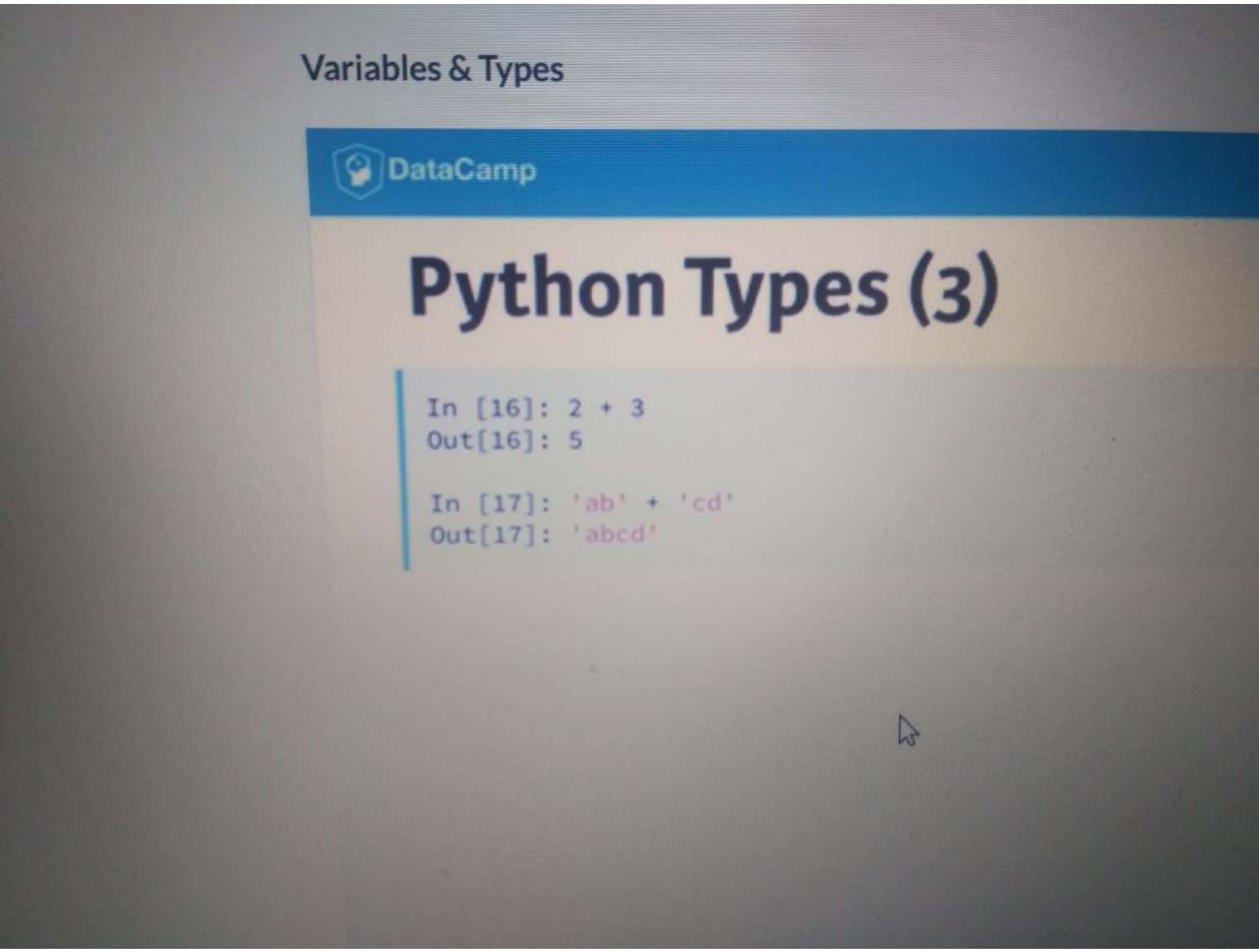


Variable Assignment | Python - Chromium

Variable Assignme ×

https://campus.datacamp.com/courses/intro-to-python-for-

DataCamp

Exercise

Variable Assignment

In Python, a variable allows you to refer to a value with a name. To create a variable use `=`, like this example:

```
x = 5
```

You can now use the name of this variable, `x`, instead of the actual value, `5`.

Remember, `=` in Python means *assignment*, it doesn't test equality!

Instructions

100 XP

- Create a variable `savings` with the value 100.
- Check out this variable by typing `print(savings)` in the script.

Take Hint (-30 XP)

uess the type | Python - Chromium

Guess the type | Pyt ×

https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapte

DataCamp

Exercise

Guess the type

To find out the type of a value or a variable that refers to that value, you can use the `type()` function. Suppose you've defined a variable `a`, but you forgot the type of this variable. To determine the type of `a`, simply execute:

```
type(a)
```

We already went ahead and created three variables: `a`, `b` and `c`. You can use the IPython shell on the right to discover their type. Which of the following options is correct?

Instructions

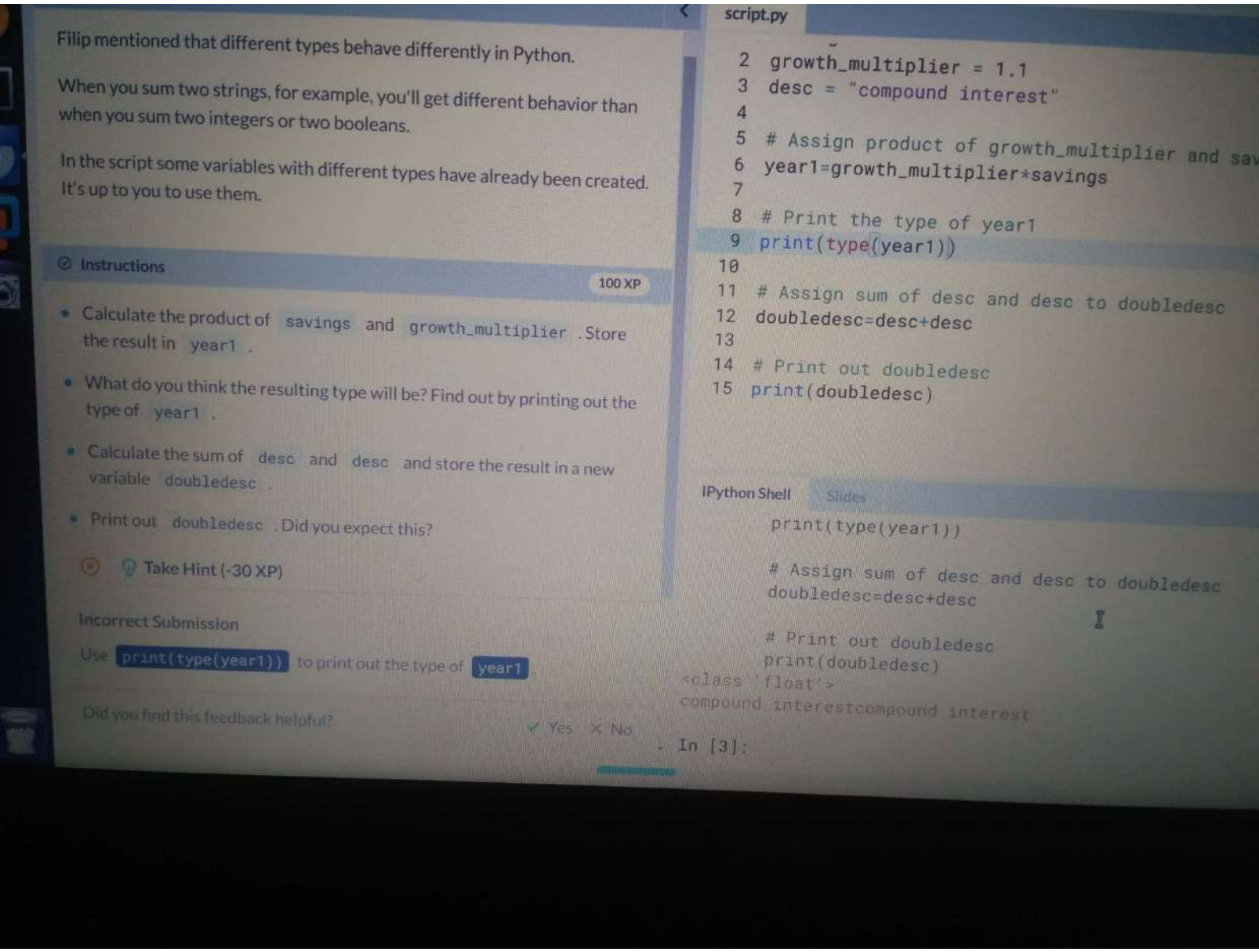
50 XP

Possible Answers

- `a` is of type `int`, `b` is of type `str`, `c` is of type `bool`
- `a` is of type `float`, `b` is of type `bool`, `c` is of type `str`
- `a` is of type `float`, `b` is of type `str`, `c` is of type `bool`
- `a` is of type `int`, `b` is of type `bool`, `c` is of type `str`

Take Hint (-15 XP)

Submit Answer



Type conversion | Python - Chromium

Type conversion | Py ×

https://campus.datacamp.com/courses/intro-to-python-for-d

DataCamp

Exercise

Type conversion

Using the `+` operator to paste together two strings can be very useful in building custom messages.

Suppose, for example, that you've calculated the return of your investment and want to summarize the results in a string. Assuming the floats `savings` and `result` are defined, you can try something like this:

```
print("I started with $" + savings + " and now have $" + re
```

This will not work, though, as you cannot simply sum strings and floats.

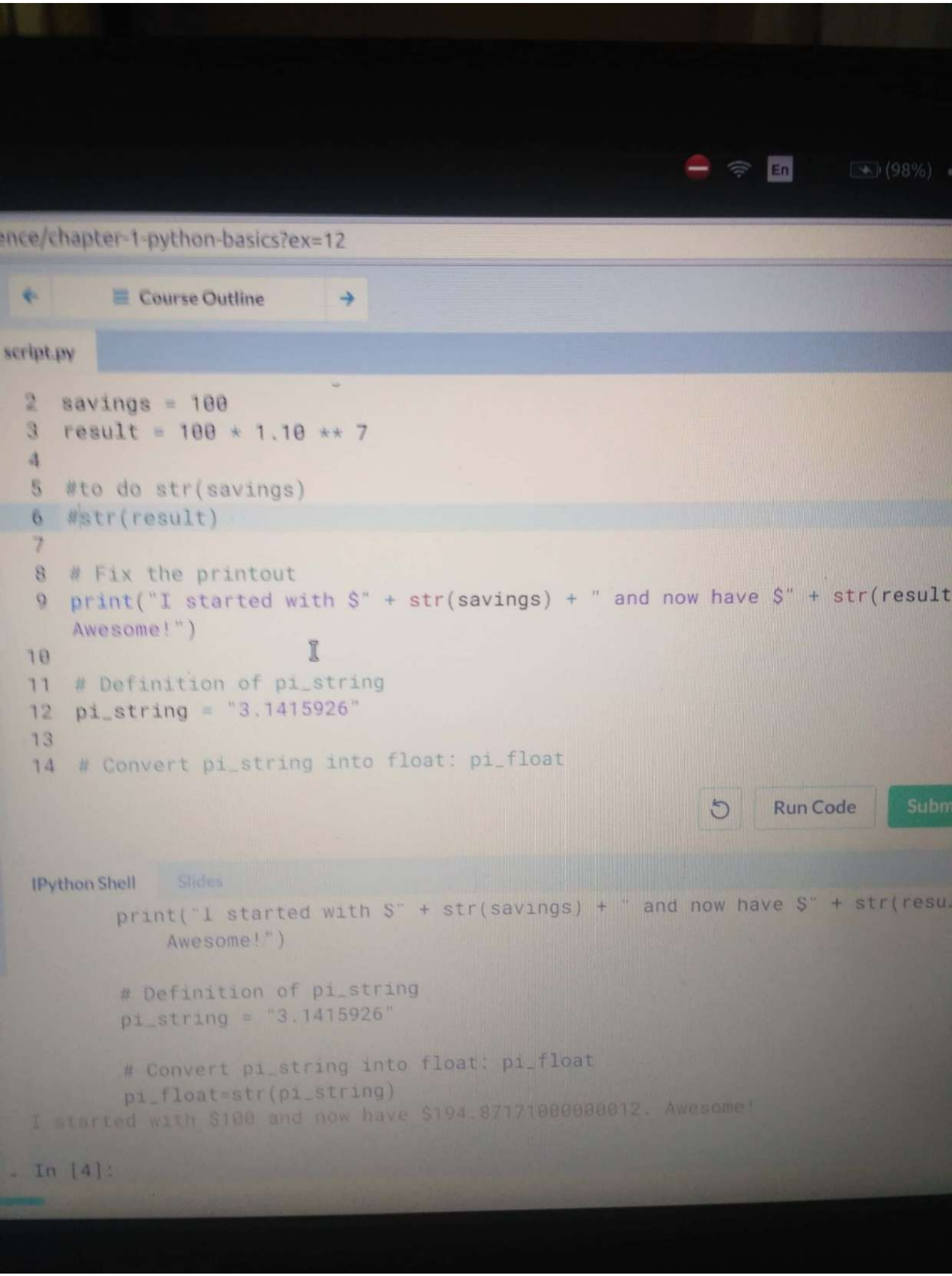
To fix the error, you'll need to explicitly convert the types of your variables. More specifically, you'll need `str()`, to convert a value into a string. `str(savings)`, for example, will convert the float `savings` to a string.

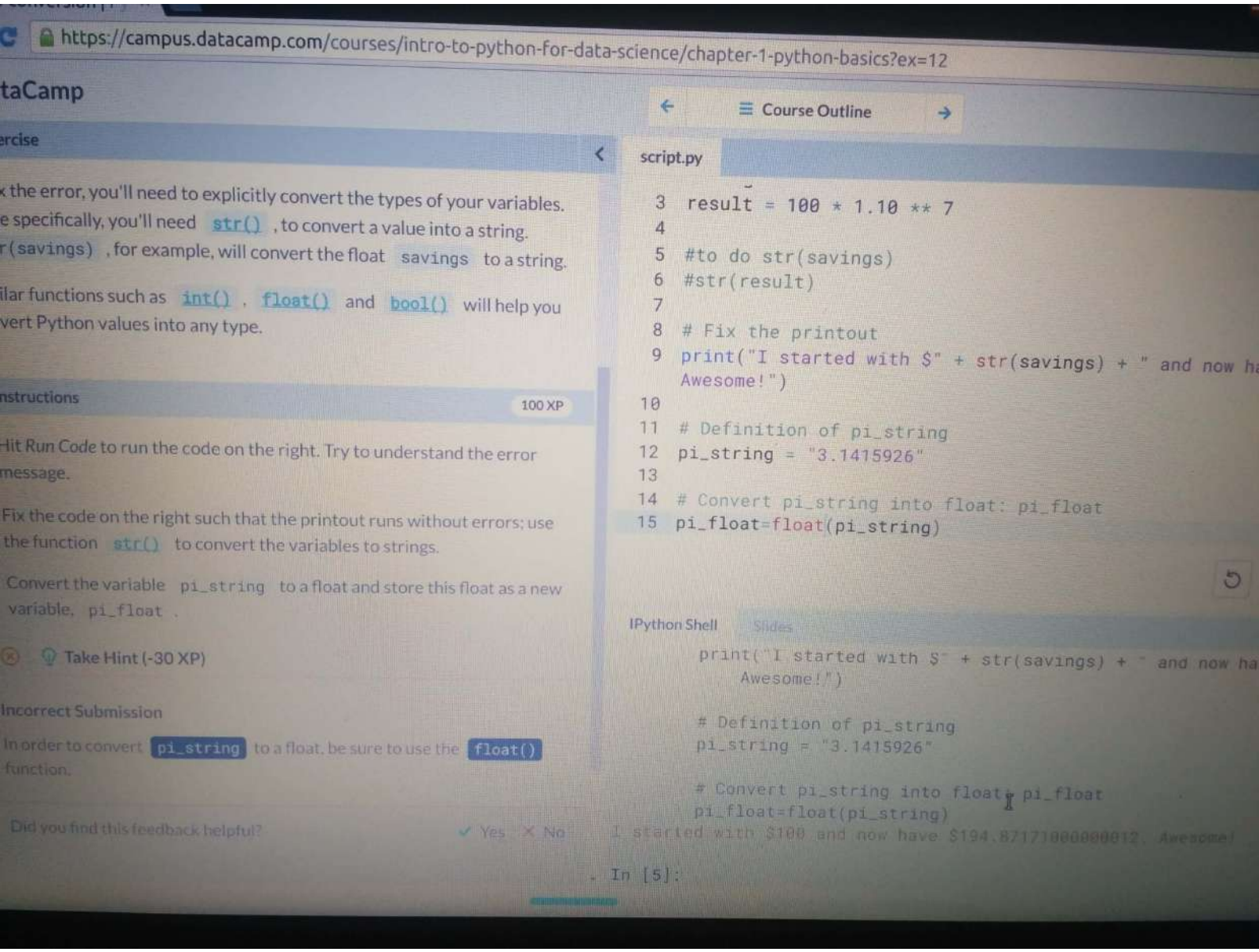
Similar functions such as `int()`, `float()` and `bool()` will help you convert Python values into any type.

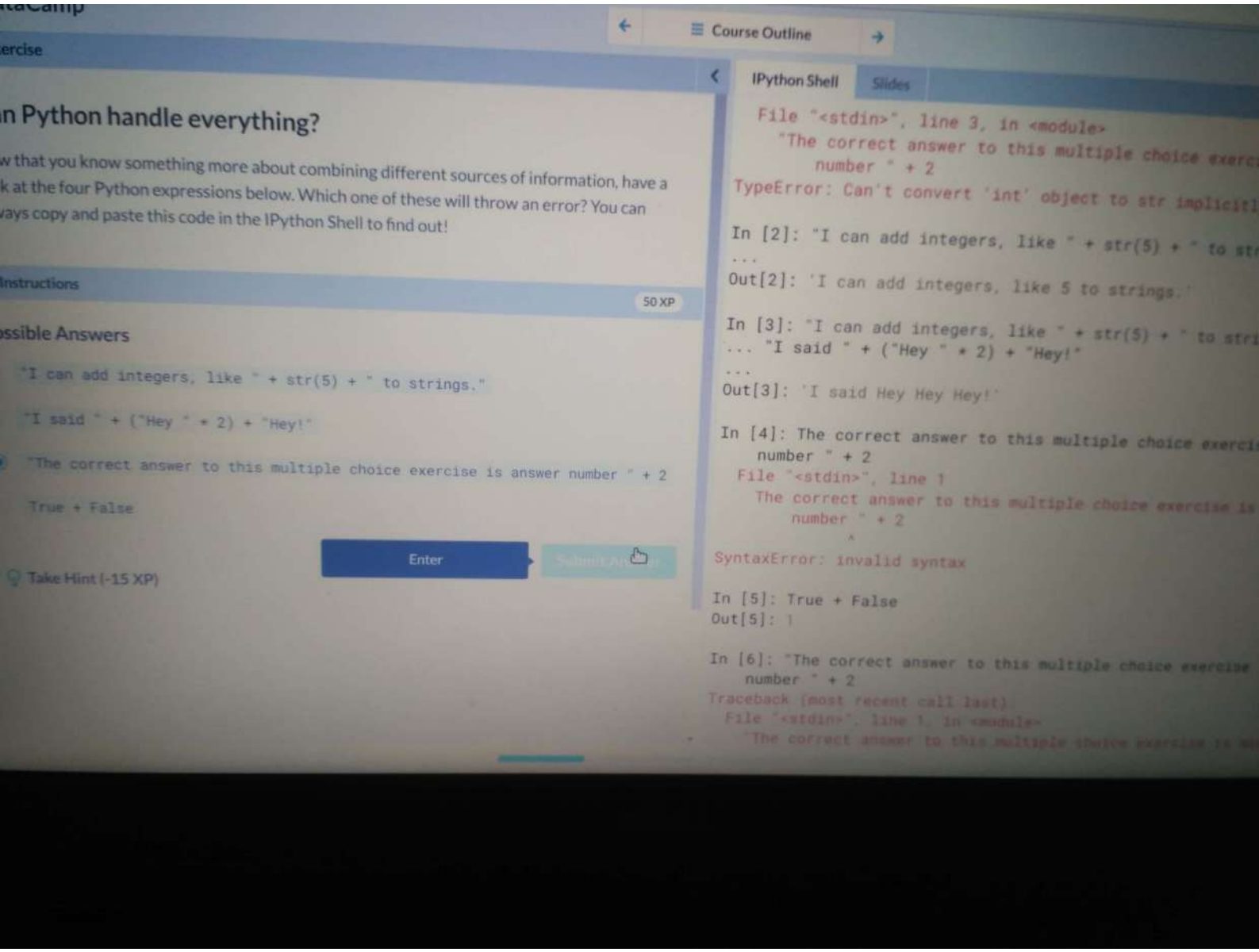
Instructions

100 XP

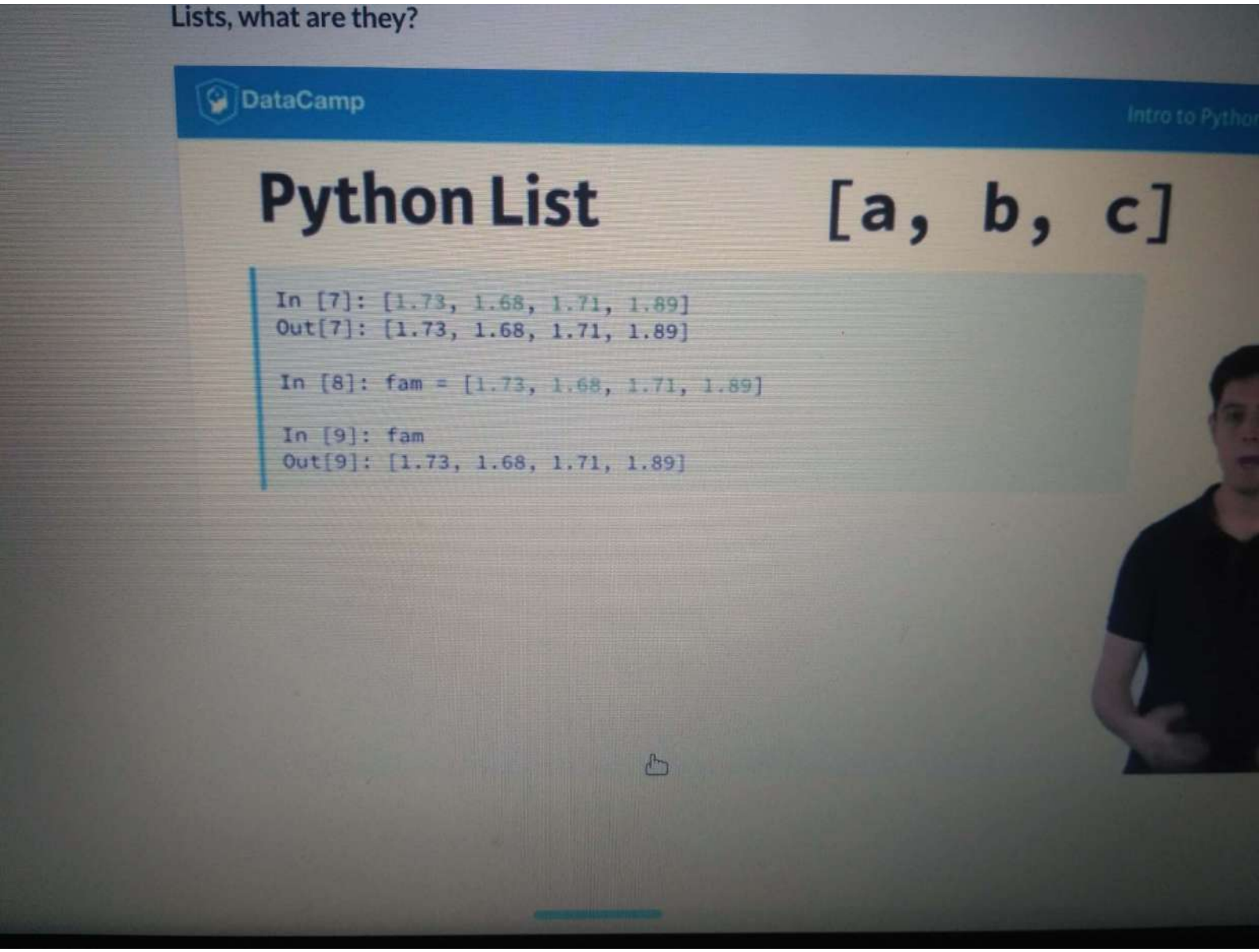
- Hit *Run Code* to run the code on the right. Try to understand the error message.



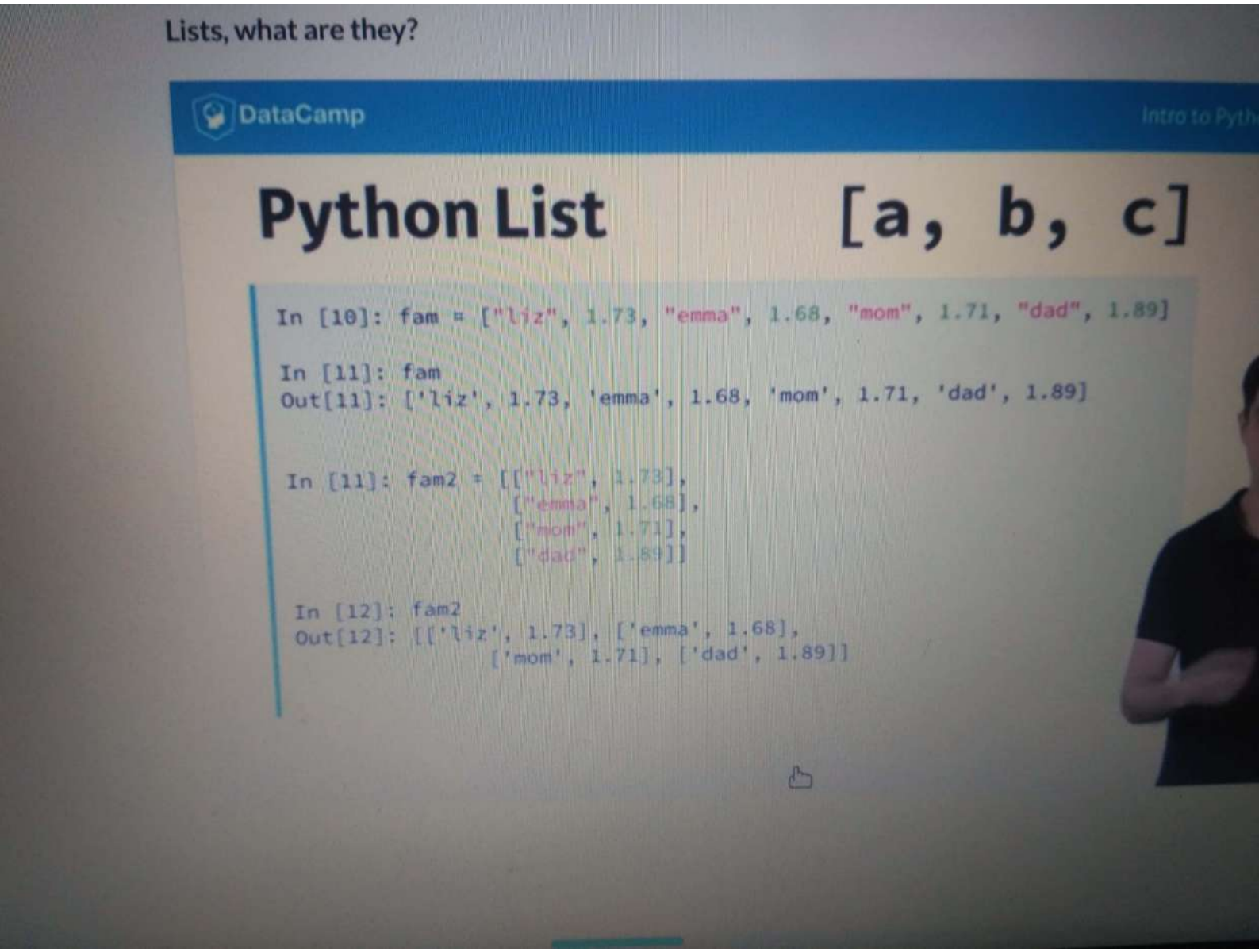


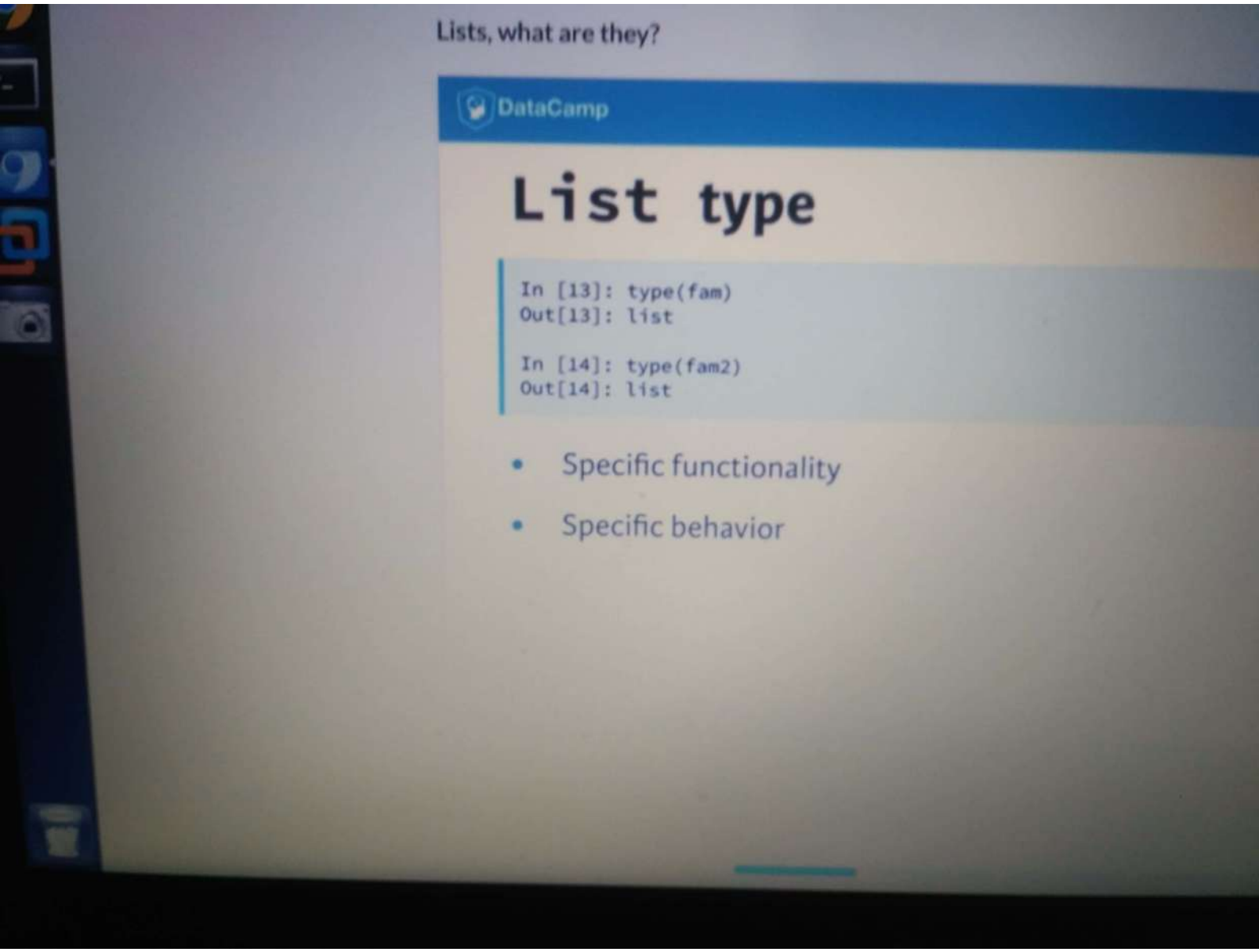


Lists, what are they?



