dive into Apache Spark Framework
- Playing with Spark RDDs
- Data Frames and Spark SQL
- Machine Learning using Spark MLlib
- Deep Dive into Spark MLlib
- Understanding Apache Kafka and Apache Flume
- Apache Spark Streaming - Processing Multiple Batches
- Apache Spark Streaming - Data Sources
- Implementing an End-to-End Project

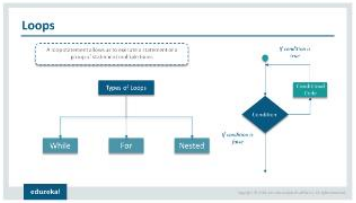
Presentation 4 / 111

# Recap



edureka!

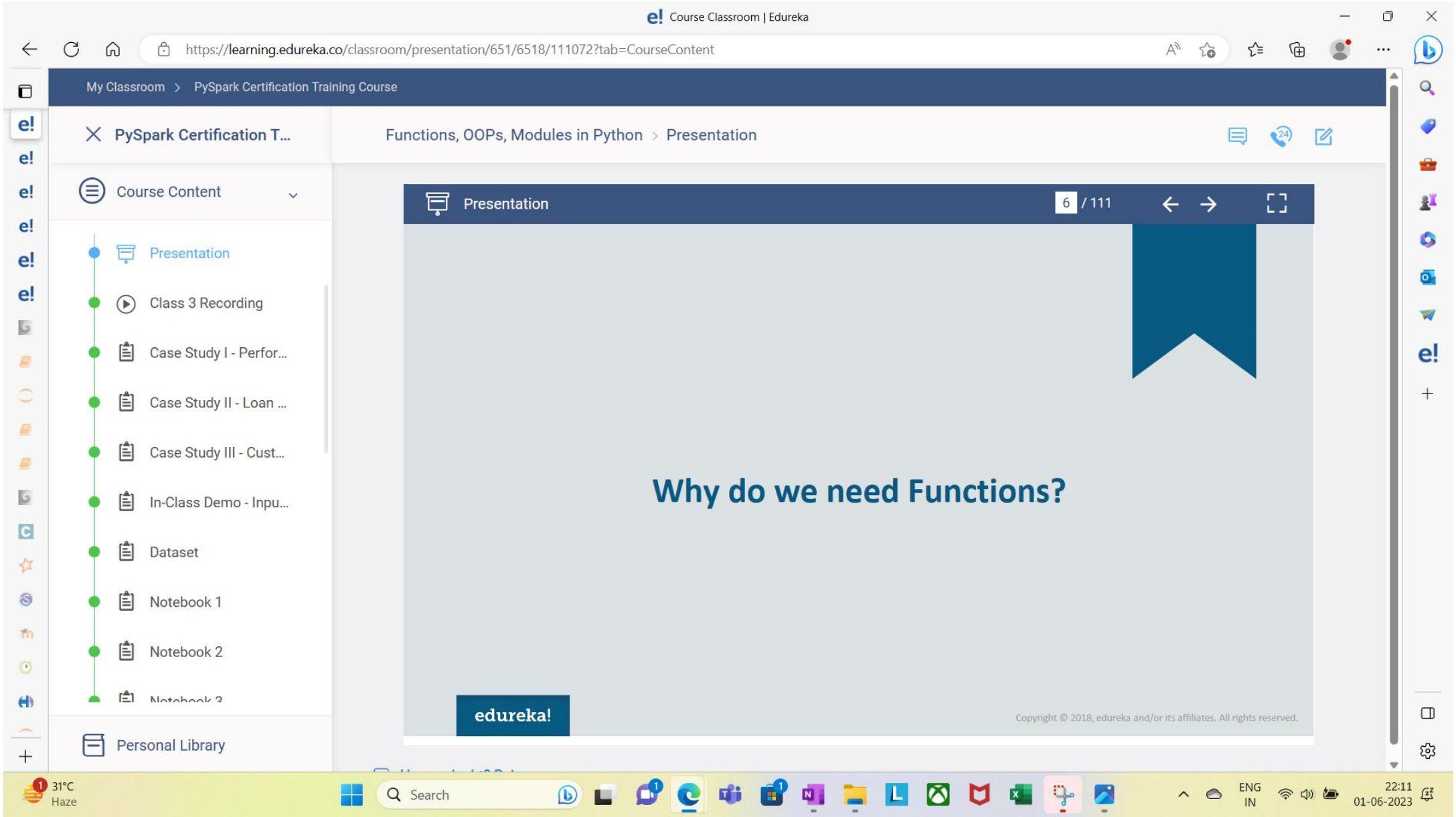
Presentation 5 / 111

# Objectives

After completing this module, you should be able to:

- Create and Execute Python Functions
- Learn Object Oriented Concepts in Python
- Understand Python Standard Libraries
- Define Modules in Python


edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
6 / 111
Why do we need Functions?
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:11
01-06-2023

Presentation 7 / 111

# Why do we need Functions?

Theon has to write a program in which he has to calculate ***Factorial*** multiple times

Factorial = n (n-1).....1

Logic to implement *Factorial*

```
num = int(input("Number: "))

factorial = 1

if num < 0:
   print("Must be positive")
elif num == 0:
   print("Factorial = 1")
else:
   for i in range(1, num+1):
      factorial = factorial * i

print(factorial)
```

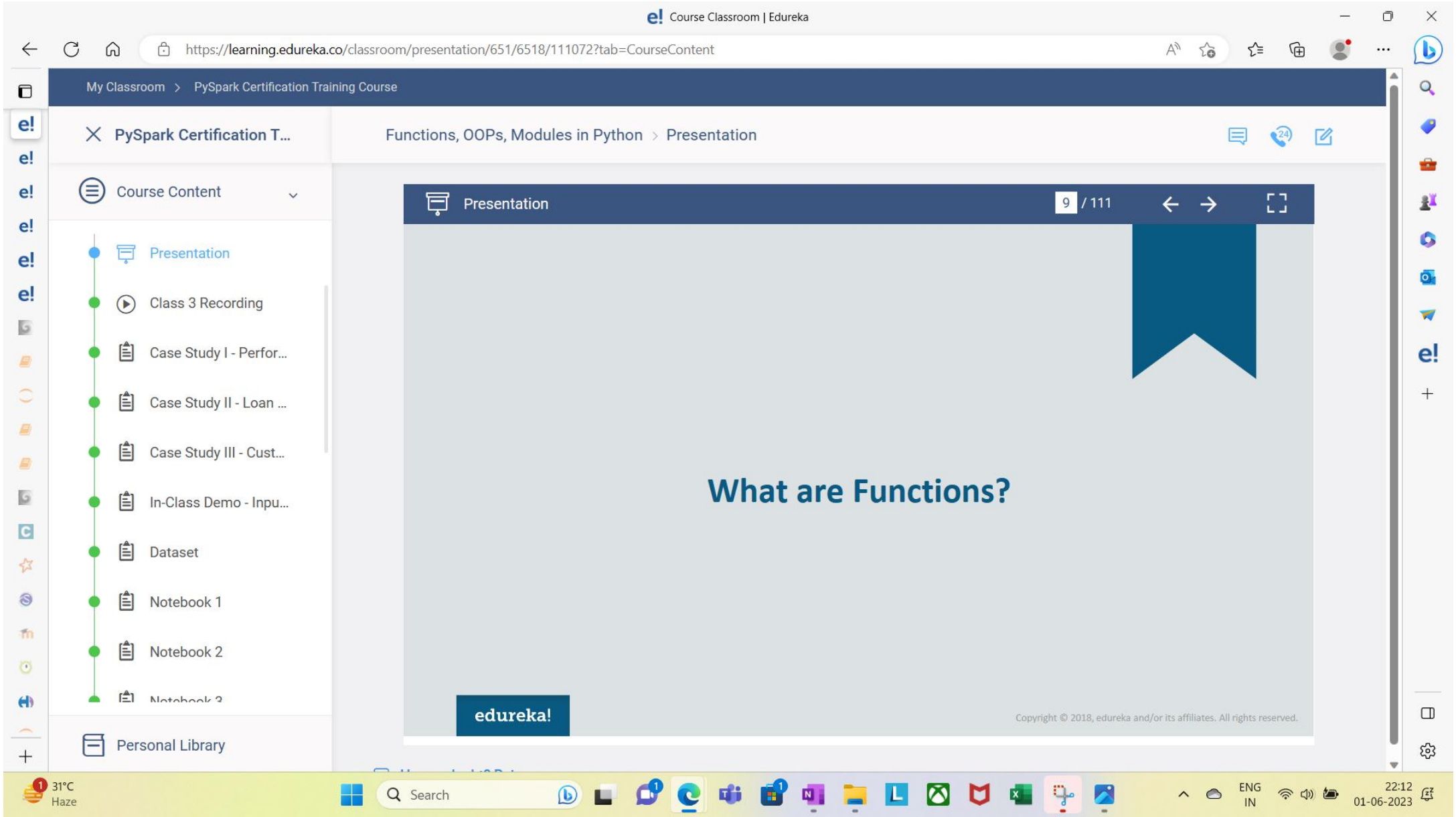
edureka!

Presentation 8 / 111

# Why do we need Functions?


```
num = int(input("Number: "))

factorial = 1

if num < 0:
   print("Must be positive")
elif num == 0:
   print("Factorial = 1")
else:
   for i in range(1, num+1):
      factorial = factorial * i

print(factorial)
```

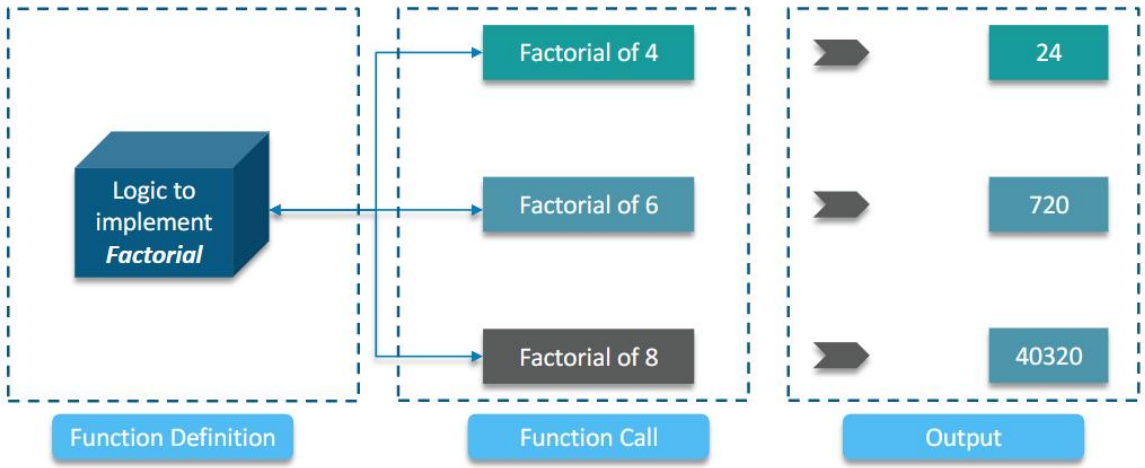
edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
9 / 111
What are Functions?
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:12
01-06-2023

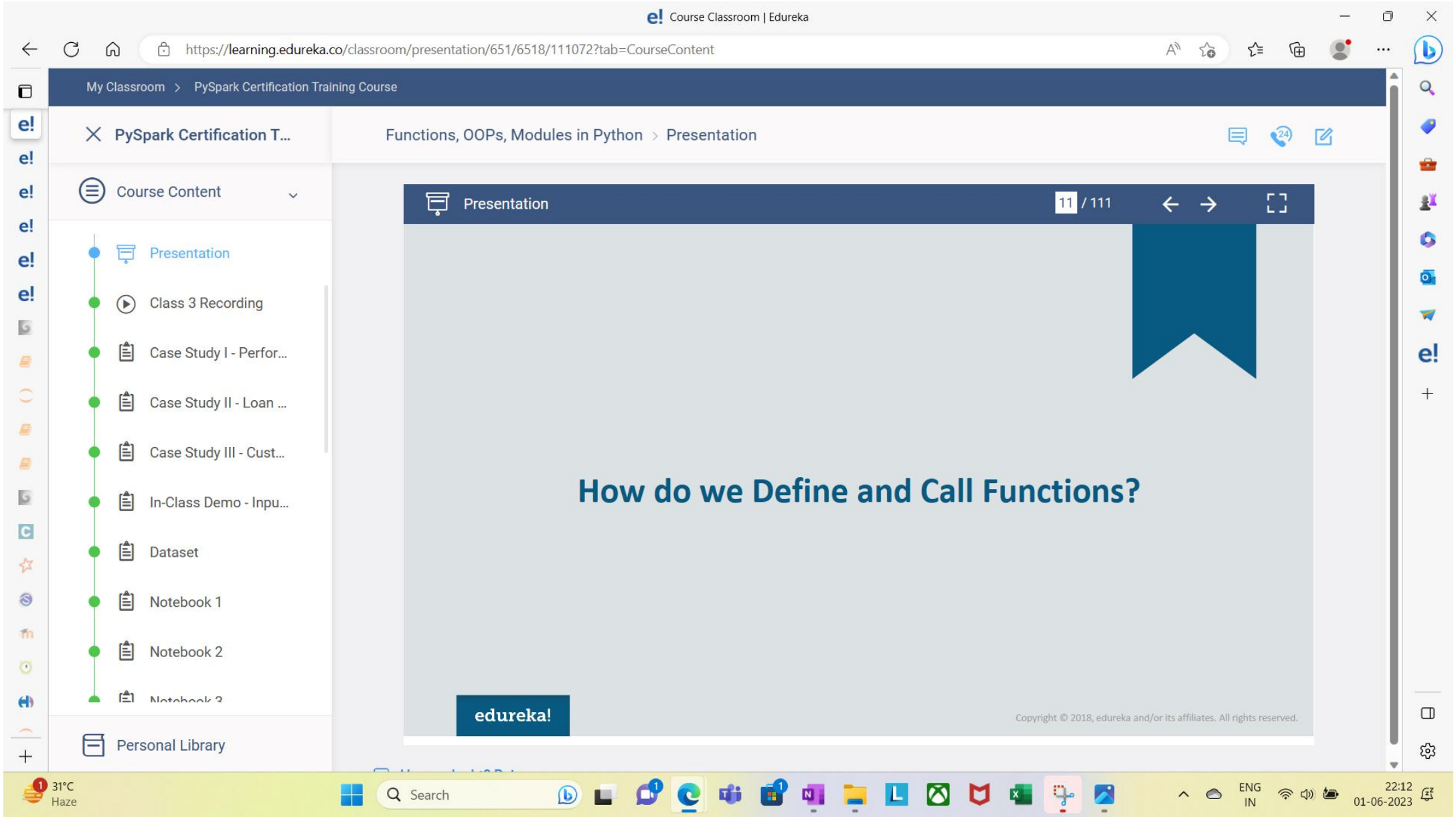
Presentation 10 / 111

# What are Functions?

A **Function** is a block of organized, reusable code that is used to perform a single and related action




edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
11 / 111
How do we Define and Call Functions?
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:12
01-06-2023

# Function Definition and Function Call

**Function Definition**

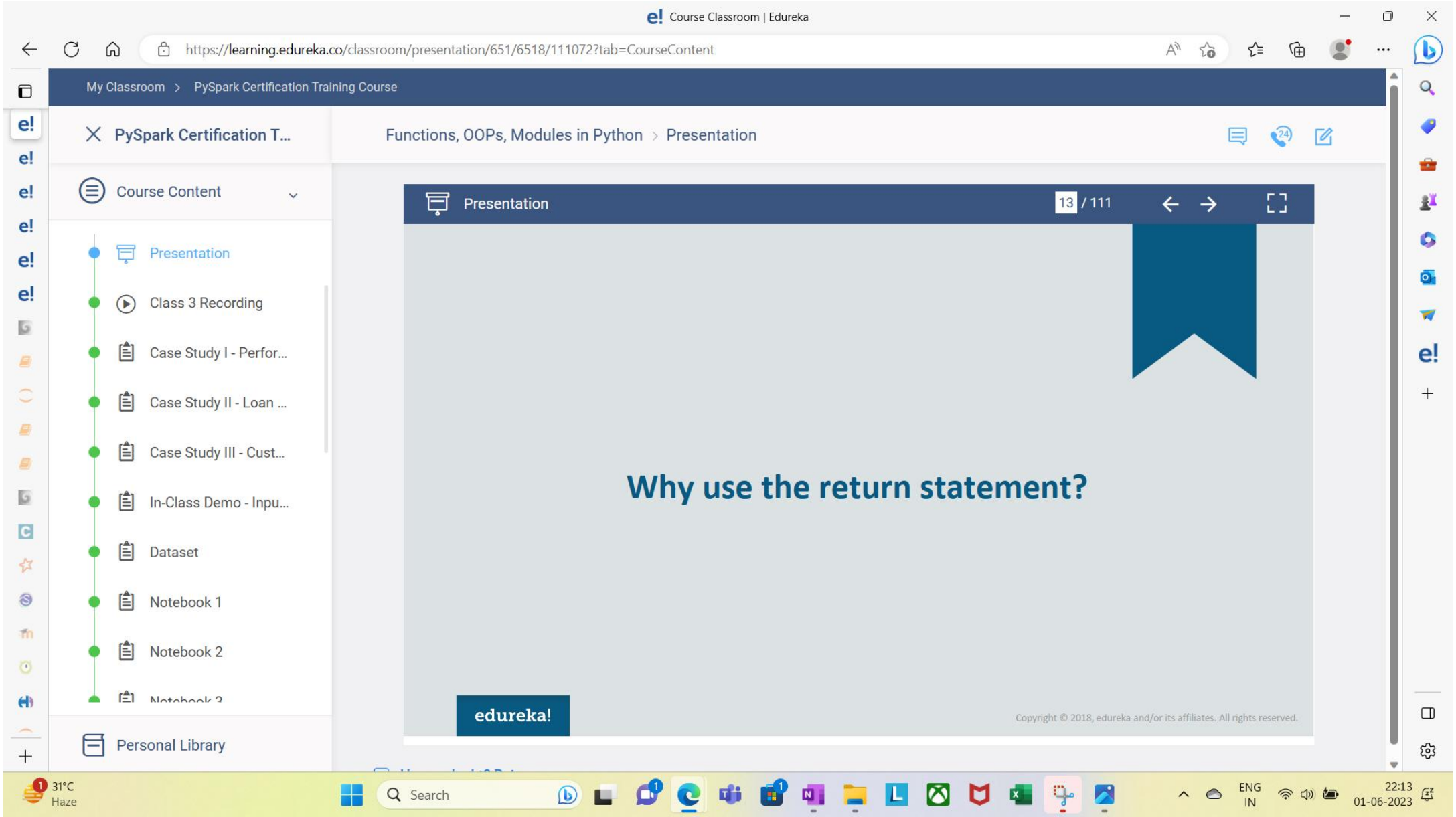
The set of statements written once executed multiple times

**Function Call**

Once we define the Function, we can call it multiple times

```
def print_name(str):
    print("Welcome to Python, ",str)
    return()

str=input("Enter your name :")
print_name(str)
```

edureka!


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
13 / 111
Why use the return statement?
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:13
01-06-2023

