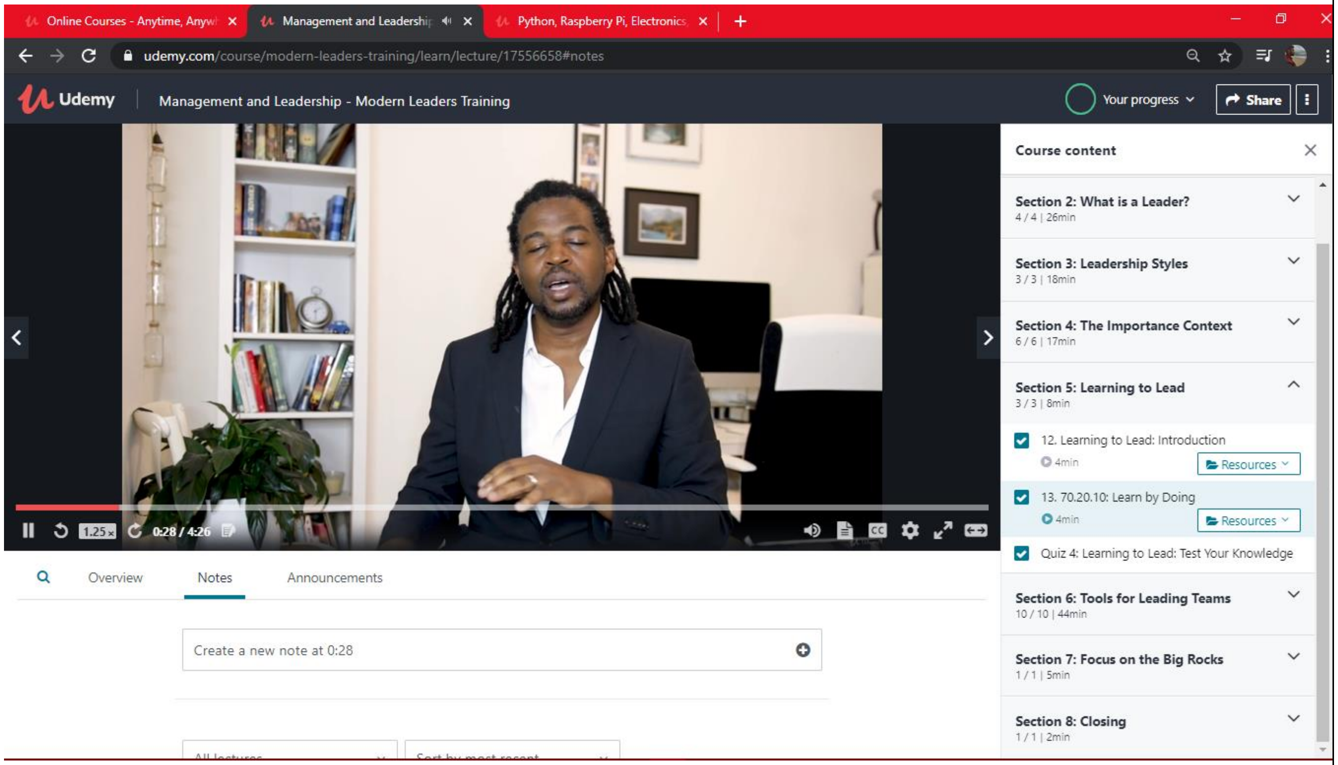
DAILY ASSESSMENT FORMAT

Date:	12/06/2020	Name:	Akshay
Course:	Management and Learning - Modern Leaders Training	USN:	4AL17EC008
Topic:	Learning to lead	Semester & Section:	6TH & A
Github Repository:	Akshay-Online-Course		