Lists, what are they?





Create a list | Python - Chromium

Create a list | Pytho ×

https://campus.datacamp.com/courses/intro-to-python-for

DataCamp

Exercise

Create a list

As opposed to `int`, `bool` etc., a list is a **compound data type**; you can group values together:

```
a = "is"
b = "nice"
my_list = ["my", "list", a, b]
```

After measuring the height of your family, you decide to collect some information on the house you're living in. The areas of the different parts of your house are stored in separate variables for now, as shown in the script.

Instructions

100 XP

- Create a list, `areas`, that contains the area of the hallway (`hall`), kitchen (`kit`), living room (`liv`), bedroom (`bed`) and bathroom (`bath`), in this order. Use the predefined variables.
- Print `areas` with the `print()` function.

Take Hint (-30 XP)

DataCamp

Exercise

Select the valid list

A list can contain any Python type. But a list itself is also a Python type. That means that a list can also contain a list! Python is getting funkier by the minute, but fear not, just remember the list syntax:

```
my_list = [el1, el2, el3]
```

Can you tell which ones of the following lines of Python code are valid ways to build a list?

A. `[1, 3, 4, 2]` B. `[[1, 2, 3], [4, 5, 7]]` C. `[1 + 2, "a" * 5, 3]`

Instructions 50 XP

Possible Answers

- A, B and C

Instead of creating a flat list containing strings and floats, representing the names and areas of the rooms in your house, you can create a list of lists. The script on the right can already give you an idea.

Don't get confused here: `"hallway"` is a string, while `hall` is a variable that represents the float `11.25` you specified earlier.

Instructions 100 XP

- Finish the list of lists so that it also contains the bedroom and bathroom data. Make sure you enter these in order!
- Print out `house`; does this way of structuring your data make more sense?
- Print out the type of `house`. Are you still dealing with a list?

Take Hint (-30 XP)

Incorrect Submission

Have you used `print(type(house))` to print out the type of the `house` variable?

Did you find this feedback helpful? ✓ Yes ✗ No

script.py

```
2  hall = 11.25
3  kit = 18.0
4  liv = 20.0
5  bed = 10.75
6  bath = 9.50
7
8  # house information as list of lists
9  house = [["hallway", hall],
10          ["kitchen", kit],
11          ["living room", liv],
12          ["bedroom",bed],
13          ["bathroom",bath]]
14 print(house)
15 print(type(house))
```

IPython Shell Slides

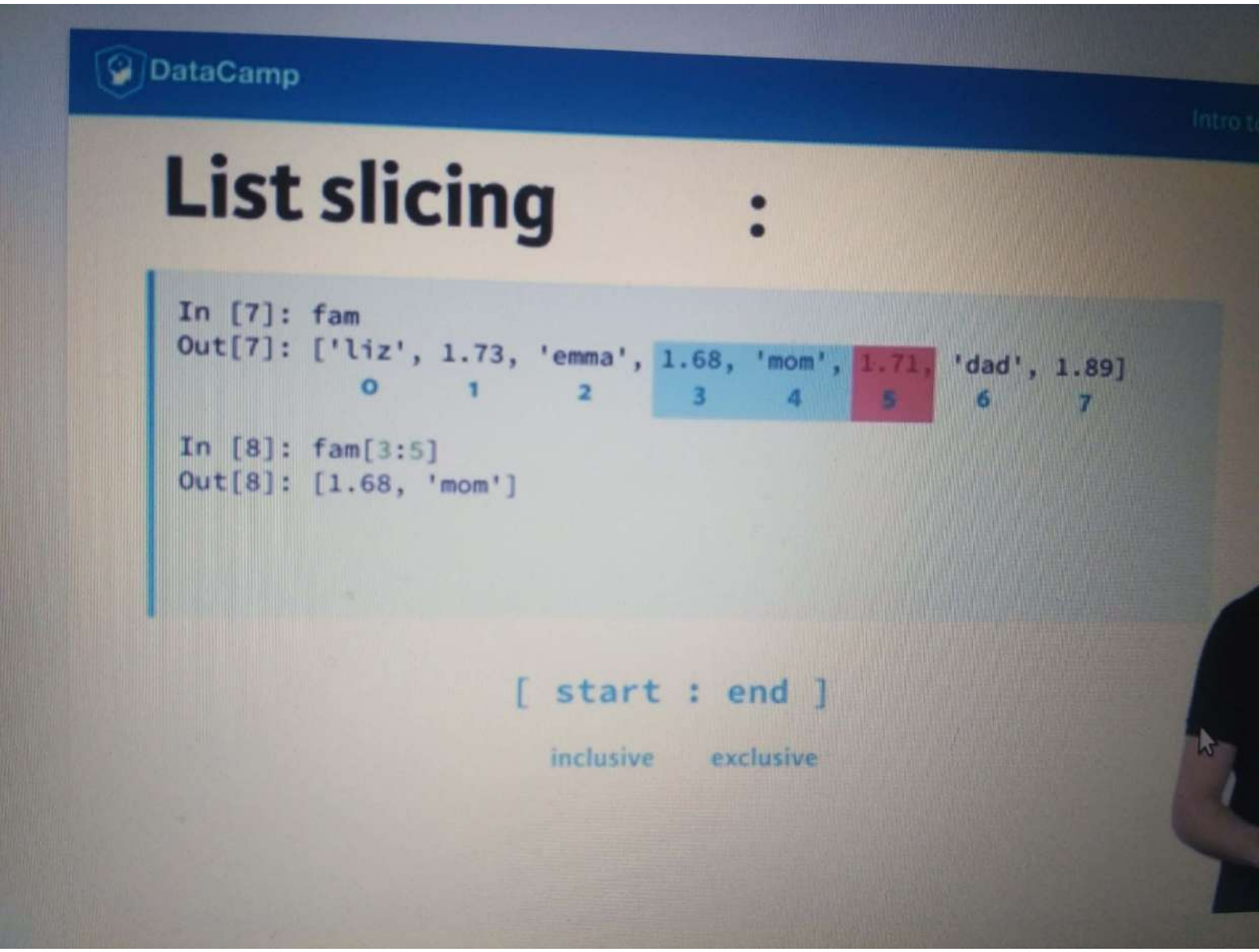
```
print(house)
print(type(house))

# Print out house

# Print out the type of house
[['hallway', 11.25], ['kitchen', 18.0], ['livi
 ['bathroom', 9.5]]
<class 'list'>
```

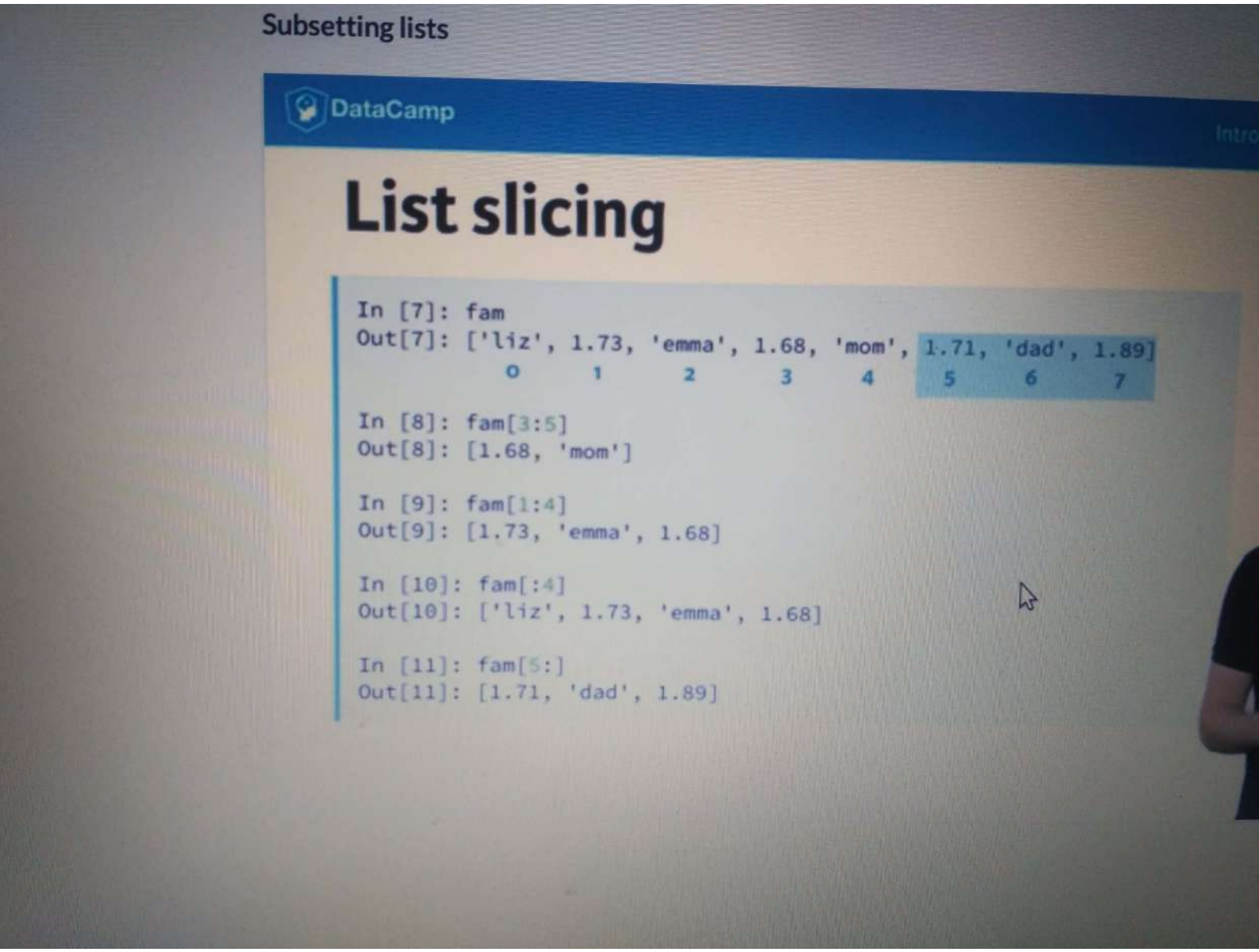
Subsetting lists

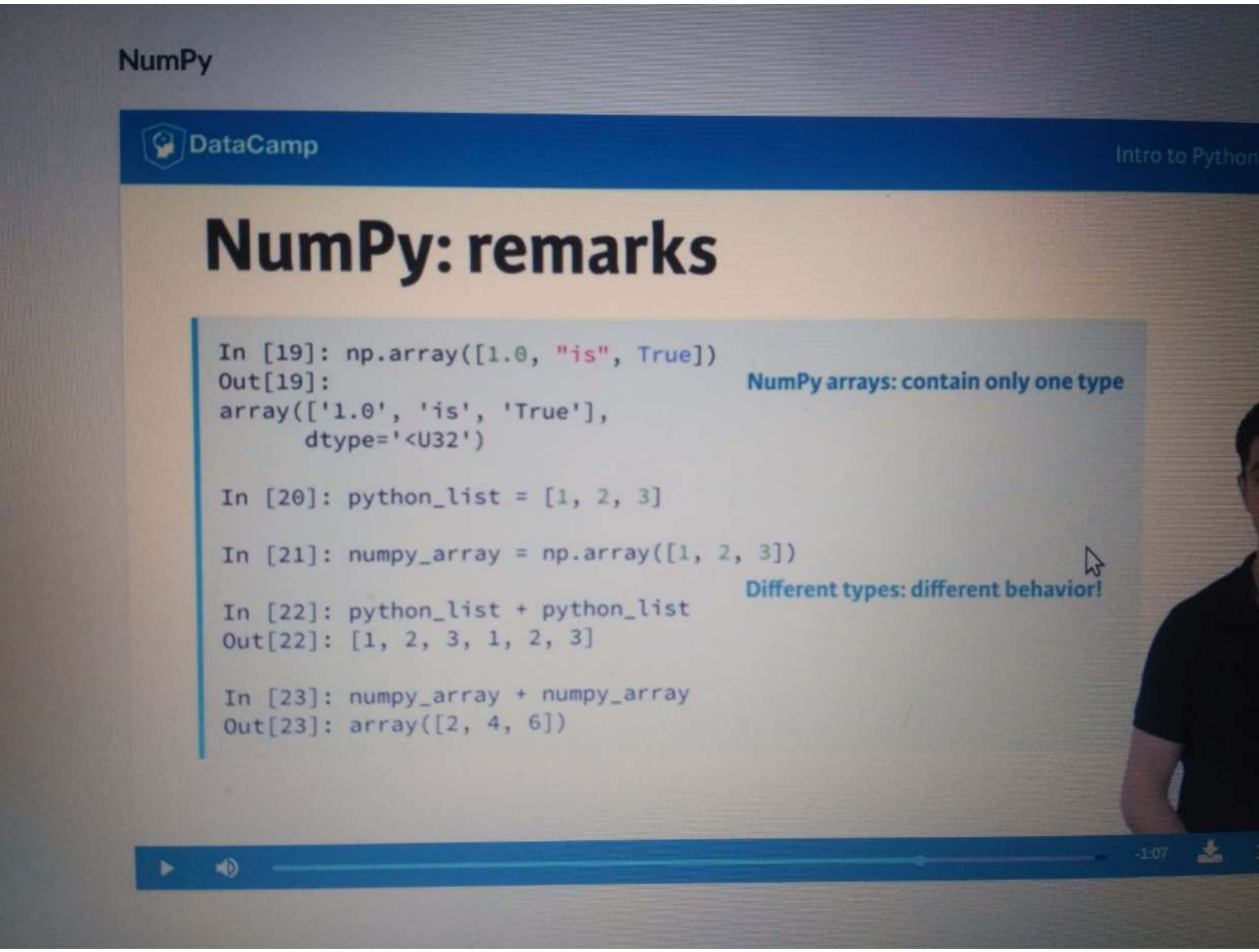
```
In [1]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [2]: fam
Out[2]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

index:	0	1	2	3	4	5	6	7
	-8	-7	-6	-5	-4	-3	-2	-1

```
In [3]: fam[3]
Out[3]: 1.68

In [4]: fam[6]
Out[4]: 'dad'

In [5]: fam[-1]
Out[5]: 1.89
```

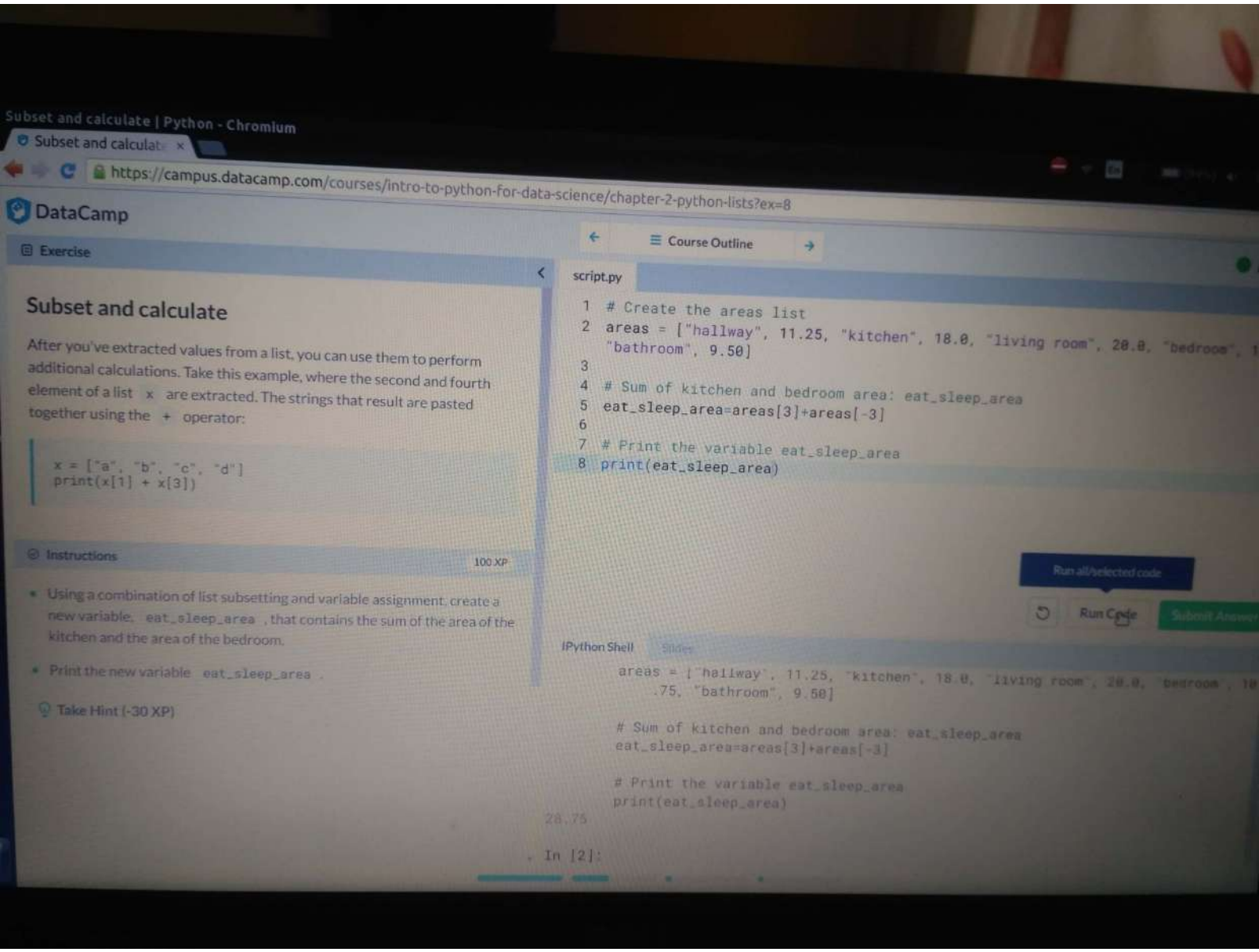


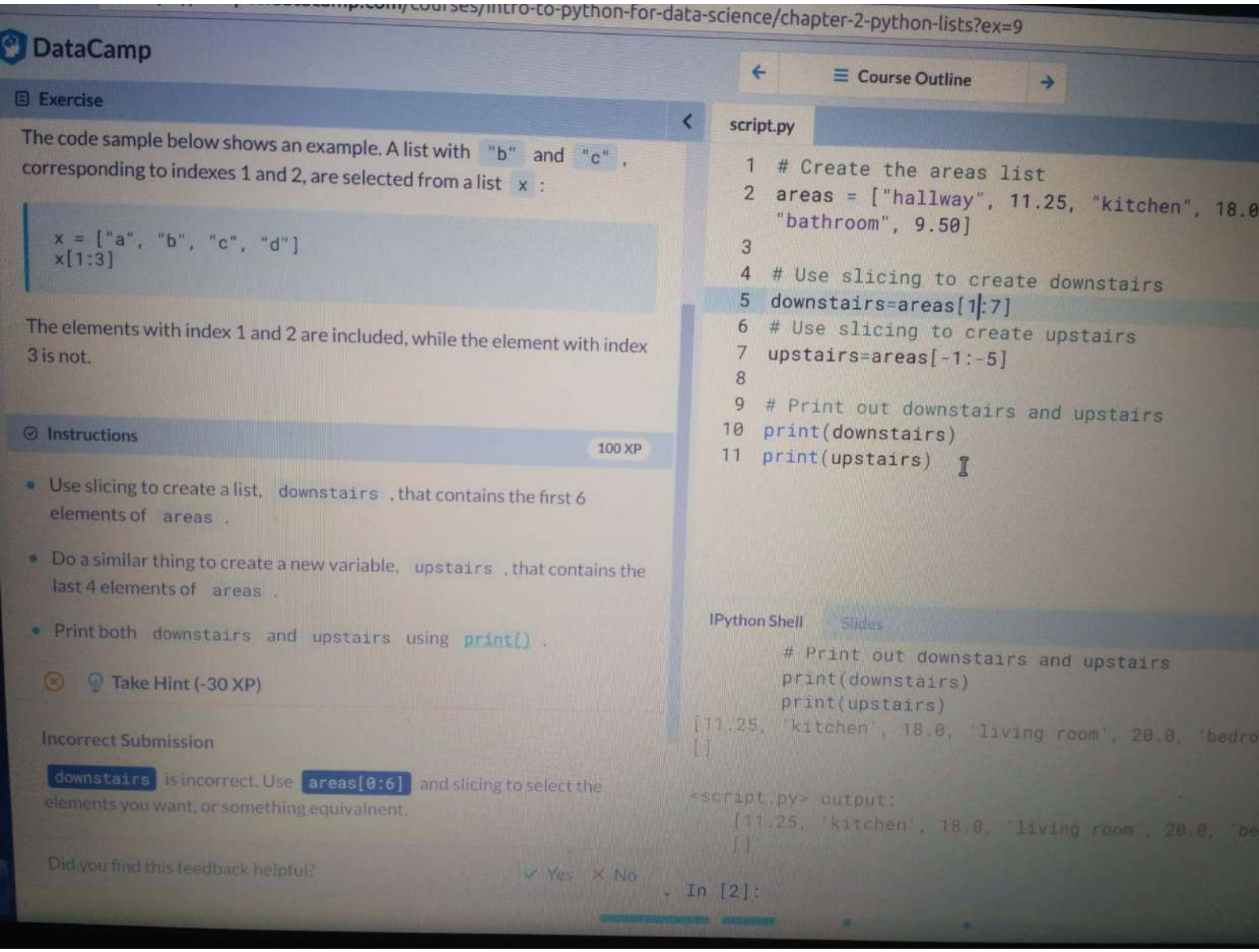


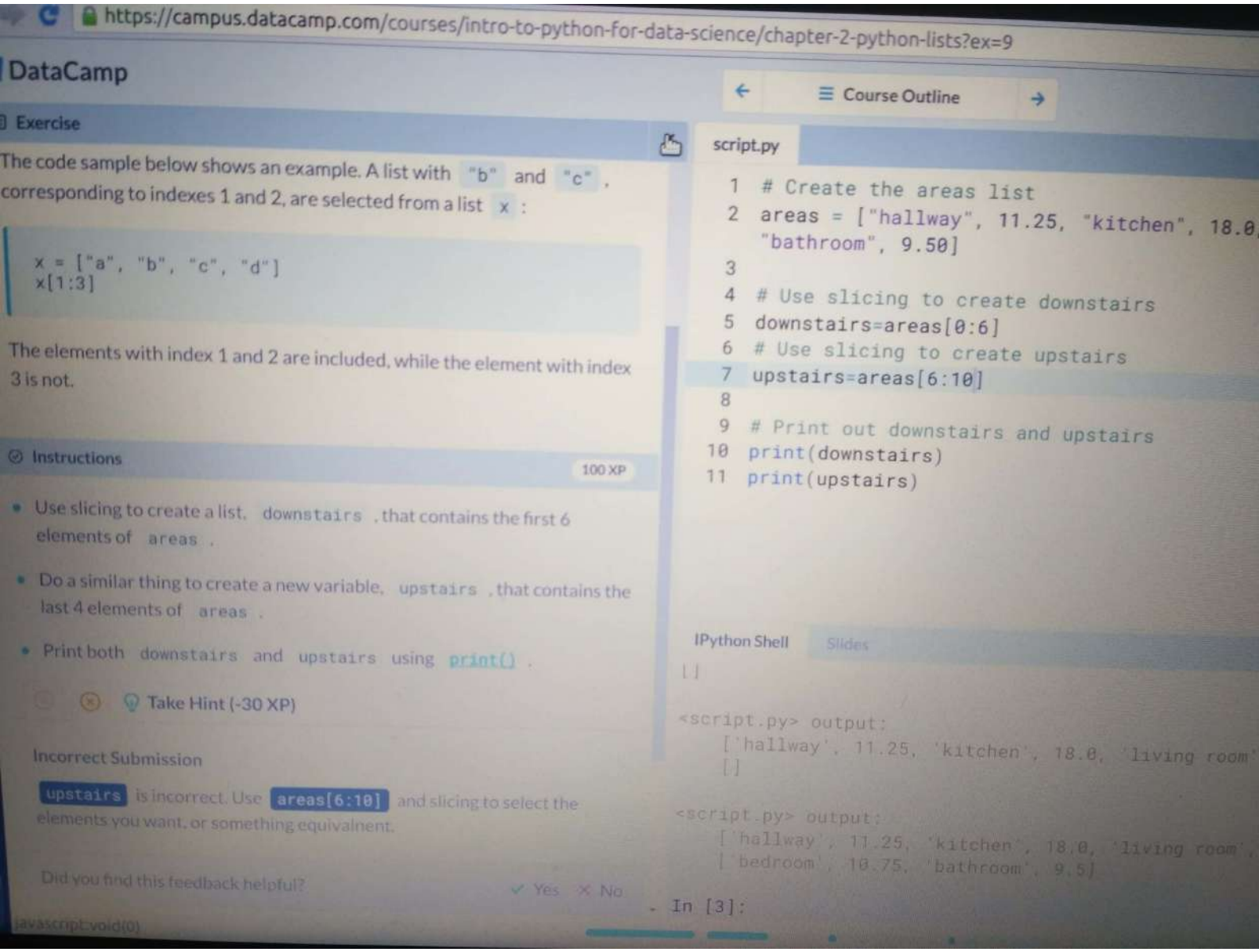


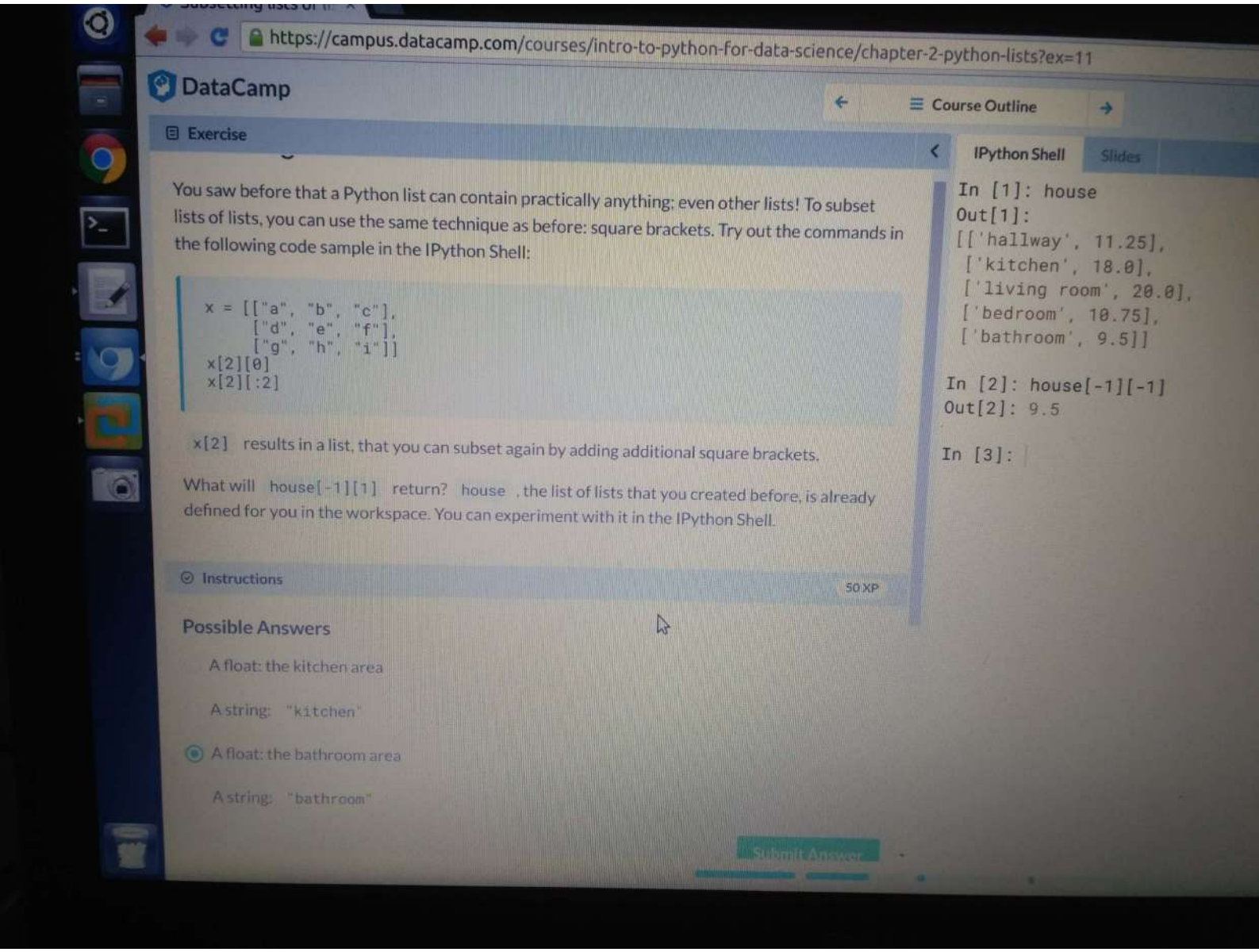
NumPy Subsetting

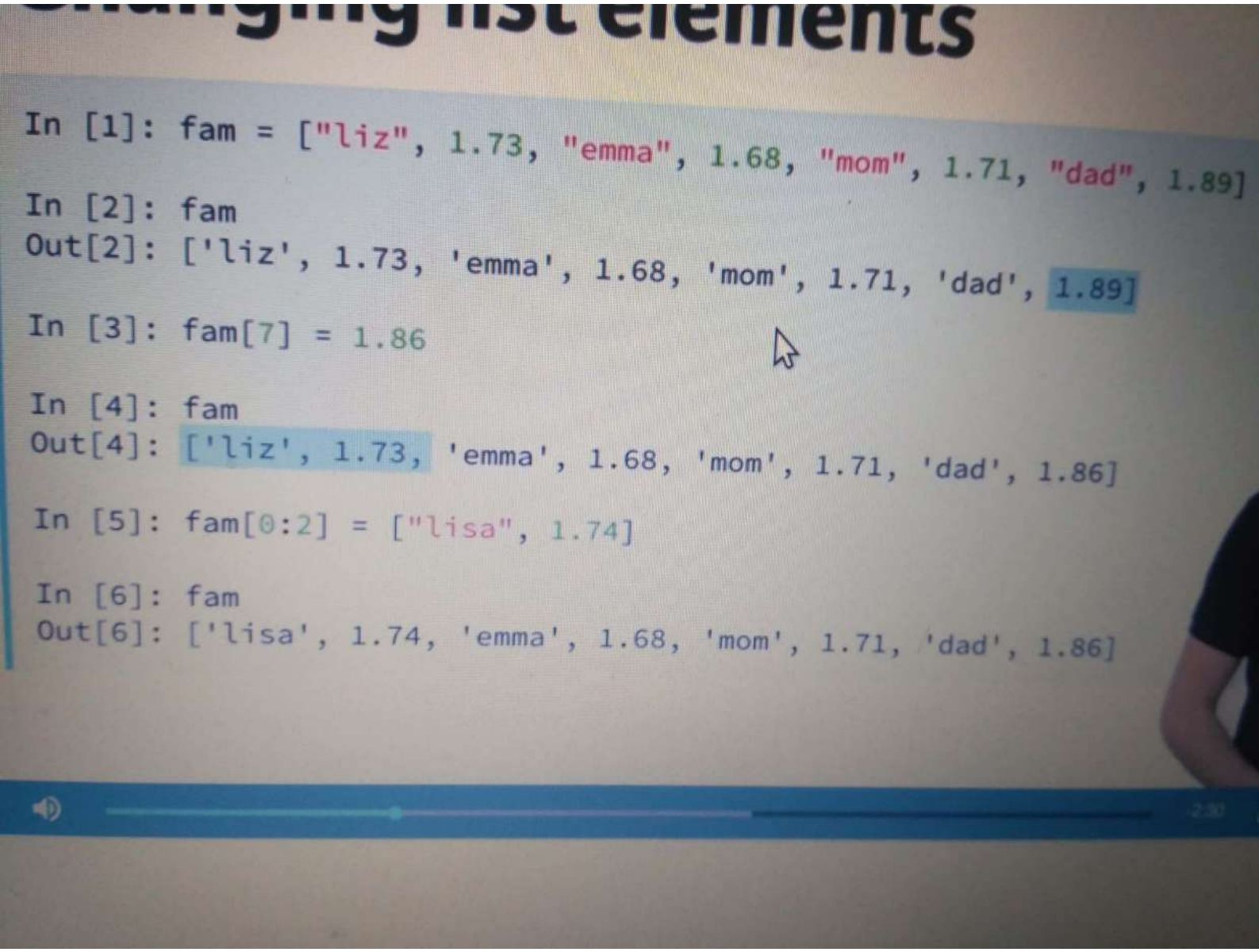
```
In [24]: bmi
Out[24]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [25]: bmi[1]
Out[25]: 20.975