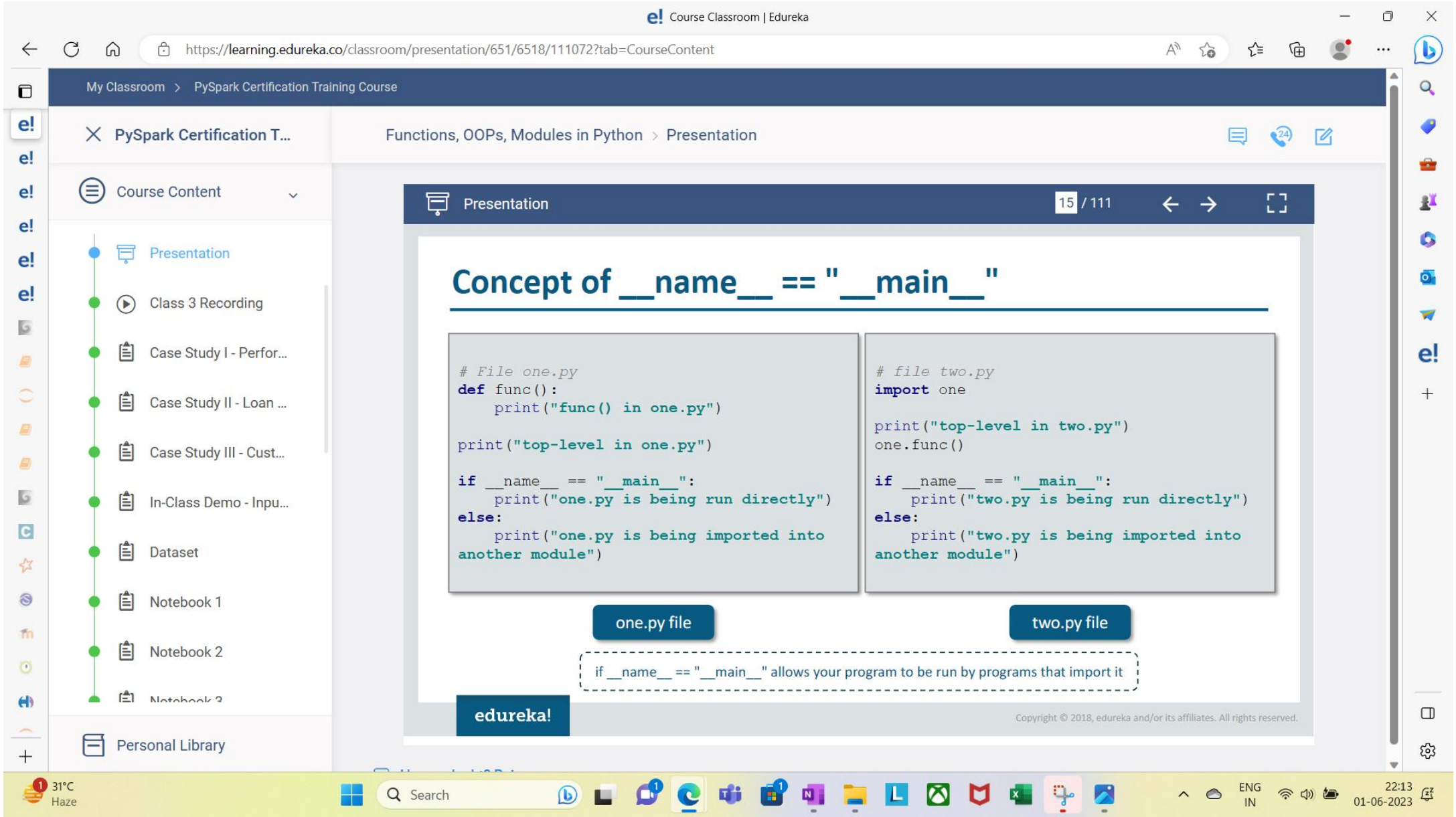
Presentation 14 / 111

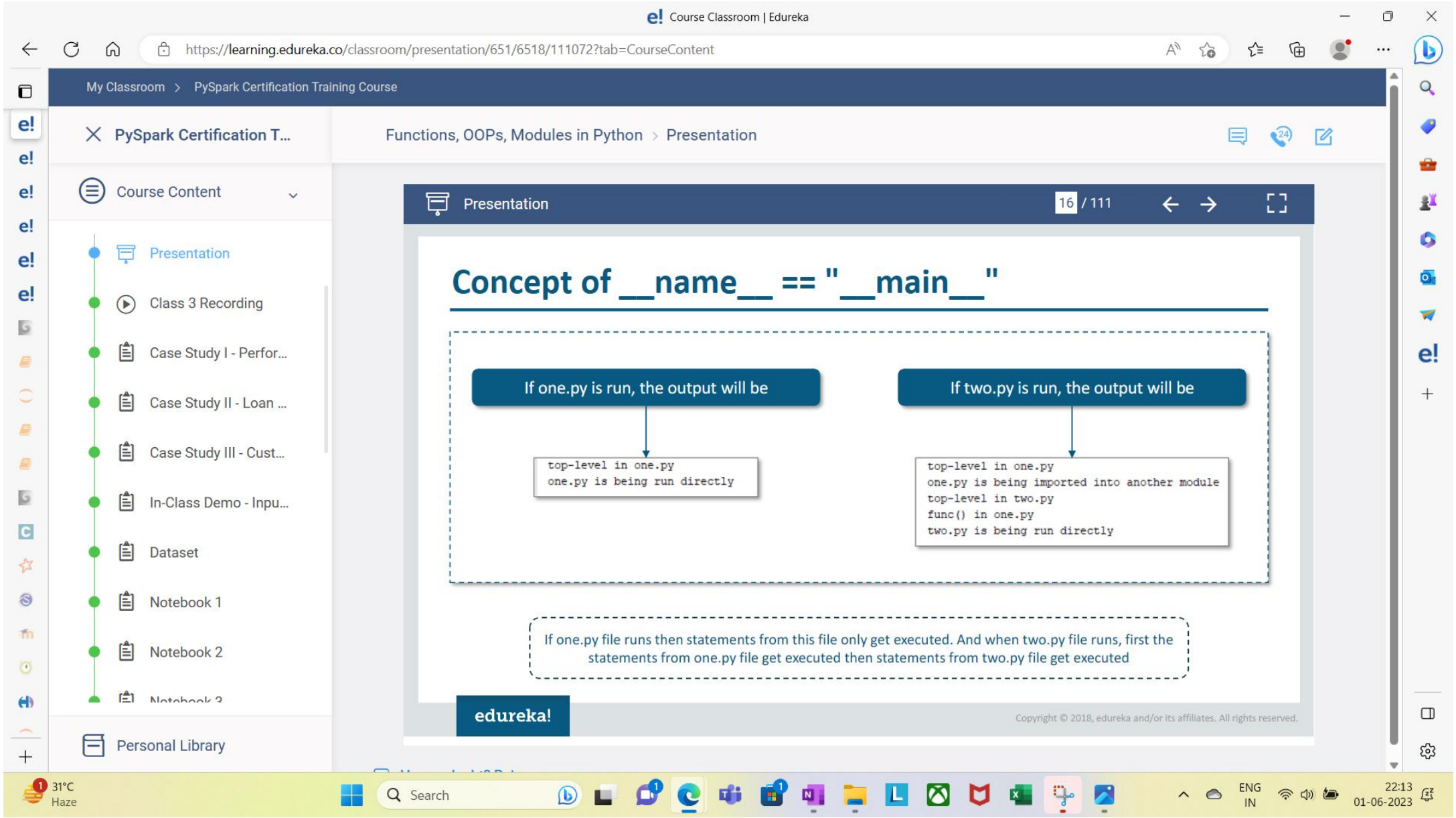
# The Return Statement

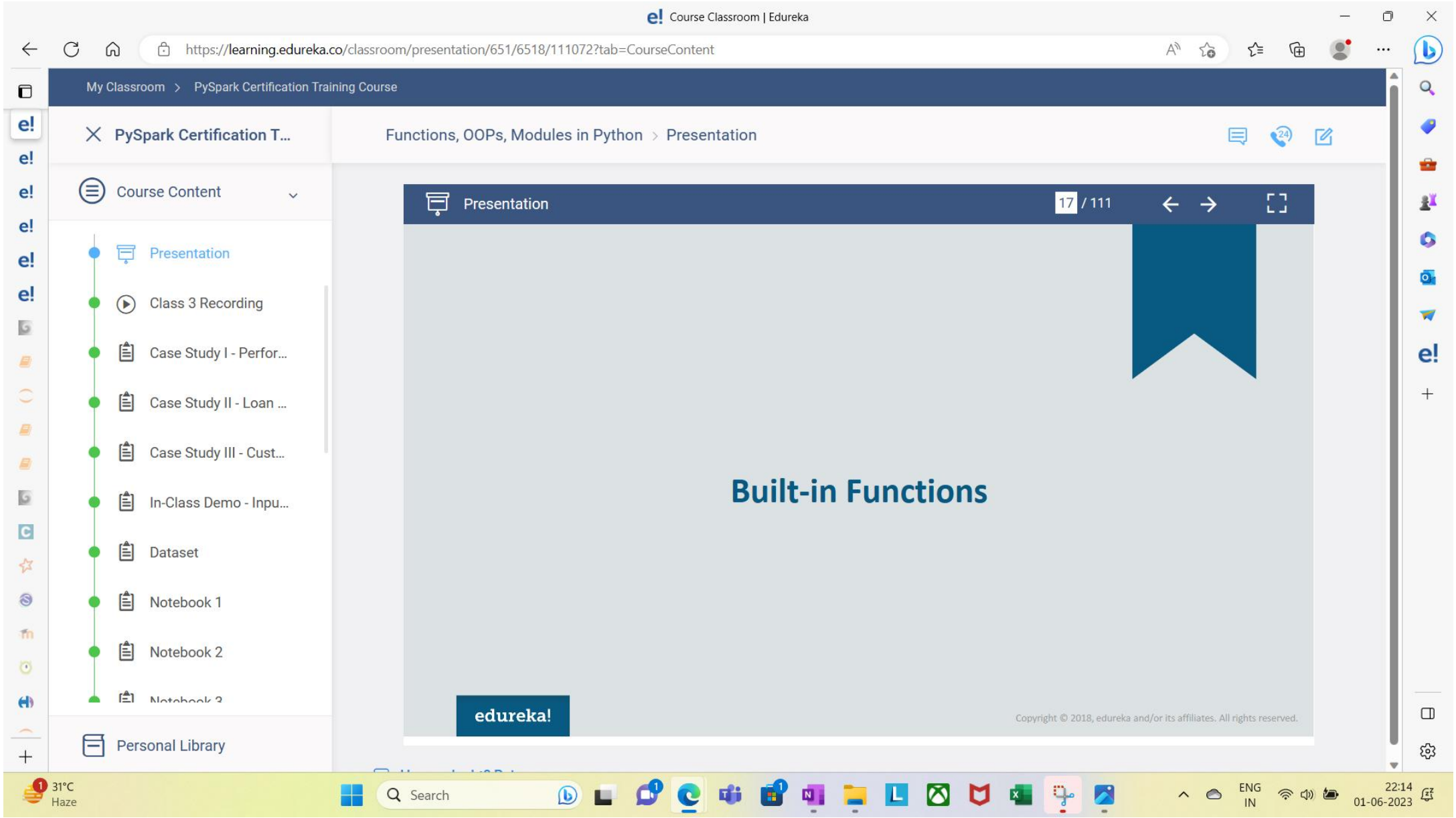
```
def add(a,b):
    sum=a+b
    return (sum)

num1=input("Enter first number")
num2=input("Enter second number")
n1=int(num1)
n2=int(num2)
ans=add(n1,n2)
print("Addition of 2 number is:",ans)
```

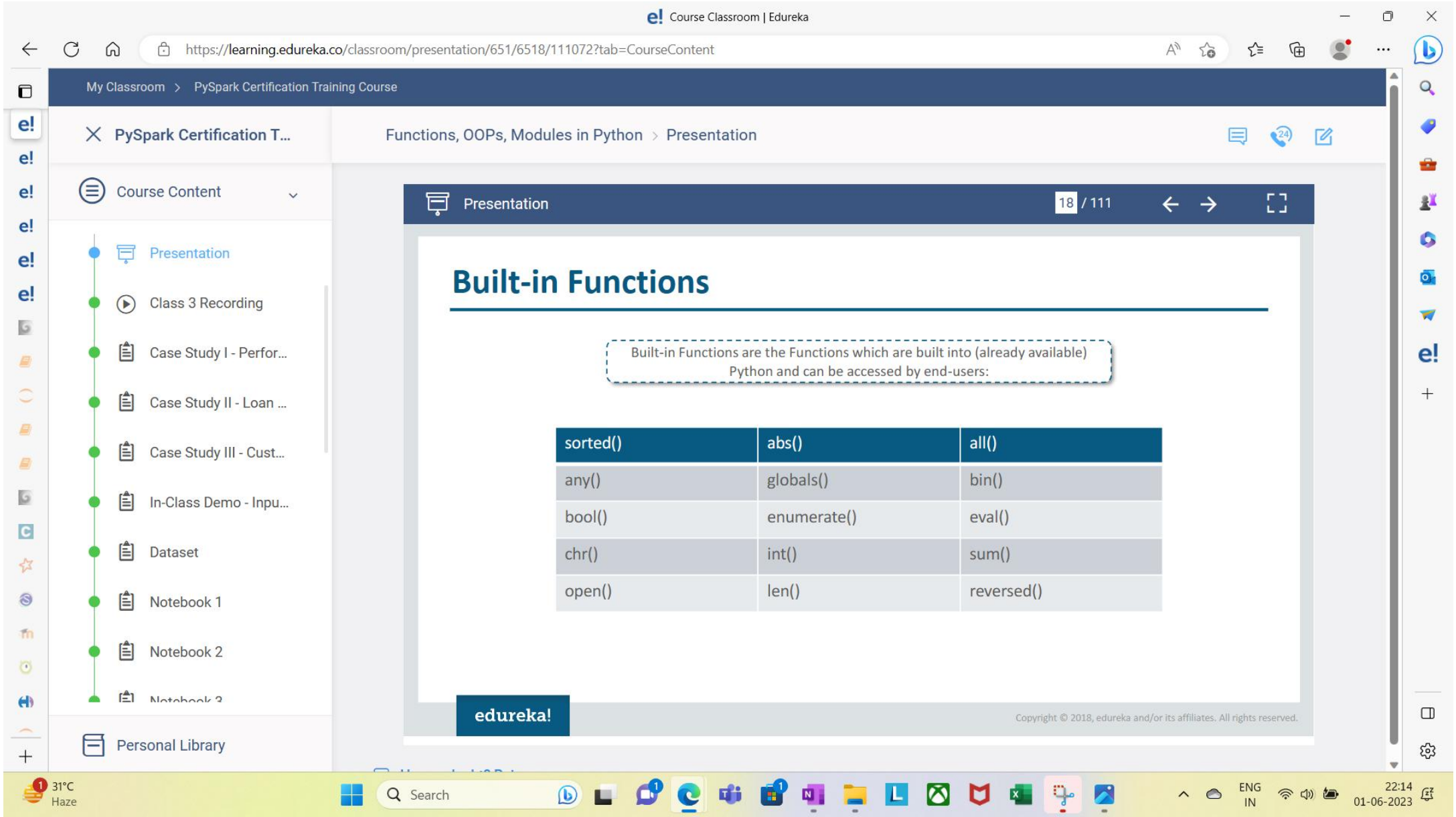
The return statement terminates the execution of a function and returns control to the calling function

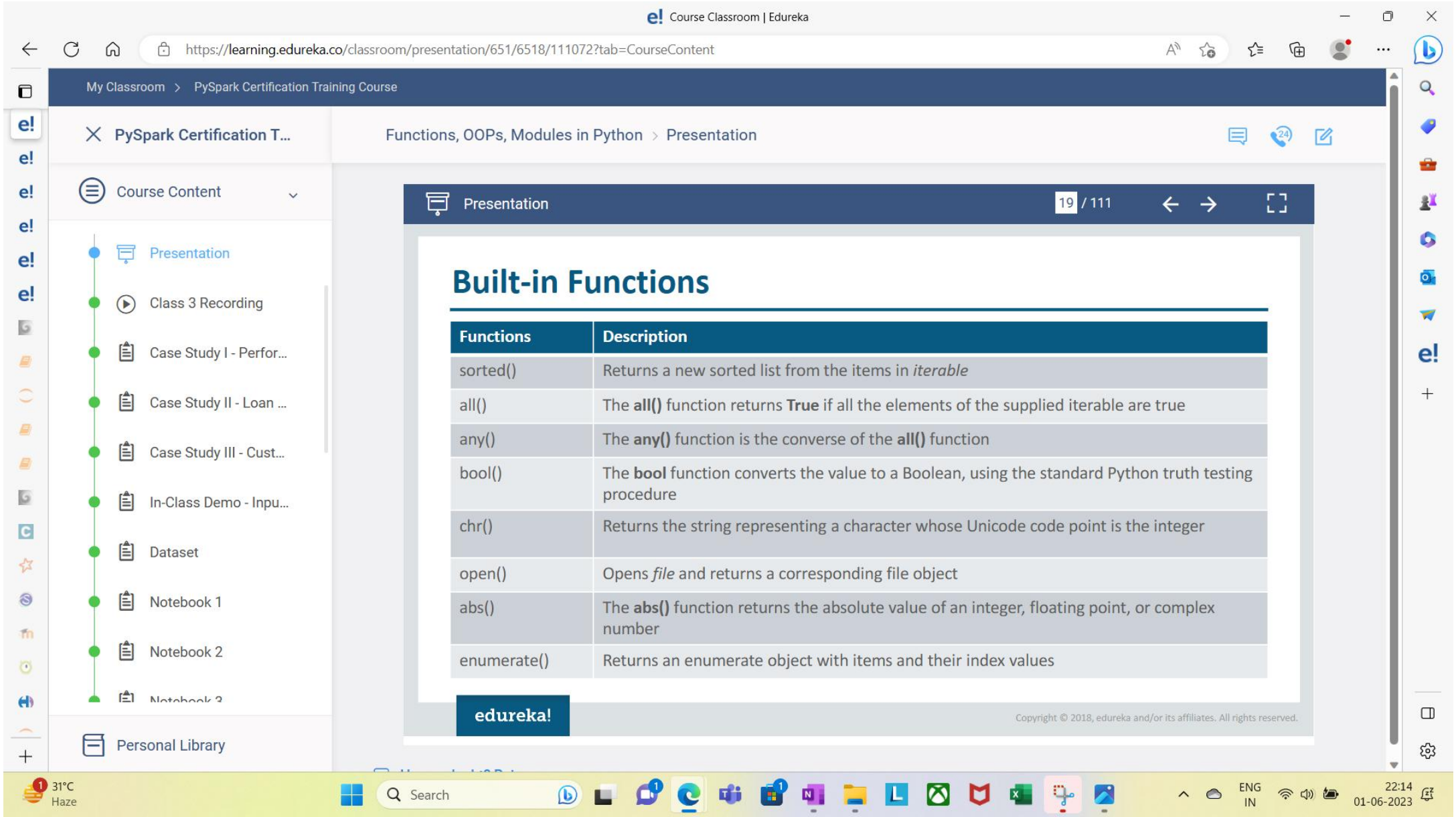
edureka!

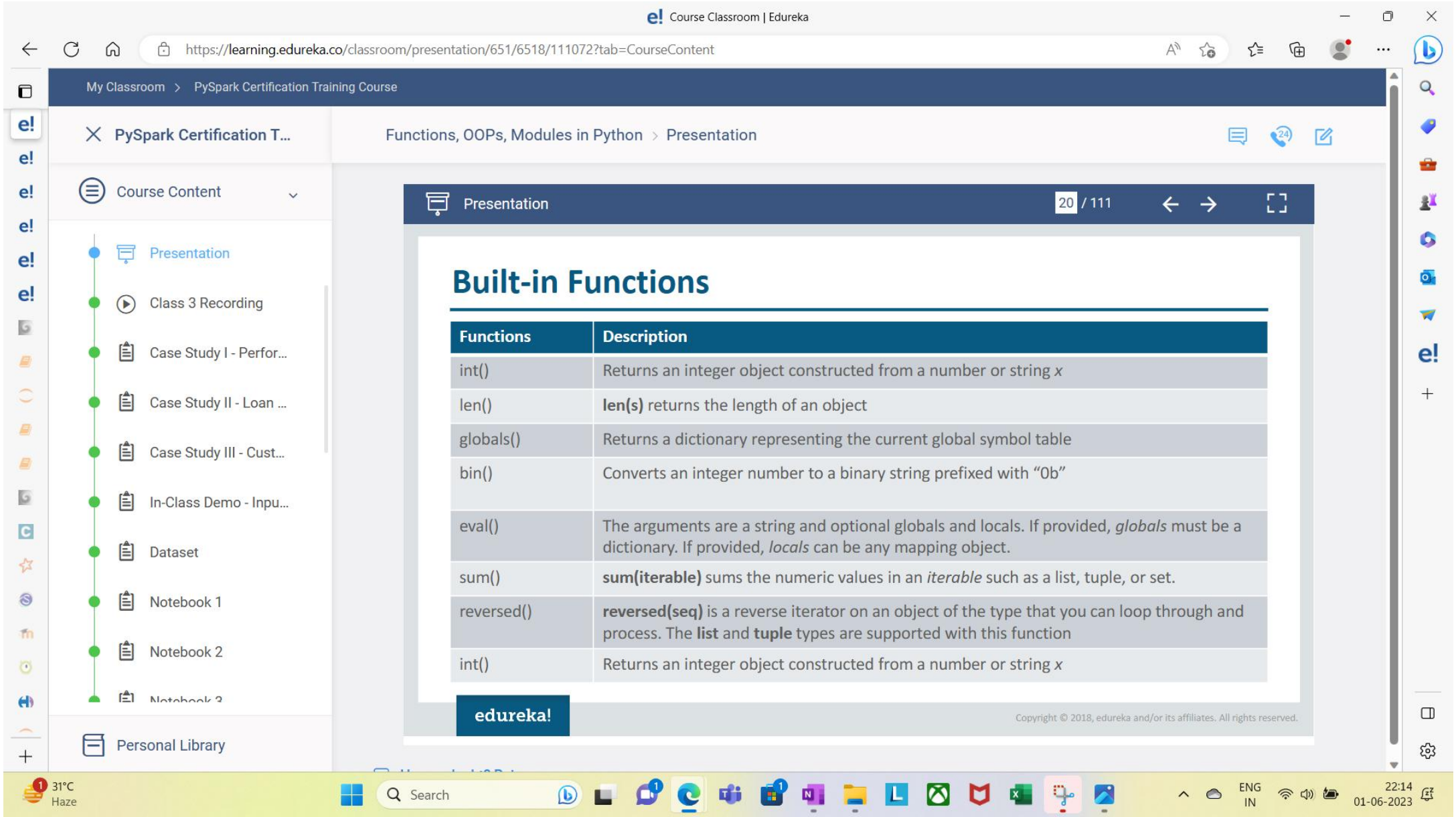

Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
15 / 111
Concept of __name__ == "__main__"
# File one.py
def func():
print("func() in one.py")
print("top-level in one.py")
if __name__ == "__main__":
print("one.py is being run directly")
else:
print("one.py is being imported into another module")
# file two.py
import one
print("top-level in two.py")
one.func()
if __name__ == "__main__":
print("two.py is being run directly")
else:
print("two.py is being imported into another module")
one.py file
two.py file
if __name__ == "__main__" allows your program to be run by programs that import it
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:13
01-06-2023

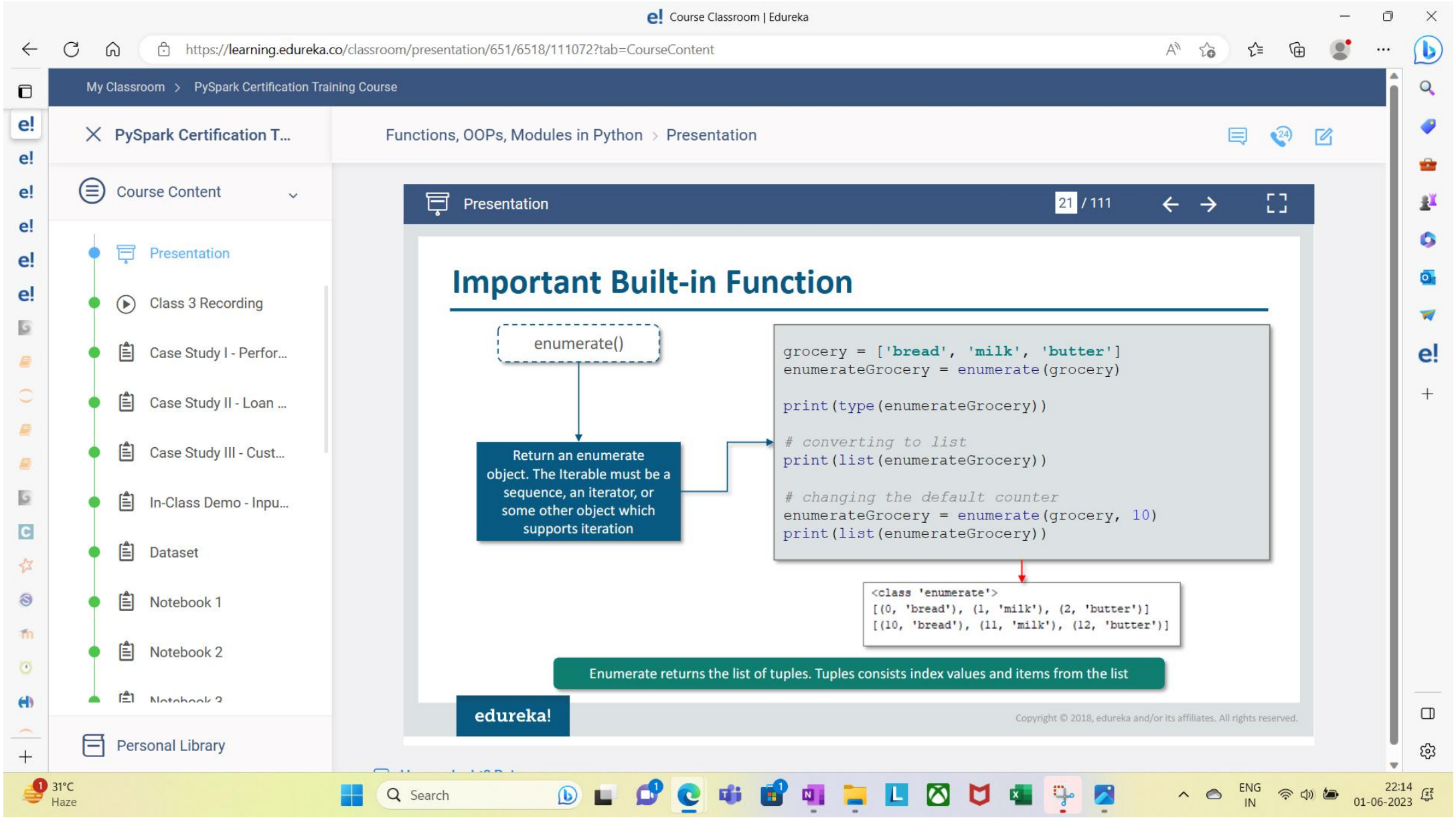

Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
16 / 111
Concept of __name__ == "__main__"
If one.py is run, the output will be
top-level in one.py
one.py is being run directly
If two.py is run, the output will be
top-level in one.py
one.py is being imported into another module
top-level in two.py
func() in one.py
two.py is being run directly
If one.py file runs then statements from this file only get executed. And when two.py file runs, first the statements from one.py file get executed then statements from two.py file get executed
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:13
01-06-2023