FORENOON SESSION DETAILS



Online Courses - Anytime, Anywh
Management and Leadership
Python, Raspberry Pi, Electronics,
udemy.com/course/modern-leaders-training/learn/lecture/17556658#notes
Udemy
Management and Leadership - Modern Leaders Training
Your progress
Share
Course content
Section 2: What is a Leader?
4 / 4 | 26min
Section 3: Leadership Styles
3 / 3 | 18min
Section 4: The Importance Context
6 / 6 | 17min
Section 5: Learning to Lead
3 / 3 | 8min
12. Learning to Lead: Introduction
4min
Resources
13. 70.20.10: Learn by Doing
4min
Resources
Quiz 4: Learning to Lead: Test Your Knowledge
Section 6: Tools for Leading Teams
10 / 10 | 44min
Section 7: Focus on the Big Rocks
1 / 1 | 5min
Section 8: Closing
1 / 1 | 2min
1.25x
0:28 / 4:26
Overview
Notes
Announcements
Create a new note at 0:28

Report:
There is, in many senses, a degree of formality about the term ‘leader’. A whole organisation accepts one person as its ‘leader’ and, by virtue of that badge of rank, the leader has formal authority and power. This can lead new leaders to infer that they need to be able to do everything, and do it right every time. However, taken to extremes, there are some very real dangers in this view. For example, if an organisation has only one leader, that person can come to be seen as the source of all

ideas and the maker of all decisions. The rest of the organisation must, therefore, be followers, who take no initiative and make no decisions. These people are also free of responsibility for the outcomes of their actions. This presents a big problem for the organisation as a whole and followers as individuals:

• There is no synergy, in that the whole never becomes greater than the sum of its parts, because its parts do not work together;

• People do not take the initiative;

• There is little incentive for anyone to do anything "good" except follow orders; and

• There is little reason for people to not do "bad" things so long as they are within the letter of the law.

This is not going to make for a pleasant workplace, or an inspiring one. It is also an issue for the leader. As the only one leading, he or she may not be less than perfect at any time, and not just right every time, but seen to be right every time. This is an impossible state to achieve: we are human and therefore fallible. To err is human, as the saying goes. What's more, the longer such a formal leader is in post, the greater the gap becomes between the leader and their followers. What can happen is that the leader becomes less tolerant of independent thought, and the followers become less capable of it. At this point, if it is to survive 'After the Leader', an organisation has to look seriously at succession planning. In such an organisation, succession planning must be the responsibility of the leader, otherwise it is likely to be interpreted as mutiny. That, in itself, is likely to be a problem, as the leader will often look for someone who will 'carry on their legacy', rather than someone to take the organisation forward in a new direction, responding to current issues and needs. History, unfortunately, shows time and again that:

• Families with a commanding father or mother are often at least slightly dysfunctional;

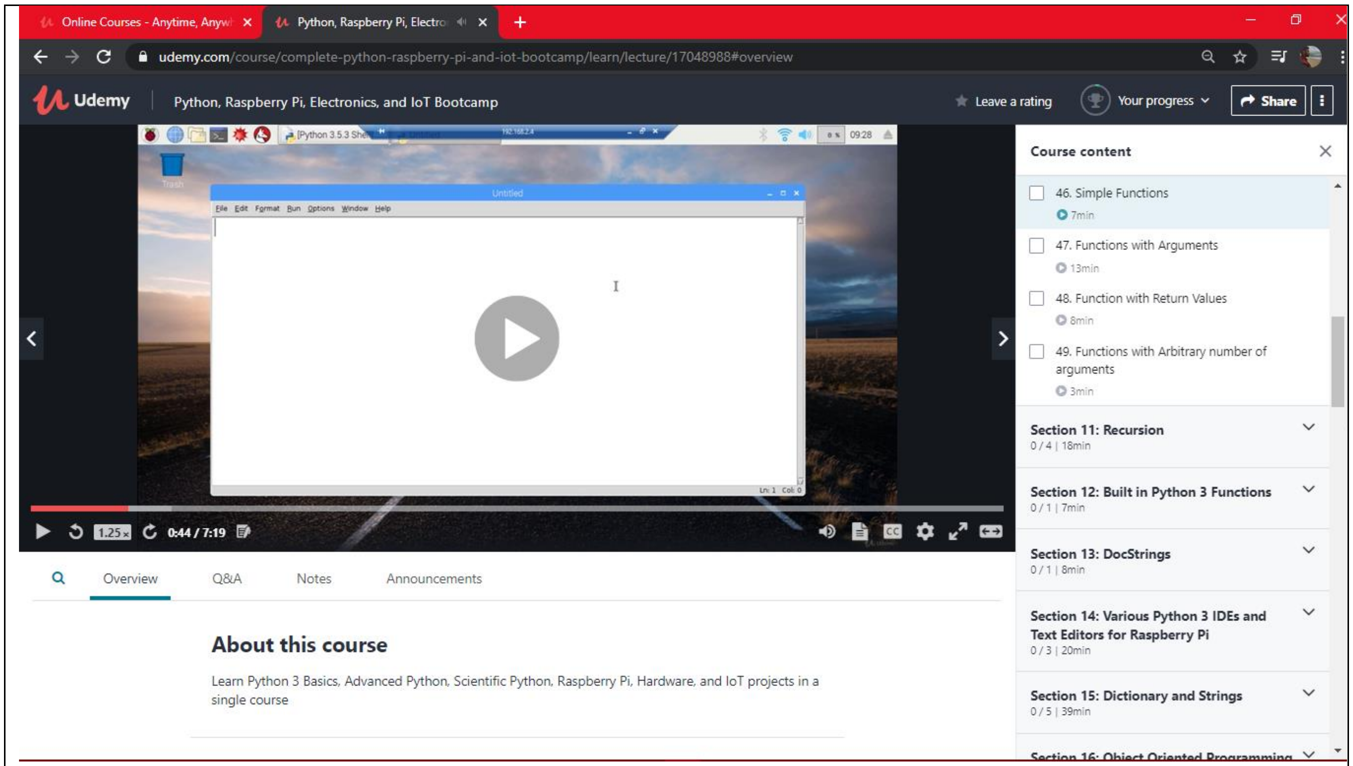
• Nations with a cult of personality around a single "great helmsman" tend to suffer in the long run; and