In [26]: bmi > 23
Out[26]: array([False, False, False,  True, False], dtype=bool)

In [27]: bmi[bmi > 23]
Out[27]: array([ 24.747])
```

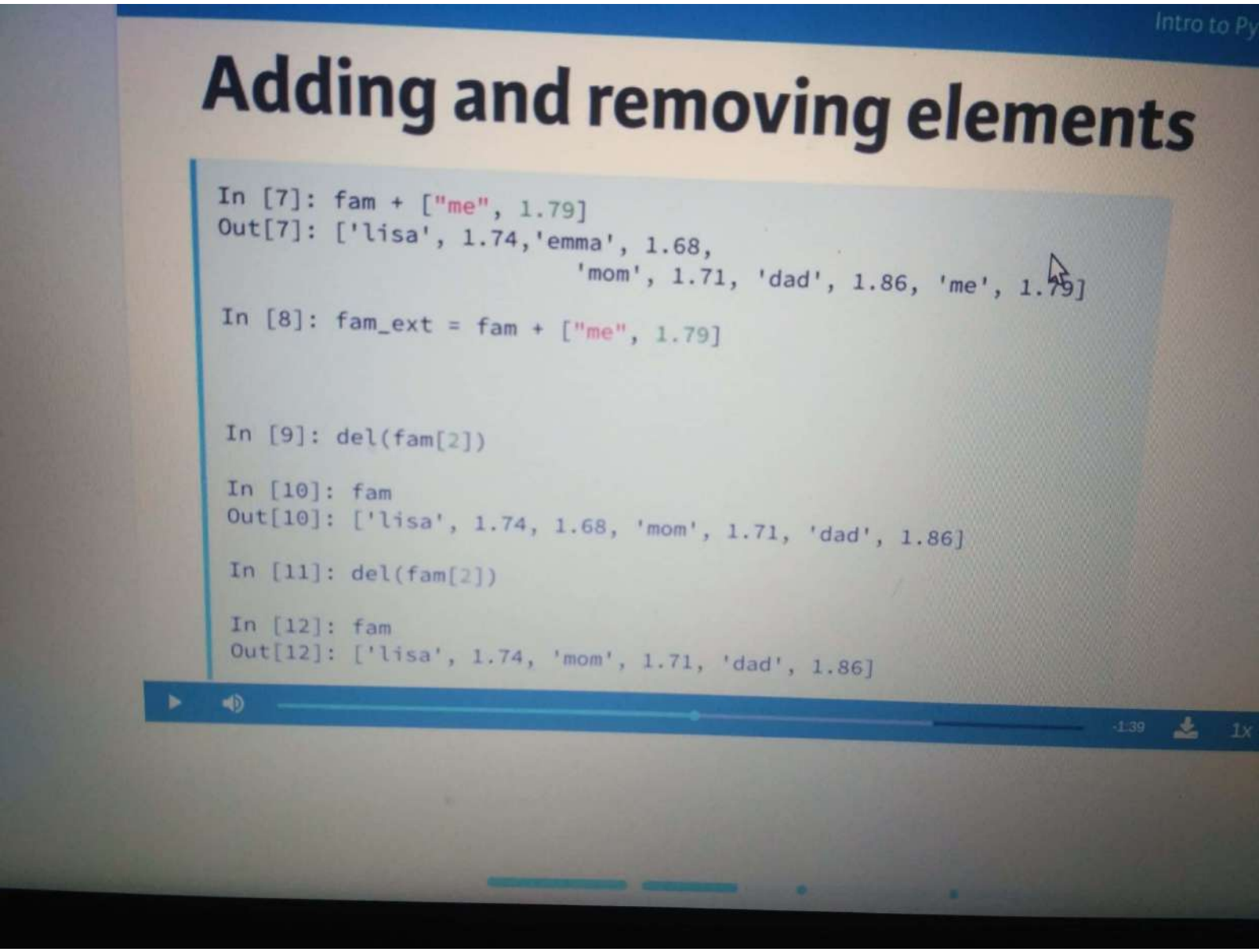


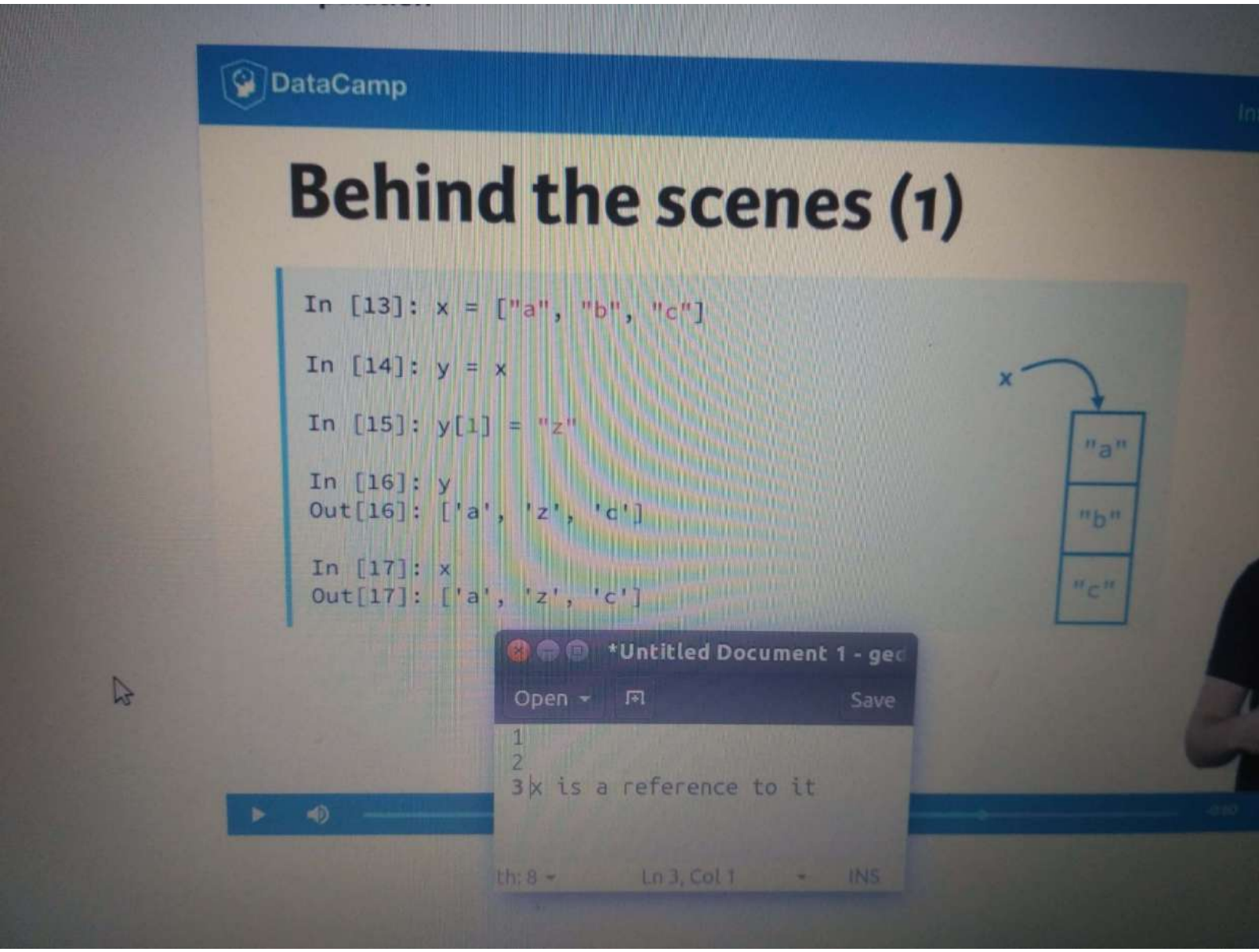






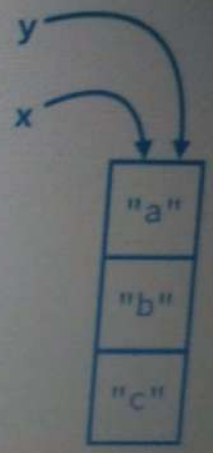






Behind the scenes (1)

```
In [13]: x = ["a", "b", "c"]

In [14]: y = x

In [15]: y[1] = "z"

In [16]: y
Out[16]: ['a', 'z', 'c']

In [17]: x
Out[17]: ['a', 'z', 'c']
```



*Untitled Document 1 - gec

Open Save

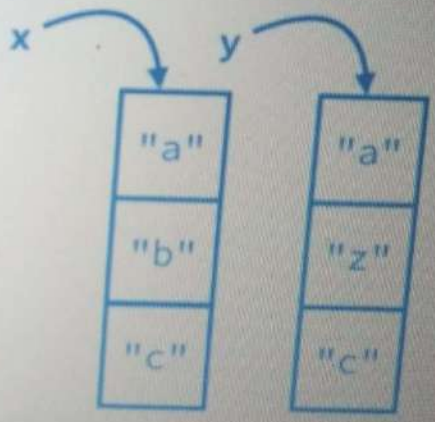
1
2
3 x is a reference to it

th: 8 Ln 3, Col 1 INS

DataCamp

Behind the scenes (2)

```
In [18]: x = ["a", "b", "c"]

In [19]: y = list(x)

In [20]: y = x[:]

In [21]: y[1] = "z"