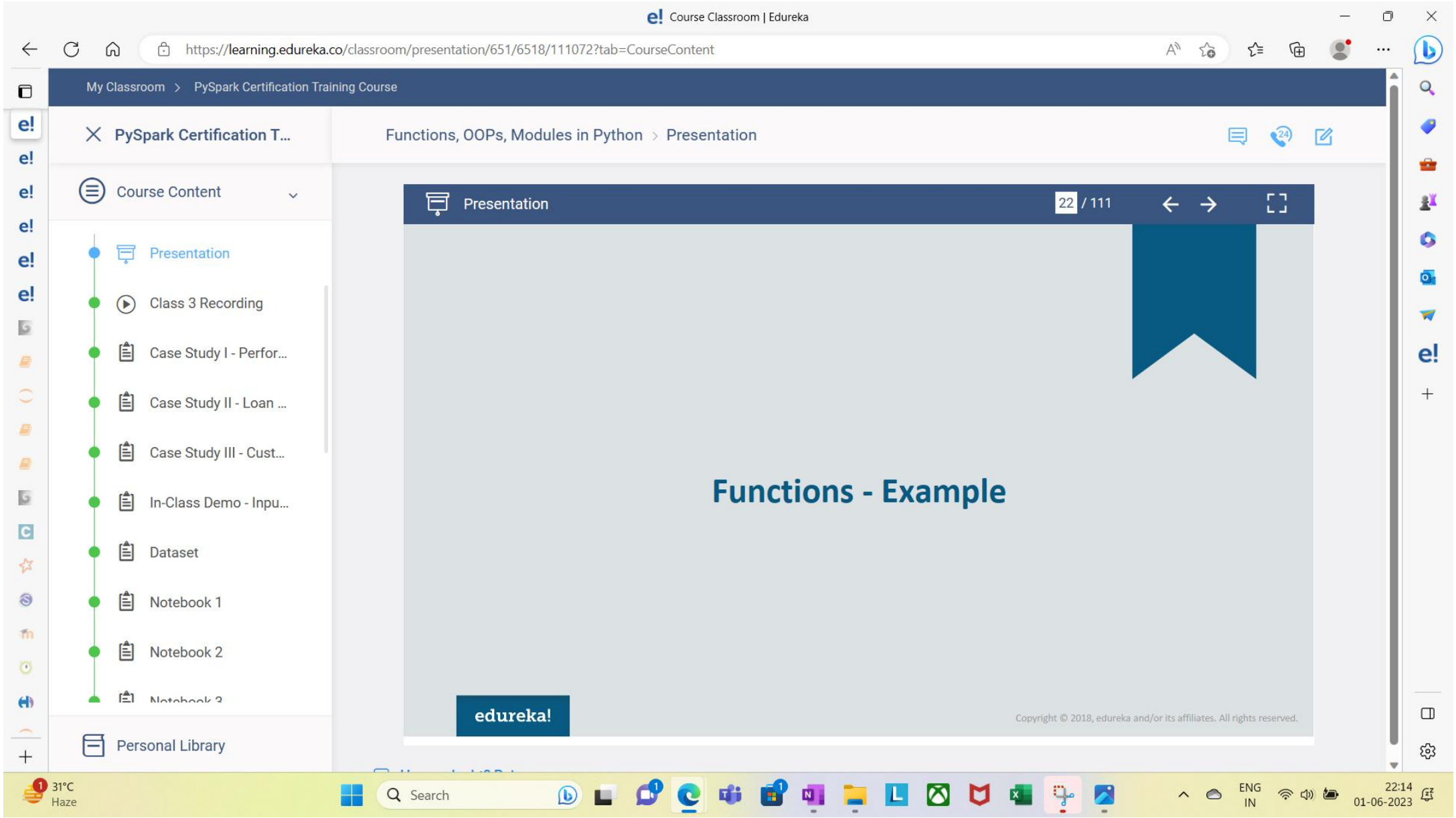

Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
17 / 111
Built-in Functions
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023

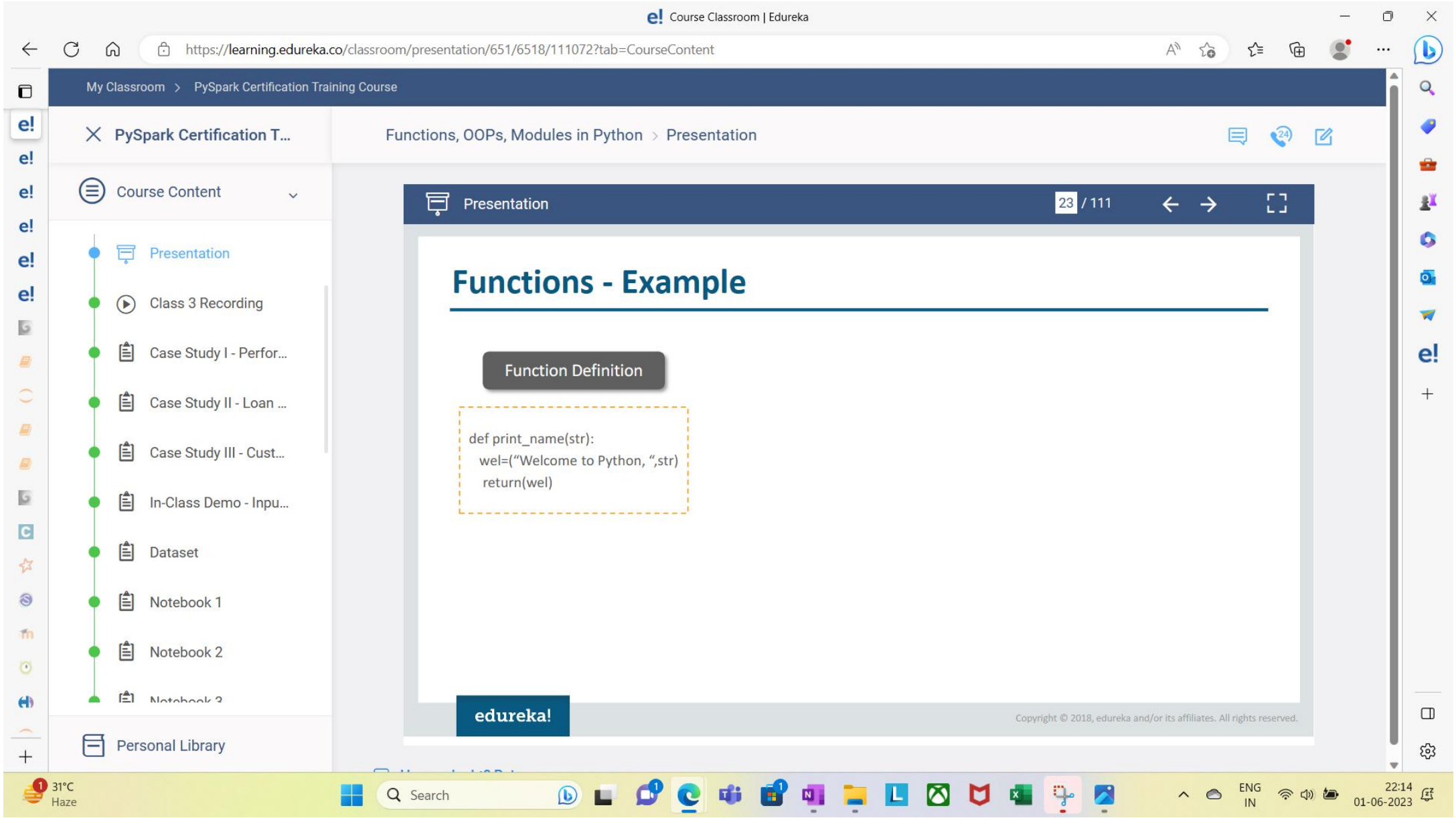

e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
18 / 111
Built-in Functions
Built-in Functions are the Functions which are built into (already available) Python and can be accessed by end-users:
sorted() abs() all()
any() globals() bin()
bool() enumerate() eval()
chr() int() sum()
open() len() reversed()
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C Haze
Search
ENG IN
22:14 01-06-2023

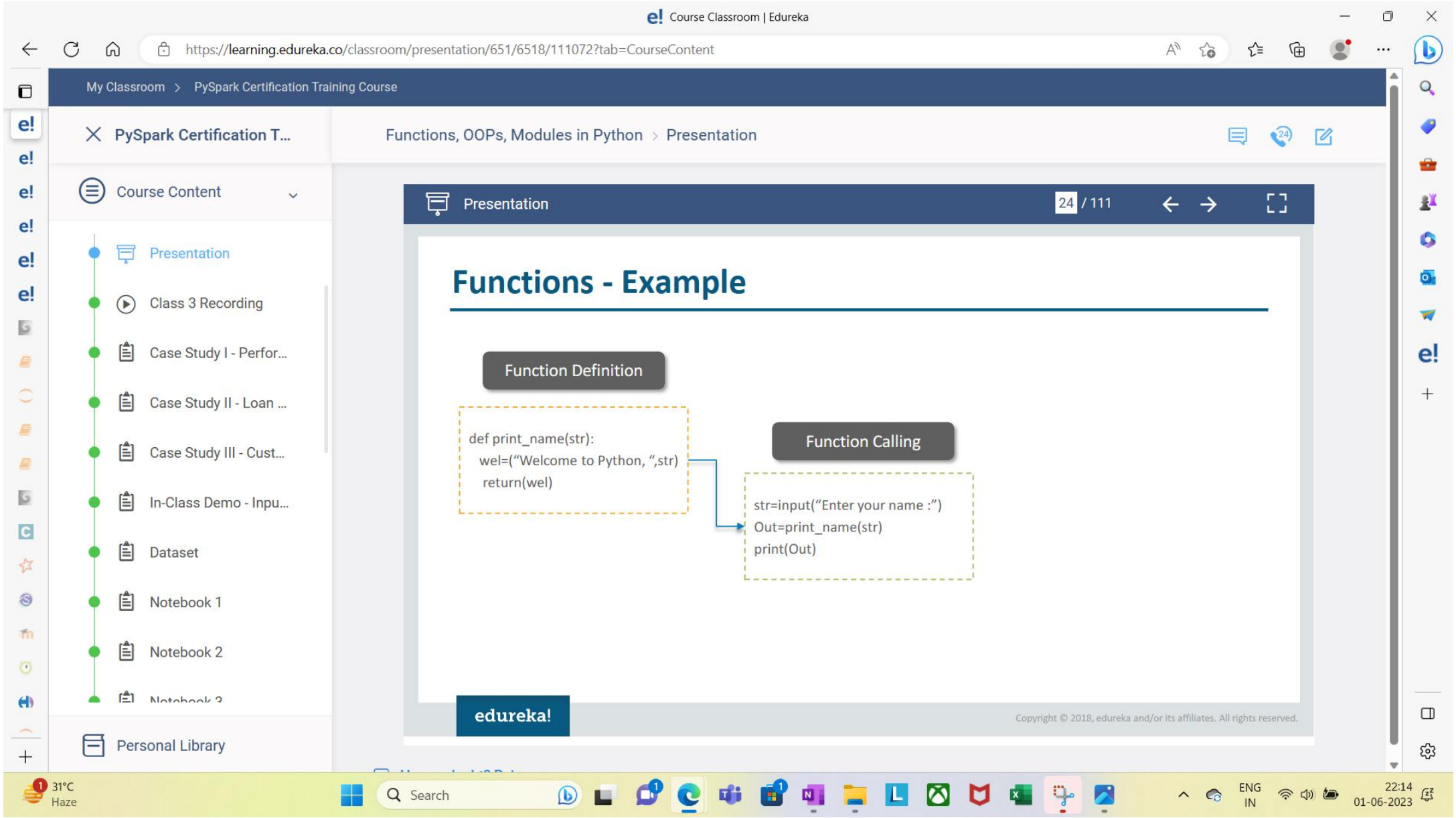

e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
19 / 111
Built-in Functions
Functions
Description
sorted()
Returns a new sorted list from the items in iterable
all()
The all() function returns True if all the elements of the supplied iterable are true
any()
The any() function is the converse of the all() function
bool()
The bool function converts the value to a Boolean, using the standard Python truth testing procedure
chr()
Returns the string representing a character whose Unicode code point is the integer
open()
Opens file and returns a corresponding file object
abs()
The abs() function returns the absolute value of an integer, floating point, or complex number
enumerate()
Returns an enumerate object with items and their index values
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023

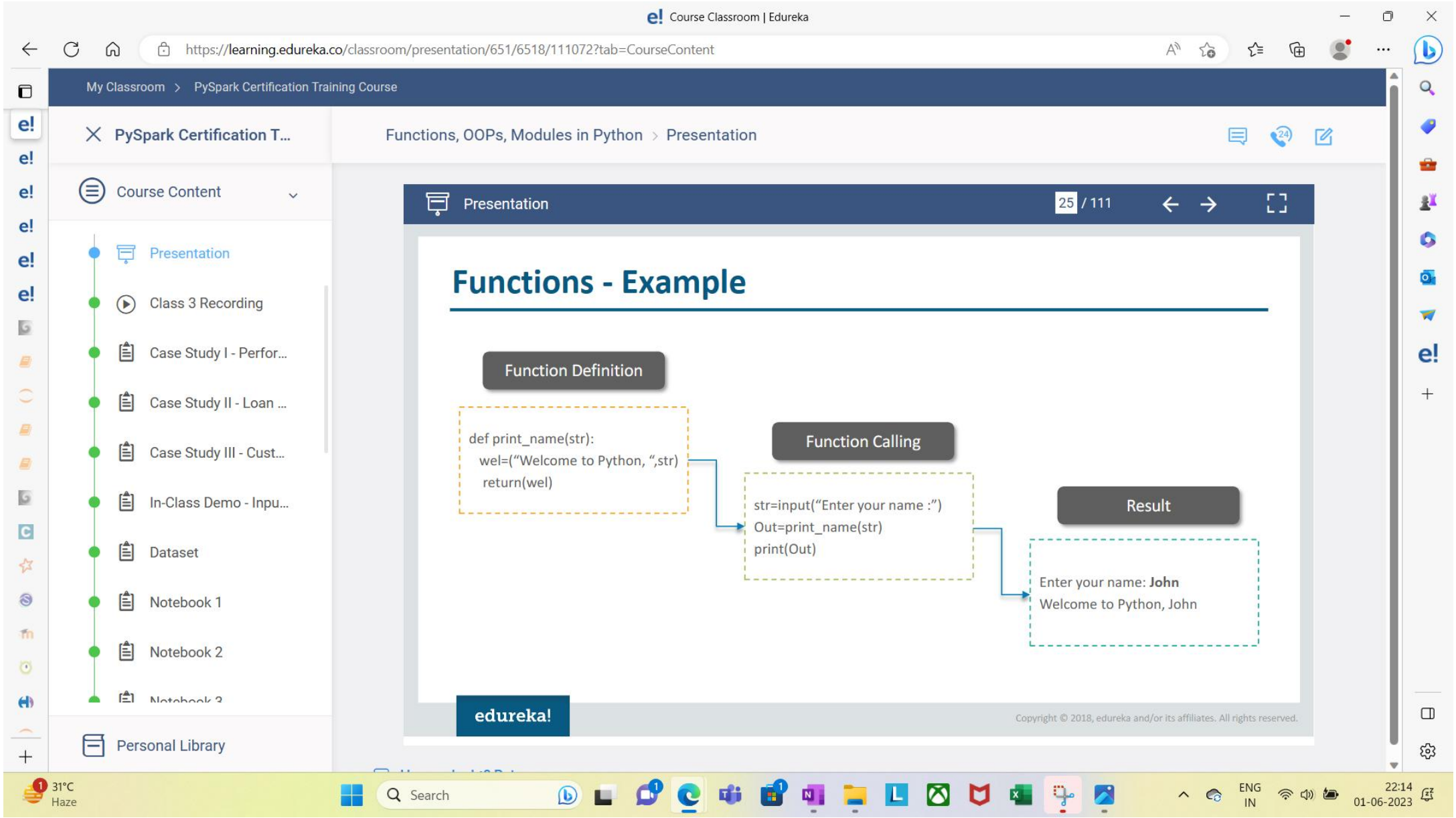

e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
20 / 111
Built-in Functions
Functions
Description
int()
Returns an integer object constructed from a number or string x
len()
len(s) returns the length of an object
globals()
Returns a dictionary representing the current global symbol table
bin()
Converts an integer number to a binary string prefixed with "0b"
eval()
The arguments are a string and optional globals and locals. If provided, globals must be a dictionary. If provided, locals can be any mapping object.
sum()
sum(iterable) sums the numeric values in an iterable such as a list, tuple, or set.
reversed()
reversed(seq) is a reverse iterator on an object of the type that you can loop through and process. The list and tuple types are supported with this function
int()
Returns an integer object constructed from a number or string x
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023

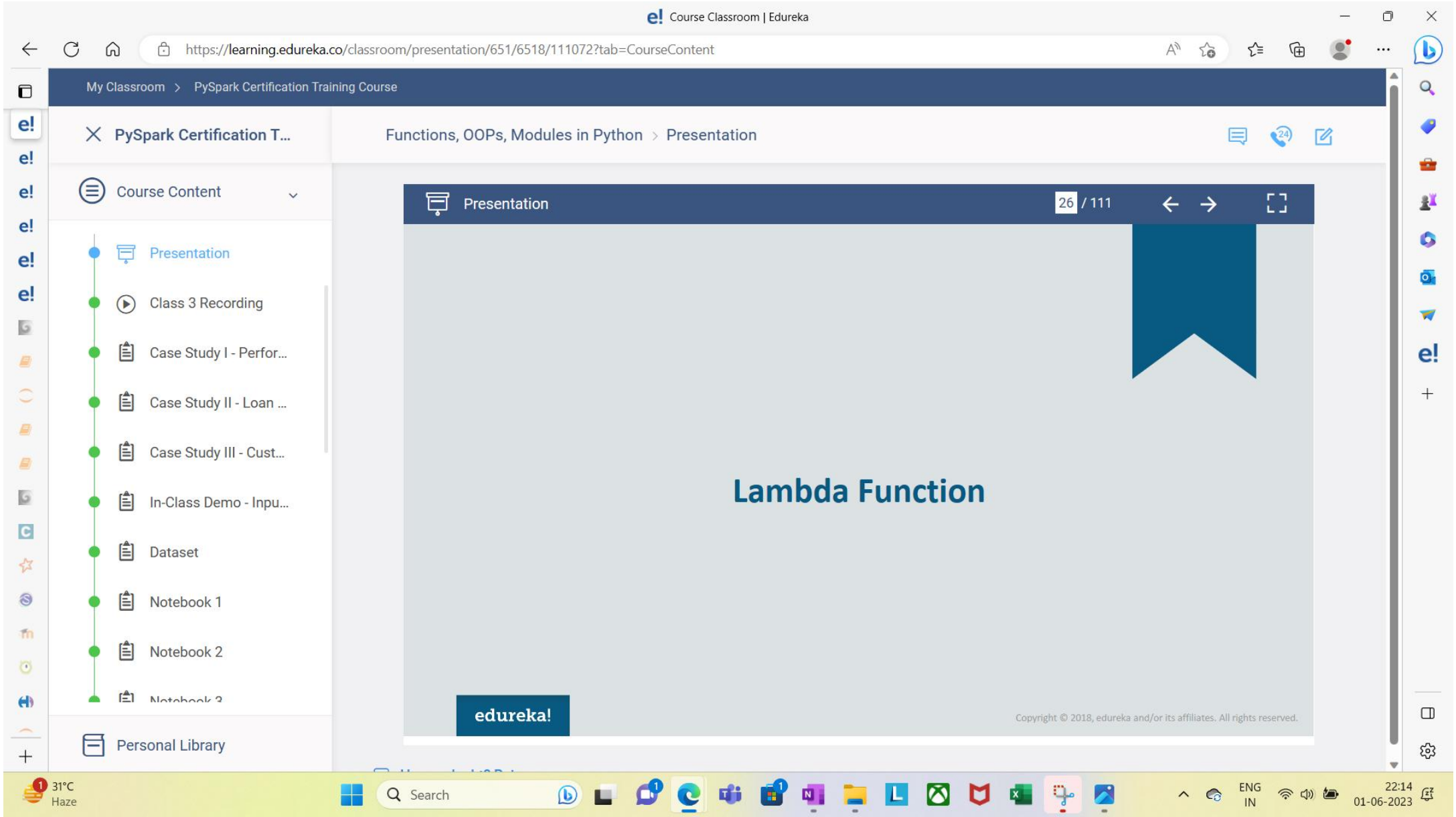

Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
21 / 111
Important Built-in Function
enumerate()
Return an enumerate object. The Iterable must be a sequence, an iterator, or some other object which supports iteration
grocery = ['bread', 'milk', 'butter']
enumerateGrocery = enumerate(grocery)
print(type(enumerateGrocery))
# converting to list
print(list(enumerateGrocery))
# changing the default counter
enumerateGrocery = enumerate(grocery, 10)
print(list(enumerateGrocery))
<class 'enumerate'>
[(0, 'bread'), (1, 'milk'), (2, 'butter')]
[(10, 'bread'), (11, 'milk'), (12, 'butter')]
Enumerate returns the list of tuples. Tuples consists index values and items from the list
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C Haze
Search
ENG IN
22:14 01-06-2023


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
22 / 111
Functions - Example
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
23 / 111
Functions - Example
Function Definition
def print_name(str):
wel=("Welcome to Python, ",str)
return(wel)
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
24 / 111
Functions - Example
Function Definition
def print_name(str):
wel=("Welcome to Python, ",str)
return(wel)
Function Calling
str=input("Enter your name :")
Out=print_name(str)
print(Out)
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023