• Companies ruled by the iron hand of their founder are lost when the founder dies or is shown to have had feet of clay.

This kind of leadership simply does not work. The lesson is obvious: anyone stepping into a leadership position needs a different model of leadership.

Date:	12/06/2020	Name: Akshay
Course:	Complete-Python-raspberry-pi-and-IOT-bootcamp	USN: 4AL17EC008
Topic:	Function	Semester & Section: 6TH A SEC

AFTERNOON SESSION DETAILS
Image of session


Online Courses - Anytime, Anywh
Python, Raspberry Pi, Electro
udemy.com/course/complete-python-raspberry-pi-and-iot-bootcamp/learn/lecture/17048988#overview
Udemy
Python, Raspberry Pi, Electronics, and IoT Bootcamp
Leave a rating
Your progress
Share
Untitled
File Edit Format Run Options Window Help
1.25x
0:44 / 7:19
Overview
Q&A
Notes
Announcements

About this course

Learn Python 3 Basics, Advanced Python, Scientific Python, Raspberry Pi, Hardware, and IoT projects in a single course

Course content

- 46. Simple Functions — 7min
- 47. Functions with Arguments — 13min
- 48. Function with Return Values — 8min
- 49. Functions with Arbitrary number of arguments — 3min

Section 11: Recursion
0 / 4 | 18min

Section 12: Built in Python 3 Functions
0 / 1 | 7min

Section 13: DocStrings
0 / 1 | 8min

Section 14: Various Python 3 IDEs and Text Editors for Raspberry Pi
0 / 3 | 20min

Section 15: Dictionary and Strings
0 / 5 | 39min

Section 16: Object Oriented Programming

Report – Report can be typed or hand written for up to two pages.

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

Defining a Function

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword def followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax

```
def functionname( parameters ):
  "function_docstring"
  function_suite
  return [expression]
```

By default, parameters have a positional behavior and you need to inform them in the same order that they were defined.

Example

The following function takes a string as input parameter and prints it on standard screen.

```
def printme( str ):
   "This prints a passed string into this function"
   print (str)
   return
```

Calling a Function

Defining a function gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is an example to call the printme() function −

```
#!/usr/bin/python3

# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print (str)
   return

# Now you can call printme function
printme("This is first call to the user defined function!")
printme("Again second call to the same function")
```

When the above code is executed, it produces the following result −

```
This is first call to the user defined function!
Again second call to the same function
```