In [22]: x
Out[22]: ['a', 'b', 'c']
```

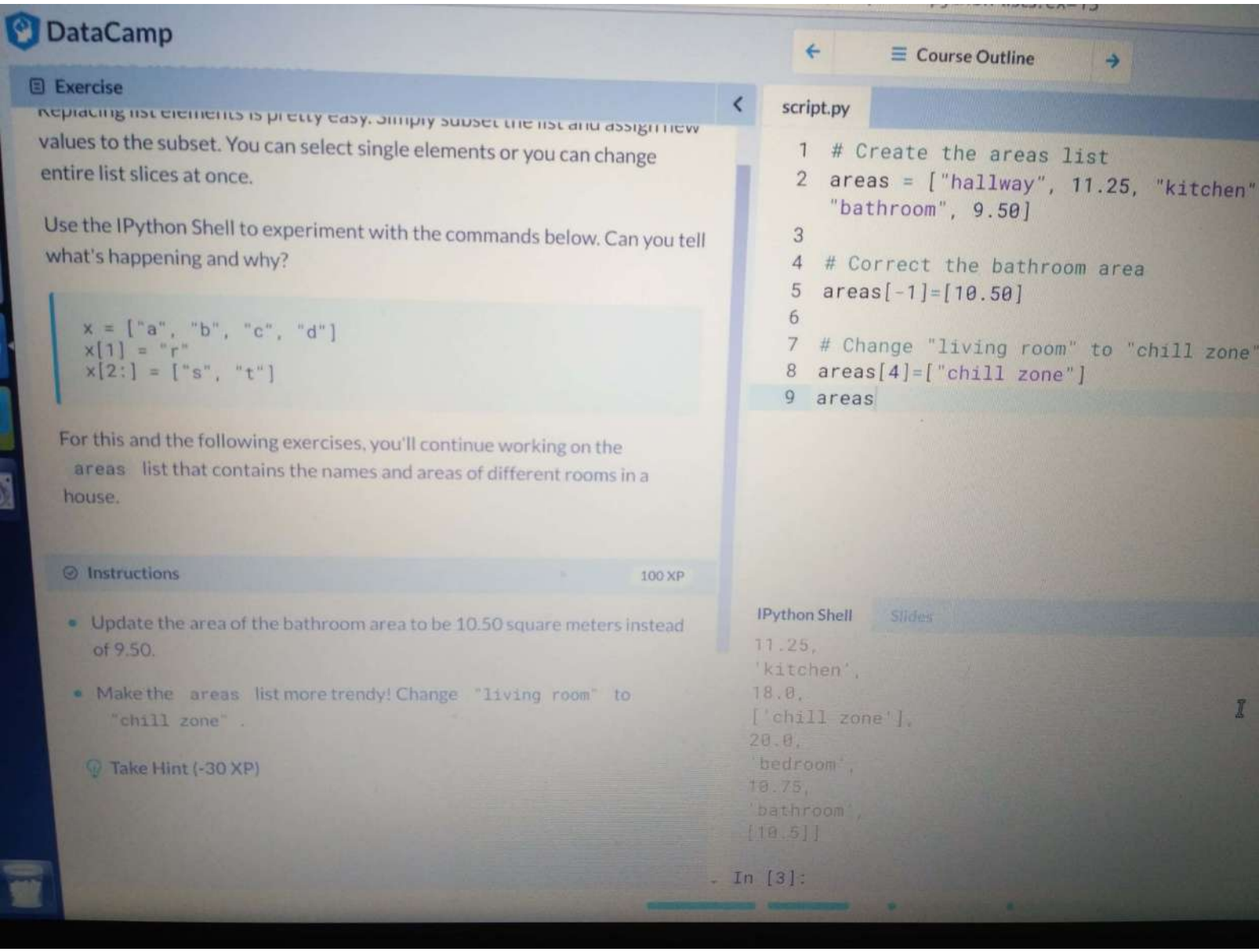


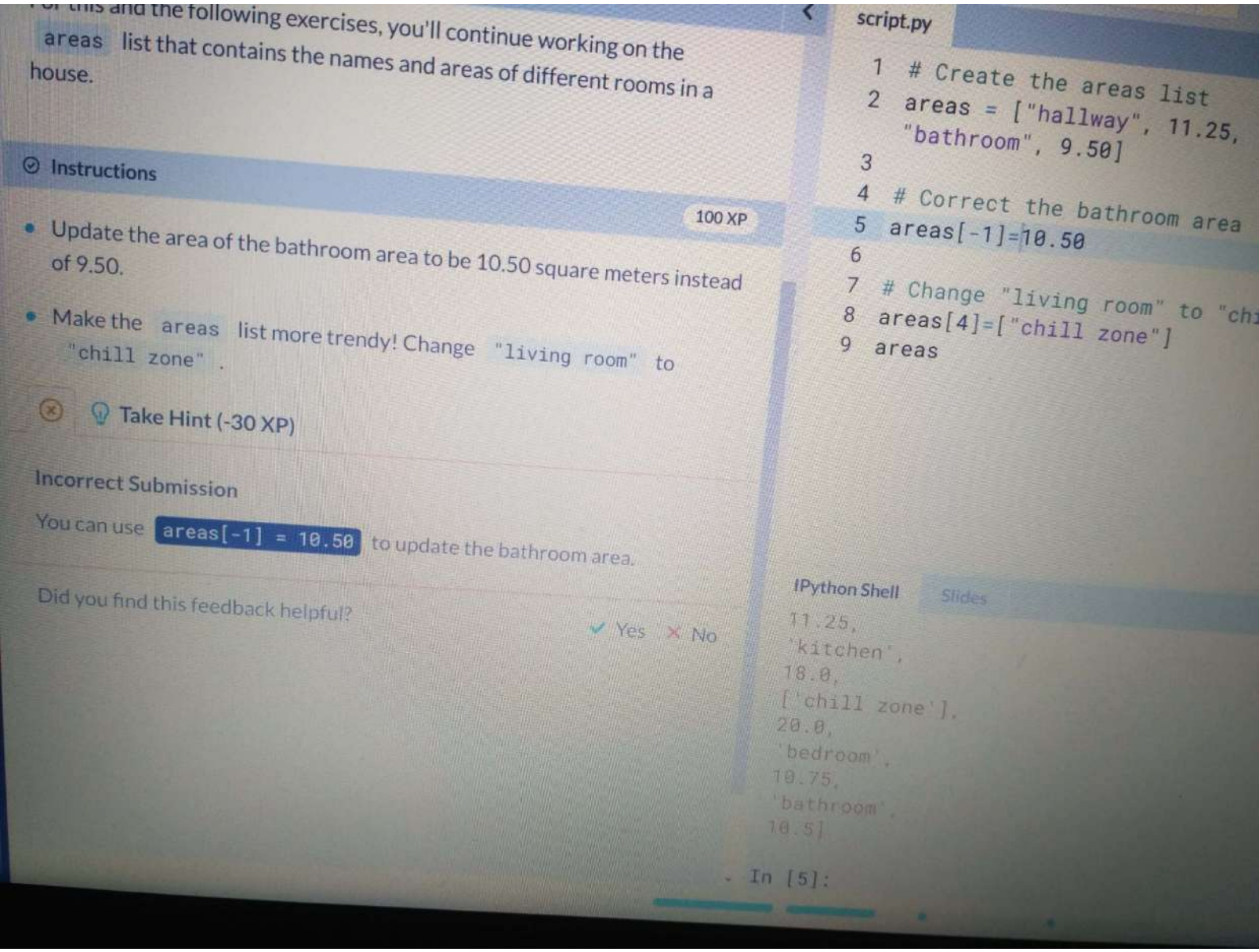
*Untitled Document 1 - ged

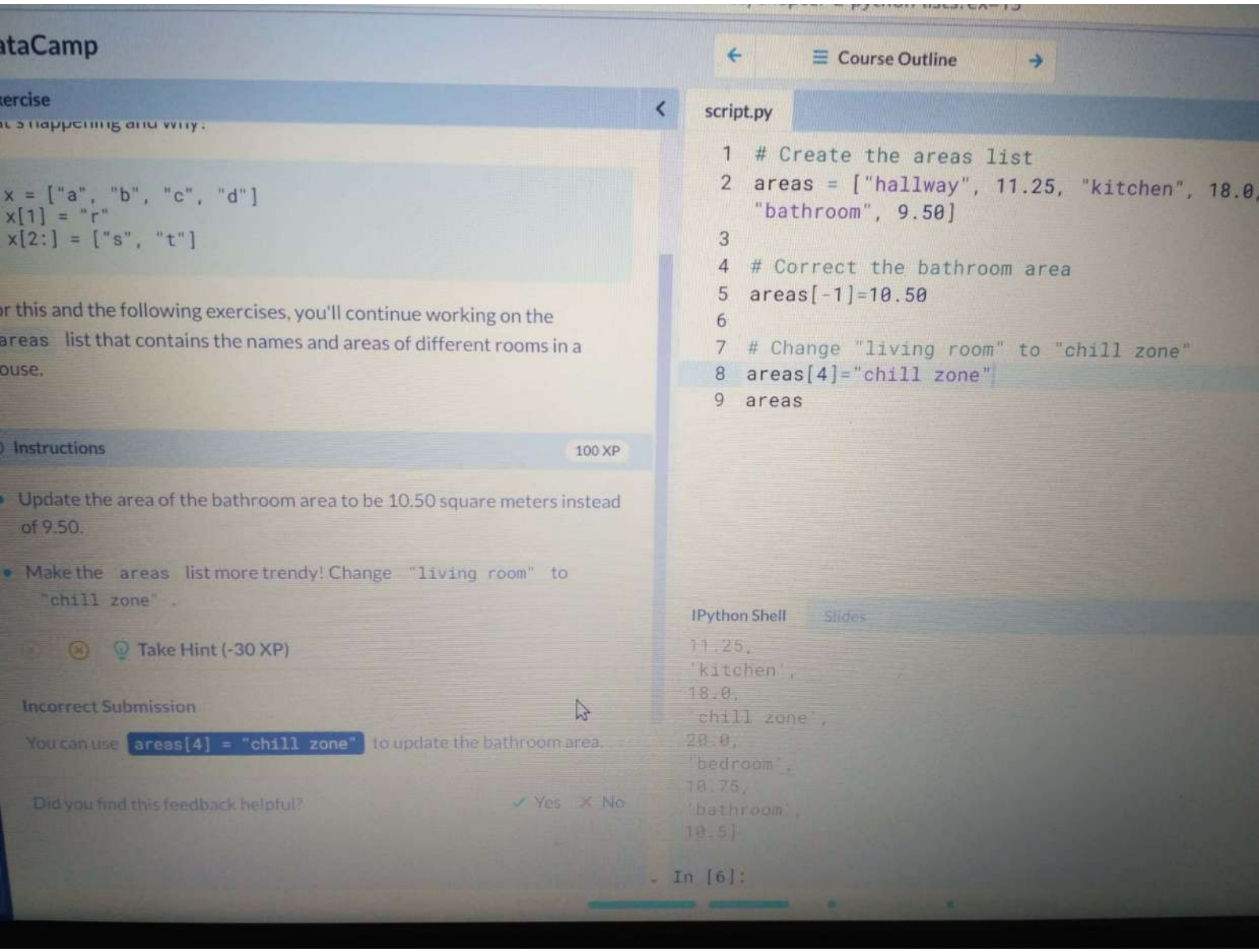
Open Save

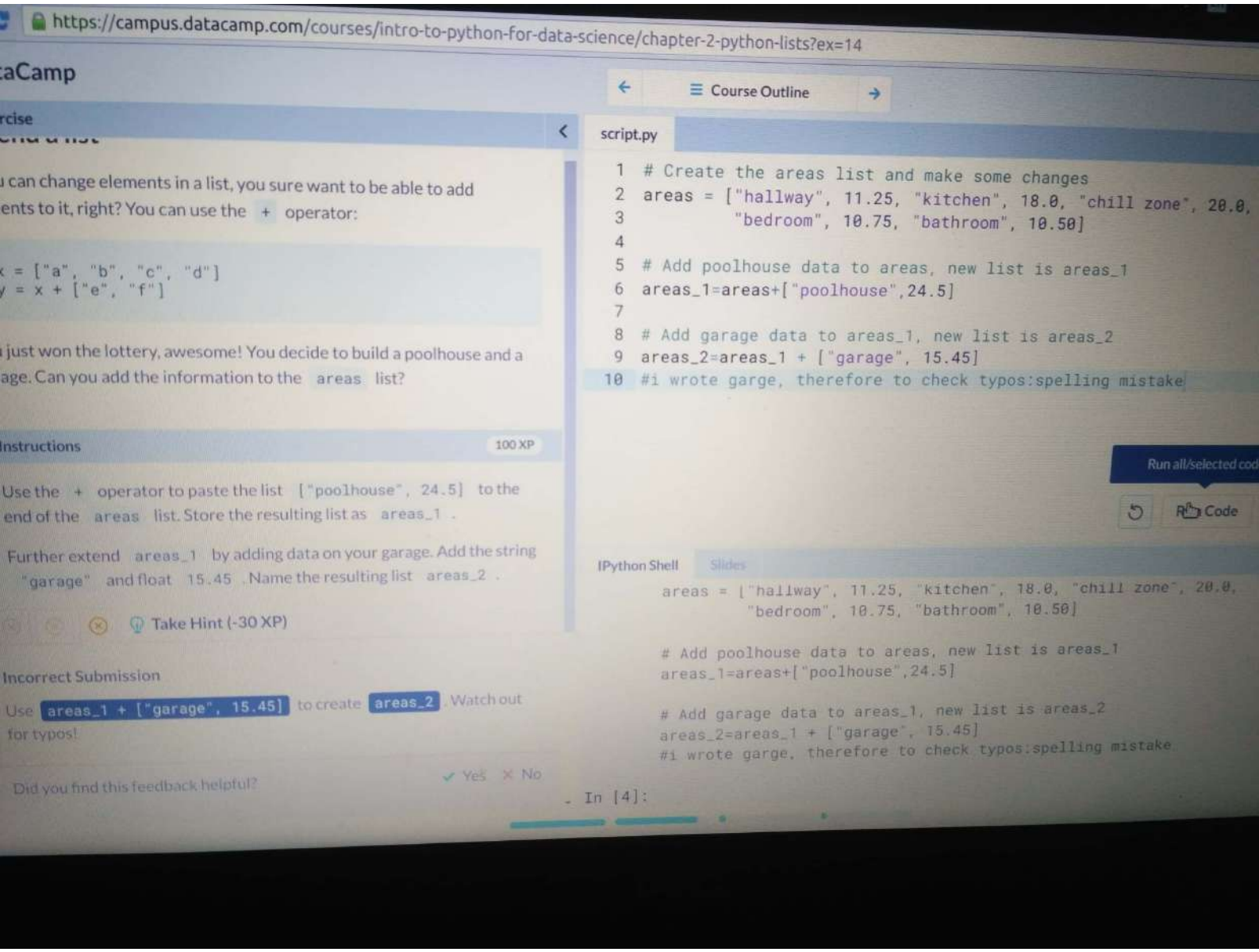
1
2
3 now, x not affected

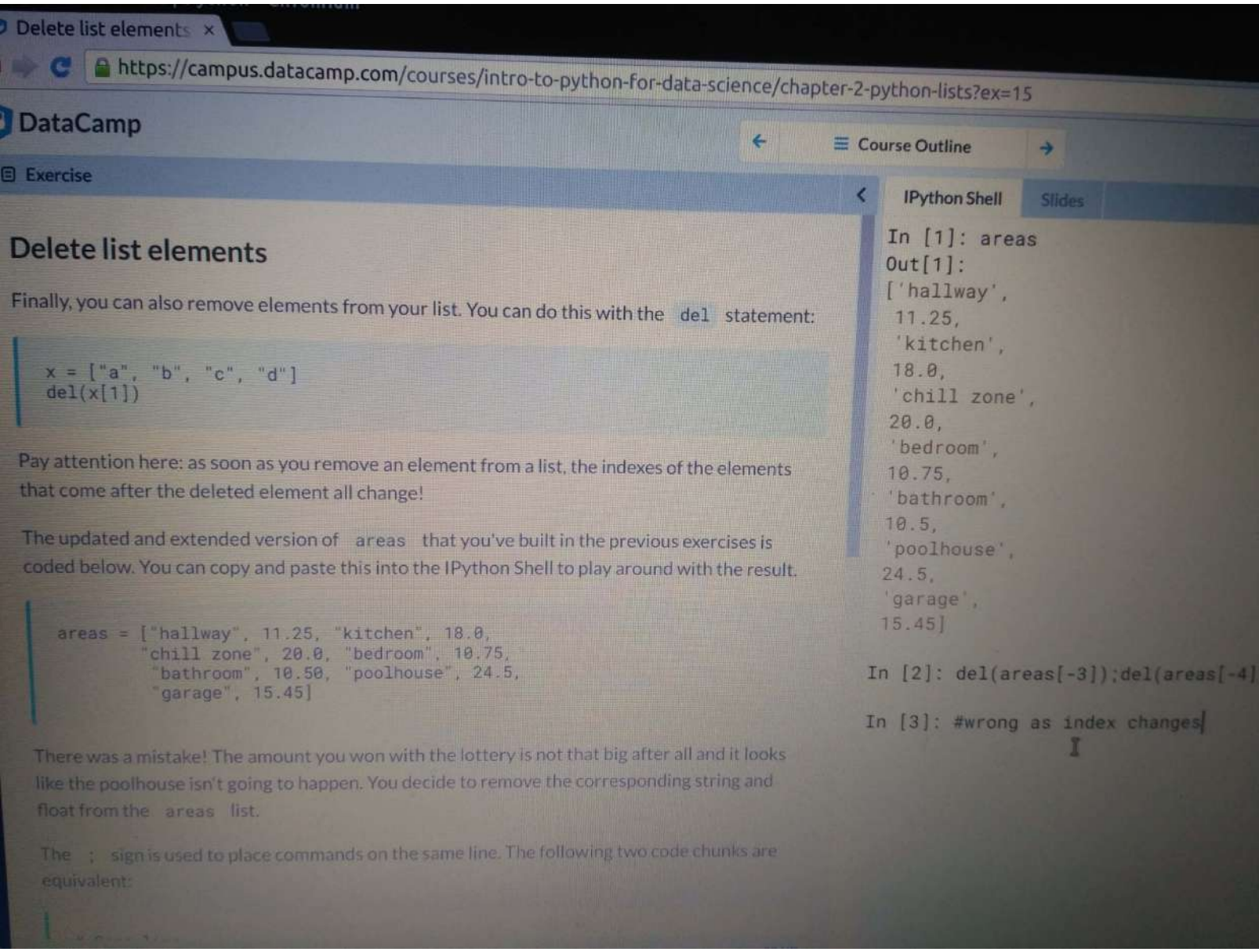
3:11

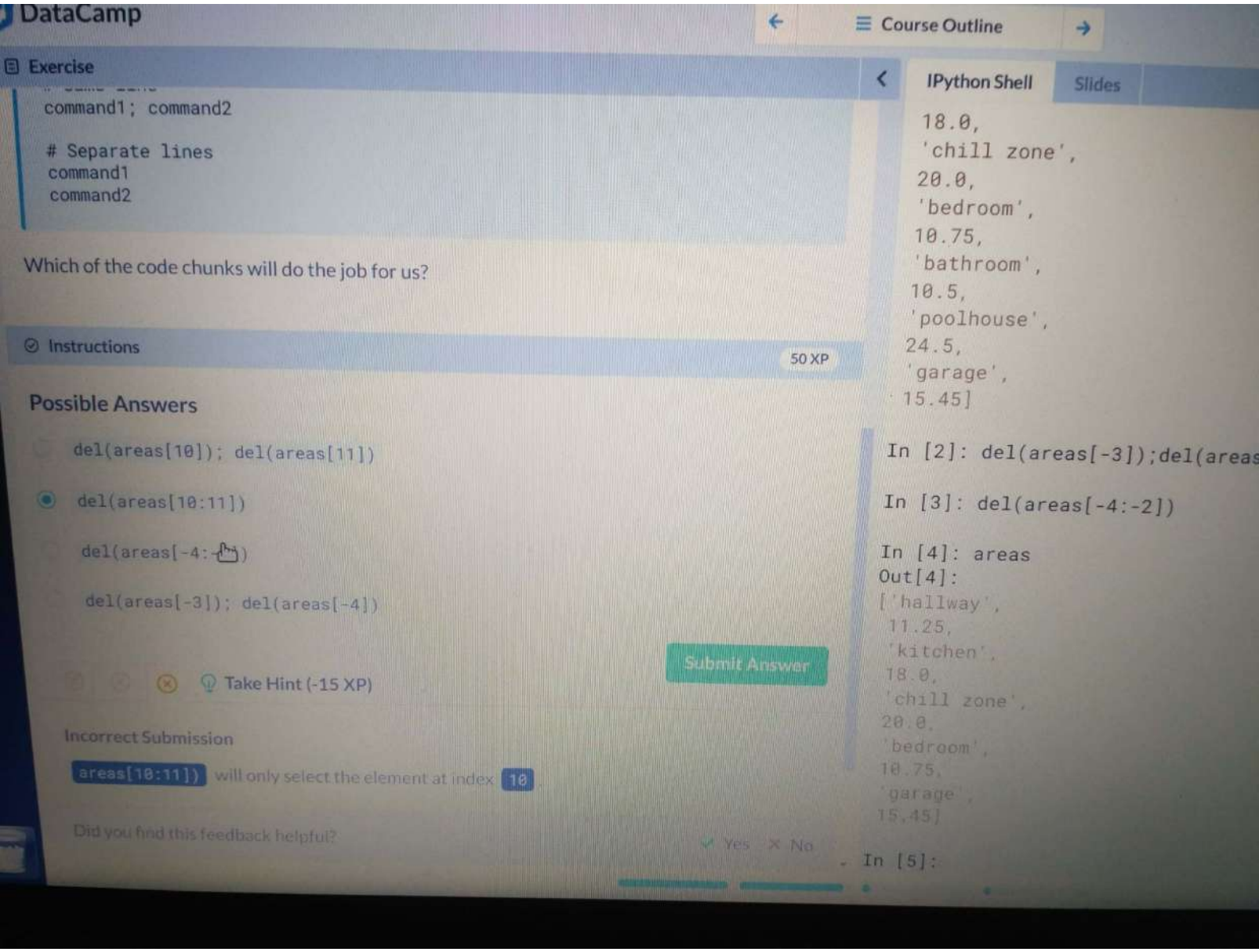


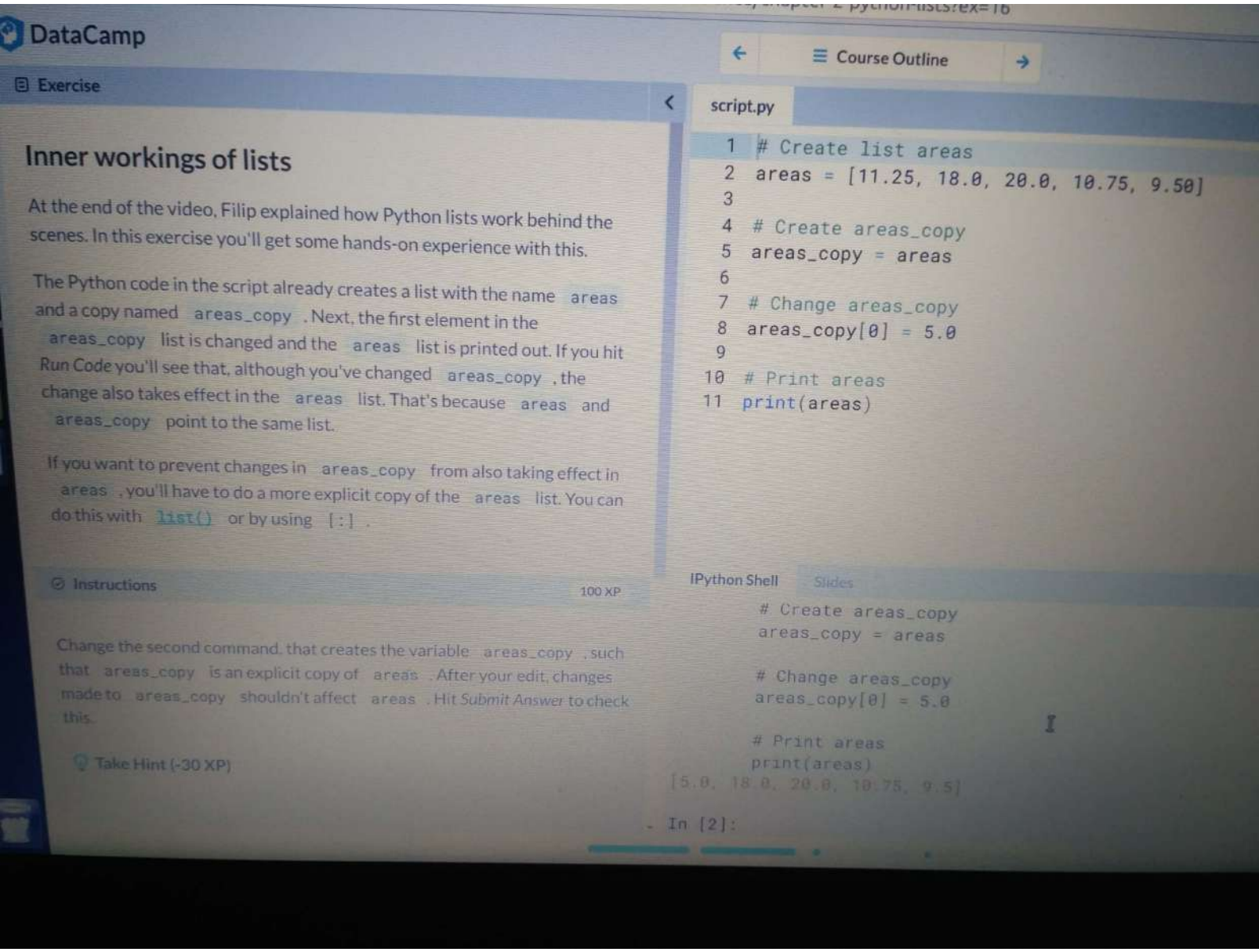


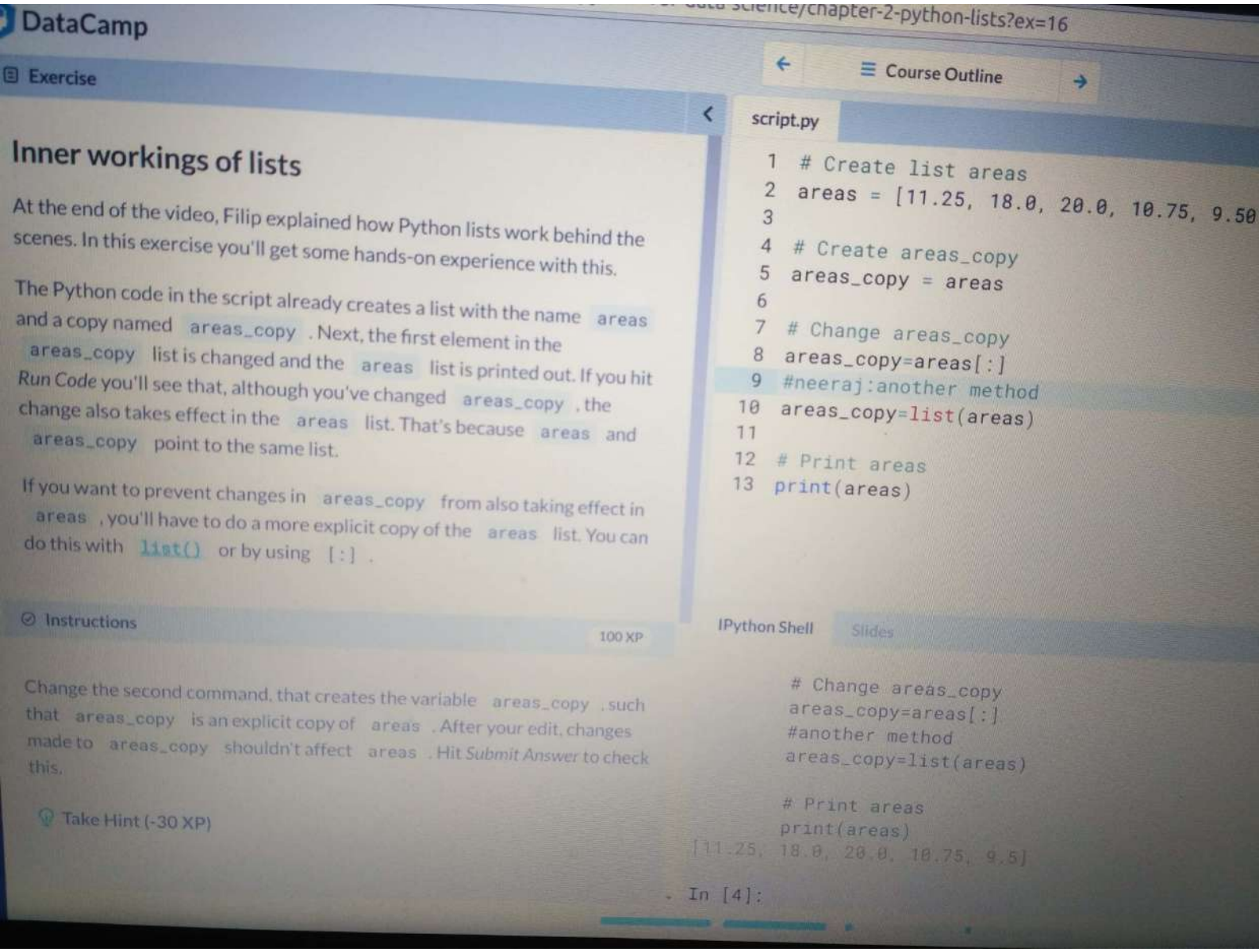


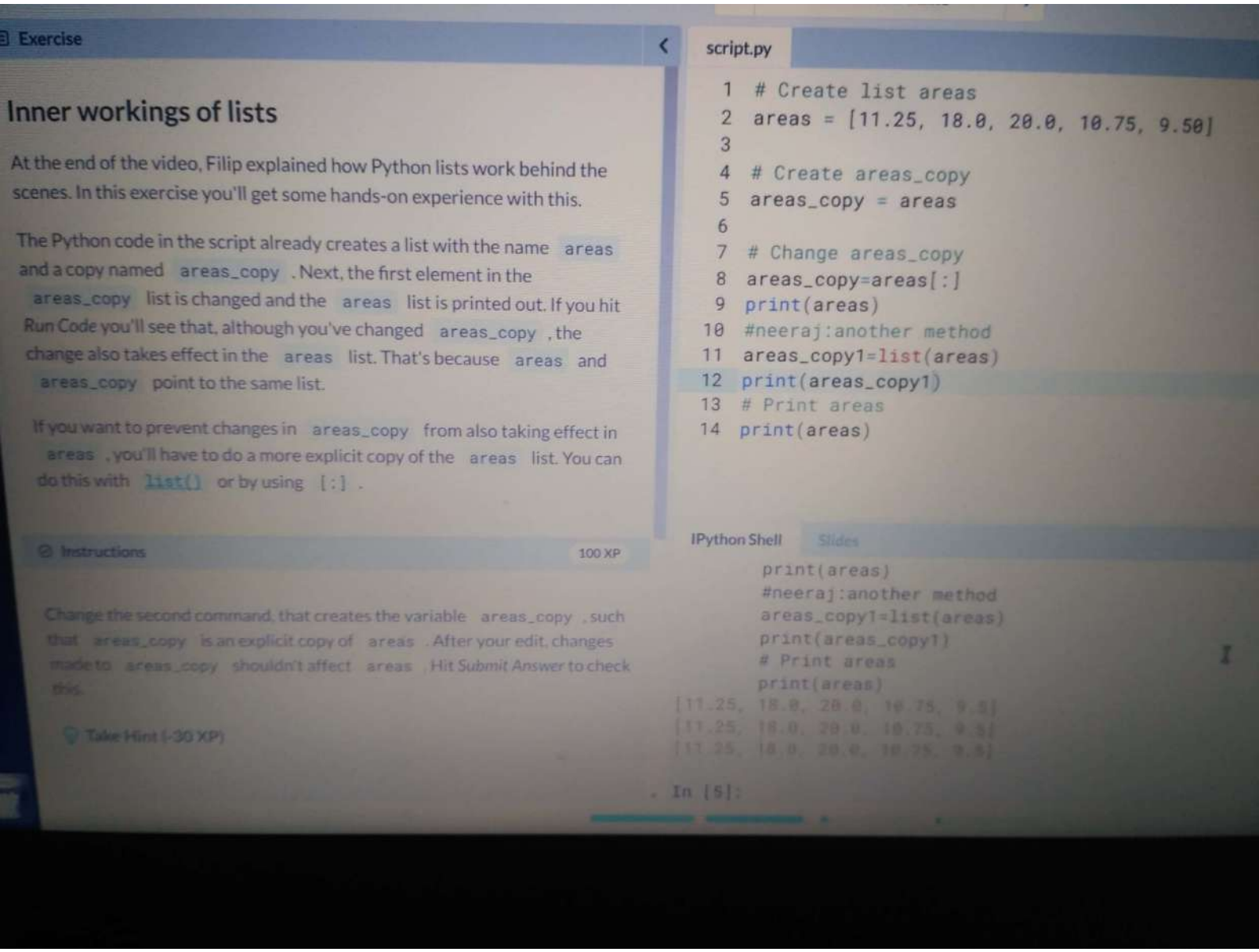


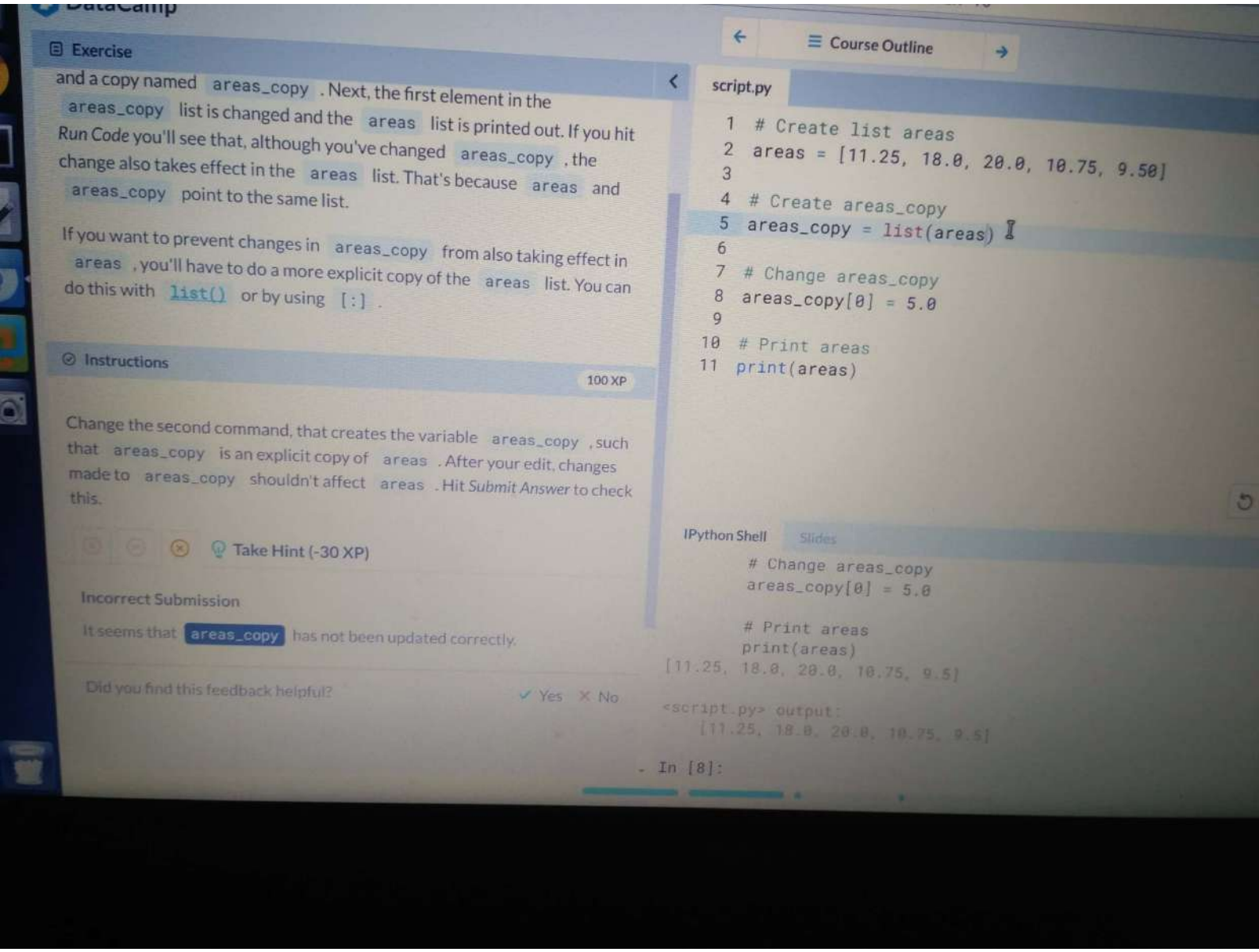










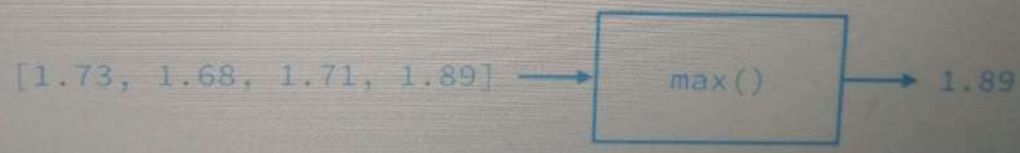


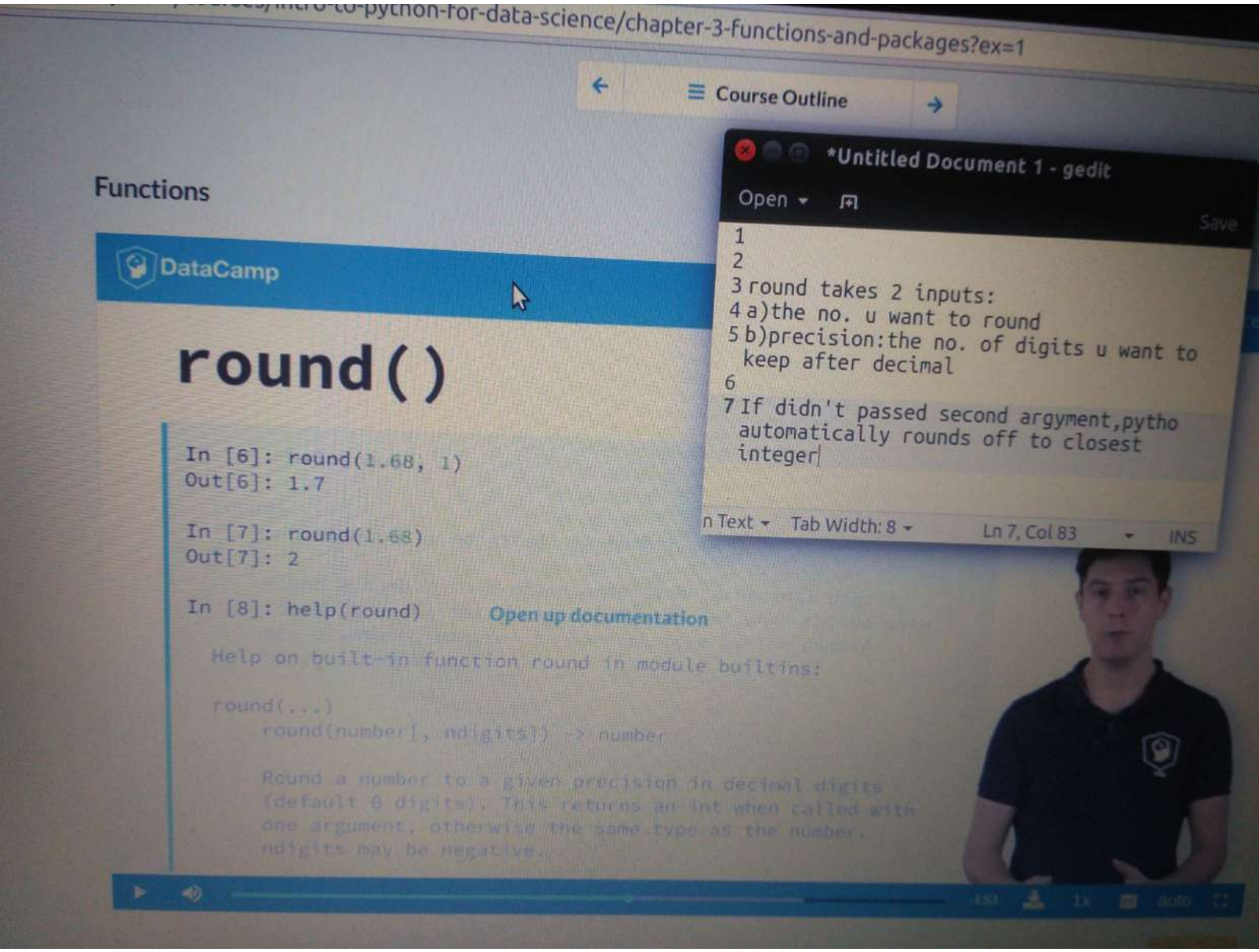
Functions

DataCamp

Example

```
In [1]: fam = [1.73, 1.68, 1.71, 1.89]

In [2]: fam
Out[2]: [1.73, 1.68, 1.71, 1.89]

In [3]: max(fam)
Out[3]: 1.89
```





round()

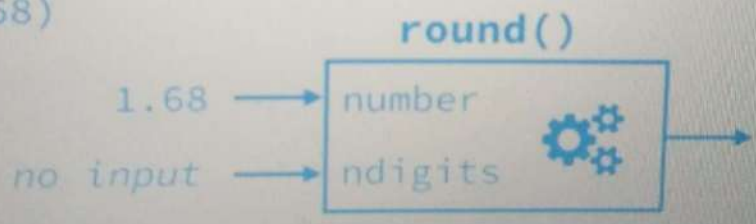
```
In [8]: help(round)