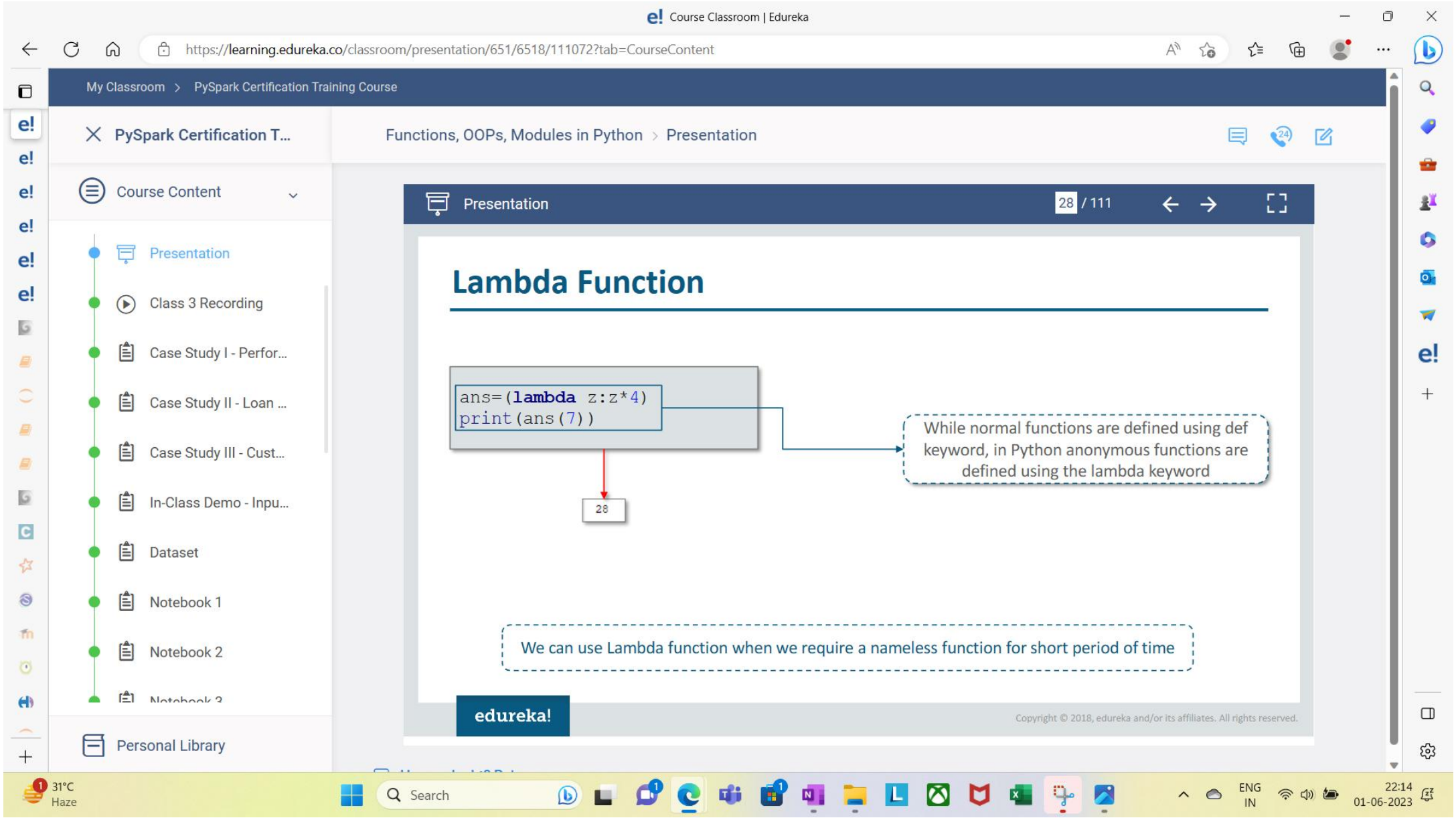

Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
25 / 111
Functions - Example
Function Definition
def print_name(str):
wel=("Welcome to Python, ",str)
return(wel)
Function Calling
str=input("Enter your name :")
Out=print_name(str)
print(Out)
Result
Enter your name: John
Welcome to Python, John
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
26 / 111
Lambda Function
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023

# Lambda Function

- In case you wish to make your functions more concise, easy to write and read, you can create Lambda functions
- Anonymous Lambda function can be defined using the keyword lambda
- Lambda functions cannot contain commands, and they can not contain more than one expression
- Lambda function can take any number of arguments (including optional arguments) and returns the value of a single expression

edureka!


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
28 / 111
Lambda Function
ans=(lambda z:z*4)
print(ans(7))
28
While normal functions are defined using def keyword, in Python anonymous functions are defined using the lambda keyword
We can use Lambda function when we require a nameless function for short period of time
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023

# Map, Filter, Reduce using Lambda

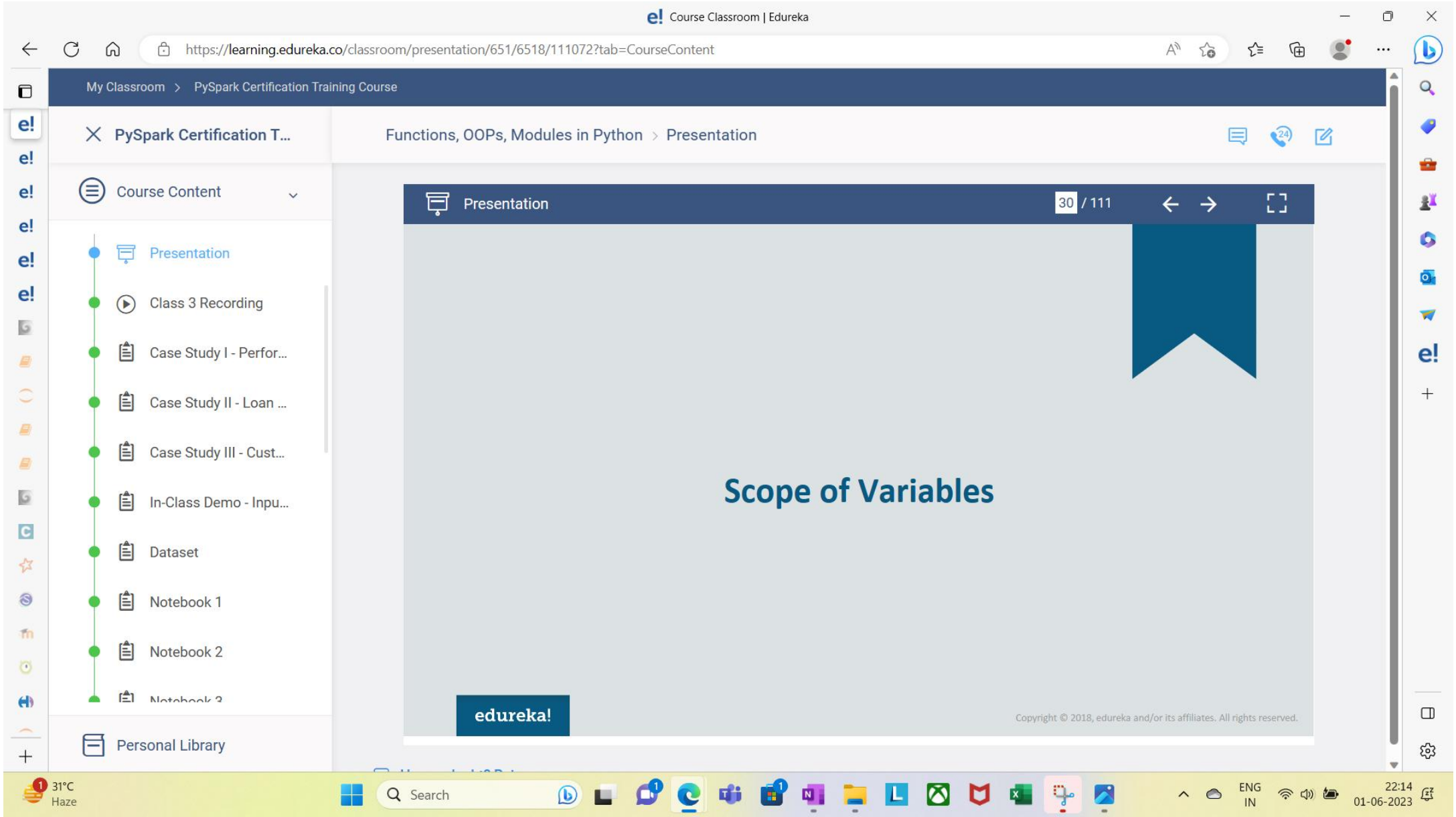
```
#Map
items = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, items))
print(squared)

#filter
number_list = range(-5, 5)
less_than_zero = list(filter(lambda x: x < 0, number_list))
print(less_than_zero)

#reduce
from functools import reduce
product = reduce((lambda x, y: x * y), [1, 2, 3, 4])
print(product)
```

*map* applies a function to all the items in an input_list

As the name suggests, *filter* creates a list of elements for which a function returns true

*reduce* is a really useful function for performing some computation on a list and returning the result

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
30 / 111
Scope of Variables
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:14
01-06-2023

Presentation 31 / 111

# Scope of a Variable


Global Variables

Local Variables

The variables which are not declared in the function (outside the function) and can be used anywhere in the program

```
a=50
def number():
    b=30
    print(b)

print(a)
number()
```

edureka!

Presentation 32 / 111

# Scope of a Variable


Global Variables

Local Variables

```
a=50
def number():
    b=30
    print(b)

print(a)
number()
```

The variables which are declared within any function can be used within that function only

edureka!

Presentation 33 / 111

# Scope of a Variable

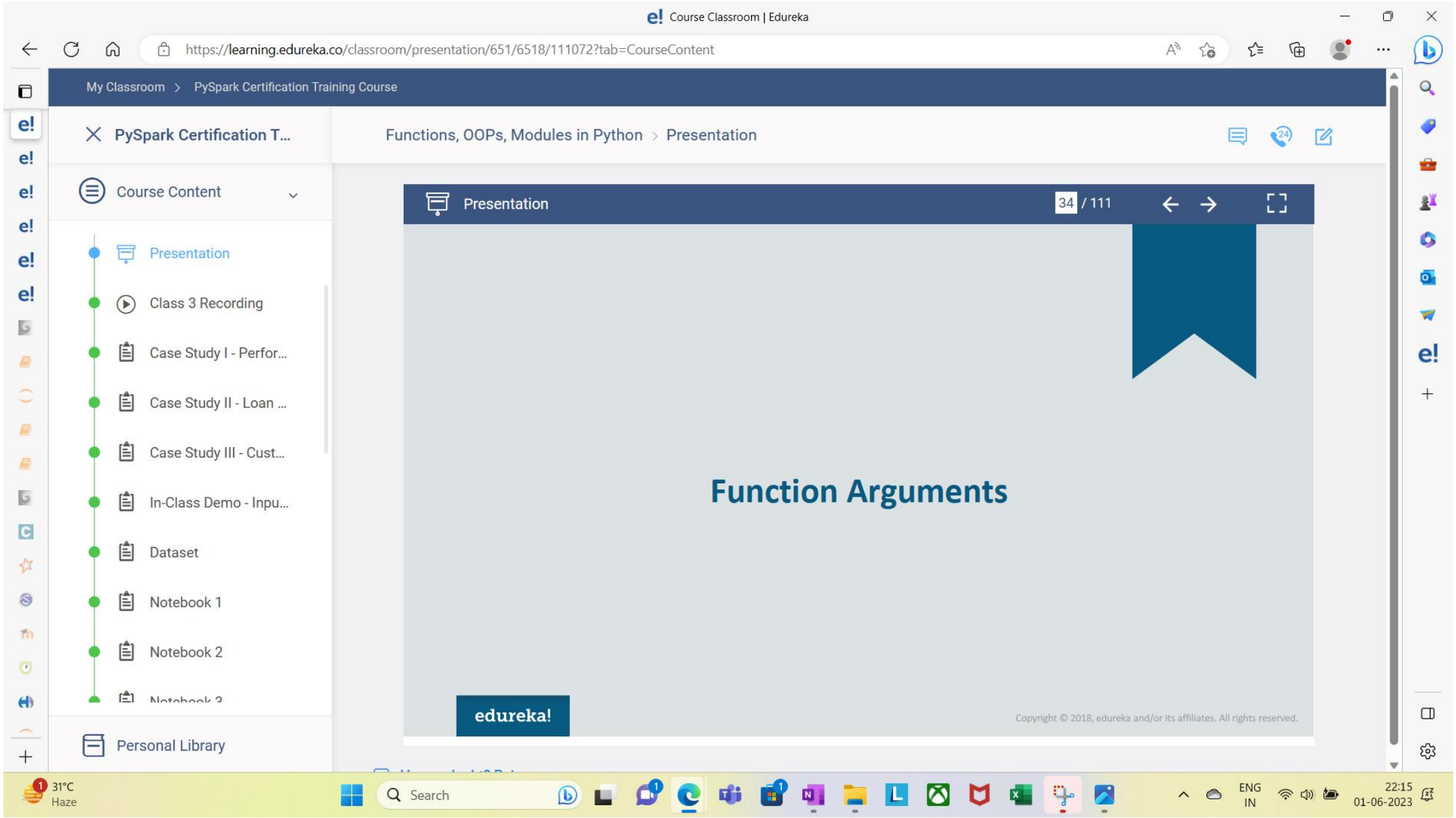
```
a=30
def add(b):
    c=30
    print("c=",c)
    sum=b+c
    print("Addition is: ",sum)

print(a)
add(40)
print(c)
```

Variable **a** is a global variable. It can be accessed anywhere in the program

Variable **c** is a Local variable. So, it can not be used outside the function

```
30
c= 30
Addition is:  70

NameError                                 Traceback (most recent call last)
<ipython-input-2-21e0e25f0ce1> in <module>()
      8 print(a)
      9 add(40)
---> 10 print(c)

NameError: name 'c' is not defined
```

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
34 / 111
Function Arguments
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023

Presentation 35 / 111

# Required Arguments

The number of arguments in the function call should match exactly with the function definition

edureka!

Presentation 36 / 111

## Required Arguments - Example

```
def print_name(str):
    print("Welcome to
Python, ",str)
    return()

print_name()
```

Arguments are not given when the function is called. So, an error is obtained.

```
TypeError                                 Traceback (most recent call last)
<ipython-input-4-53c5e89ae19a> in <module>()
      3     return()
      4
----> 5 print_name()
      6

TypeError: print_name() missing 1 required positional argument: 'str'
```

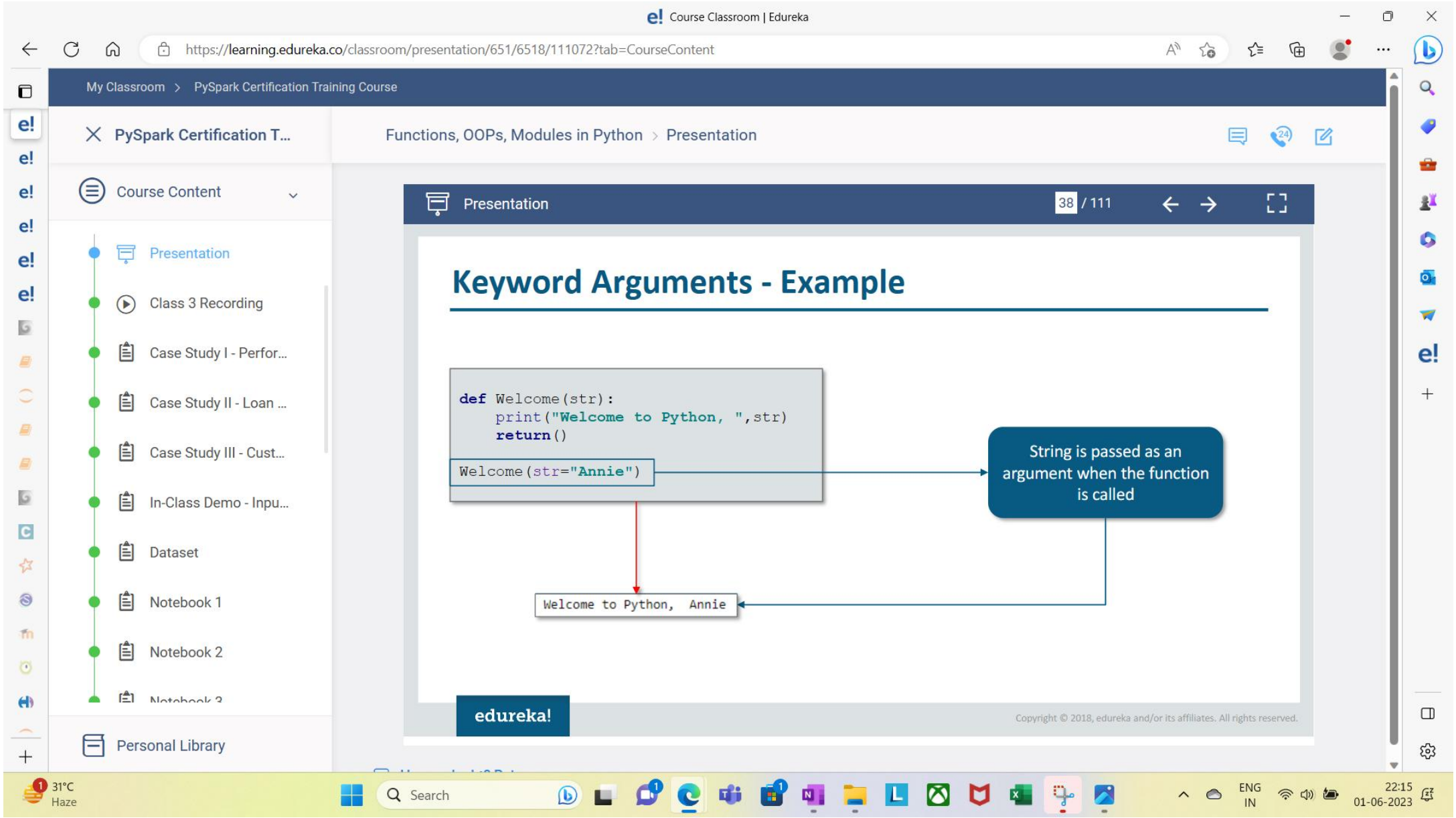
edureka!

Presentation 37 / 111

# Keyword Arguments

When keyword arguments are used in a function call, the caller identifies the arguments by the parameter name

edureka!


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
38 / 111
Keyword Arguments - Example
def Welcome(str):
print("Welcome to Python, ",str)
return()
Welcome(str="Annie")
String is passed as an argument when the function is called
Welcome to Python, Annie
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C Haze
Search
ENG IN
22:15 01-06-2023

Presentation 39 / 111

## Keyword Arguments - Example

```
def add(a,b,c,d):
    print(a)
    print(b)
    print(c)
    print(d)
    sum=a+b+c+d
    return(sum)

ans=add(d=10,a=50,c=5,b=5)
print("Sum is : ",ans)
```

Integer values are passed as arguments when the function is called

```
50
5
5
10
Sum is :  70
```

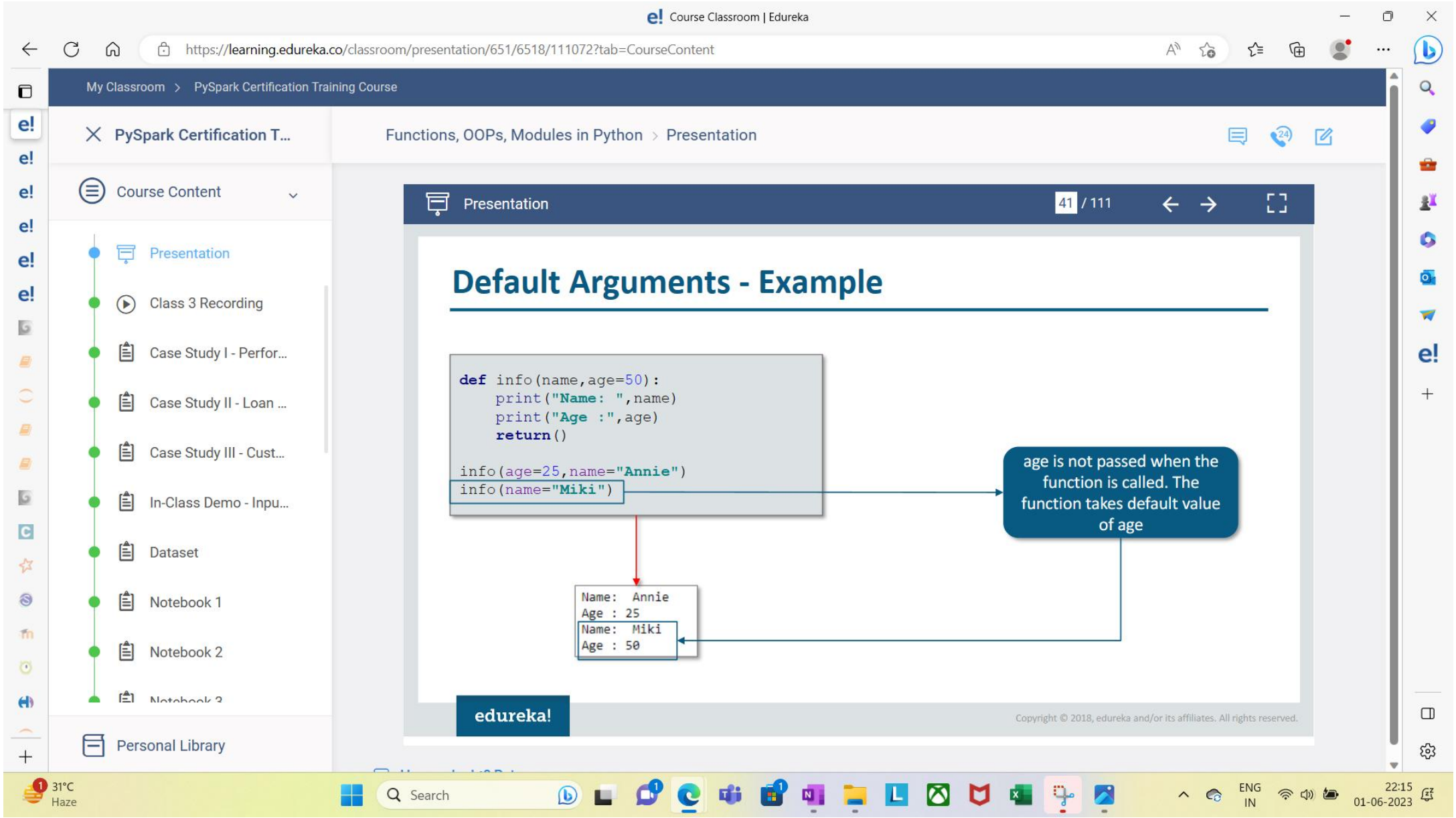
edureka!

Presentation 40 / 111

# Default Arguments

A default argument is an argument, that assumes a default value if a value is not provided in the function call for that argument

edureka!