  round(...)
      round(number[, ndigits]) -> number

      Round a number to a given precision in decimal digits
      (default 0 digits). This returns an int when called with
      one argument, otherwise the same type as the number.
      ndigits may be negative.
```

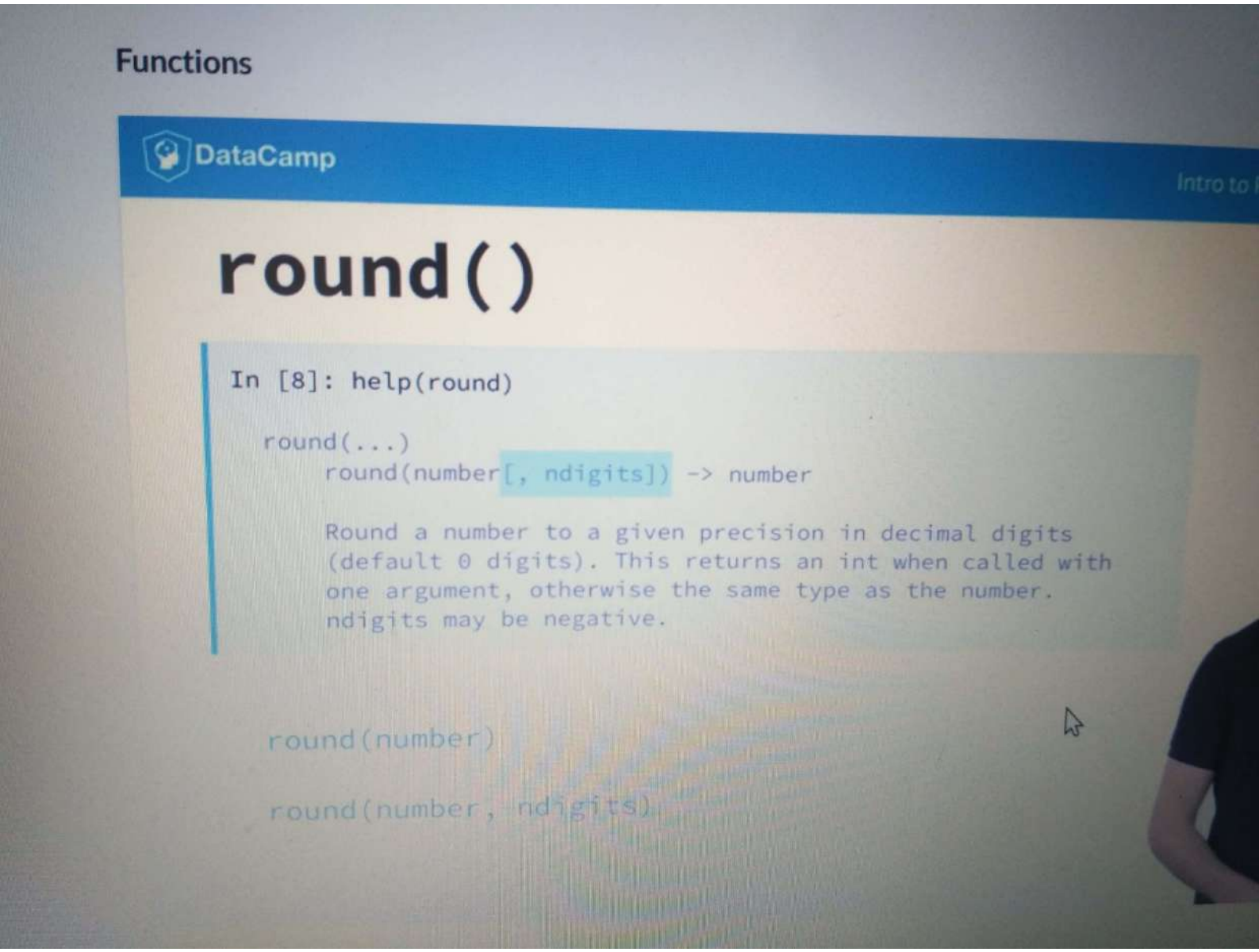
```
round(1.68, 1)
```

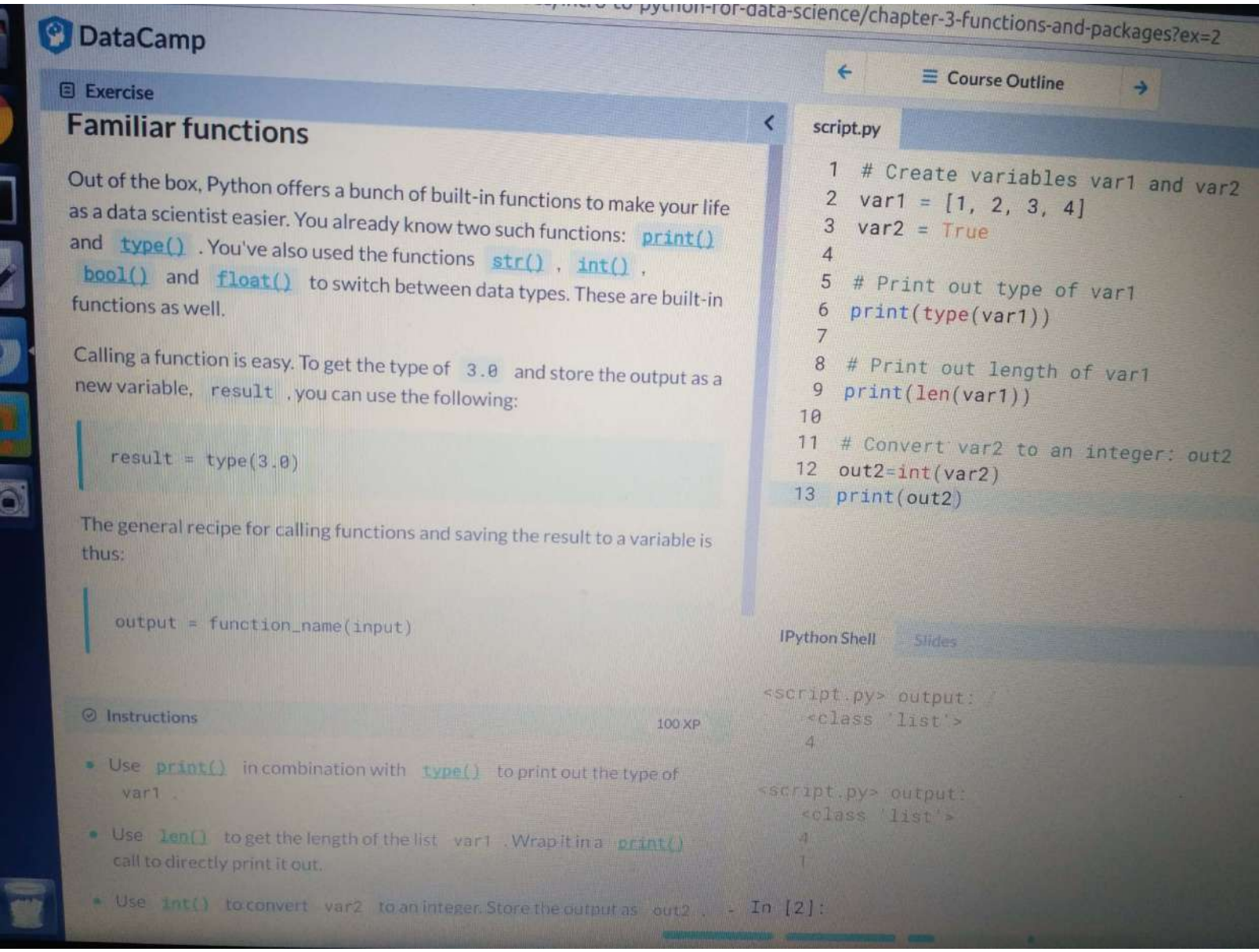
round()

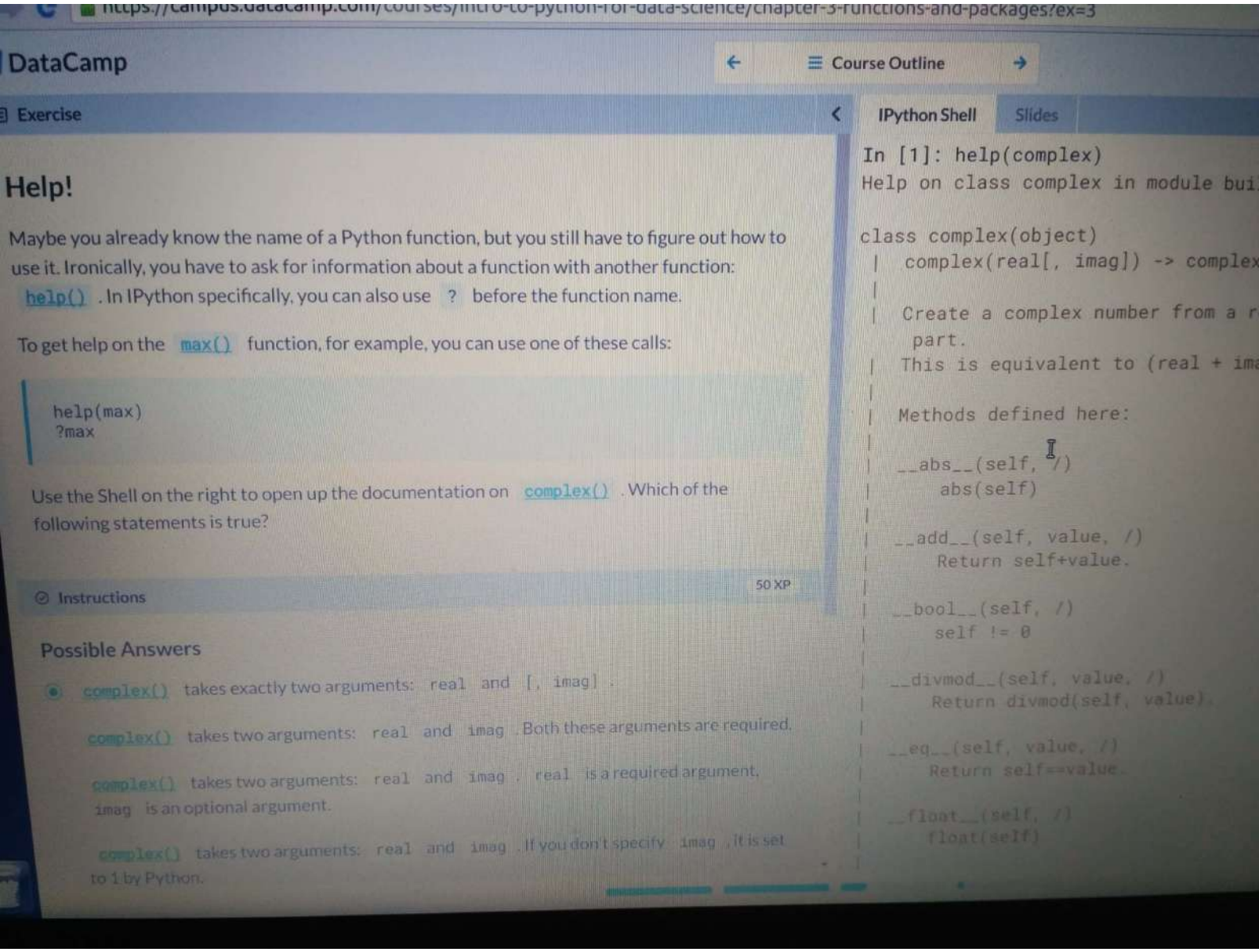
```
In [8]: help(round)

  round(...)
      round(number[, ndigits]) -> number

      Round a number to a given precision in decimal digits
      (default 0 digits). This returns an int when called with
      one argument, otherwise the same type as the number.
      ndigits may be negative.
```

round(1.68)









```
help(max)
?max
```

Use the Shell on the right to open up the documentation on `complex()`. Which of the following statements is true?

Instructions 50 XP

Possible Answers

- `complex()` takes exactly two arguments: `real` and `[, imag]`.
- `complex()` takes two arguments: `real` and `imag`. Both these arguments are required.
- ◉ `complex()` takes two arguments: `real` and `imag`. `real` is a required argument, `imag` is an optional argument.
- `complex()` takes two arguments: `real` and `imag`. If you don't specify `imag`, it is set to 1 by Python.

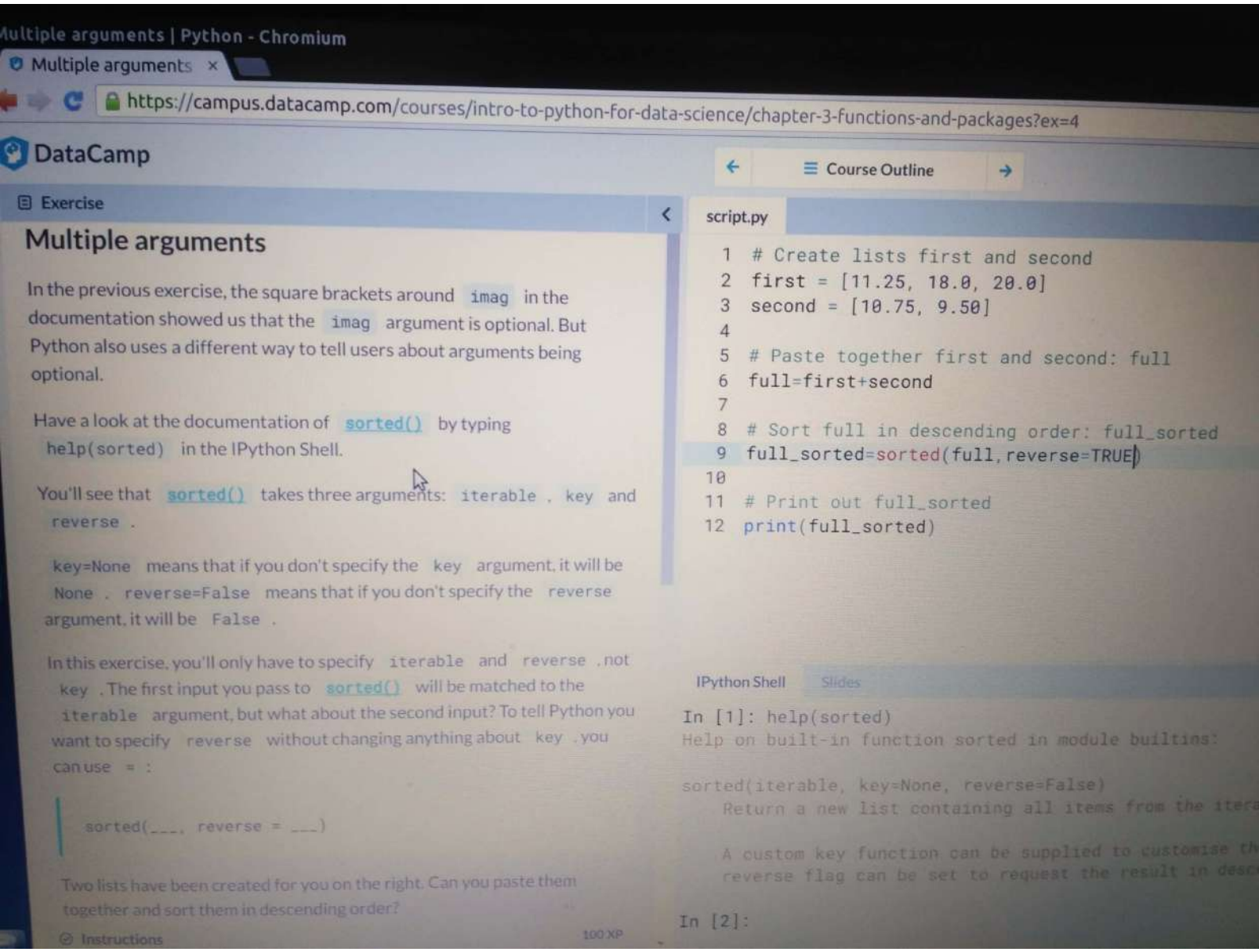
Submit Answer

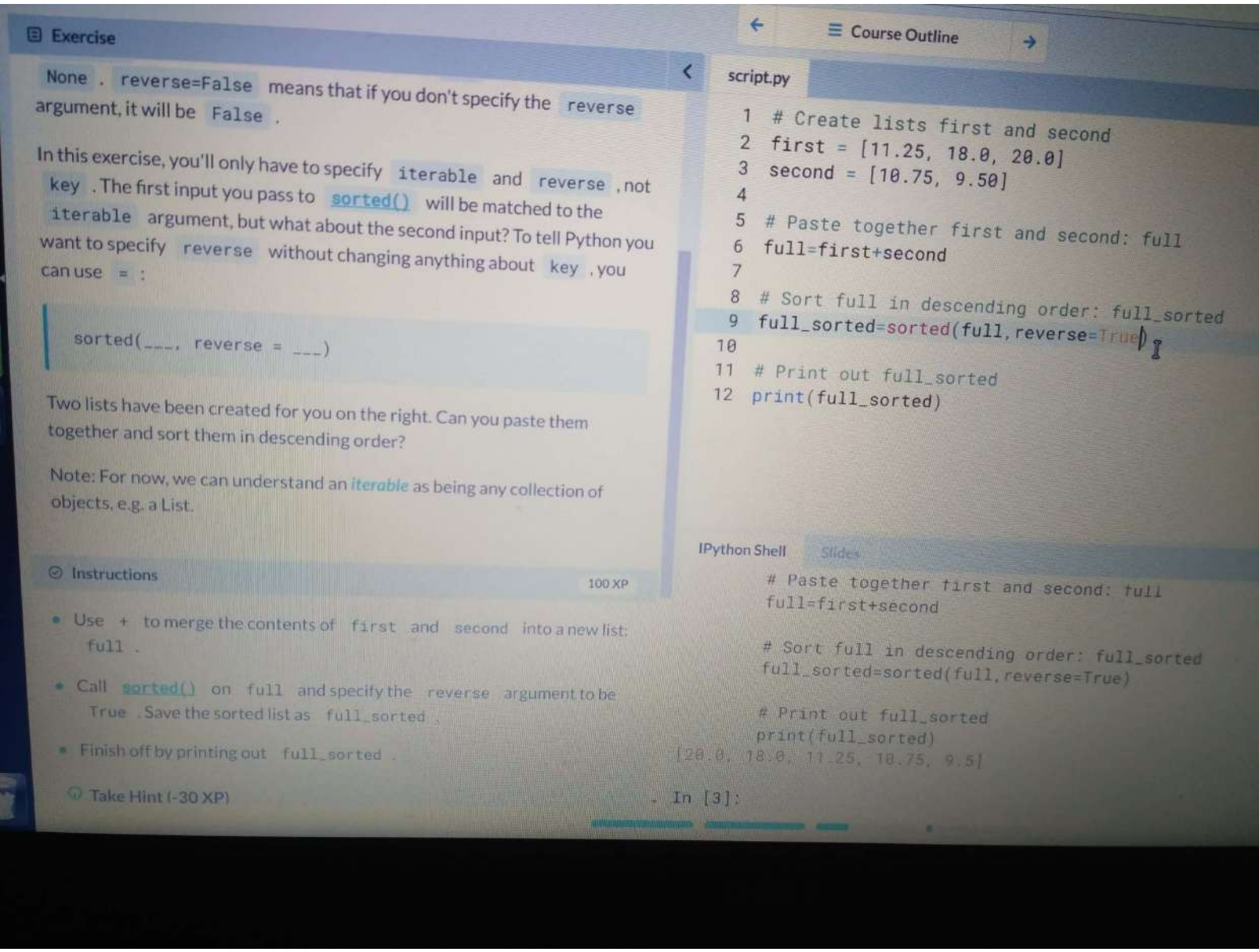
Take Hint (-15 XP)

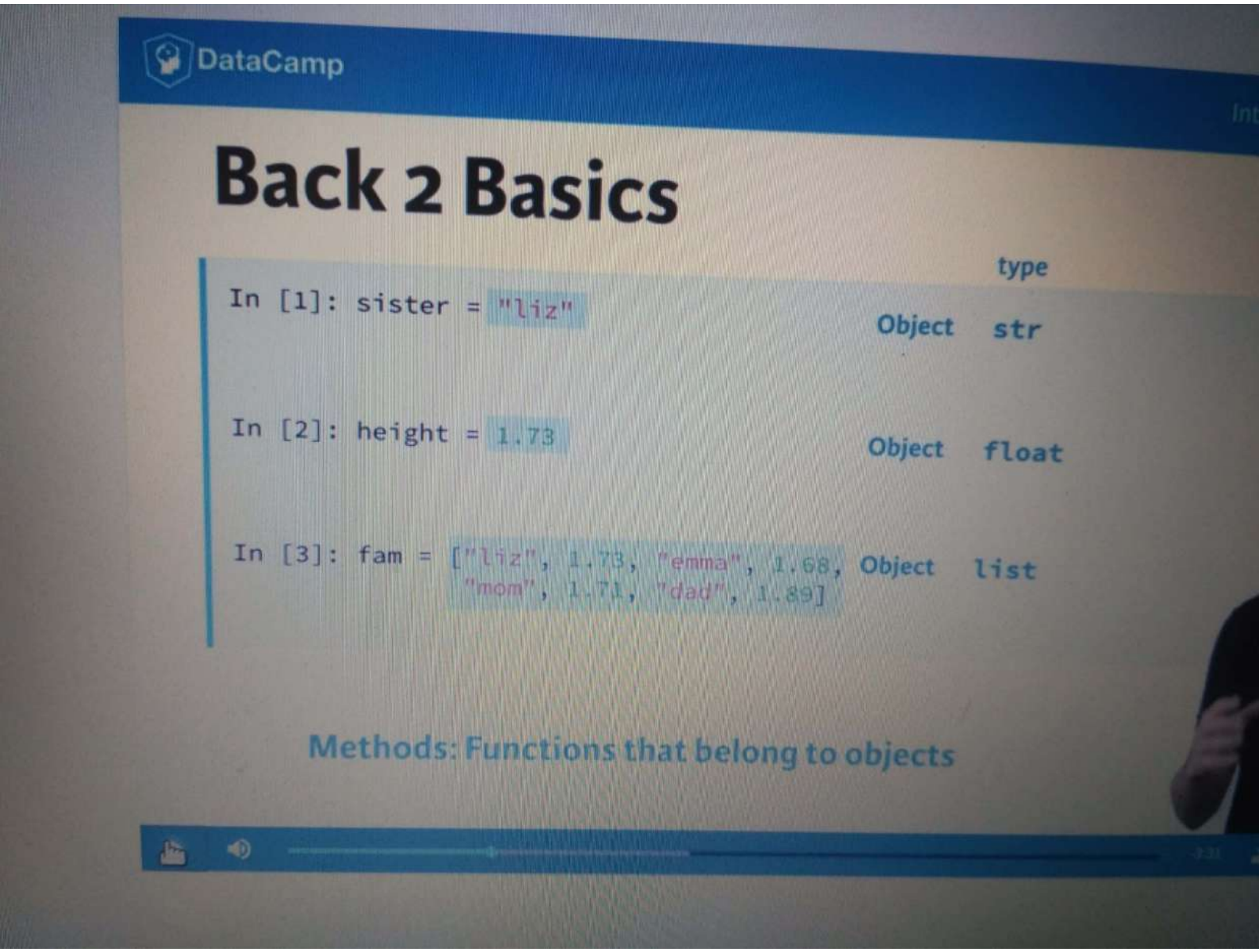
Incorrect Submission

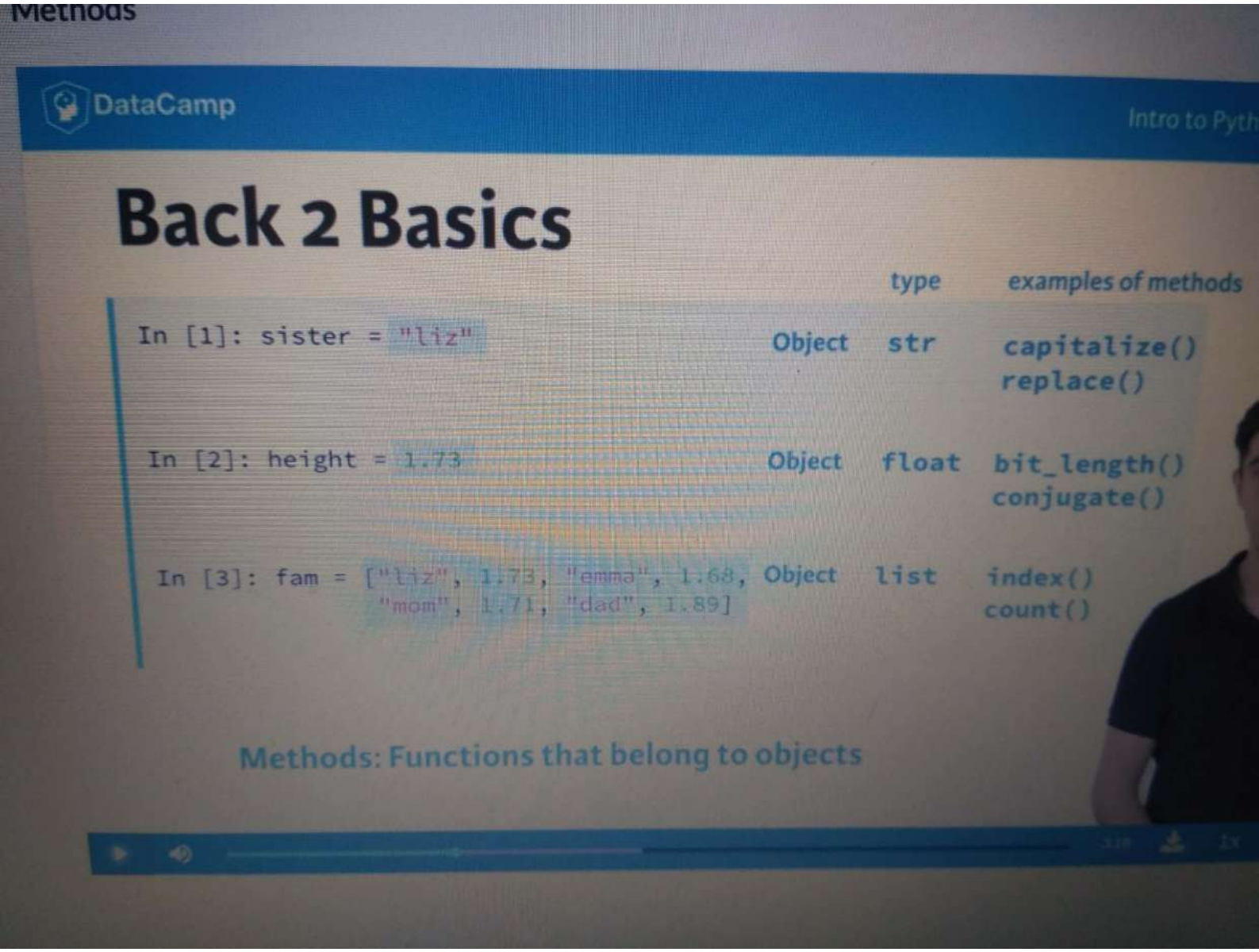
Incorrect. `[, imag]` shows that `imag` is an optional argument.

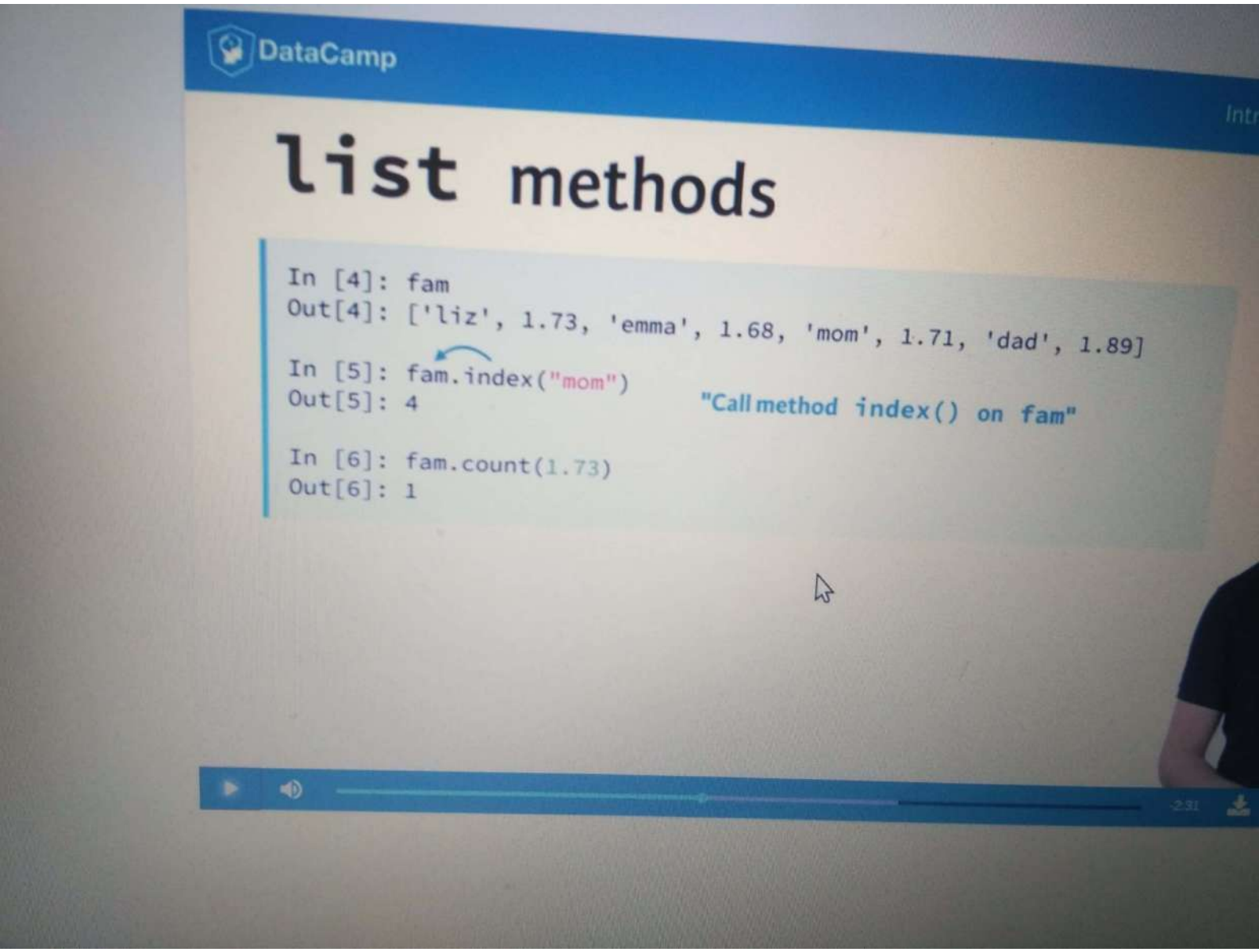
```
In [1]: help(comp
Help on class com