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
41 / 111
Default Arguments - Example
def info(name,age=50):
print("Name: ",name)
print("Age :",age)
return()
info(age=25,name="Annie")
info(name="Miki")
age is not passed when the function is called. The function takes default value of age
Name: Annie
Age : 25
Name: Miki
Age : 50
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023

# Default Arguments - Example

```
def employee(name,id,salary=10000):
    print("Name of Employee: ",name)
    print("ID no of Employee: ",id)
    print("Salary of Employe: ",salary)

employee("Sara",101,20000)
employee("Alex",98)
```

salary is not passed when the function is called. The function takes the default value of salary

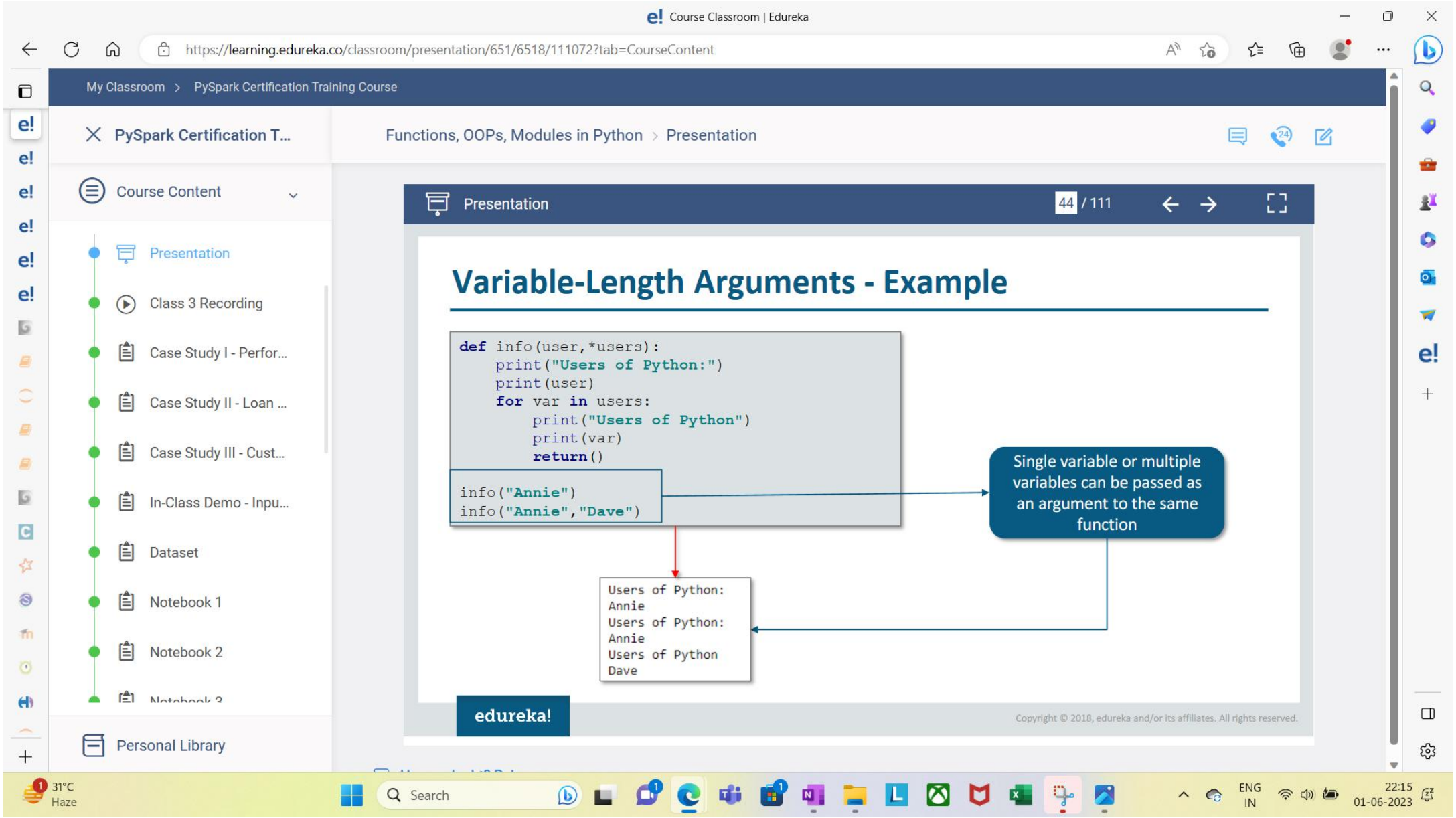
```
Name of Employee:  Sara
ID no of Employee:  101
Salary of Employe:  20000
Name of Employee:  Alex
ID no of Employee:  98
Salary of Employe:  10000
```

edureka!

# Variable-Length Arguments

You may need to process a function for more arguments than you have specified while defining the function, so variable length arguments can be used.

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
44 / 111
Variable-Length Arguments - Example
def info(user,*users):
print("Users of Python:")
print(user)
for var in users:
print("Users of Python")
print(var)
return()
info("Annie")
info("Annie","Dave")
Single variable or multiple variables can be passed as an argument to the same function
Users of Python:
Annie
Users of Python:
Annie
Users of Python
Dave
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023

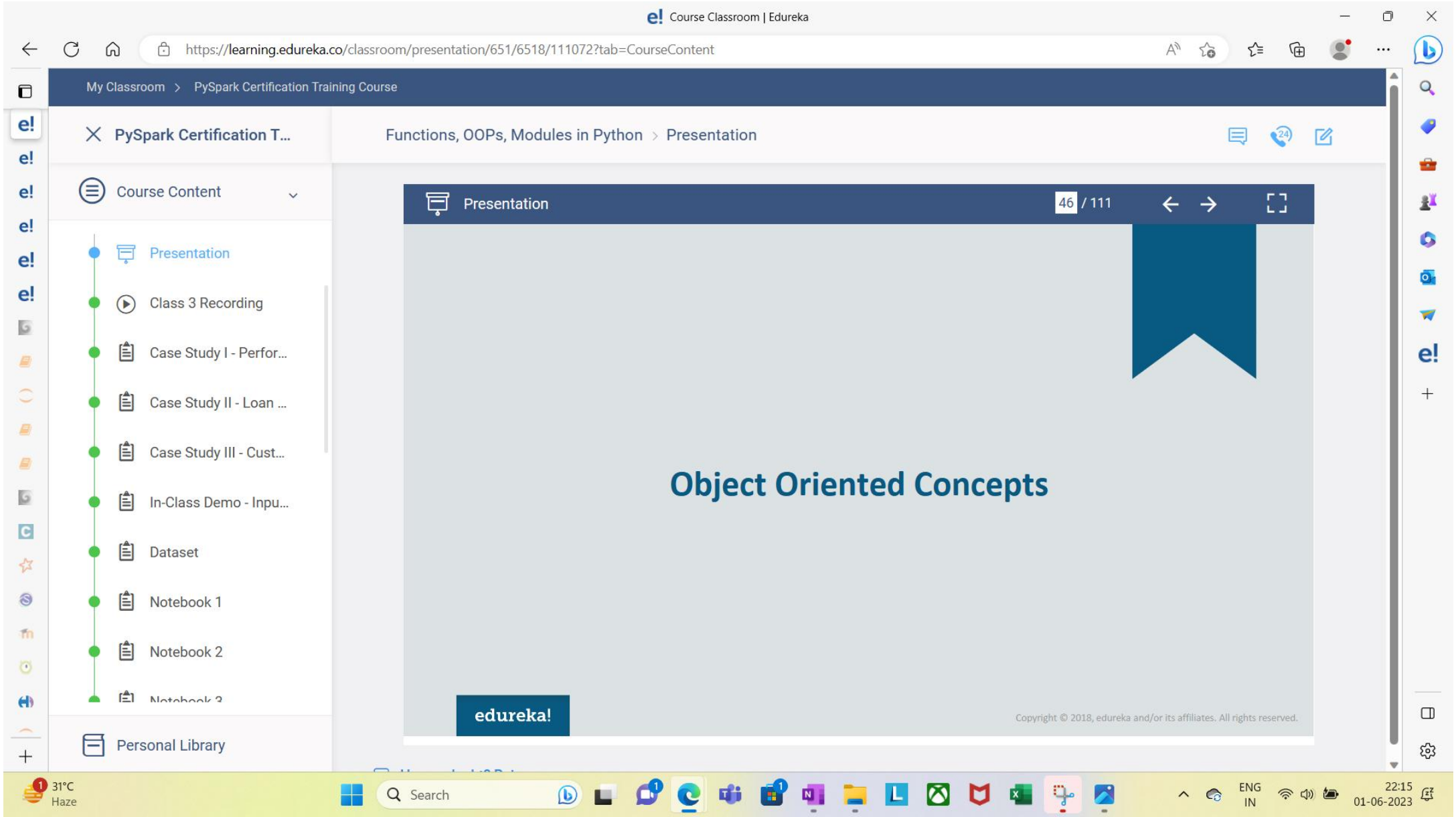
# Variable-Length Arguments - Example

```
def myFunction(arg1, arg2, arg3, *args, **kwargs):
    print('First Normal Argument : ' + str(arg1))
    print('Second Normal Argument : ' + str(arg2))
    print('Third Normal Argument : ' + str(arg3))
    print('Non-keyworded Argument : ' + str(args))
    print('Keyworded Argument : ' + str(kwargs))

myFunction(1, 2, 3, 4, 5, 6, 7, name='Mandar',
country='India', age=25)
```

When you pass a dictionary, it is collected in function using **kwargs

```
First Normal Argument : 1
Second Normal Argument : 2
Third Normal Argument : 3
Non-keyworded Argument : (4, 5, 6, 7)
Keyworded Argument : {'name': 'Mandar', 'country': 'India', 'age': 25}
```

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
46 / 111
Object Oriented Concepts
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023

Presentation 47 / 111

# Object Oriented Concepts

Python is an Object Oriented language, because of this creating and using classes and objects becomes easy

Data becomes active

Code is reusable, thus less code may have to be written

Ability to simulate real world events much more effectively

Programmer is able to produce faster, more accurate, and better written applications

edureka!

Presentation 48 / 111

# Python OOP is Different than Other OOPs

You can't write a single piece of your code in Java without using *class*, but you can do so in python, even many projects don't use it

Python is an interpreted languages, i.e., it execute code line-by-line, if your program has some error, that error will be shown when your line of code will execute, which is not the case with Java

edureka!

Presentation 49 / 111

# Use Case

SRT is a Multi National Company. This Company wants to create employee information sheet, which will include name, employee ID, and their progress


Employee

| Weekly | Monthly | Progress |
|---|---|---|
| | | |
| | | |
| | | |

edureka!

Presentation 50 / 111

# Use Case - Solution

However, it will be difficult to create a separate sheet for every employee. So they decided to create one class of Employee's Information and create object of every employee and call class for new employee


Class

Objects

edureka!

Presentation 51 / 111

# Class and Objects

*Class* is a blueprint used to create objects having same property or attribute as its class


Class

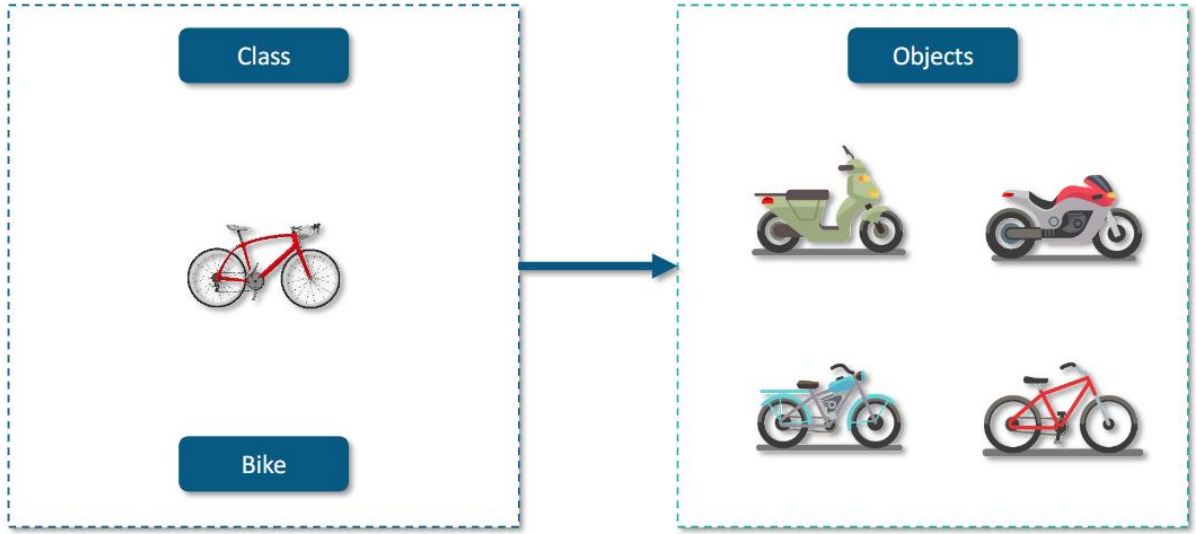
An *Object* is an instance of a class which contains variables and methods

Objects

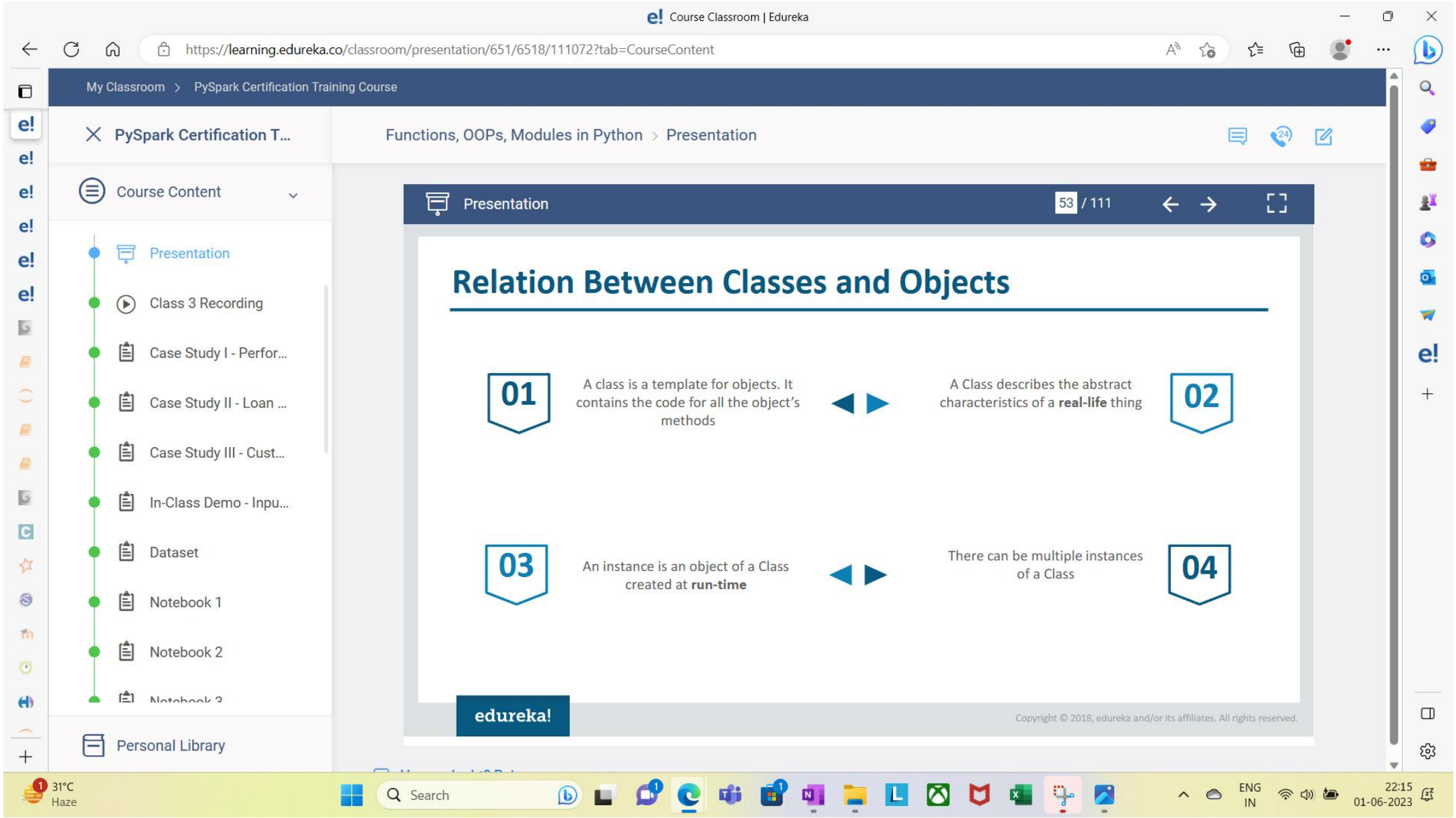
edureka!

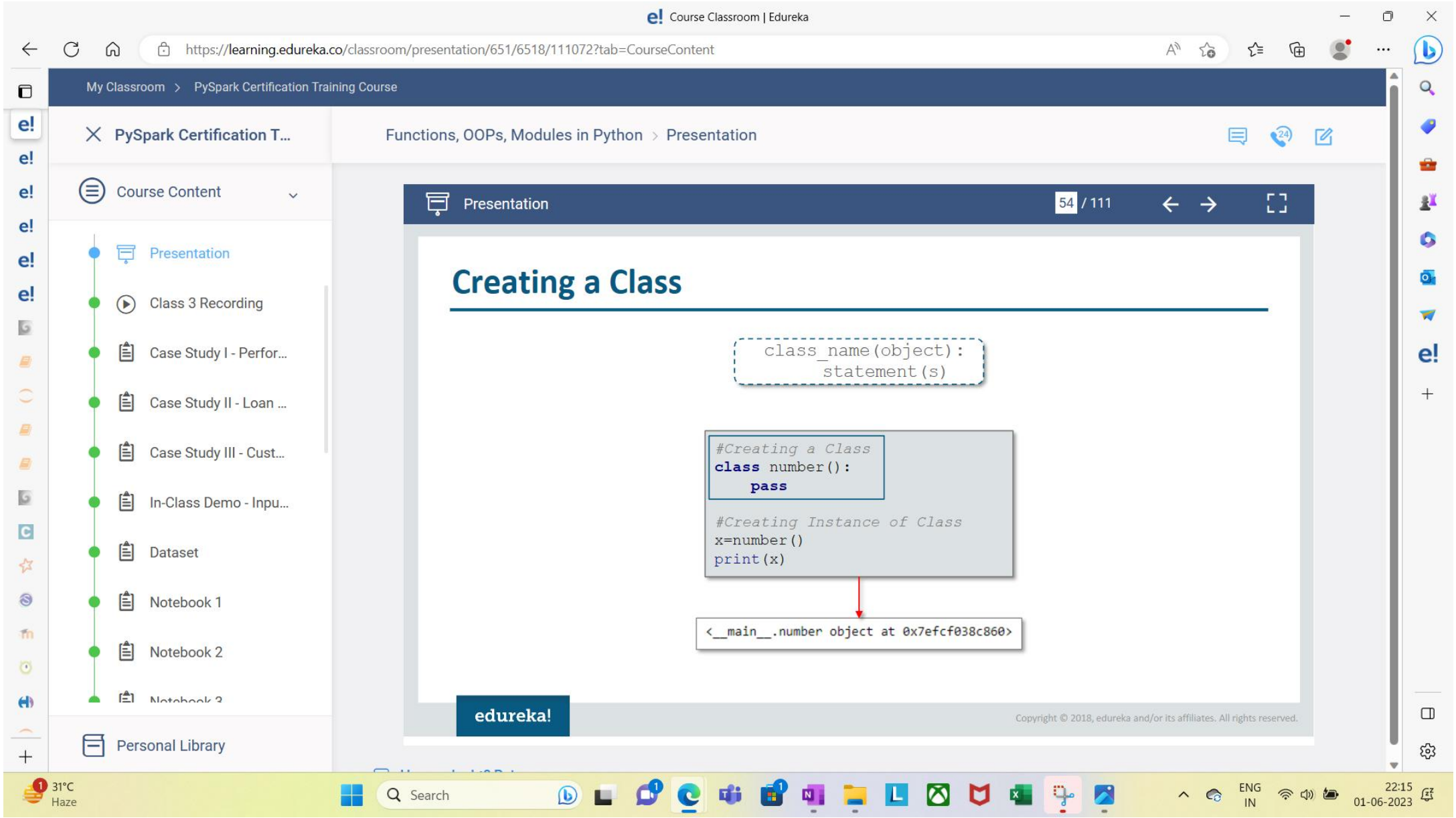
Presentation 52 / 111

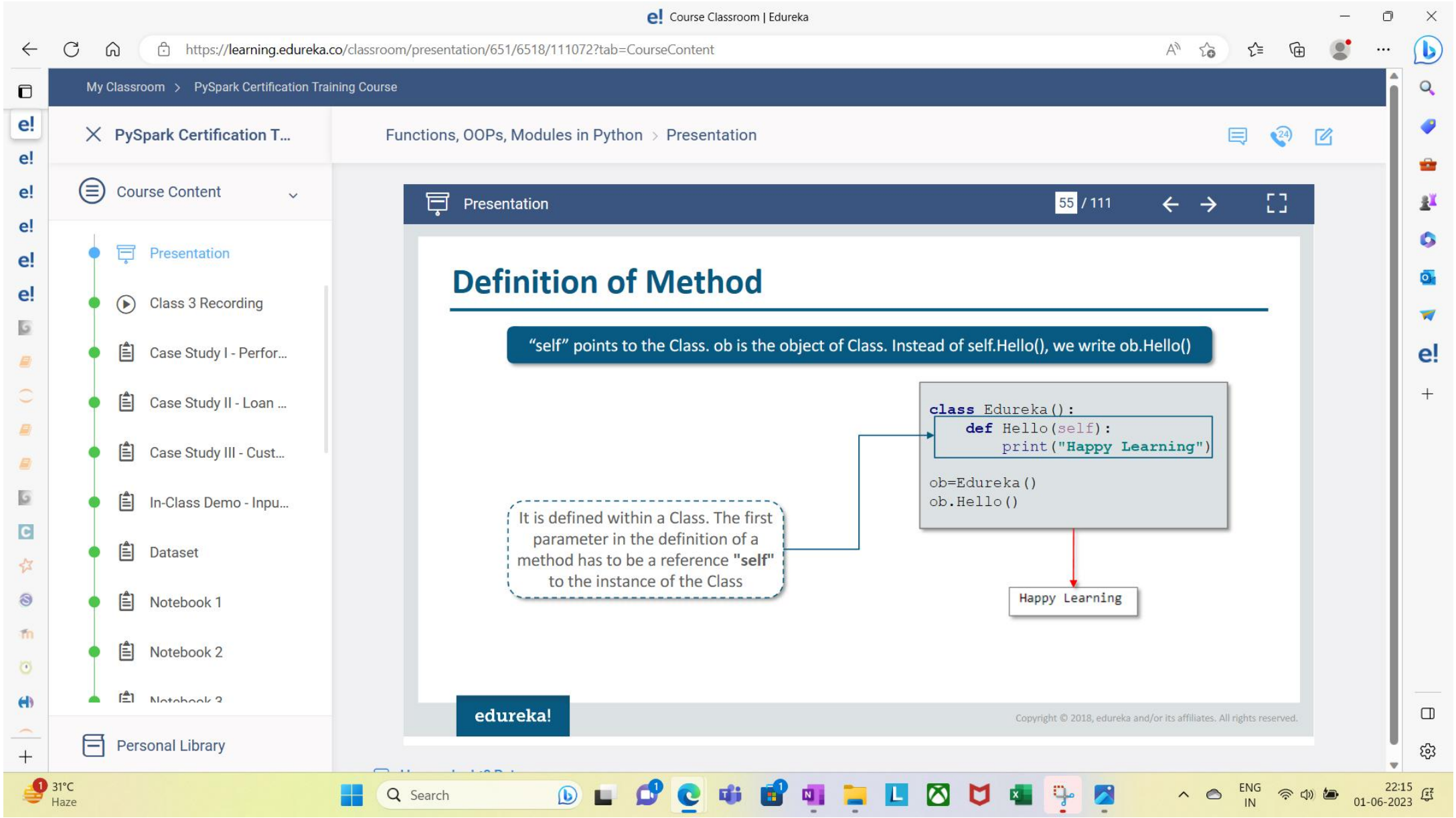
## Class and Objects




edureka!