class complex(obj
 |  complex(real[,
 |
 |  Create a compl
 |      part.
 |  This is equival
 |
 |  Methods defined
 |
 |  __abs__(self, /)
 |      abs(self)
 |
 |  __add__(self, va
 |      Return self+v
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __divmod__(self, v
 |      Return divmod(
 |
 |  __eq__(self, value
 |      Return self==va
 |
 |  __float__(self, /)
 |      float(self)
```



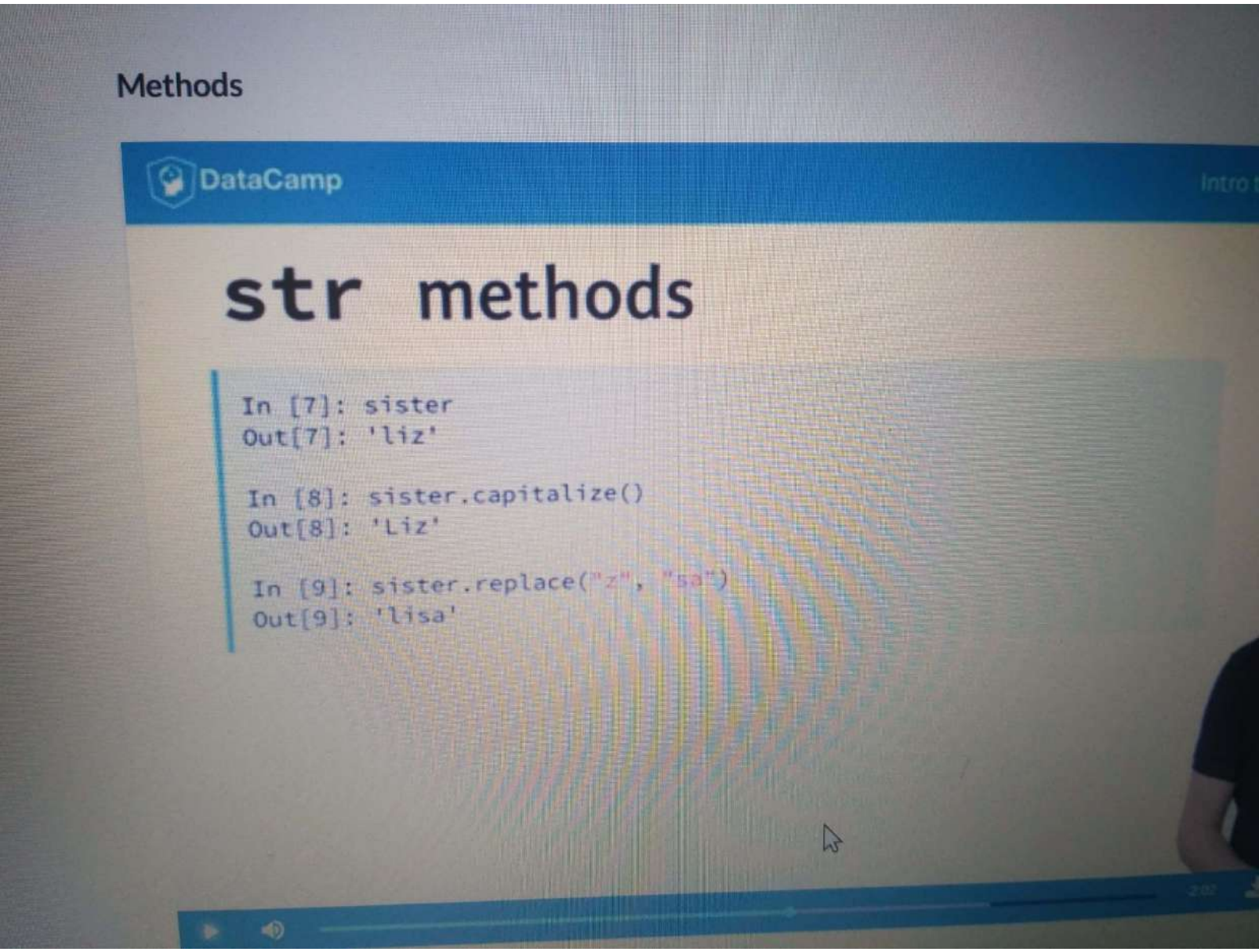




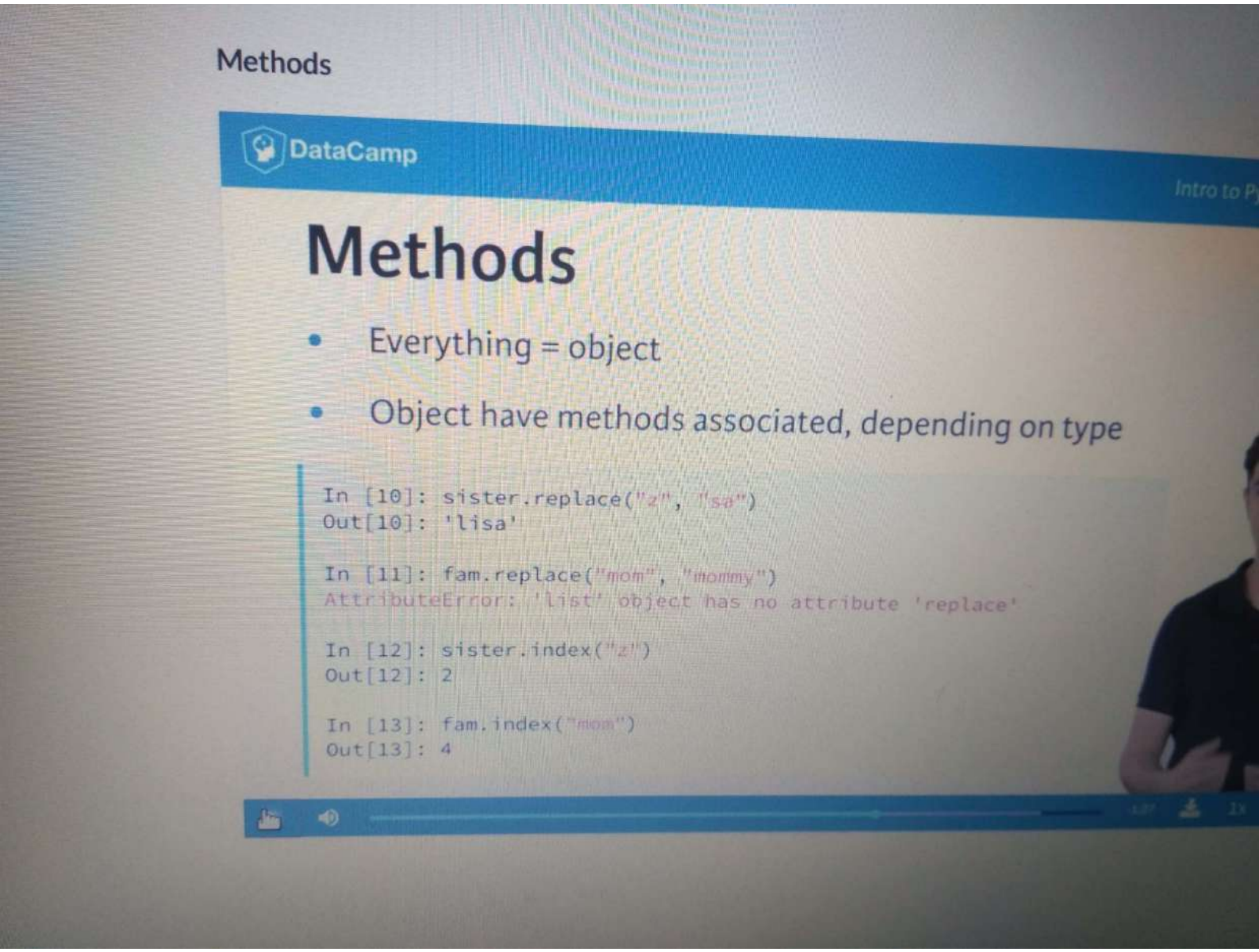




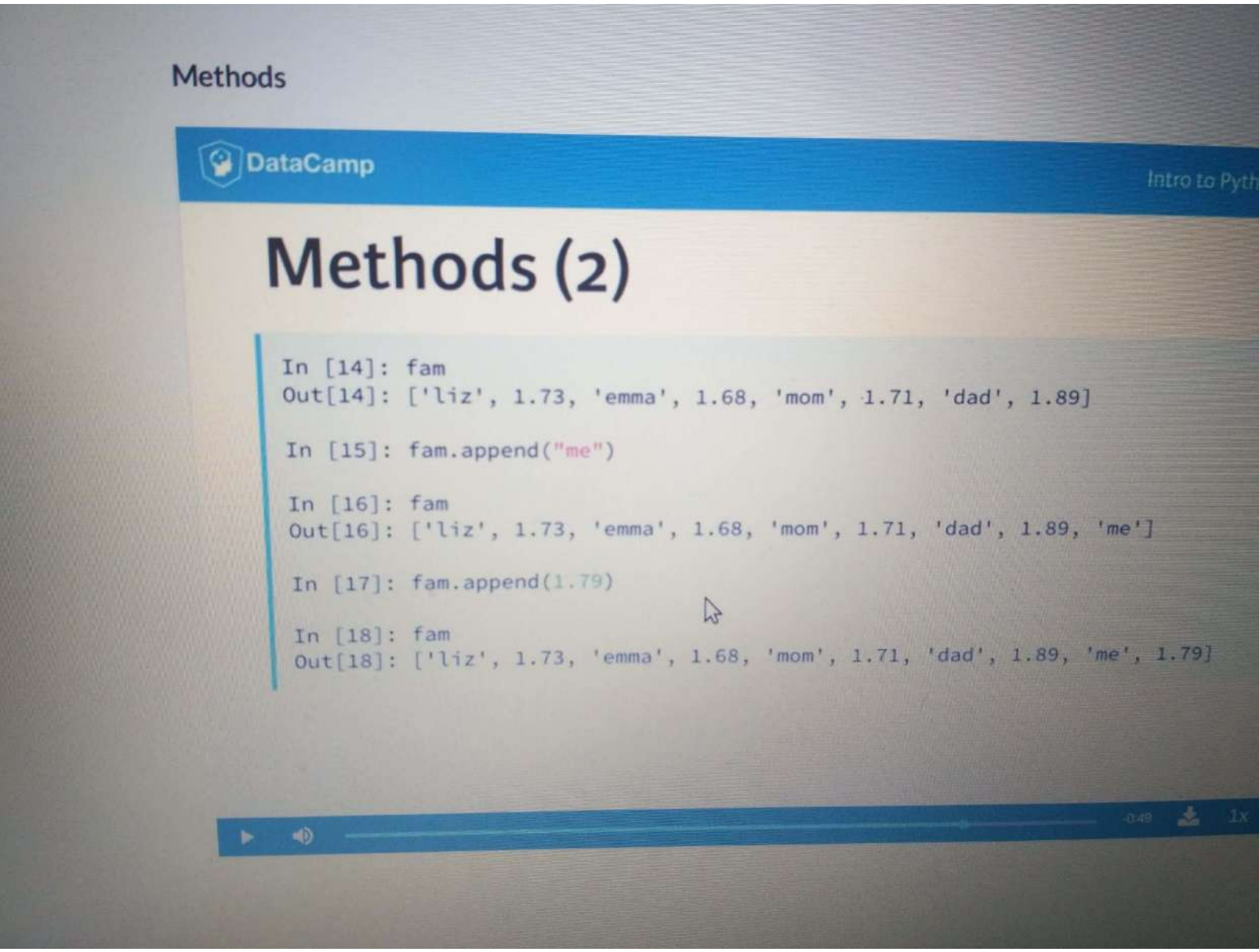
Methods

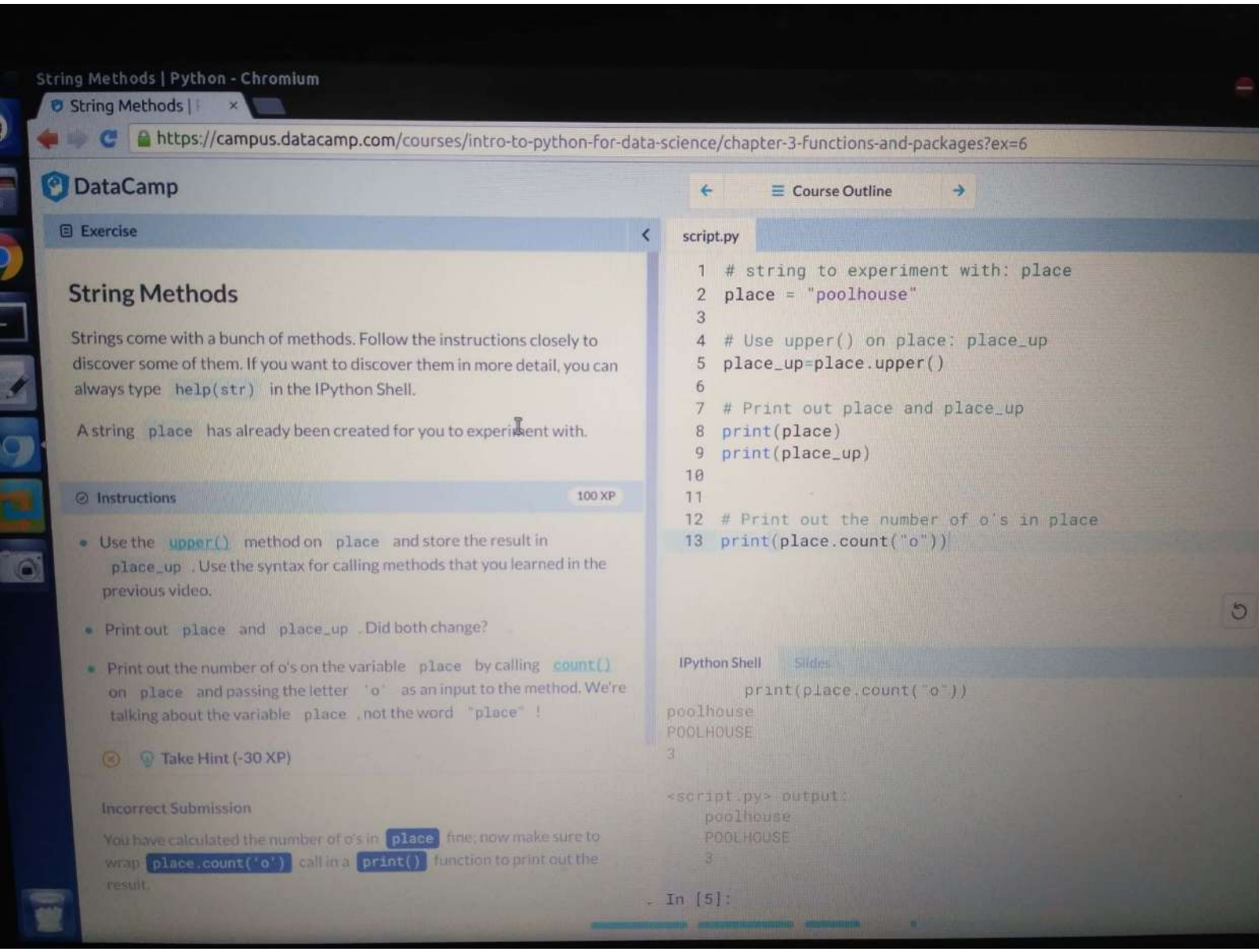


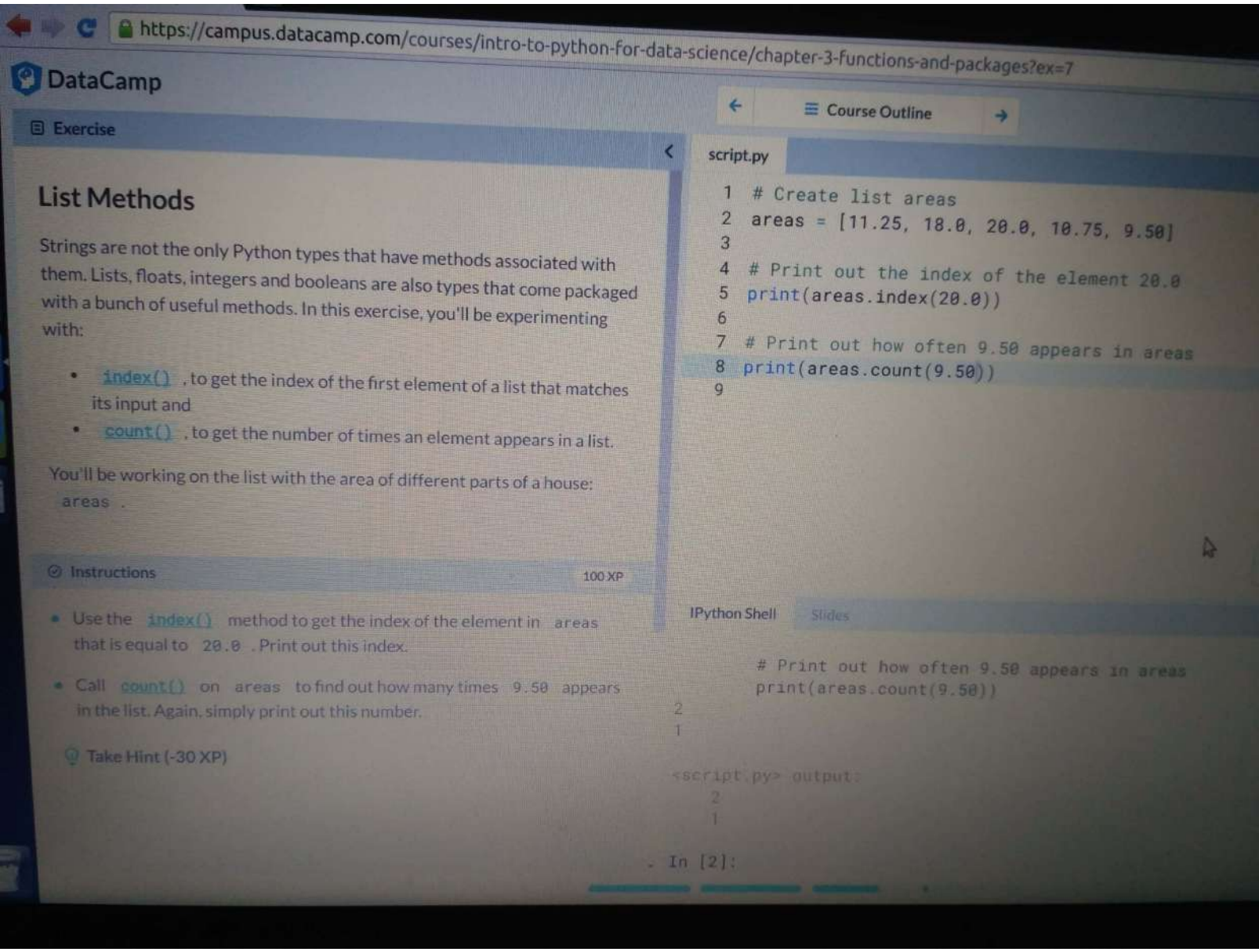
Methods

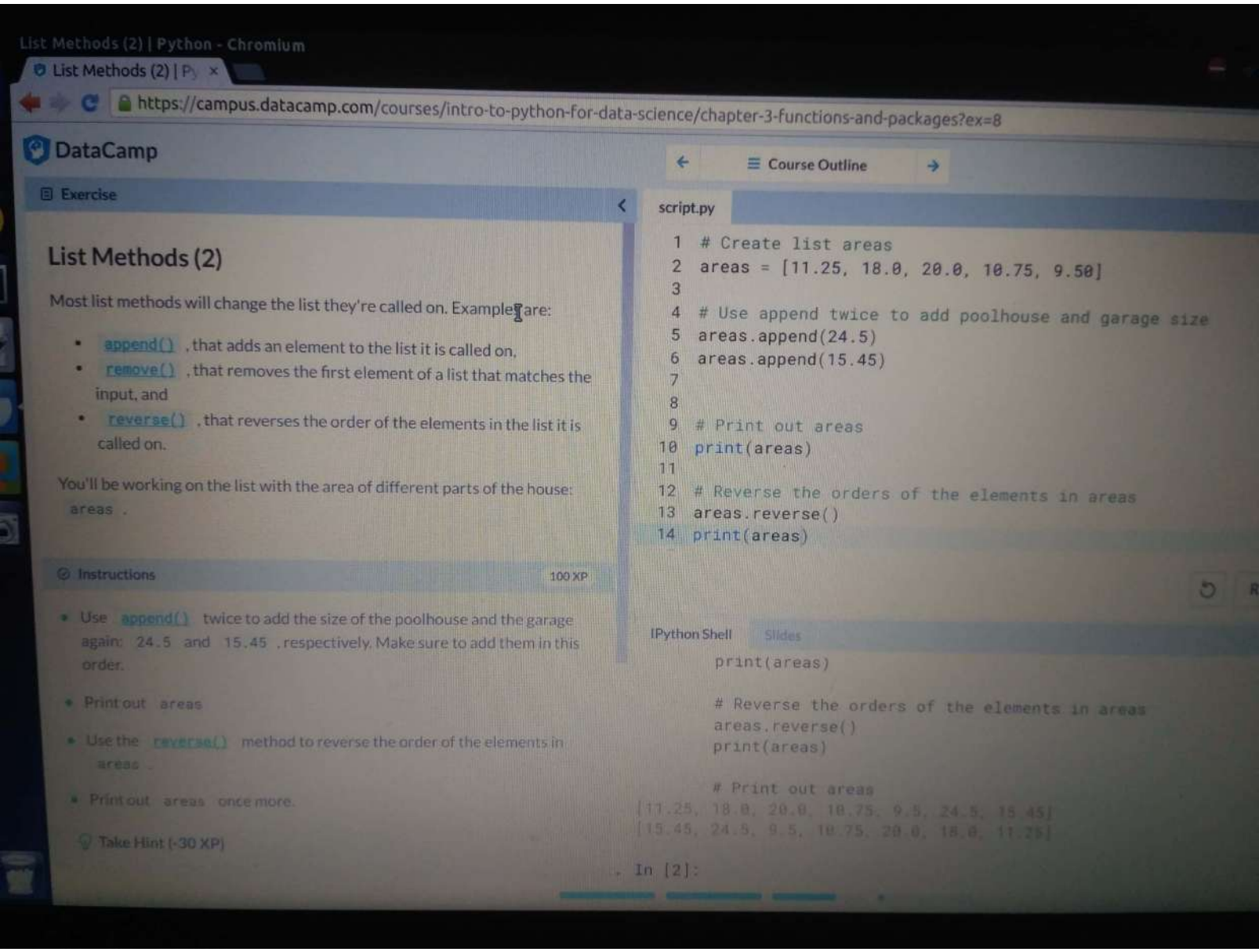


Methods

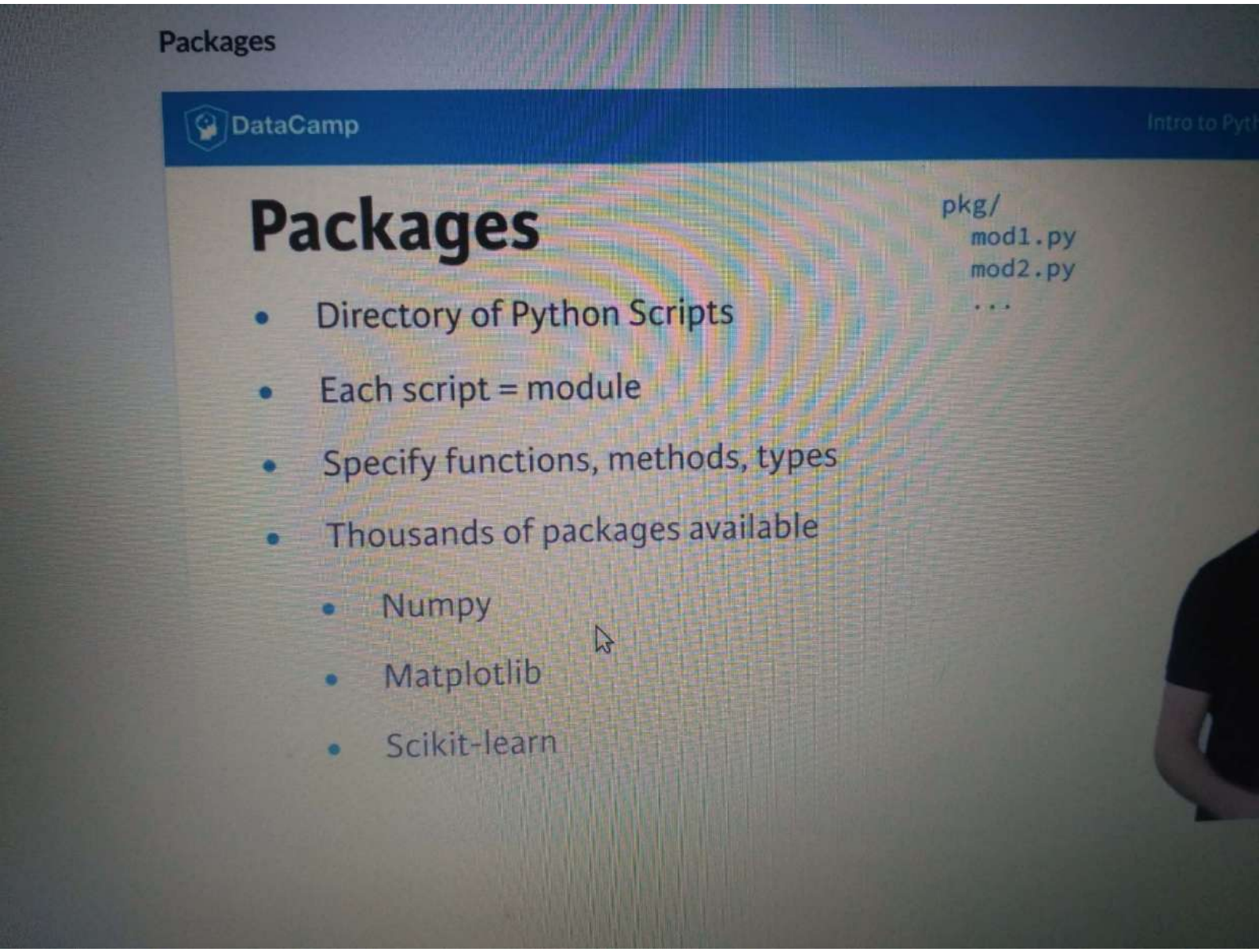


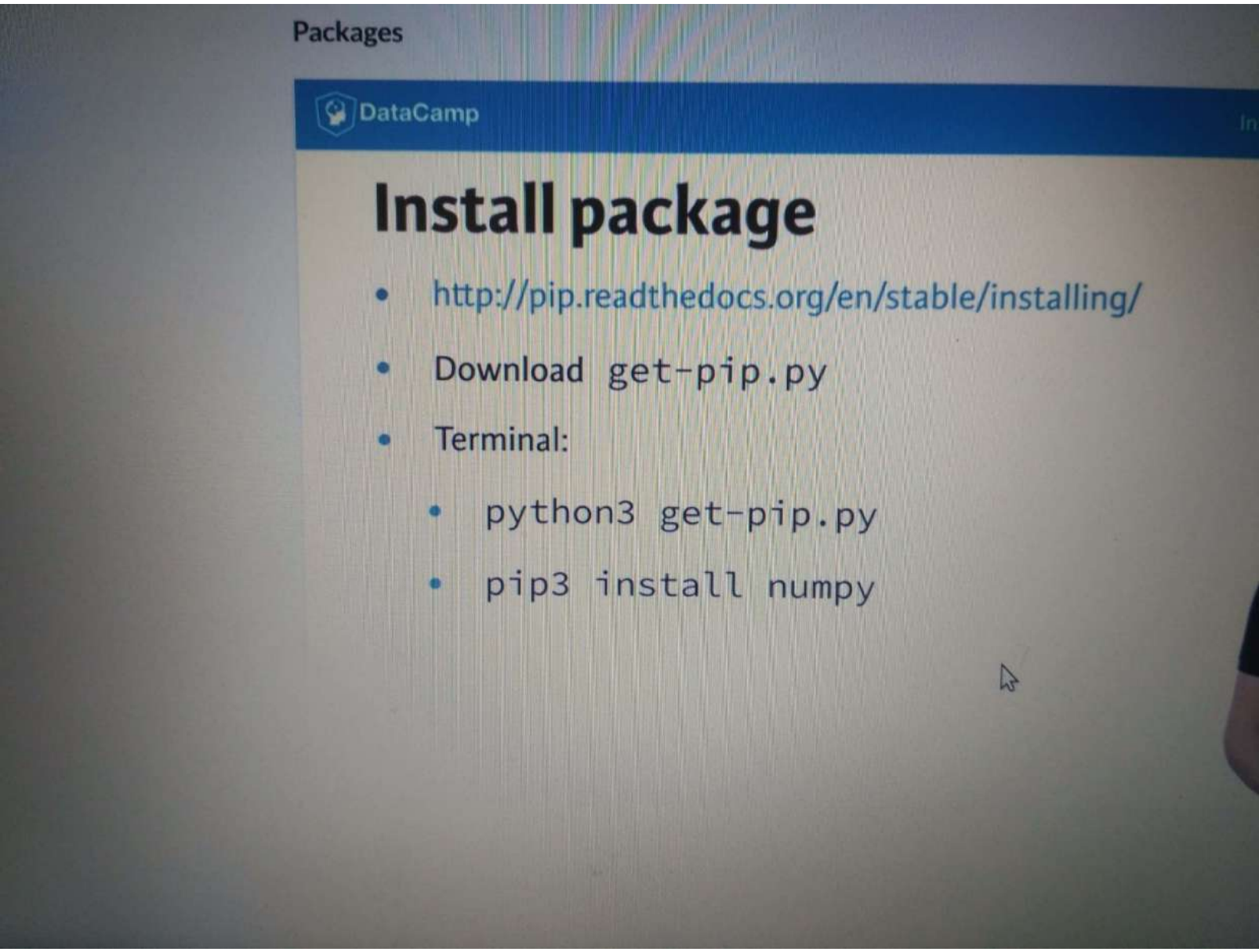


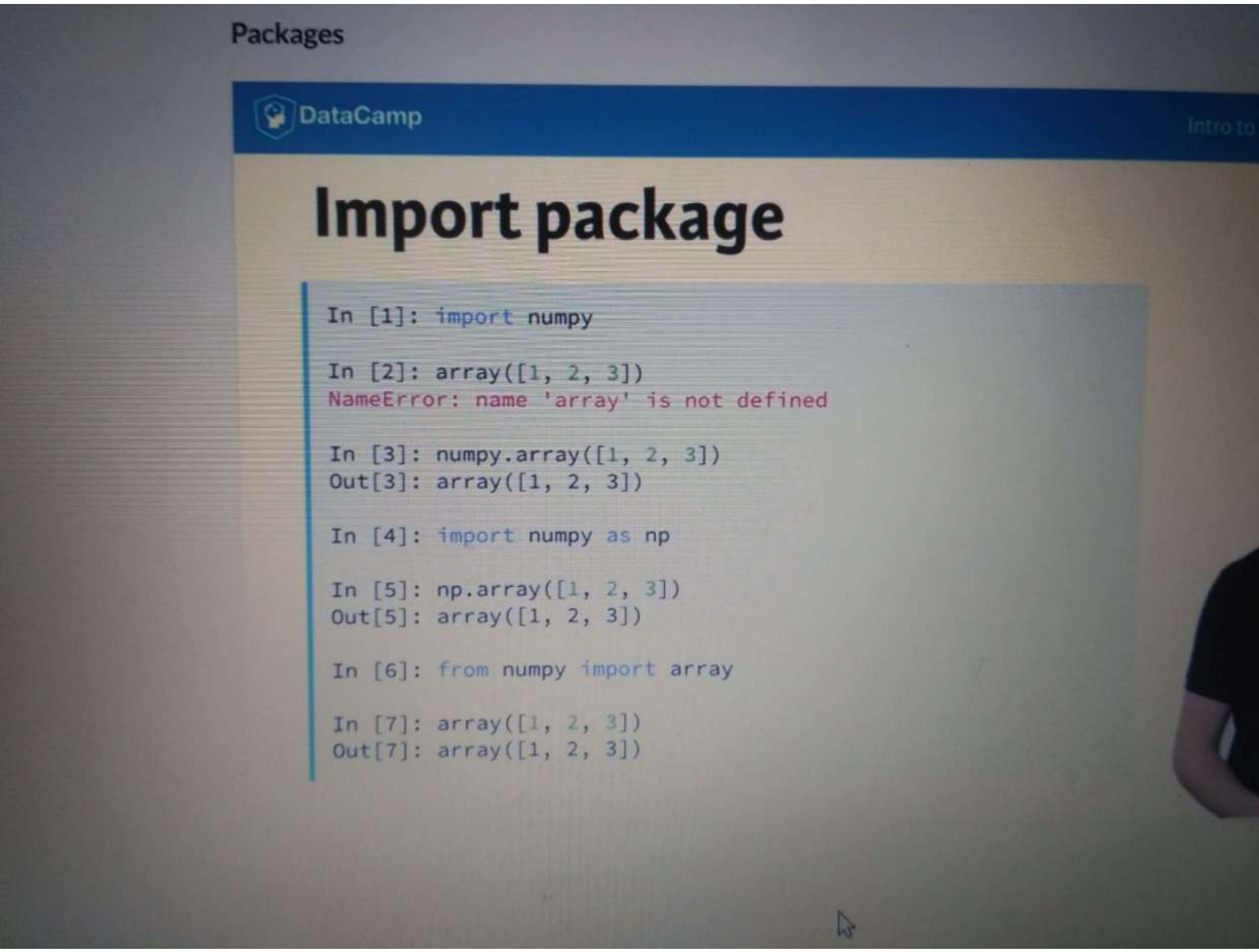




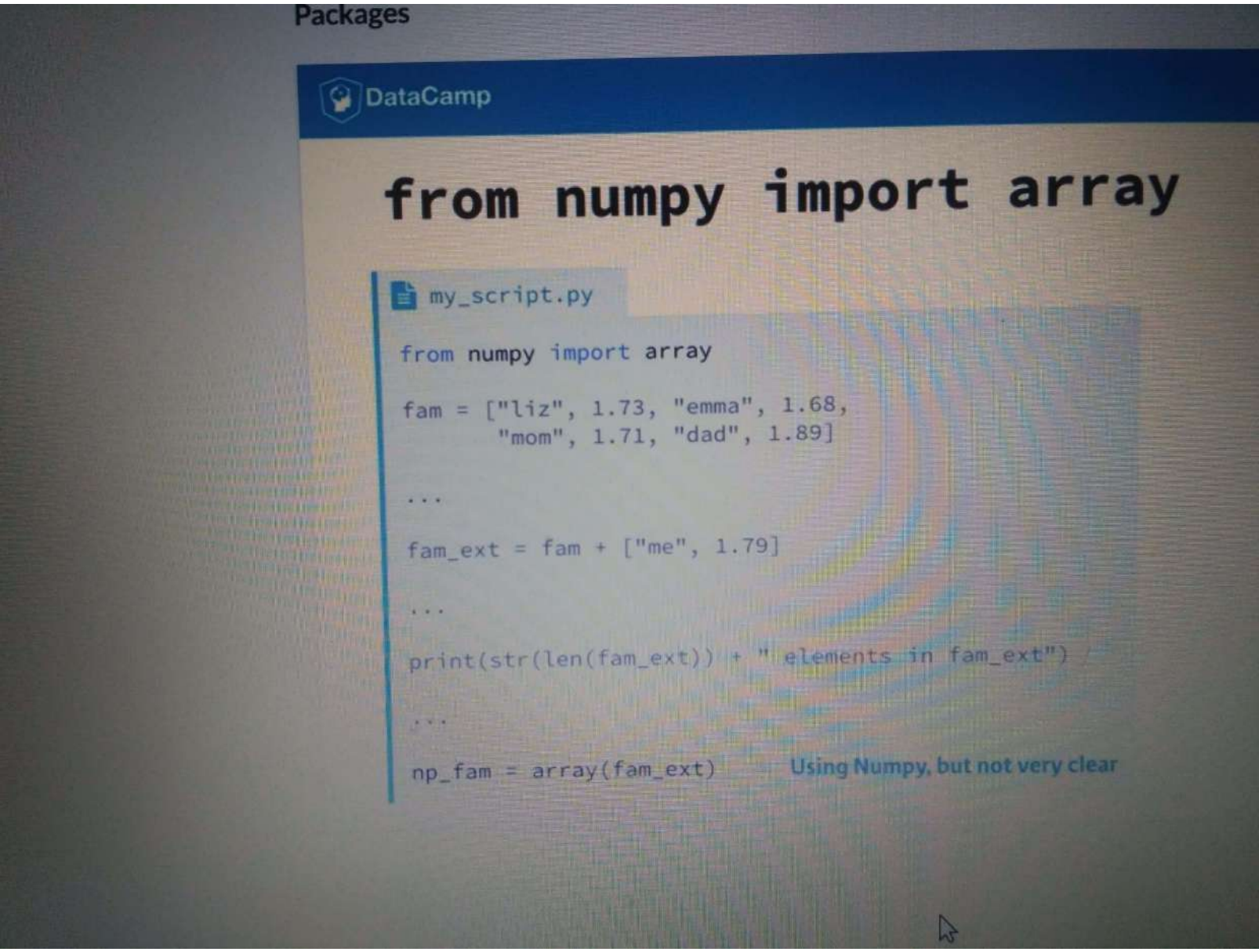
Packages







Packages

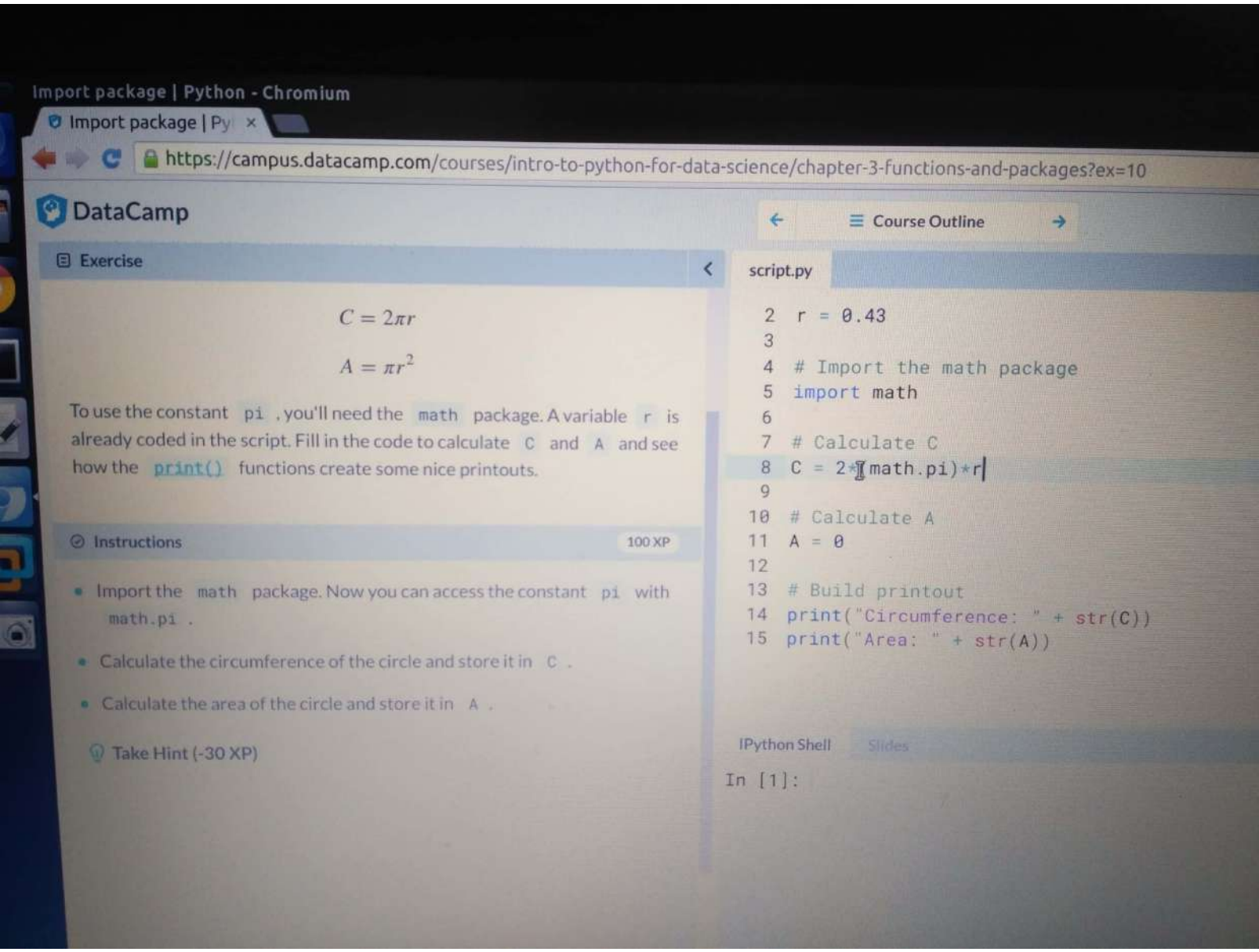


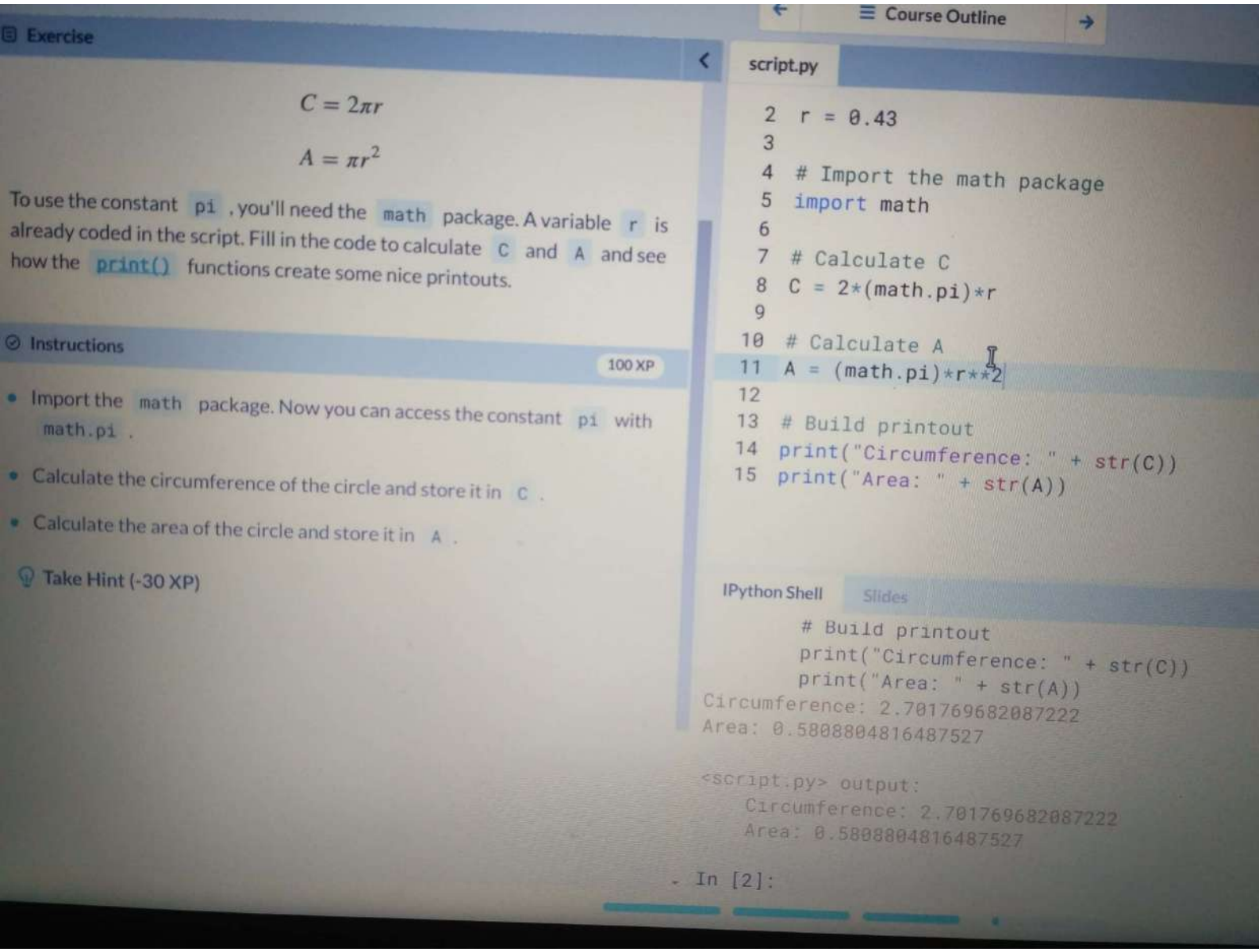
import numpy

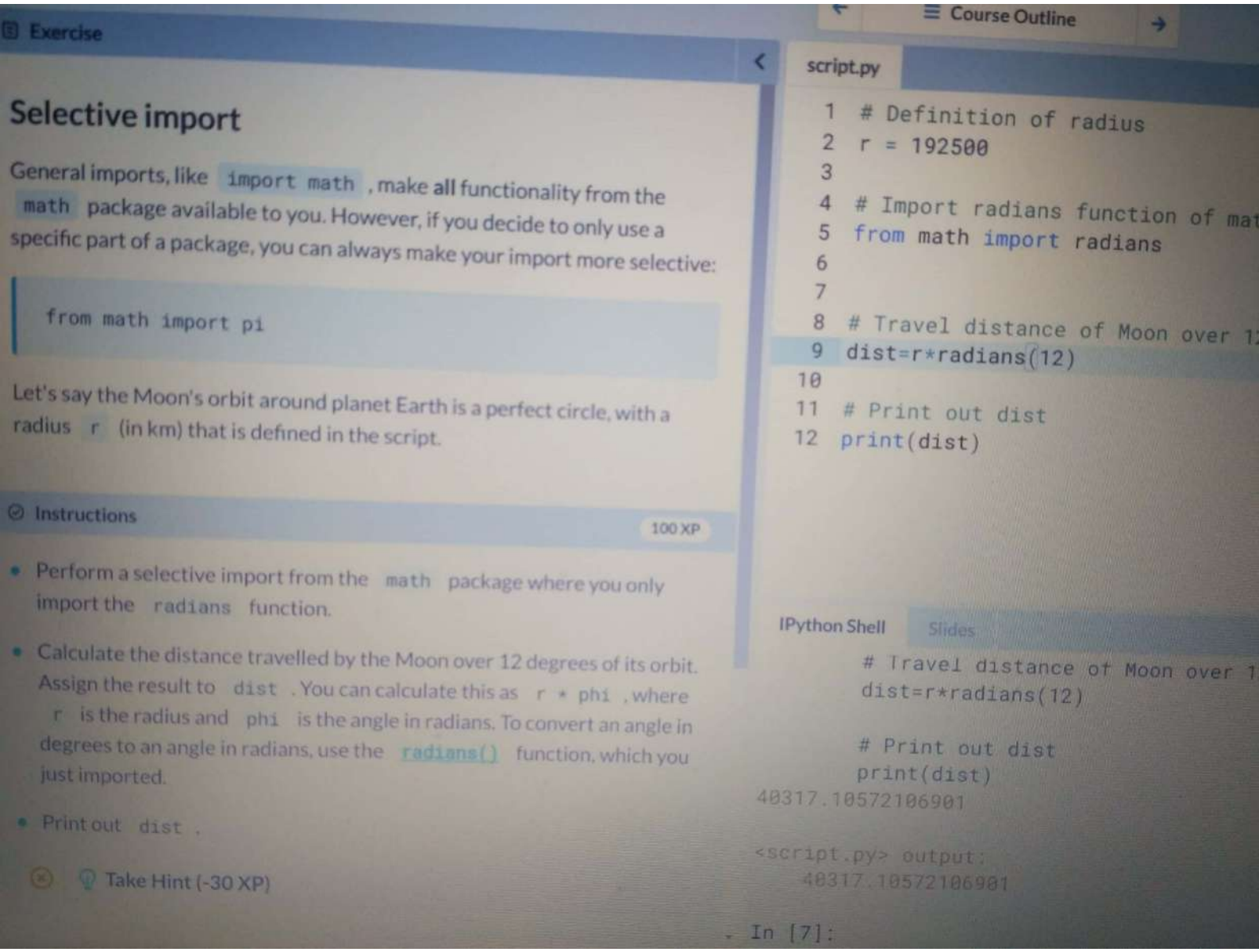
my_script.py

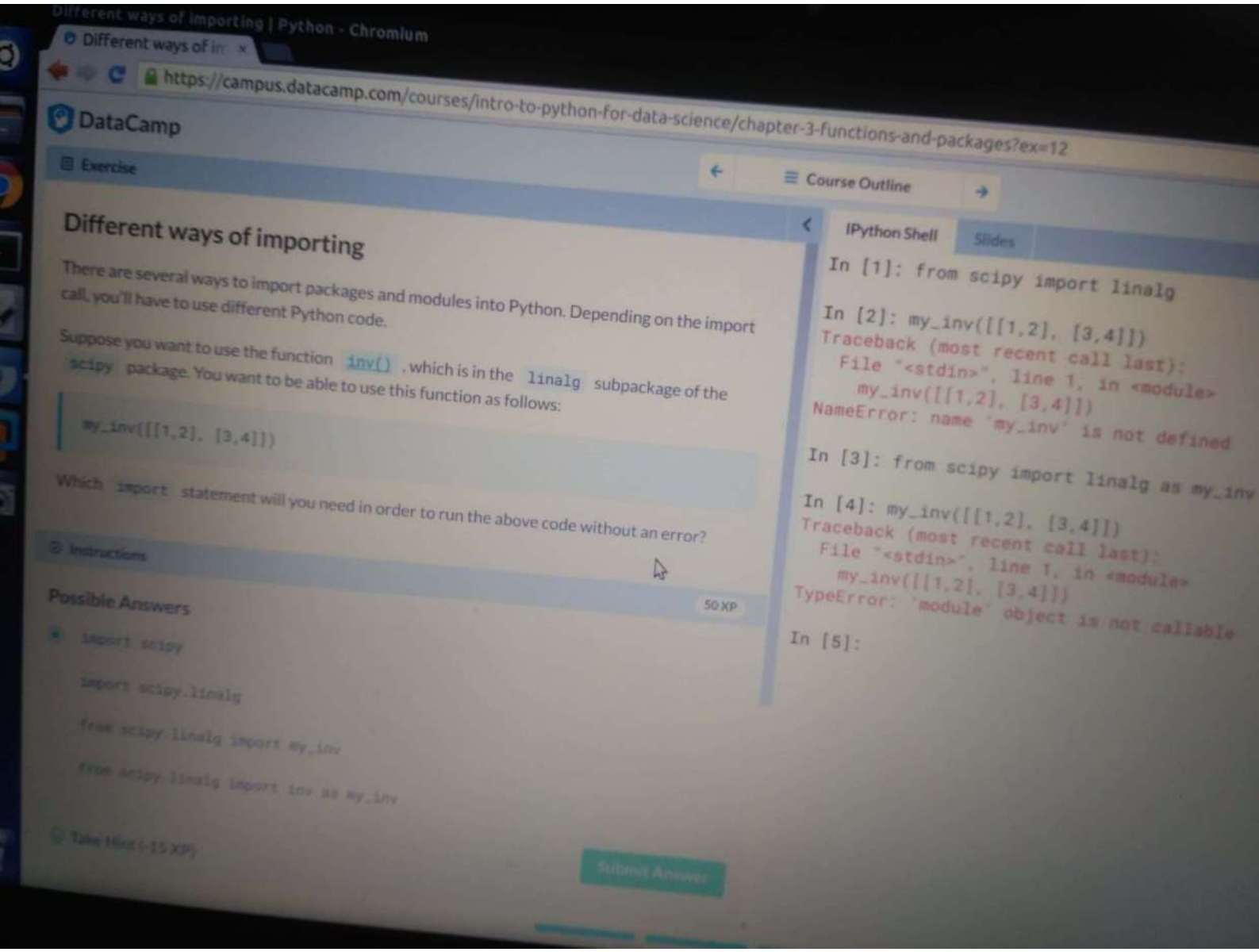
```
import numpy