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
53 / 111
Relation Between Classes and Objects
01
A class is a template for objects. It contains the code for all the object's methods
A Class describes the abstract characteristics of a real-life thing
02
03
An instance is an object of a Class created at run-time
There can be multiple instances of a Class
04
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
54 / 111
Creating a Class
class_name(object):
statement(s)
#Creating a Class
class number():
pass
#Creating Instance of Class
x=number()
print(x)
<__main__.number object at 0x7efcf038c860>
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
55 / 111
Definition of Method
"self" points to the Class. ob is the object of Class. Instead of self.Hello(), we write ob.Hello()
class Edureka():
def Hello(self):
print("Happy Learning")
ob=Edureka()
ob.Hello()
It is defined within a Class. The first parameter in the definition of a method has to be a reference "self" to the instance of the Class
Happy Learning
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023

Presentation 56 / 111

# Scope of Variables


Global Variables

Local Variables

The variables which are not declared in the Class (outside the class) and can be used anywhere in the program

```
a=50
class Edureka():
    b=60
    print(b)

ob=Edureka()
```

edureka!

Presentation 57 / 111

# Scope of Variables


Global Variables

Local Variables

```
a=50
class Edureka():
    b=60
    print(b)

ob=Edureka()
```

The variables which are declared within any Class can be used within that Class only

edureka!

Presentation 58 / 111

# Attributes

Class attributes are attributes which are owned by the class itself. They will be shared by all the instances of the class

Attributes

| Name | Employee ID | Progress |
|---|---|---|
| | | |
| | | |
| | | |

There are two types of Attributes:

Built-in Attributes

User Defined Attributes

edureka!

# Built-in Class Attributes

```
class Edureka:
    empcount=0

print("Edureka.__dict__:",Edureka.__dict__)
print("Edureka.__doc__:",Edureka.__doc__)
print("Edureka.__name__",Edureka.__name__)
print("Edureka.__module__:",Edureka.__module__)
print("Edureka.__bases__:",Edureka.__bases__)
```

```
Edureka.__dict__: {'__module__': '__main__', 'empcount': 0, '__dict__': <attribute '__dict__' of 'Edureka'
Edureka.__doc__: None
Edureka.__name__ Edureka
Edureka.__module__: __main__
Edureka.__bases__: (<class 'object'>,)
```

- Dictionary containing the class's namespace
- Class documentation string or None if undefined
- Class name
- Module name in which the class is defined
- A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list

edureka!

Presentation 60 / 111

# Attributes Defined by Users

```
class Foo(object):
    def math(self):
        pass
```

Attributes are created inside the Class definition

We can dynamically create new attributes for existing instances of a Class

edureka!

Presentation 61 / 111

# Private, Public and Protected Attributes

*Private* attributes can only be accessed inside of the class definition

*Public* attributes can and should be, freely used

*Protected* attributes are accessible **only** from within the class and it's subclasses

edureka!

# Public, Protected and Private Attributes- Example

```
class Edureka():
    def __init__(self):
        self.__pri=("I am Private")
        self._pro=("I am Protected")
        self.pub=("I am Public")

ob=Edureka()
print(ob.pub)
print(ob._pro)
print(ob.__pri)
```

Accessing public attribute

Accessing protected attribute

Accessing private attribute

```
I am Public
I am Protected
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-12-53c5a0bf6501> in <module>()
      8 print(ob.pub)
      9 print(ob._pro)
---> 10 print(ob.__pri)

AttributeError: 'Edureka' object has no attribute '__pri'
```

edureka!

Presentation 63 / 111

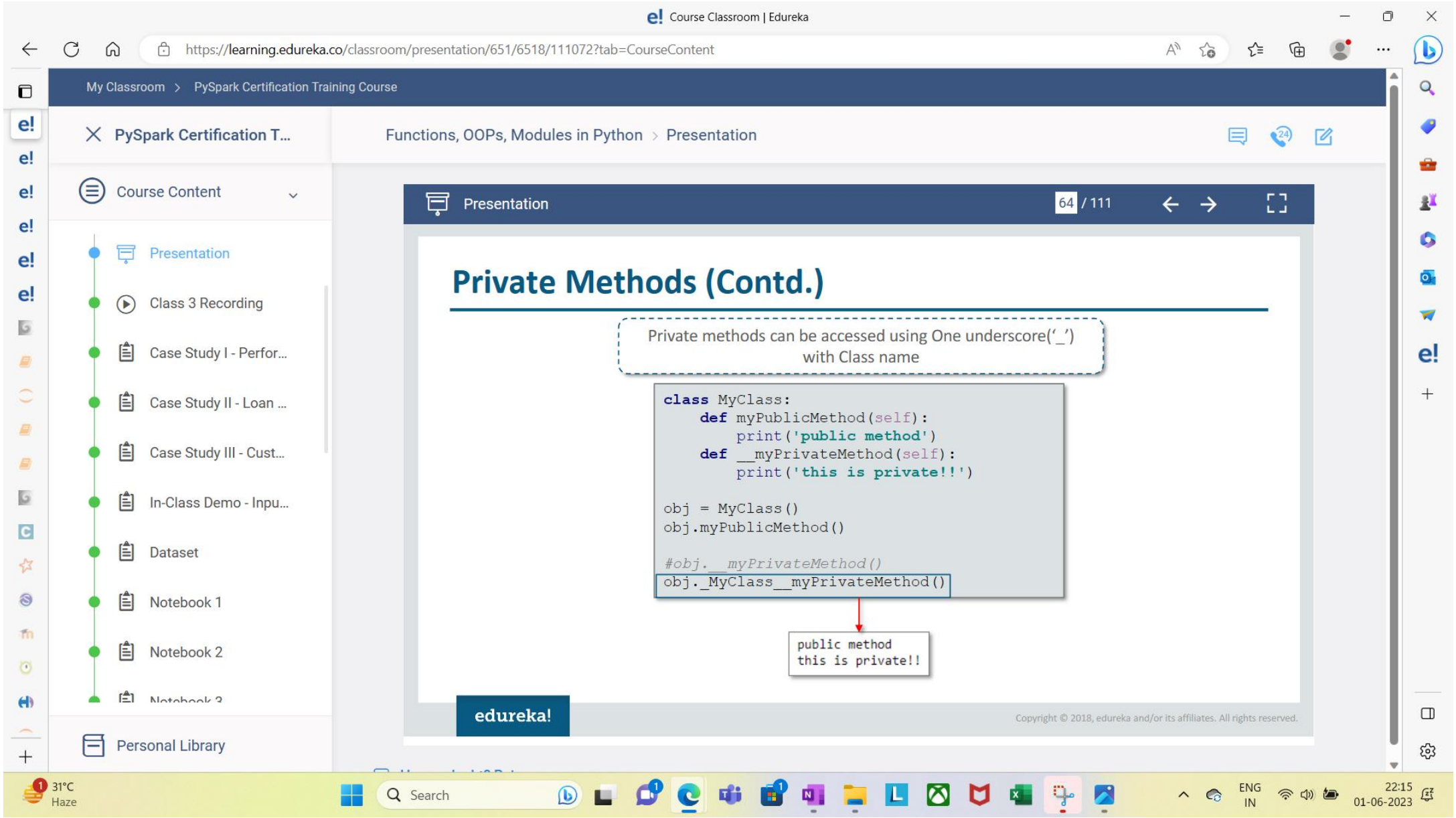
# Private Methods

When the attributes of an object can only be accessed inside the Class, it is called a **Private Class**. Python use **two underscores** to hide a Method. Two underscores can also be used to hide a Variable

```
class MyClass:
    def myPublicMethod(self):
        print('public method')
    def __myPrivateMethod(self):
        print('this is private!!')

obj = MyClass()
obj.myPublicMethod()

obj.__myPrivateMethod()
```

```
public method
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-14-c702710dbcff> in <module>()
      8 obj.myPublicMethod()
      9
---> 10 obj.__myPrivateMethod()

AttributeError: 'MyClass' object has no attribute '__myPrivateMethod'
```

edureka!


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
64 / 111
Private Methods (Contd.)
Private methods can be accessed using One underscore('_') with Class name
class MyClass:
def myPublicMethod(self):
print('public method')
def __myPrivateMethod(self):
print('this is private!!')
obj = MyClass()
obj.myPublicMethod()
#obj.__myPrivateMethod()
obj._MyClass__myPrivateMethod()
public method
this is private!!
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C Haze
Search
ENG IN
22:15 01-06-2023

# Class Variable and Instance Variable

```
class Edureka:
    domain=("Data Science")
    def Setcourse(self,name):
        self.name=name

ob1=Edureka()
ob2=Edureka()

ob1.Setcourse("Python")
ob2.Setcourse("Machine Learning")

print(ob1.domain)
ob1.domain = 'GOD'
print(ob1.domain)
print(ob2.domain)

print(ob1.name)
print(ob2.name)
```

Defining a class variable

Defining an instance of class Edureka

Class variable is shared by both instances ob1 and ob2

Instance variable is unique to each instance

```
Data Science
GOD
Data Science
Python
Machine Learning
```

edureka!

# Constructor and Destructor

```
class TestClass:
    def __init__(self):
        print("constructor")

    def __del__(self):
        print("destructor")

if __name__ == "__main__":
    obj = TestClass()
    del obj
```

The constructor is implemented using *__init__(self)* which you can define parameters that follows the self

The destructor is defined using *__del__(self)*. In the example, the *obj* is created and manually deleted, therefore, both messages will be displayed

```
constructor
destructor
```

edureka!

Presentation 67 / 111

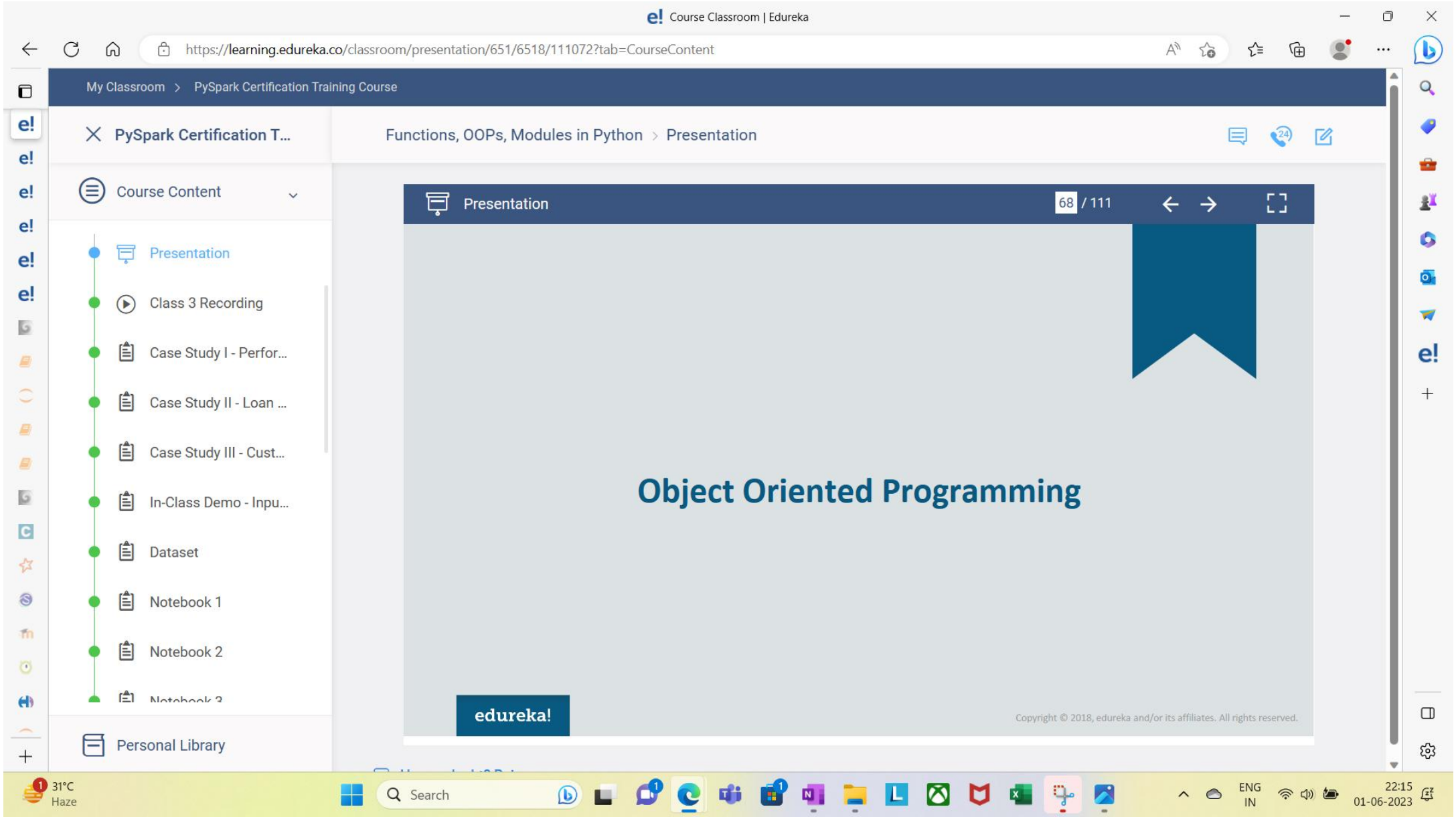
# Multiple Constructors

```
class Date:
   def __init__(self, year, month, day): #year-month-day
       self.year = year
       self.month = month
       self.day = day
       # print("init")

   @classmethod
   def dmy(cls, day, month, year): #day-month-year
       # print("dmy")
       cls.year = year
       cls.month = month
       cls.day = day
       #order of return should be same as init
       return cls(cls.year, cls.month, cls.day)
```

```
  @classmethod
    def mdy(cls, month, day, year): #month-day-year
        # print("mdy")
        cls.year = year
        cls.month = month
        cls.day = day
        #order of return should be same as init
        return cls(cls.year, cls.month, cls.day)

a=Date(2016, 12, 11)
print(a.year) #2016

b=Date.dmy(9, 10, 2015)
print(b.year) #2015

a=Date.mdy(7, 8, 2014)
print(a.year) #2014
```

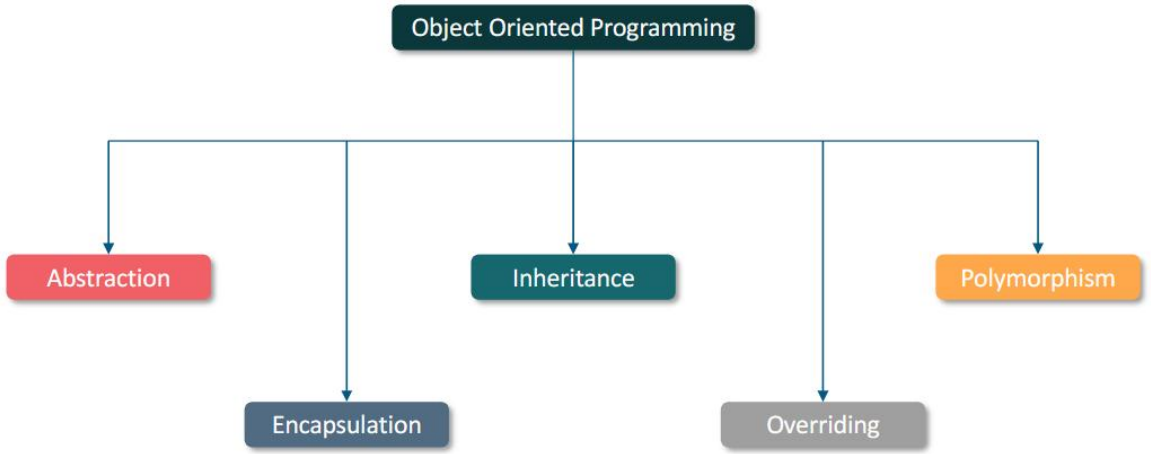
2016
2015
2014

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
68 / 111
Object Oriented Programming
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023

Presentation 69 / 111

## Key Concepts - Object Oriented Programming

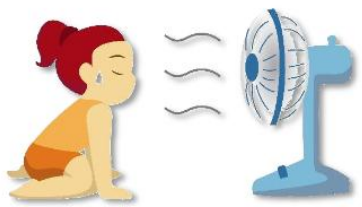



edureka!

Presentation 70 / 111

# Abstraction

*Abstraction* is simplifying complex reality by modelling classes appropriate to the problem
Class abstraction means to separate class implementation from the use of the class

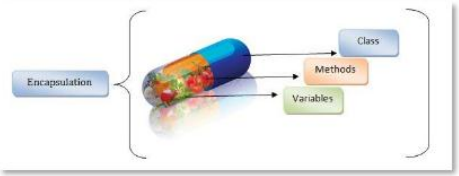


When you turn on the fan, you only know the fan is rotating. But, you do not know the actual mechanism behind it

edureka!

Presentation 71 / 111

# Encapsulation




**Encapsulation:** Combining the code into a public **interface**, and a private **implementation** of that interface

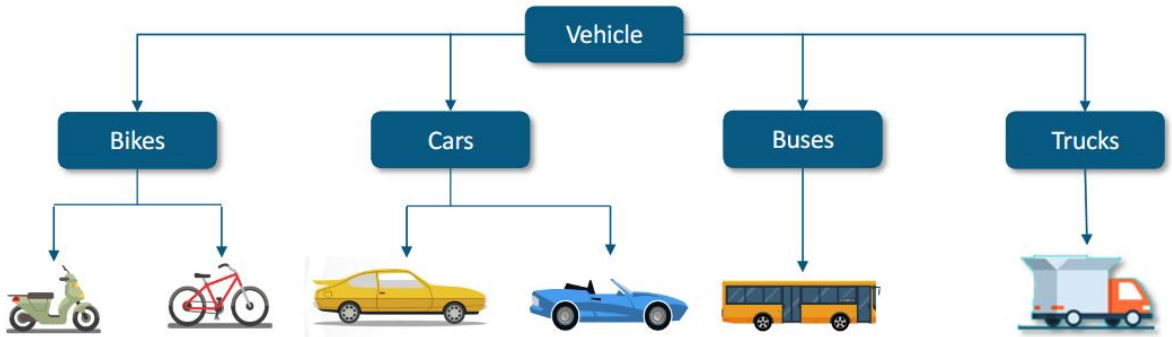
It is a Mechanism for restricting the access to some of an obmects components, which means that the internal representation of an object can not be seen from outside of the objects definition

edureka!

Presentation 72 / 111

# Inheritance

Inheritance is the powerful feature of Object Oriented Programming
It refers to deriving a class from the base class with little or no modification in it



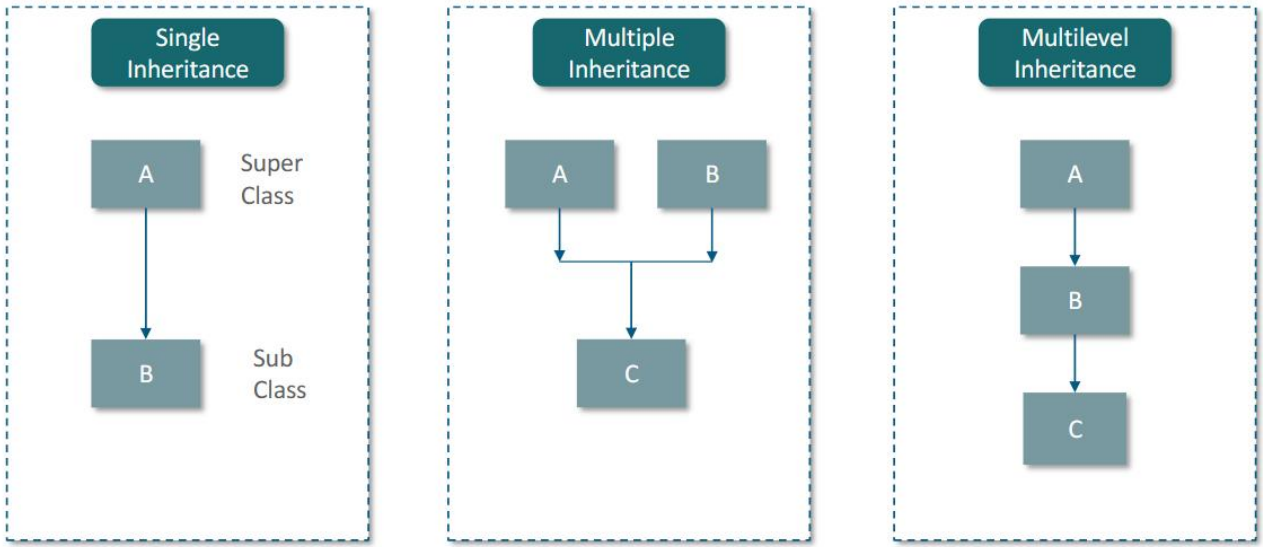

edureka!

PySpark Certification T...
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library



Presentation 73 / 111

## Types of Inheritance




edureka!

# Single Inheritance

Important benefits of inheritance are code reuse and reduction in the complexity of a program

```
class base1:
    def fun(self):
        print("In Class Base 1")

class sub(base1):
    pass

ob=sub()
ob.fun()
```

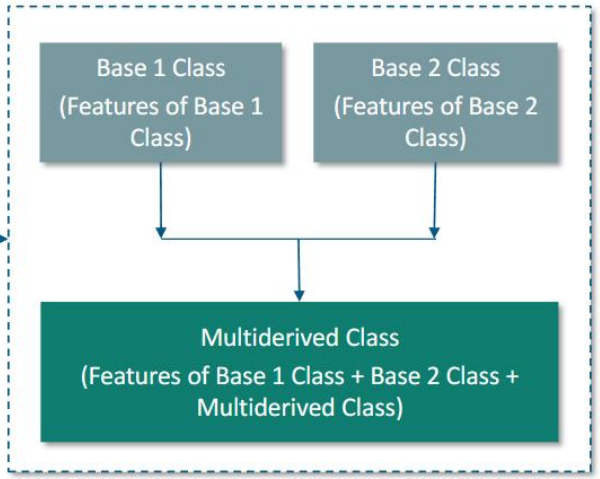
def fun function from base class in inherited in sub class and sub class can access the method inside that function

```
In Class Base 1
```

edureka!

# Multiple Inheritance

```
#Multiple Inheritance
class First(object):
    def __init__(self):
        super(First, self).__init__()
        print("first")

class Second(object):
    def __init__(self):
        super(Second, self).__init__()
        print("second")

class Third(Second, First):
    def __init__(self):
        super(Third, self).__init__()
        print("third")

Third();
```

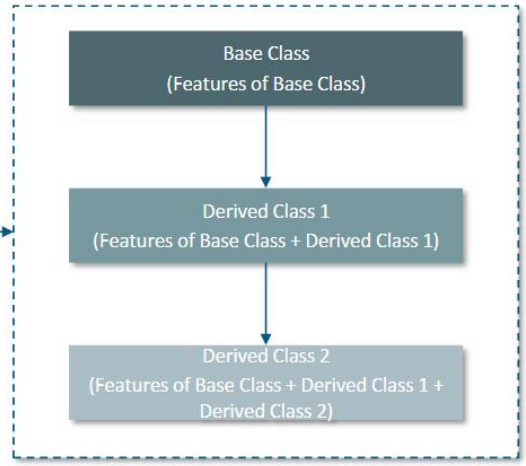



A single derived class may inherit from two or more than two base classes

edureka!

# Multilevel Inheritance

```
#Multilevel Inheritance
class Animal:
    def eat(self):
      print('Eating...')
class Dog(Animal):
   def bark(self):
      print('Barking...')
class BabyDog(Dog):
    def weep(self):
       print('Weeping...')
d=BabyDog()
d.eat()
d.bark()
d.weep()
```




```
Eating...
Barking...
Weeping...
```

The derived class inherits from a class, which in turn inherits from some other class

edureka!

Presentation 77 / 111

# Overriding Method

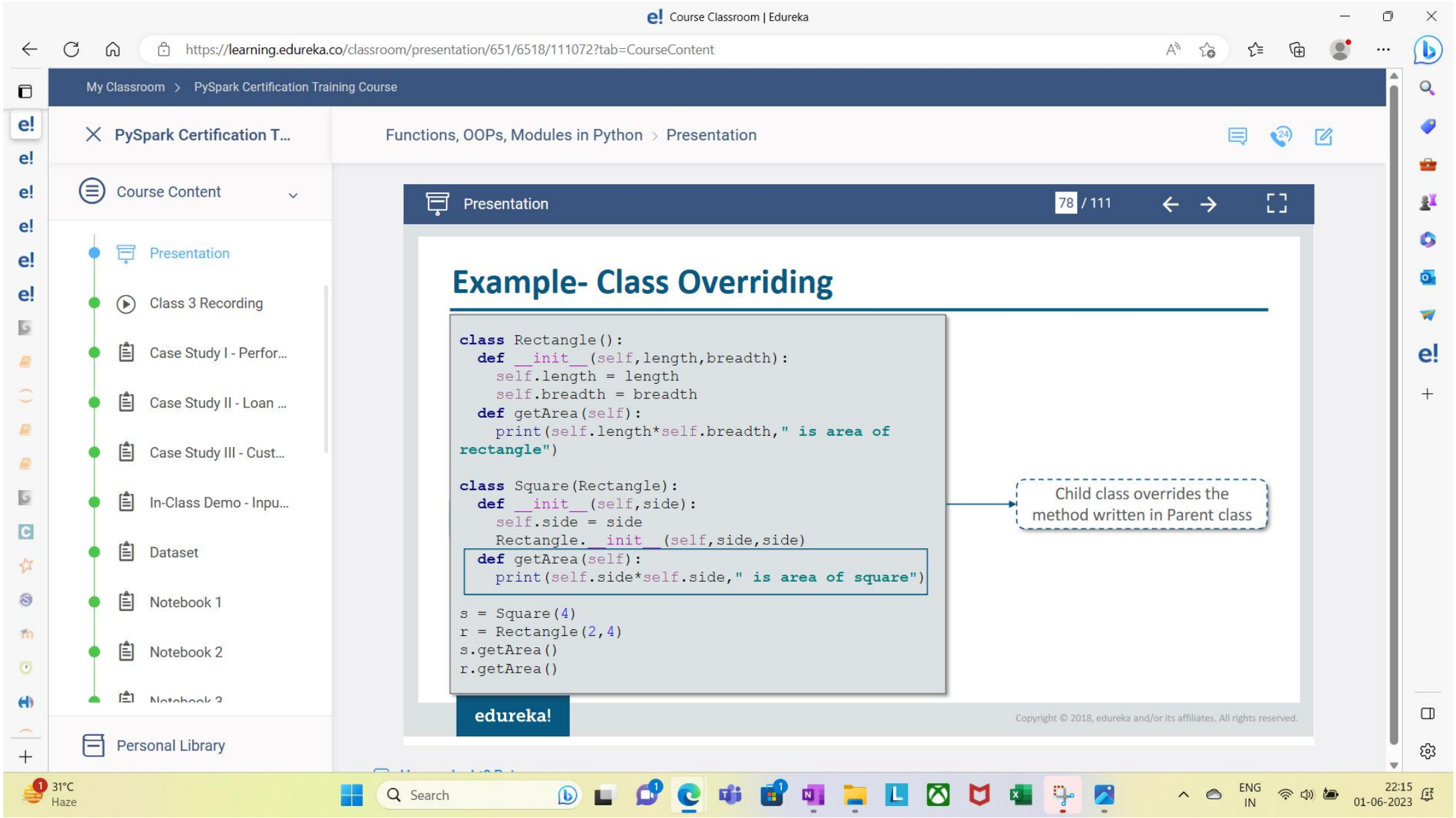
Parent class methods can always be overrided. One reason for overriding parent's methods is because you may want special or different functionality in your subclass

```
class Parent: # define parent class
  def myMethod(self):
    print("Calling parent method")
class Child(Parent): # define child class
  def myMethod(self):
    print("Calling child method")

c = Child()  # instance of child
c.myMethod() # child calls overridden method
```

Child class overrides the method written in Parent class

```
Calling child method
```

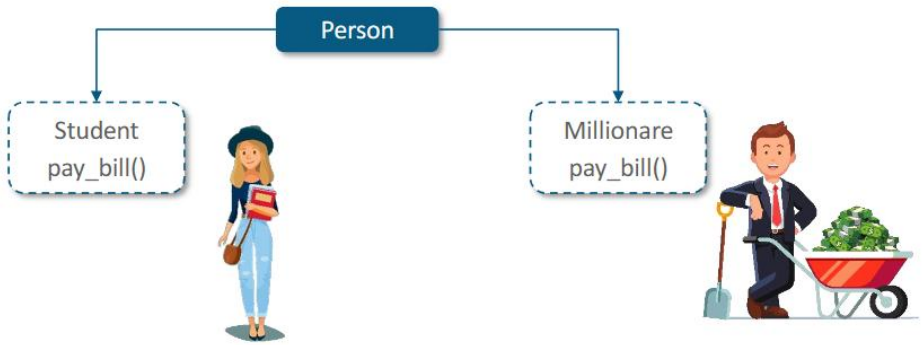
edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
78 / 111
Example- Class Overriding
class Rectangle():
def __init__(self,length,breadth):
self.length = length
self.breadth = breadth
def getArea(self):
print(self.length*self.breadth," is area of rectangle")
class Square(Rectangle):
def __init__(self,side):
self.side = side
Rectangle.__init__(self,side,side)
def getArea(self):
print(self.side*self.side," is area of square")
s = Square(4)
r = Rectangle(2,4)
s.getArea()
r.getArea()
Child class overrides the method written in Parent class
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C Haze
Search
ENG IN
22:15 01-06-2023

Presentation 79 / 111

# Polymorphism

**Polymorphism** is the ability to leverage the same interface for different underlying forms such as data types or classes




edureka!

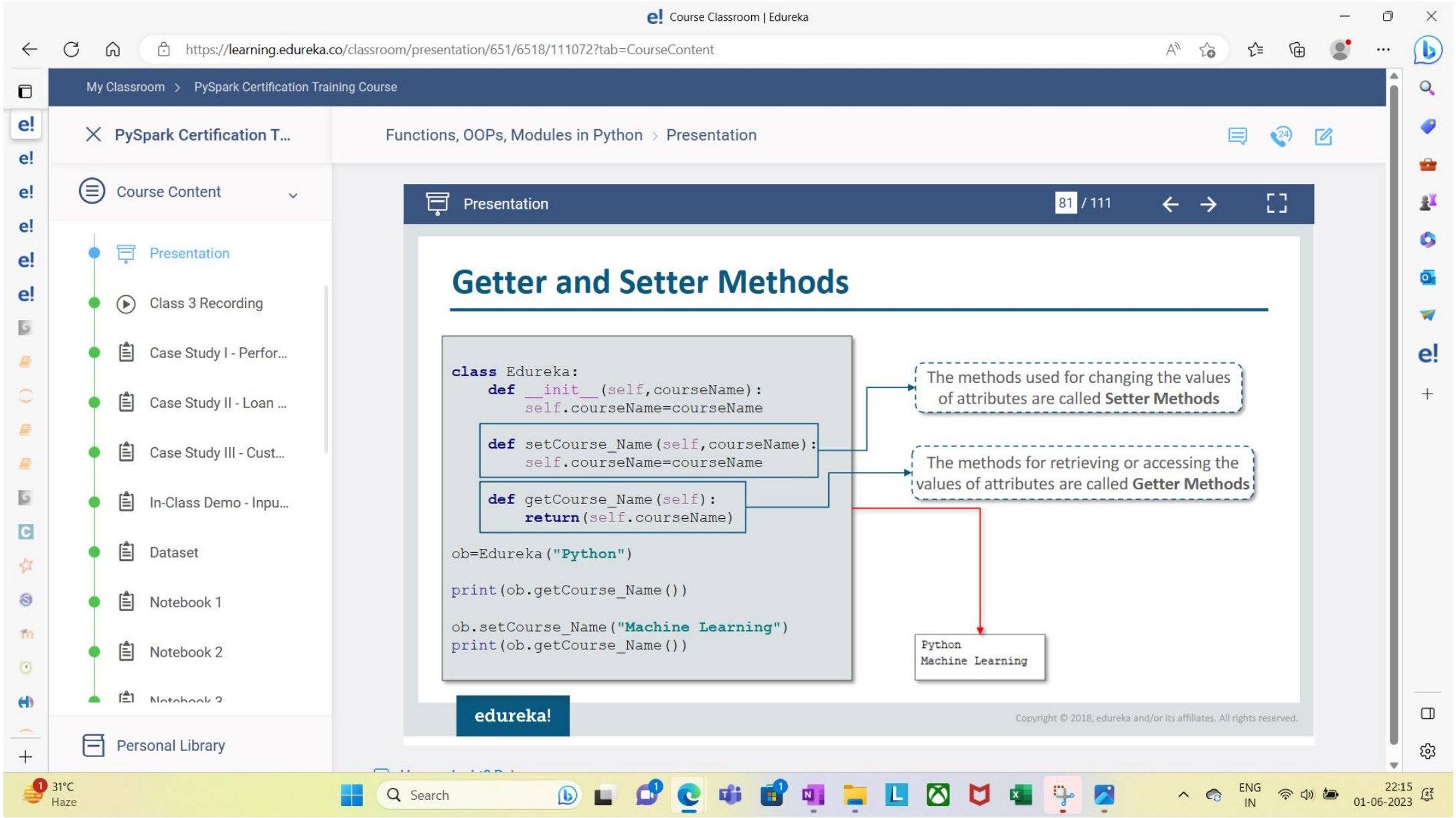
# Polymorphism (Contd.)

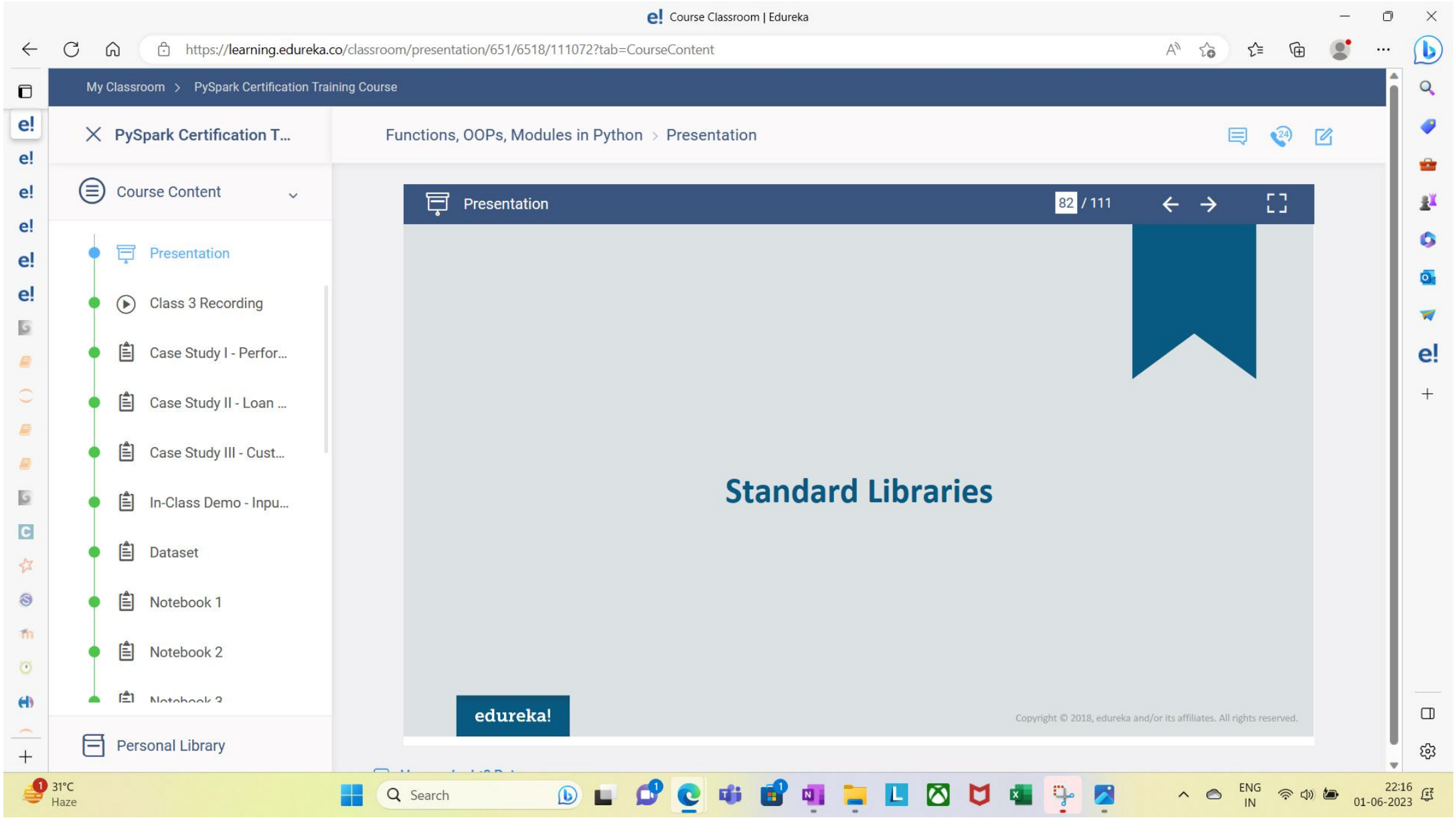
```
class Animal:
    def __init(self,name):
        self.name=name

    def talk(self):
        pass

class Cat(Animal):
    def talk(selfs):
        print("Meow")

class Dog(Animal):
    def talk(self):
        print("Woof")

c=Cat()
c.talk()

d=Dog()
d.talk()
```

Polymorphism is an important feature of class definition in Python that is utilized when you have commonly named methods across classes or subclasses

```
Meow
Woof
```

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
81 / 111
Getter and Setter Methods
class Edureka:
def __init__(self,courseName):
self.courseName=courseName
def setCourse_Name(self,courseName):
self.courseName=courseName
def getCourse_Name(self):
return(self.courseName)
ob=Edureka("Python")
print(ob.getCourse_Name())
ob.setCourse_Name("Machine Learning")
print(ob.getCourse_Name())
The methods used for changing the values of attributes are called Setter Methods
The methods for retrieving or accessing the values of attributes are called Getter Methods
Python
Machine Learning
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:15
01-06-2023


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
82 / 111
Standard Libraries
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:16
01-06-2023

Presentation 83 / 111

# Standard Libraries

Standard Library is a collection of tools that come with Python. The standard library includes the following:

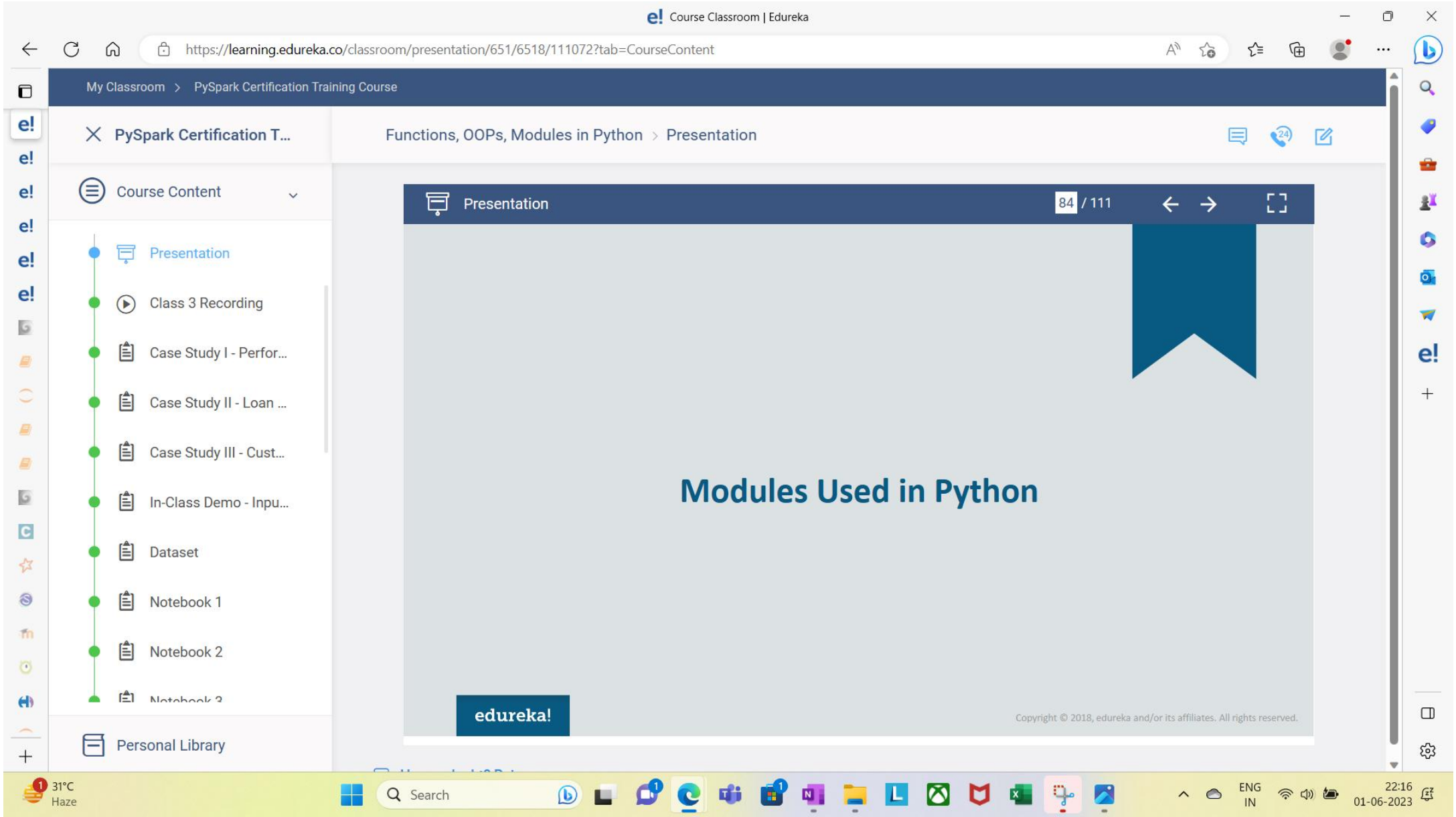
Standard Libraries

01 Built-in Functions

02 Modules

03 Packages

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
84 / 111
Modules Used in Python
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:16
01-06-2023

Presentation 85 / 111

# Modules

A Module allows you to logically organize the code

Python Modules are nothing but python files with .py extension

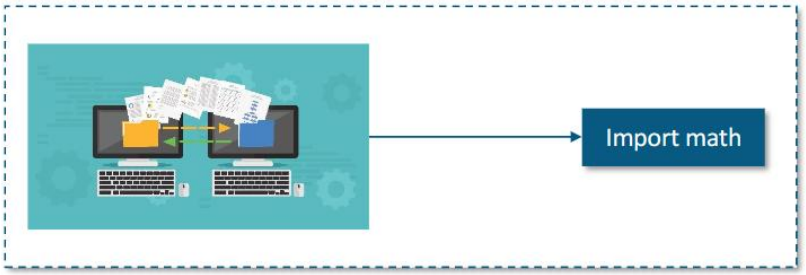
A module is a file containing Python definitions and statements

edureka!

Presentation 86 / 111

# Import Statement

Use “import” to load a module in Python. When the interpreter encounters an import statement, it imports the module, if the module is present in the search path




edureka!

Presentation 87 / 111

# Dir Function

A dir is a built-in function which returns the list of strings containing the name defined by the module. It gives the list of variables, functions defined in that module

```
import math
print(dir(math))
```

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmo
d', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log1
0', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

edureka!

Presentation 88 / 111

## From Import Statement

*From Import* statement allows you to import specific attribute from a specific module into the current namespace

```
from math import sqrt
```

This statement does not import the entire module, it just imports required files in the module

edureka!

Presentation 89 / 111

## From Import * Statement

The from..import * allows you to import all the attributes from the required module
This provides an easy way to import all the items from a module into the current namespace

```
from math import *
```

edureka!

Presentation 90 / 111

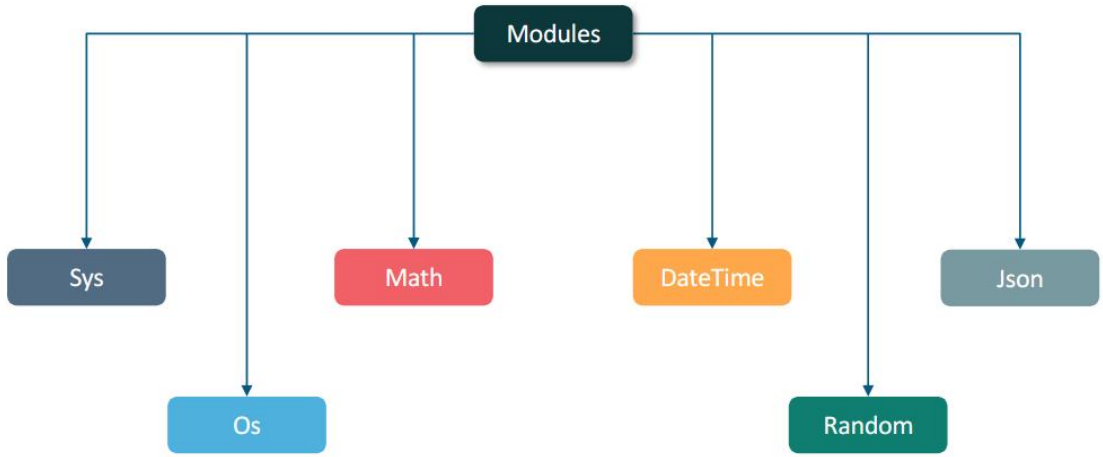
# The Reload Function

When the module is imported into a script, the code in the top-level portion of a module is executed only once. Therefore, if you want to re-execute the top-level code in a module, you can use the *reload* function. The reload function imports a previously imported module again


edureka!