fam = ["liz", 1.73, "emma", 1.68,
       "mom", 1.71, "dad", 1.89]

...

fam_ext = fam + ["me", 1.79]

...

print(str(len(fam_ext)) + " elements in fam_ext")

...

np_fam = numpy.array(fam_ext)
```

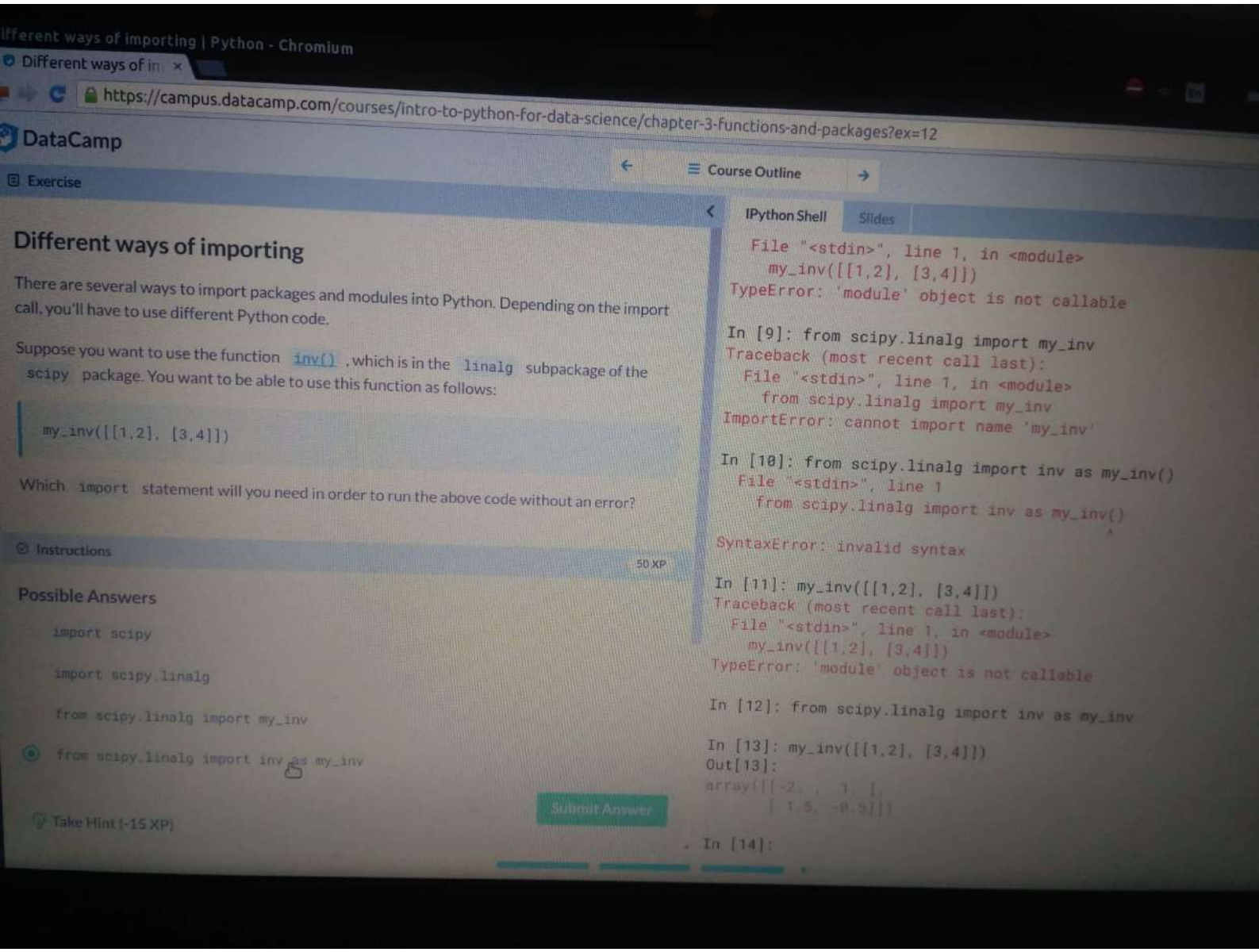
Clearly using Numpy











.com/courses/intro-to-python-for-data-science/chapter-4-numpy?ex=1

Course Outline

umPy

DataCamp

Illustration

```
In [1]: height = [1.73, 1.68, 1.71, 1.89, 1.79]

In [2]: height
Out[2]: [1.73, 1.68, 1.71, 1.89, 1.79]

In [3]: weight = [65.4, 59.2, 63.6, 88.4, 68.7]

In [4]: weight
Out[4]: [65.4, 59.2, 63.6, 88.4, 68.7]

In [5]: weight / height ** 2
TypeError: unsupported operand type(s) for **: 'list' and 'int'
```

Chromium

ps://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-4-numpy?ex=1

Course Outline

NumPy

DataCamp

Intro

Solution: NumPy

- Numeric Python
- Alternative to Python List: NumPy Array
- Calculations over entire arrays
- Easy and Fast

npus.datacamp.com/courses/intro-to-python-for-data-science/chapter-4-numpy?ex=1

Course Outline

NumPy

DataCamp

Intro to Pytho

Solution: NumPy

- Numeric Python
- Alternative to Python List: NumPy Array
- Calculations over entire arrays
- Easy and Fast
- Installation
  - In the terminal: `pip3 install numpy`

NumPy

```
In [6]: import numpy as np

In [7]: np_height = np.array(height)

In [8]: np_height
Out[8]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [9]: np_weight = np.array(weight)

In [10]: np_weight
Out[10]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])