Presentation 91 / 111

## Important Modules Used in Python




edureka!

# Sys Module

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

```
import sys
print(sys.argv)
print(sys.exit())
```

Stores any command line argument passed, when you start Python; also includes the name of the program you are running

Informs the Python interpreter to quit

```
['C:/Users/adhyapakss/PycharmProjects/DS_mod3/sysm.py']
```

edureka!

# Os Module

The **os** Module includes code that lets Python work with your operating system and run some operating system commands

```
import os

print(os.getcwd())

print(os.chdir('d:/Python'))

print(os.mkdir('d:/Python/Py'))

print(os.rmdir('d:/Python/Py'))
```

creates Directory in given path

deletes Directory from given path

edureka!

# Random Module

This module implements pseudo-random number generators for various distributions

```
import random
num = random.randrange(100)
print(num)
ran=random.randrange(0,100,20)
print(ran)
inte=random.randint(0,30)
print(inte)
```

```
39
20
30
```

Generates a random integer within the given range

Return a randomly selected element from range(start, stop, step)

Return a random integer N such that a <= N <= b

edureka!

# Date Time Module

The datetime module includes tools for working with dates, times, and combinations.
The time module includes tools for working with times and dates in the recent past to the near future

```
import datetime
print(datetime.datetime.today())

now=datetime.datetime.today()
other=datetime.datetime(1995,3,12,22,10)
print(now-other)
datetime.timedelta(18901,55547,421000)
print(now-other)
```

```
2018-01-19 10:07:37.794125
8348 days, 11:57:37.794466
8348 days, 11:57:37.794466
```

edureka!

Presentation 96 / 111

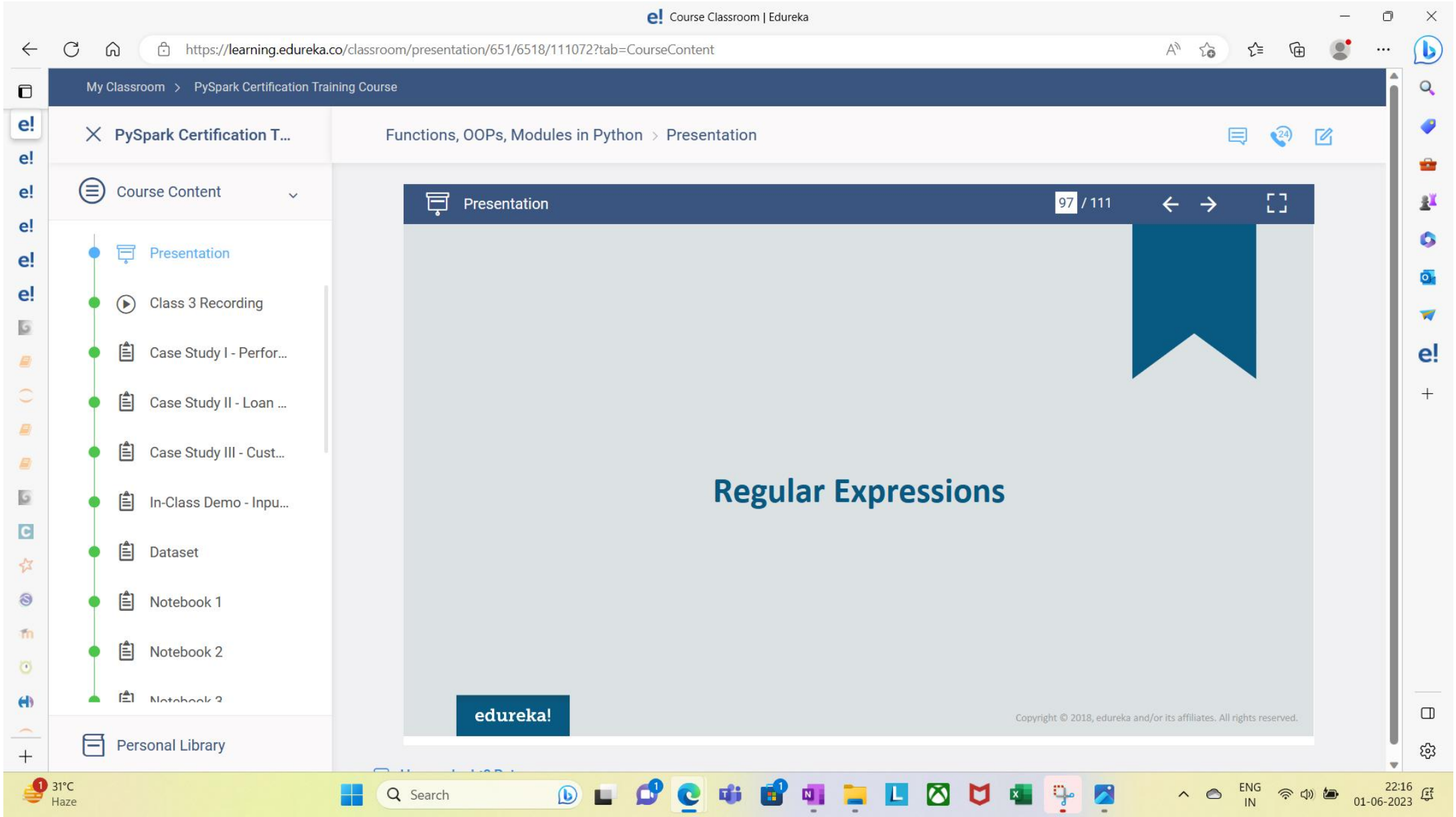
# Json Module

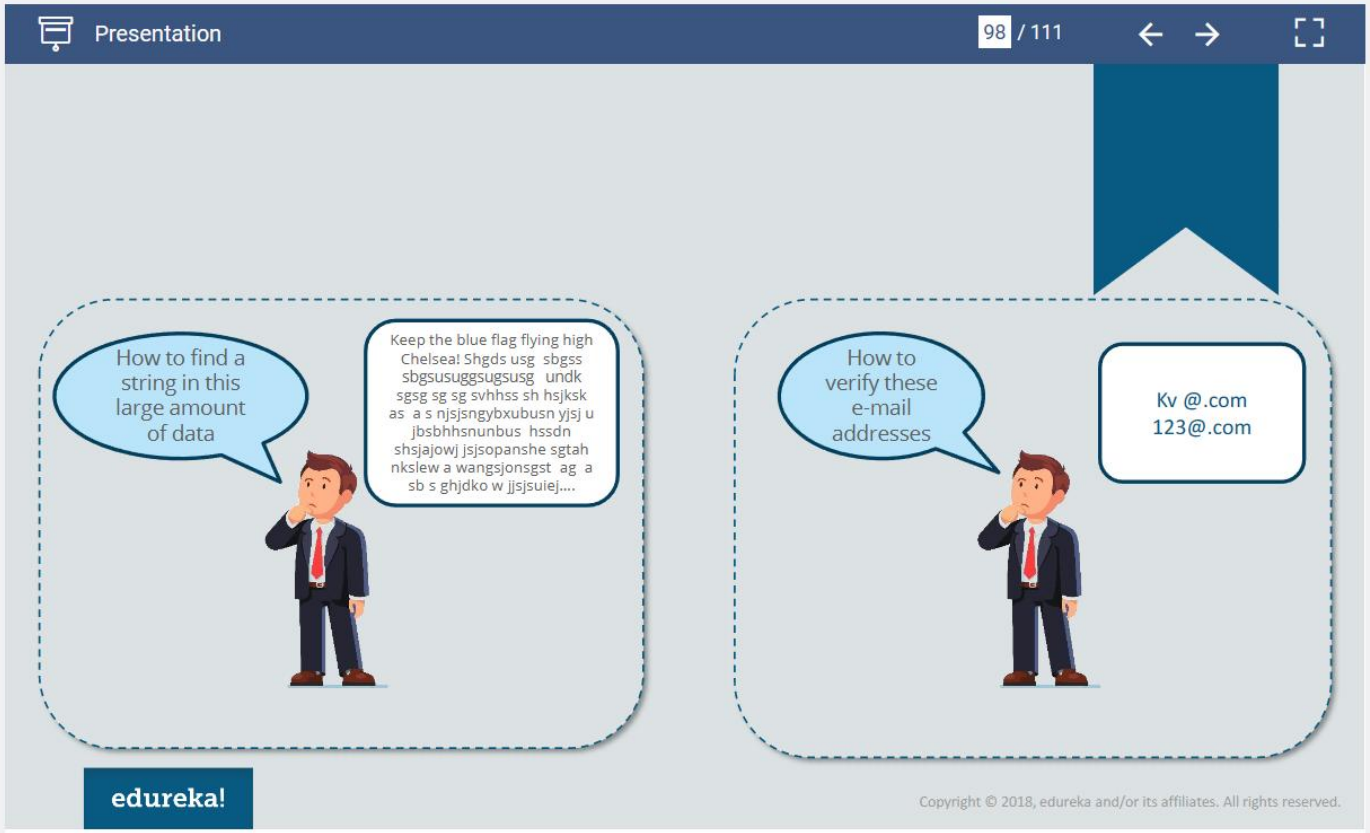
The json module provides an easy way to encode and decode data in JSON

```
import json
data={"name":"Ace","shares":100,"price":540}
json_str=json.dumps(data)
print(json_str)
```

```
{"name": "Ace", "shares": 100, "price": 540}
```

edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
97 / 111
Regular Expressions
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:16
01-06-2023



Presentation 99 / 111

# What are Regular Expressions?

A Regular Expression is a special text string for describing a search pattern

Can you identify the pattern to get the Name and Age ?

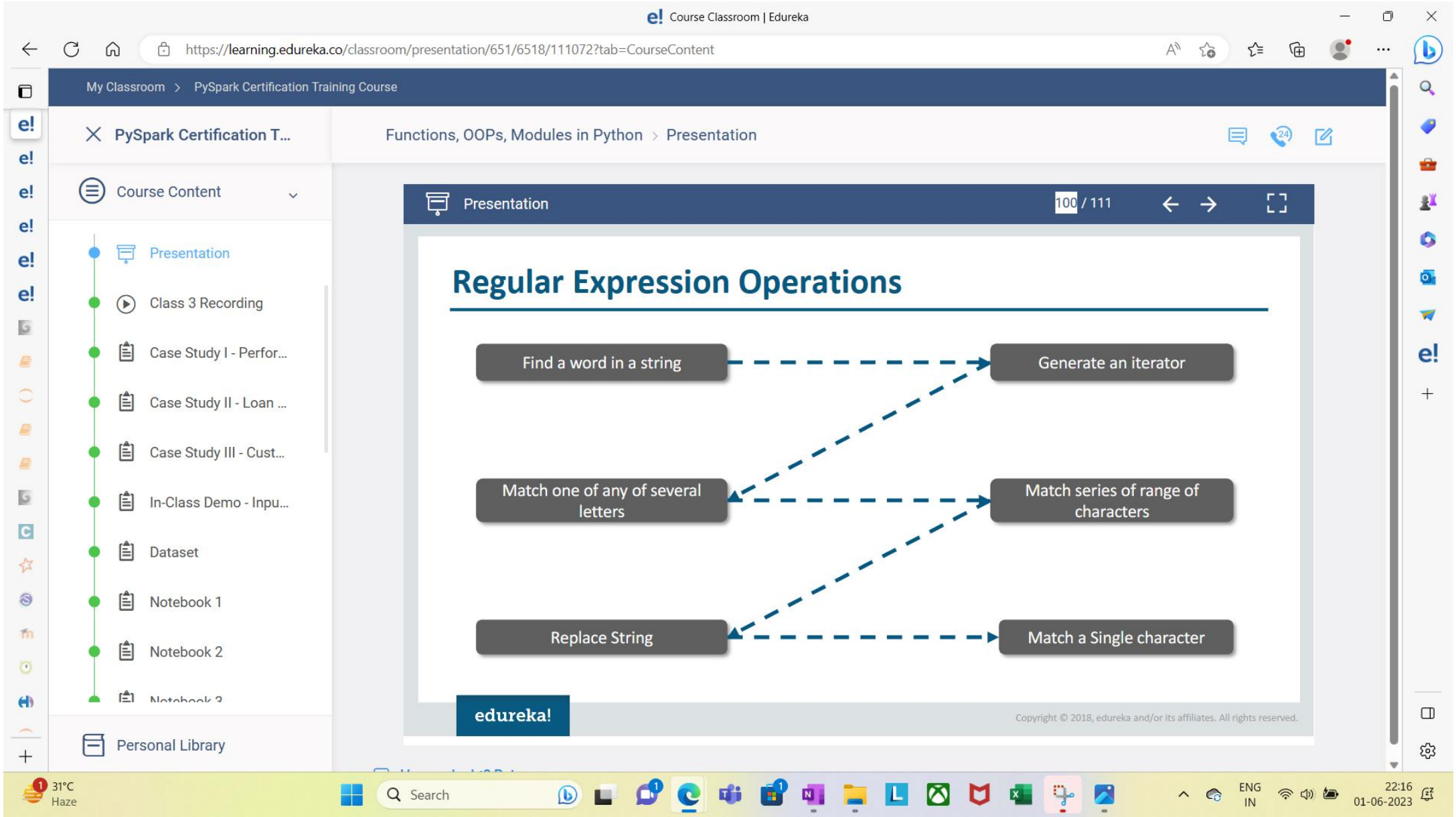
NameAge = '''
Janice is 22 and Theon is 33
Gabriel is 44 and Joey is 21

{'Janice': '22', 'Theon': '33', 'Gabriel': '44', 'Joey': '21'}

First letter of all the Names is an uppercase letter and Age is represented by numbers

NameAge = '''
Janice is 22 and Theon is 33
Gabriel is 44 and Joey is 21

edureka!


e! Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
100 / 111
Regular Expression Operations
Find a word in a string
Generate an iterator
Match one of any of several letters
Match series of range of characters
Replace String
Match a Single character
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:16
01-06-2023

Presentation 101 / 111

# Regular Expression

The replace method of strings is used to replace all occurrences of one string with another, and the index method is used to find the first occurrence of a substring in a string. But sometimes you need to do a more a sophisticated search or replace then Regular Expression is used

```
import re
print(re.sub(r'[ad]','*','abcde abcedf abcdef'))
print(re.sub(r'abc','*','abcdef abcdef'))
```

Replaces all occurrences of abc with *

We can use square brackets to indicate that we only want to match certain letters. It replaces every a and d with asterisks

```
*bc*e *bce*f *bc*ef
*def *def
```

edureka!

# Regular Expression - Example

```
print(re.sub(r'[abc][123]','*','a1+b2+d4+e5'))

print(re.sub(r'A.B', '*', 'A2B AxB AxxB A$B'))

print(re.sub(r'AB+', '*', 'ABC ABBBBBBC AC'))

print(re.sub(r'AB{3,6}', '*', 'ABB ABBB ABBBB
ABBBBBBBBB'))

print(re.sub('^abc', '*', 'abcdefgabc'))

print(re.sub('abc$', '*', 'abcdefgabc'))

print(re.sub(r'AB\+', '*', 'AB+C'))

print(re.sub(r'\d', '*', '3 + 14 = 17'))

print(re.sub(r'\D', '*', '3 + 14 = 17'))
```

```
*+*+d4+e5
* * AxxB *
*C *C AC
ABB * * *BBB
*defgabc
abcdefg*
*C
* + ** = **
3***14***17
```

edureka!

Presentation 103 / 111

## Search and Match Function

```
# search and match
print(re.search('ab','Here is an absolute string'))
print(re.match('ab','Here is an absolute string'))
```

Search looks for a pattern anywhere in a string

Match looks for a match starting at the beginning of the string

edureka!

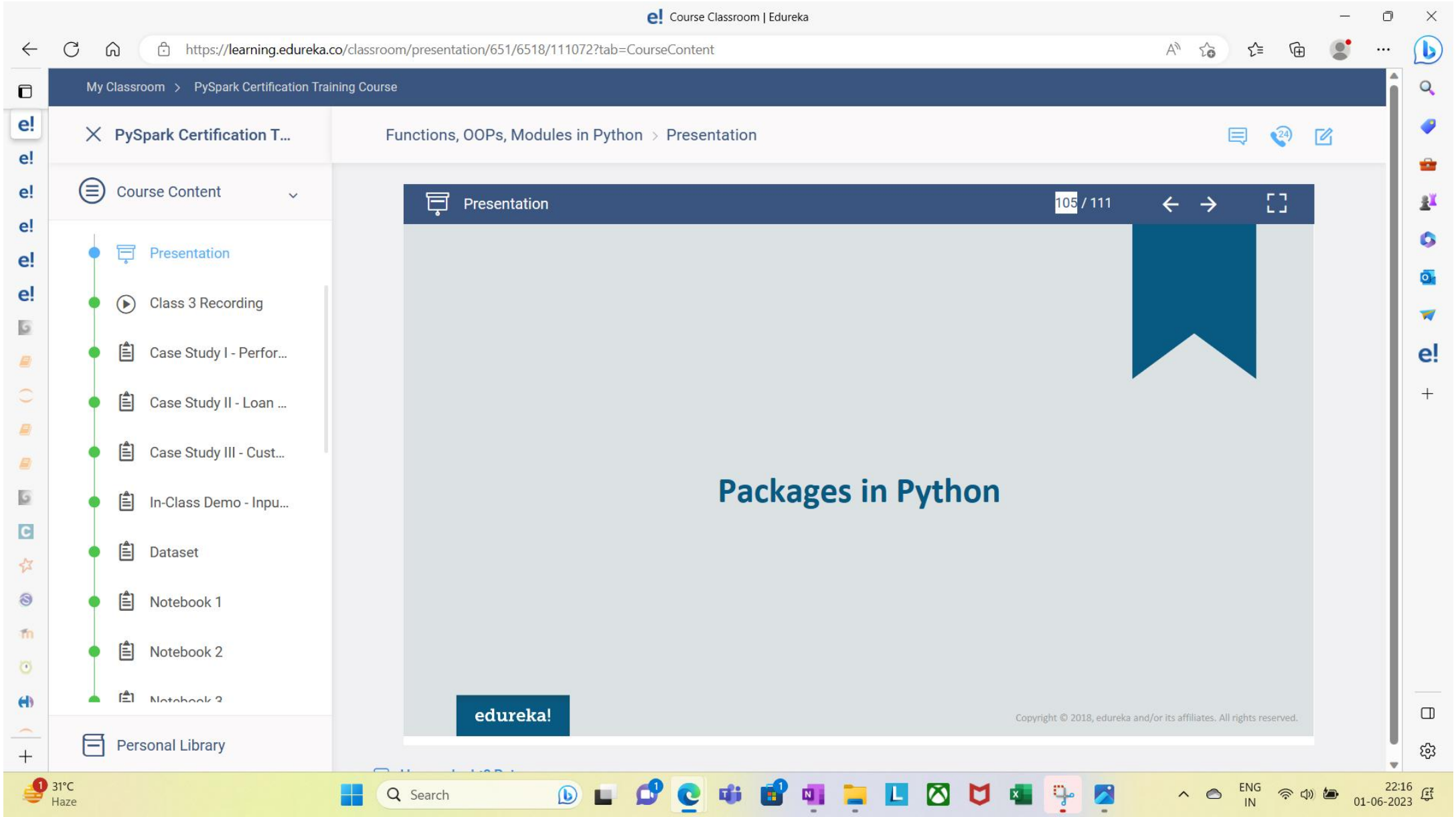
Presentation 104 / 111

# Regular Expression Application

E-mail:
Saurabh@gmail.com
Reyshma @ com
Kv @.com
123@.com

Phone Number
444-122-1234
123-122-78999
111-123-23
67-7890-2019

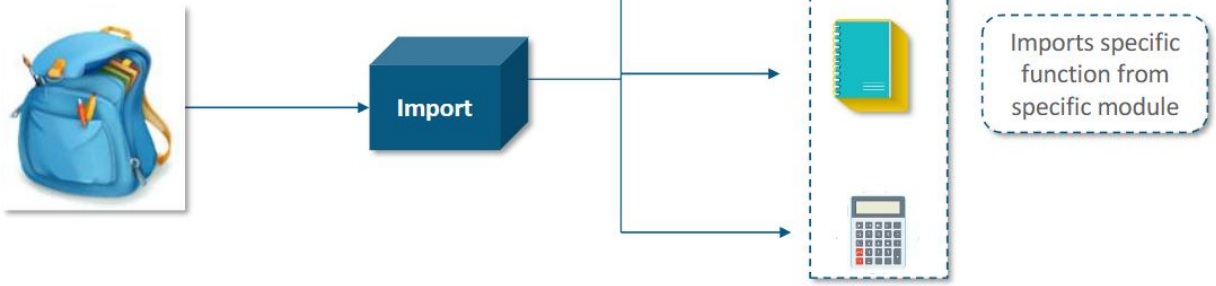
edureka!


Course Classroom | Edureka
https://learning.edureka.co/classroom/presentation/651/6518/111072?tab=CourseContent
My Classroom > PySpark Certification Training Course
PySpark Certification T...
Functions, OOPs, Modules in Python > Presentation
Course Content
Presentation
Class 3 Recording
Case Study I - Perfor...
Case Study II - Loan ...
Case Study III - Cust...
In-Class Demo - Inpu...
Dataset
Notebook 1
Notebook 2
Personal Library
Presentation
105 / 111
Packages in Python
edureka!
Copyright © 2018, edureka and/or its affiliates. All rights reserved.
31°C
Haze
Search
ENG
IN
22:16
01-06-2023

Presentation 106 / 111

# Packages in Python

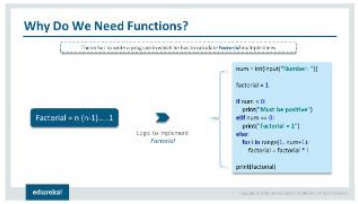
A Package is a collection of Python modules. Python packages allow you to break down large systems and organize their modules in a consistent way that you and other people can use and reuse efficiently



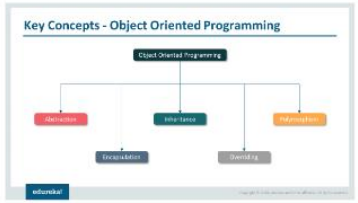

edureka!

Presentation 107 / 111

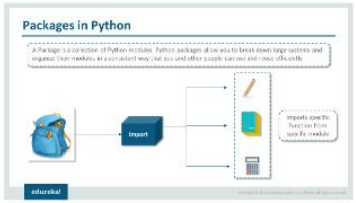
# Summary










edureka!