In [11]: bmi = np_weight / np_height ** 2

In [12]: bmi
Out[12]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```

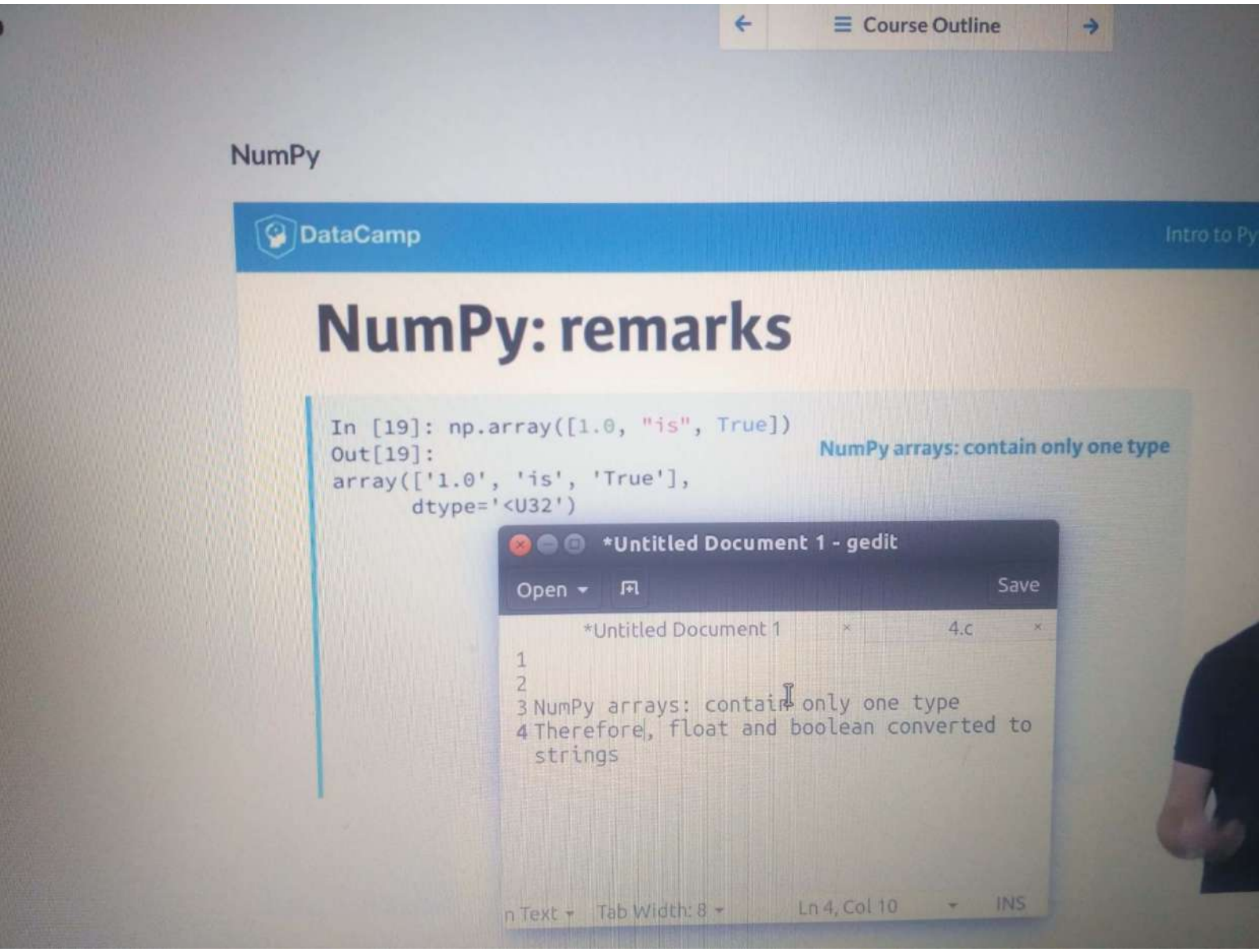
Element-wise calcula

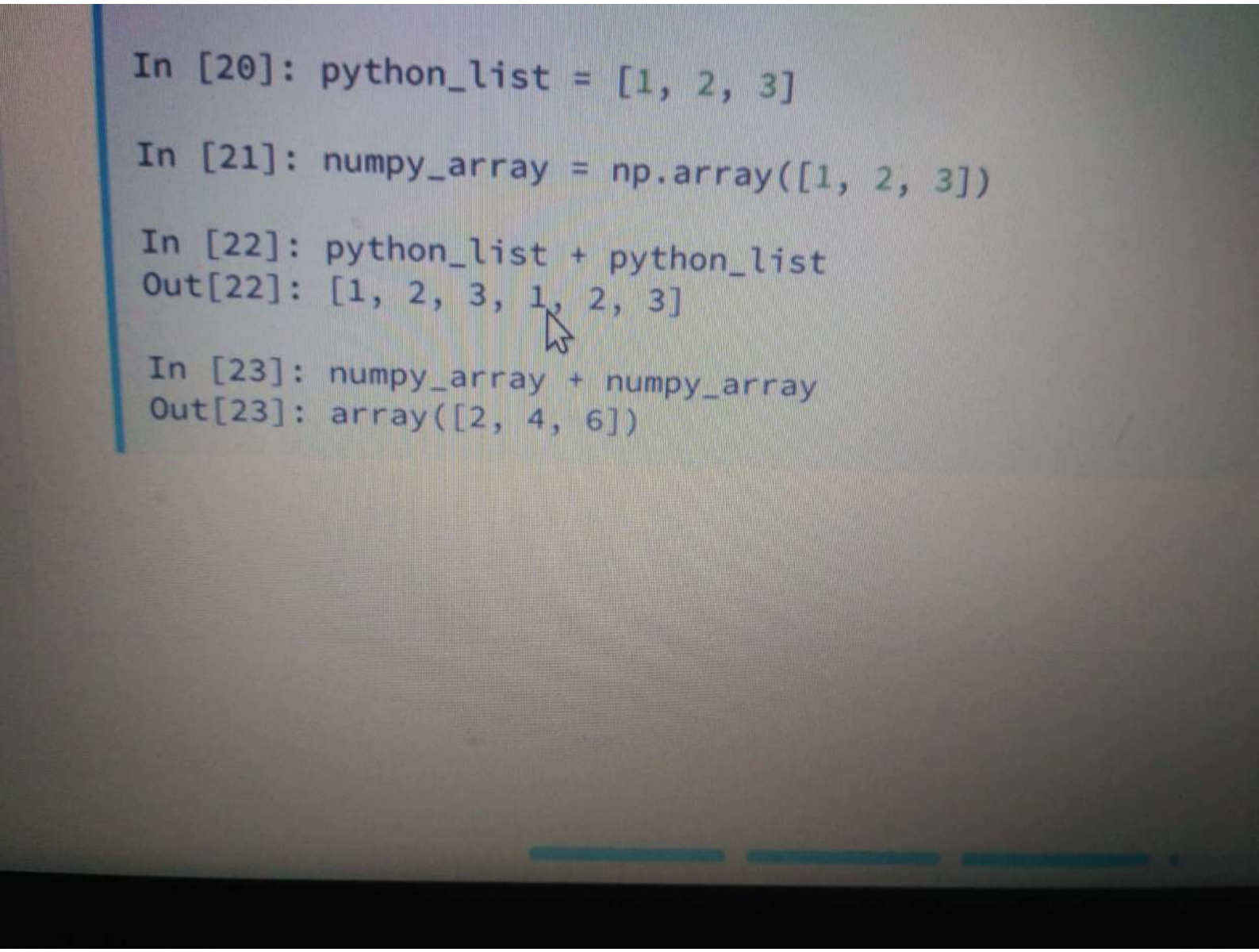
NumPy

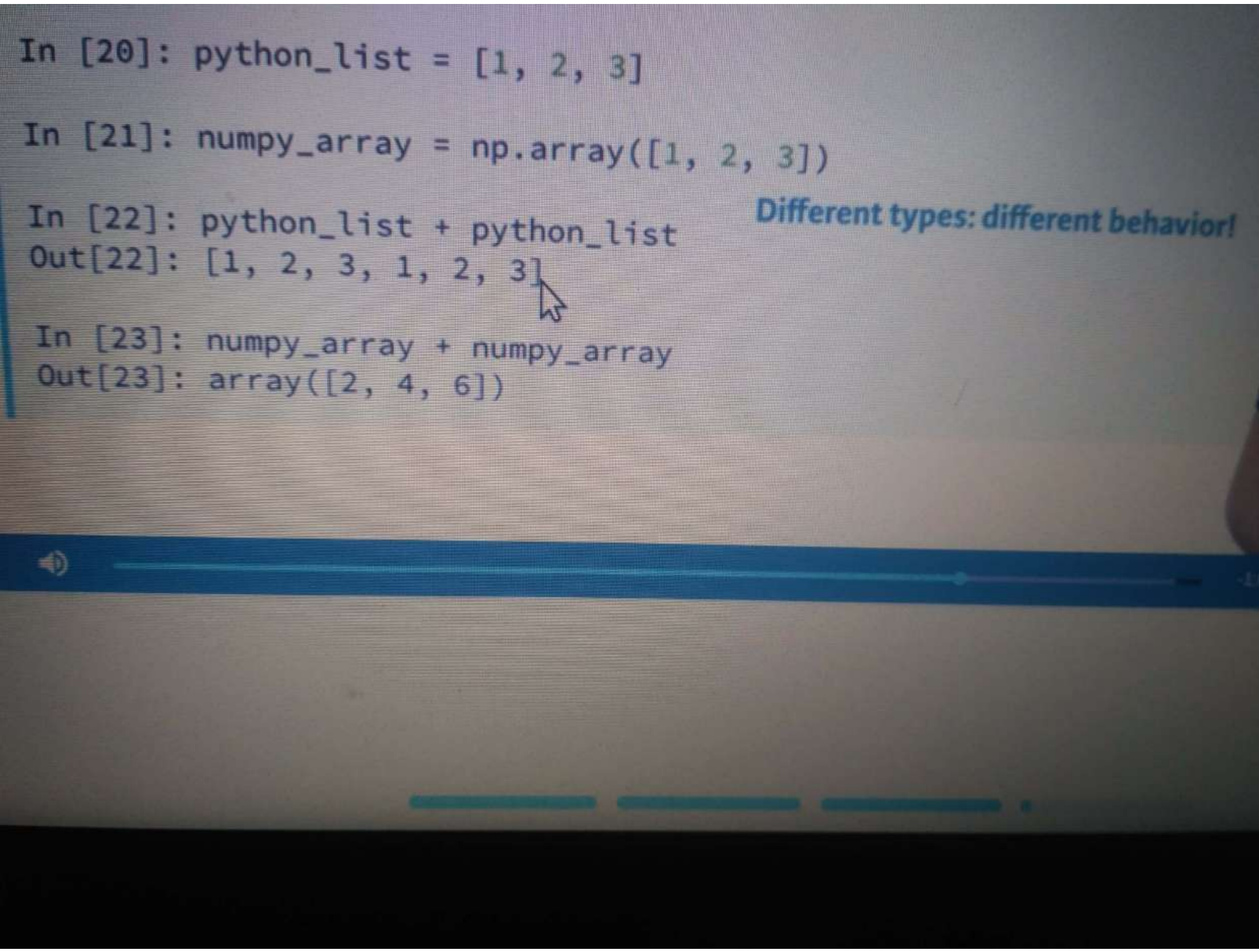
```
In [6]: import numpy as np

In [7]: np_height = np.array(height)

In [8]: np_height
Out[8]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [9]: np_weight = np.array(weight)

In [10]: np_weight
Out[10]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])

In [11]: bmi = np_weight / np_height ** 2

In [12]: bmi
Out[12]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```

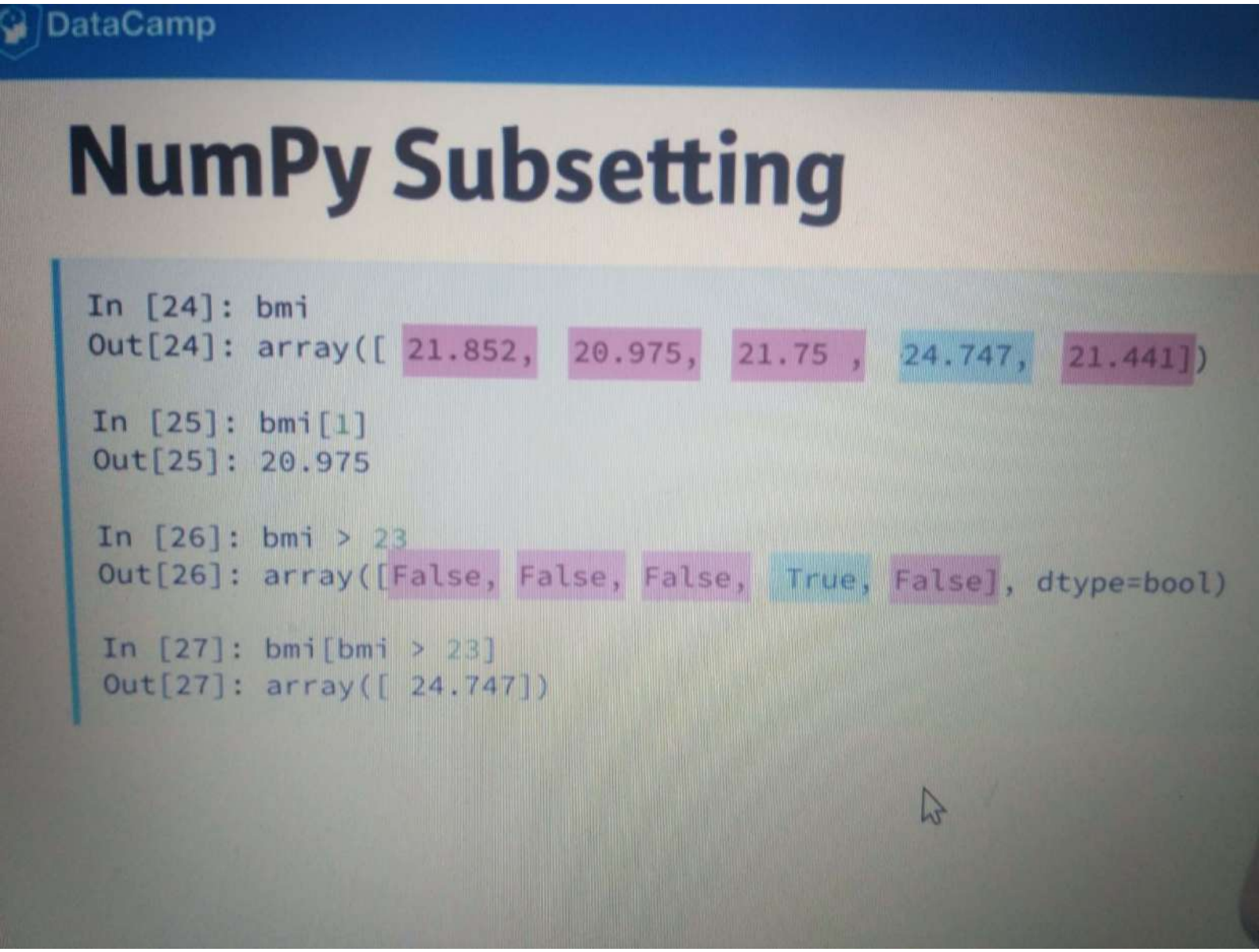
Element-wise calculation

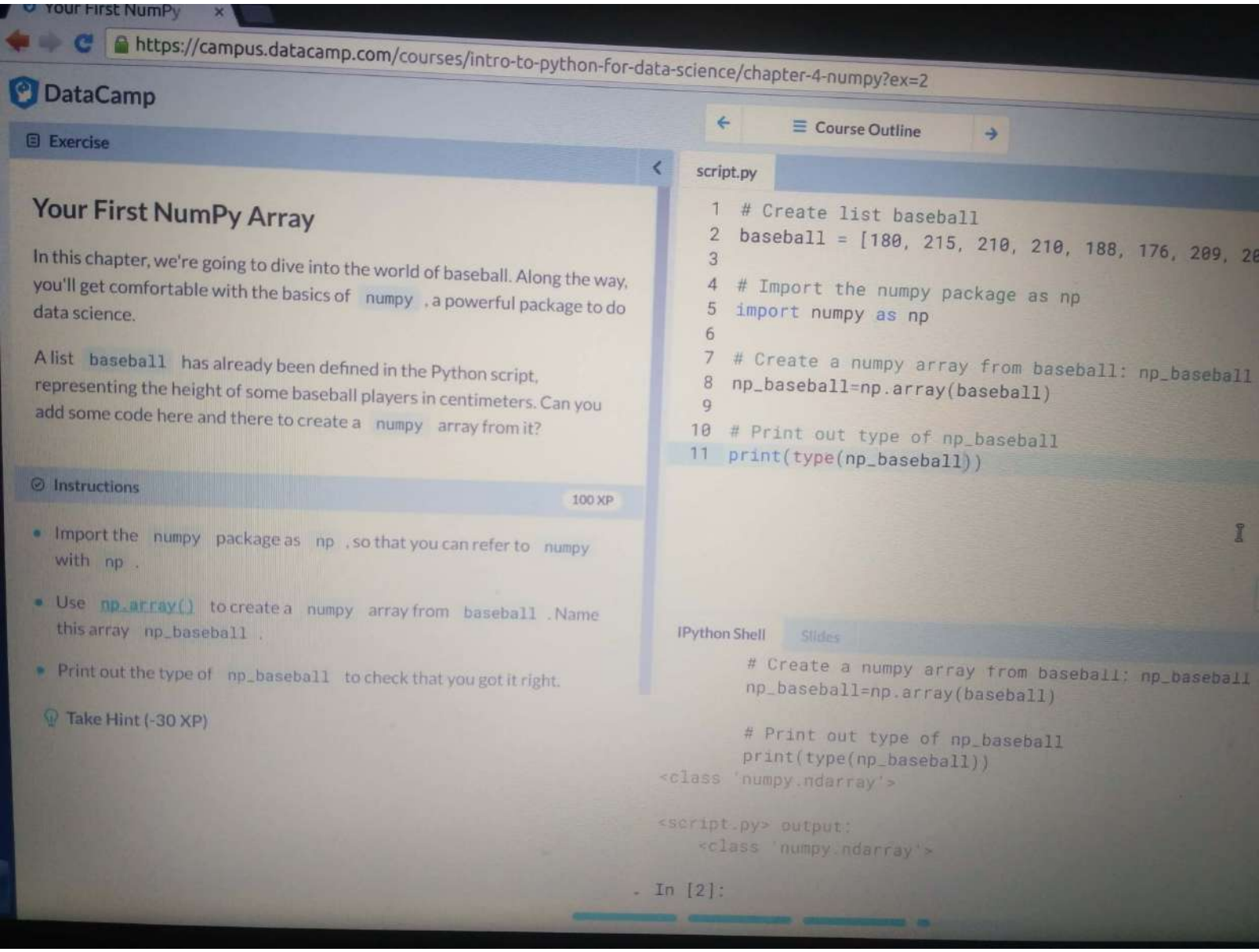
= 65.5/1.73 ** 2

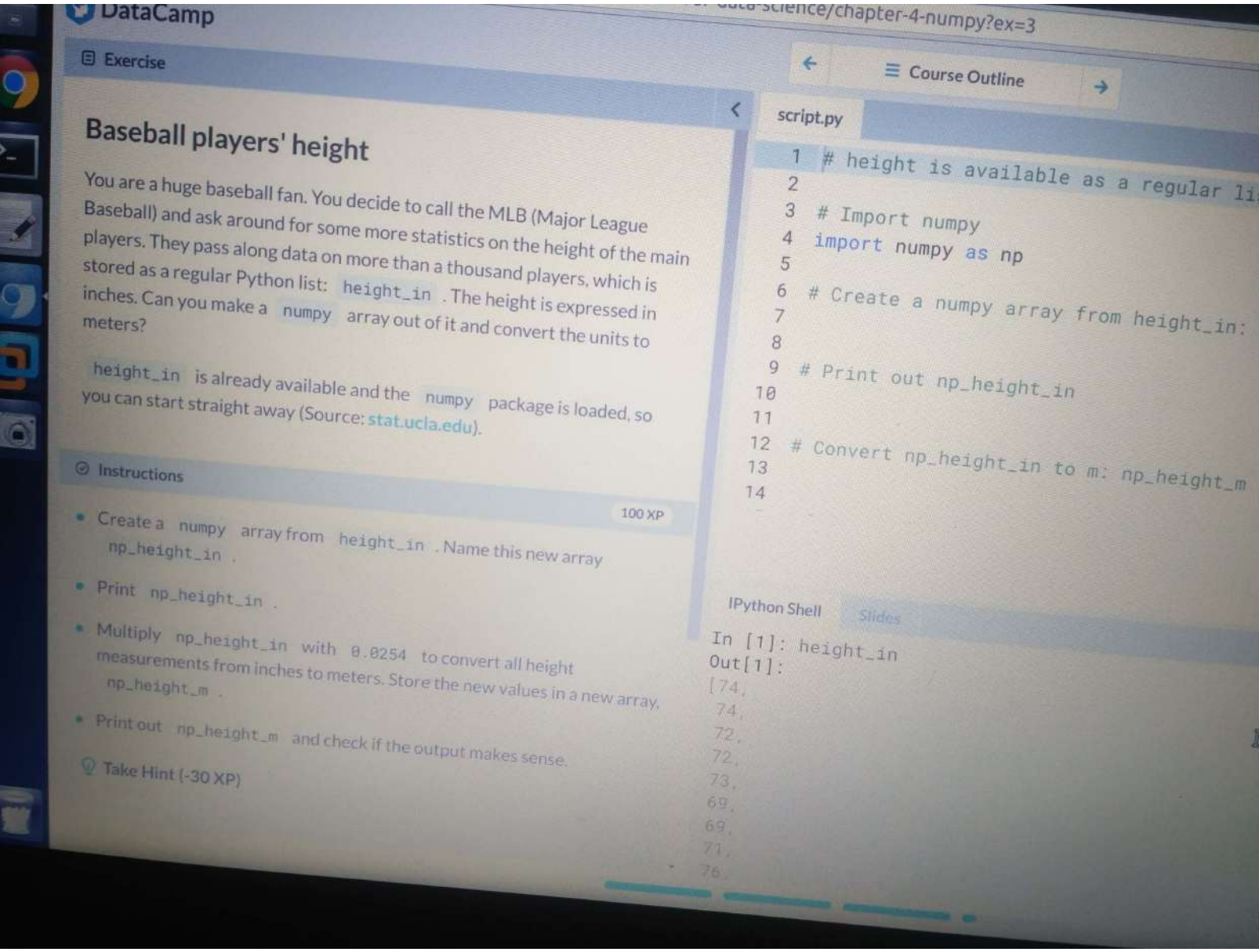


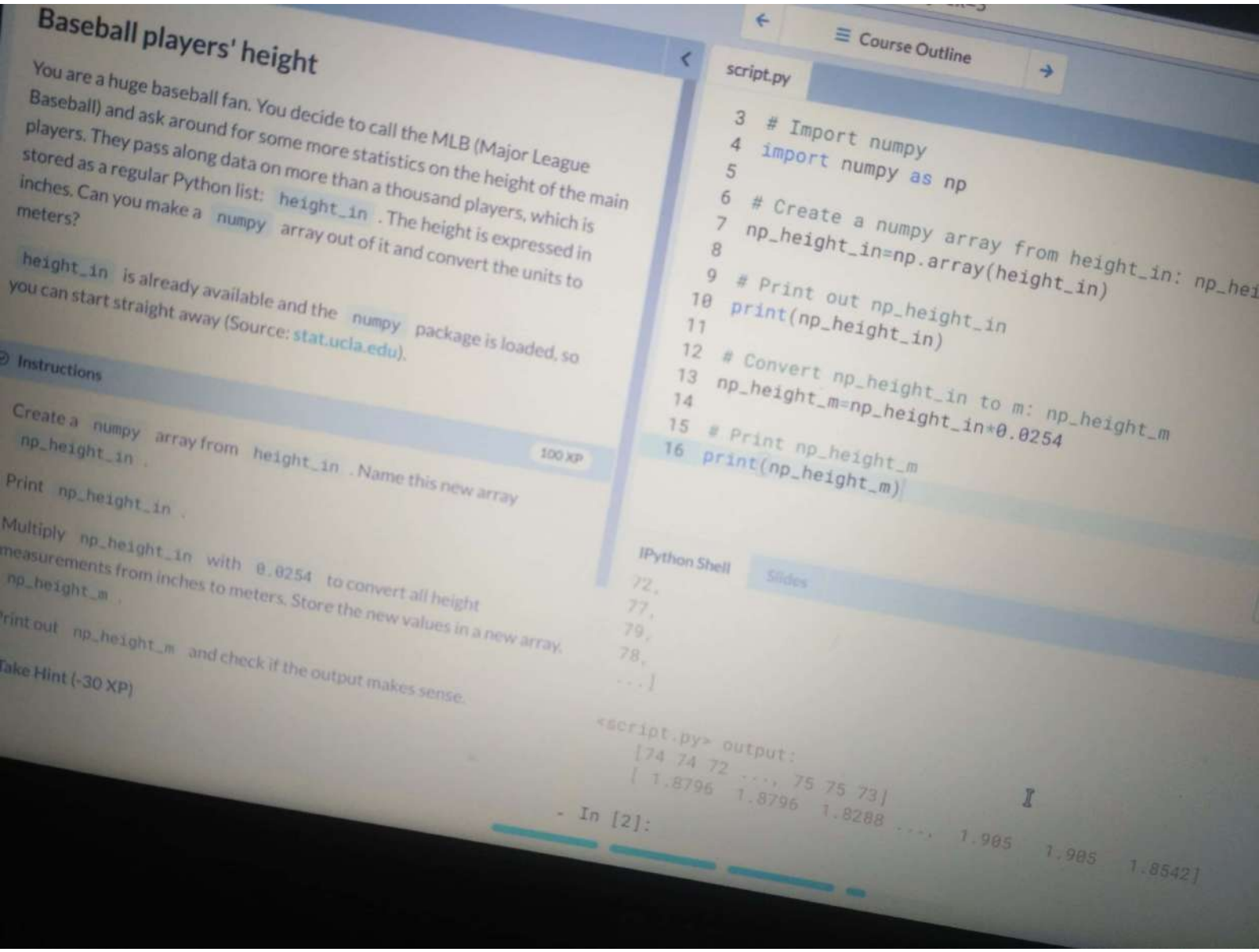












Baseball player's BMI

The MLB also offers to let you analyze their weight data. Again, both are available as regular Python lists: `height_in` and `weight`. `height_in` is in inches and `weight_lb` is in pounds.

It's now possible to calculate the BMI of each baseball player. Python code to convert `height_in` to a `numpy` array with the correct units is already available in the workspace. Follow the instructions step by step and finish the game!

Instructions

100 XP

- Create a `numpy` array from the `weight_lb` list with the correct units. Multiply by `0.453592` to go from pounds to kilograms. Store the resulting `numpy` array as `np_weight_kg`.
- Use `np_height_m` and `np_weight_kg` to calculate the BMI of each player. Use the following equation:

$$\text{BMI} = \frac{\text{weight(kg)}}{\text{height(m)}^2}$$

  Save the resulting `numpy` array as `bmi`.
- Print out `bmi`.

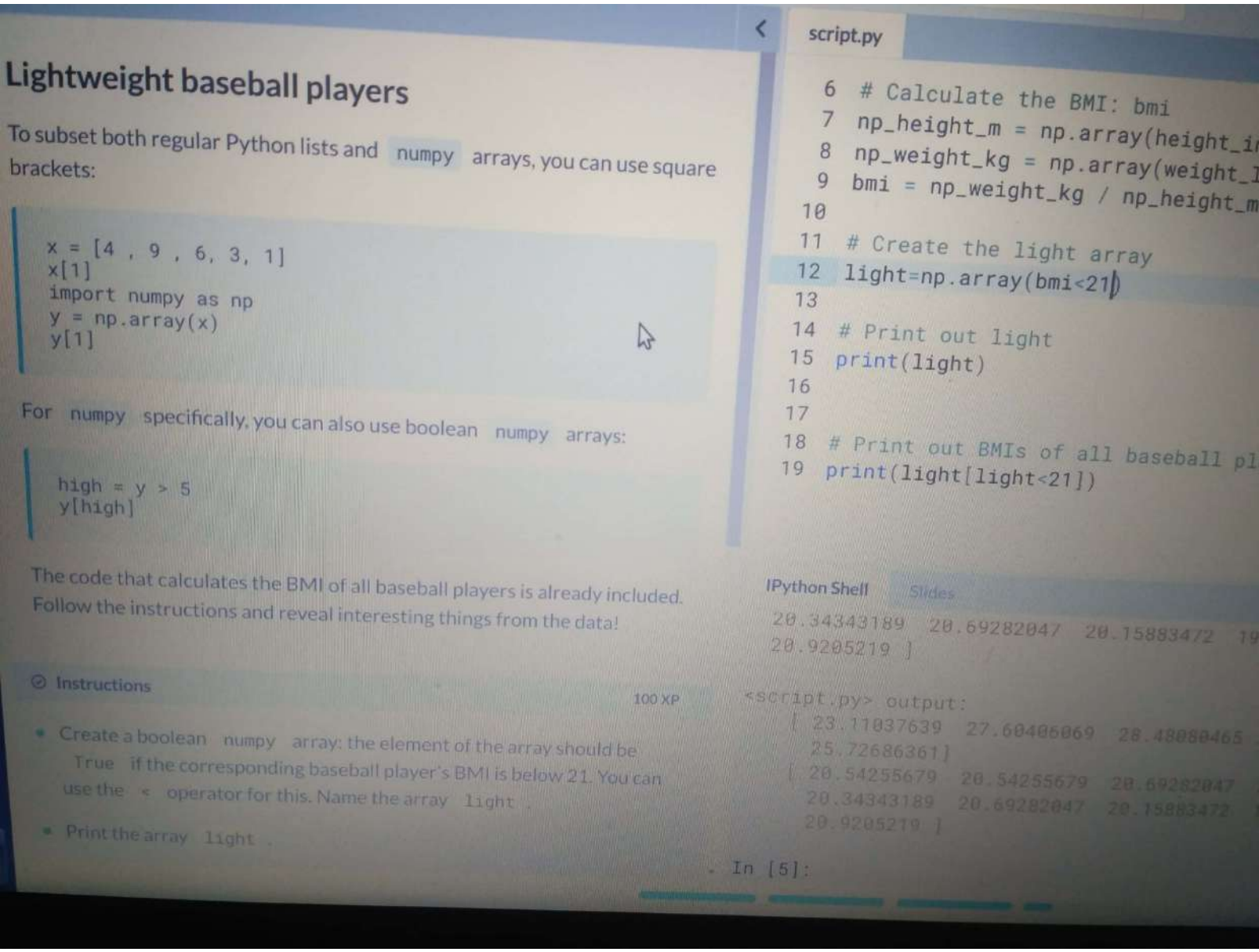
Take Hint (-30 XP)

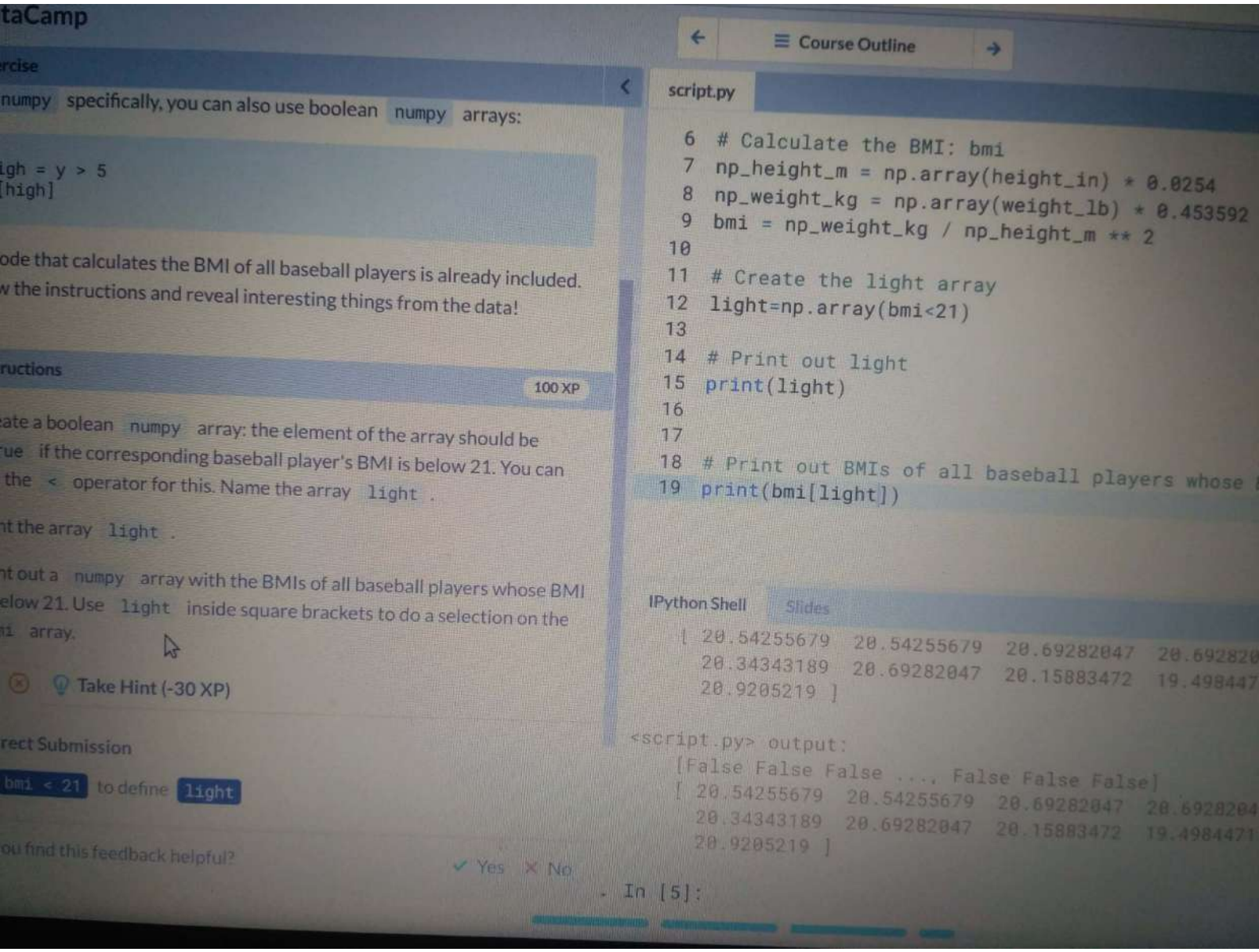
script.py

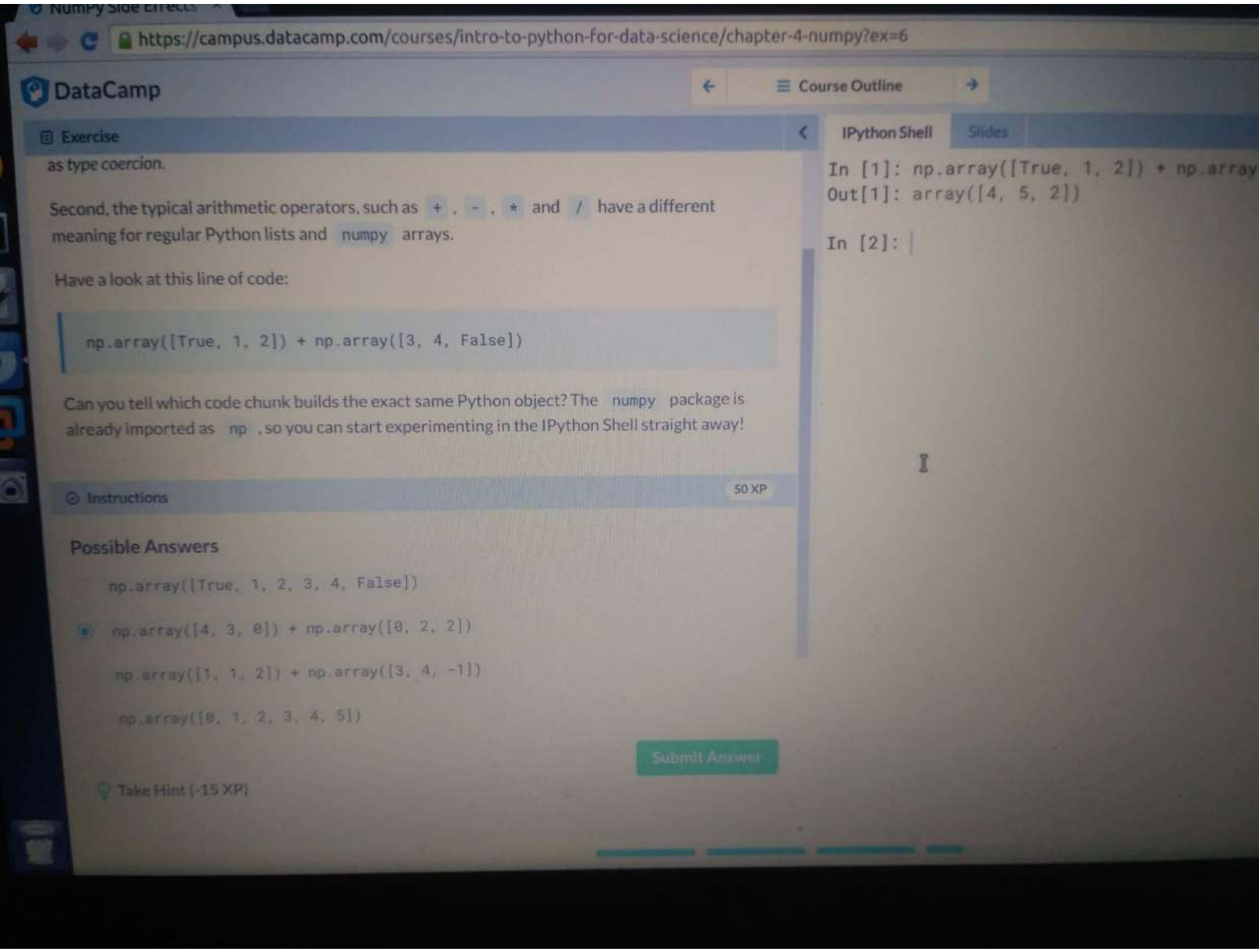
```
# Import numpy
import numpy as np

# Create array from height_in with metric units:
np_height_m = np.array(height_in) * 0.0254

# Create array from weight_lb with metric units:
np_weight_kg=np.array(weight_lb)*0.453592

# Calculate the BMI: bmi
bmi=np_weight_kg/np_height_m**2

# Print out bmi
print(bmi)
```

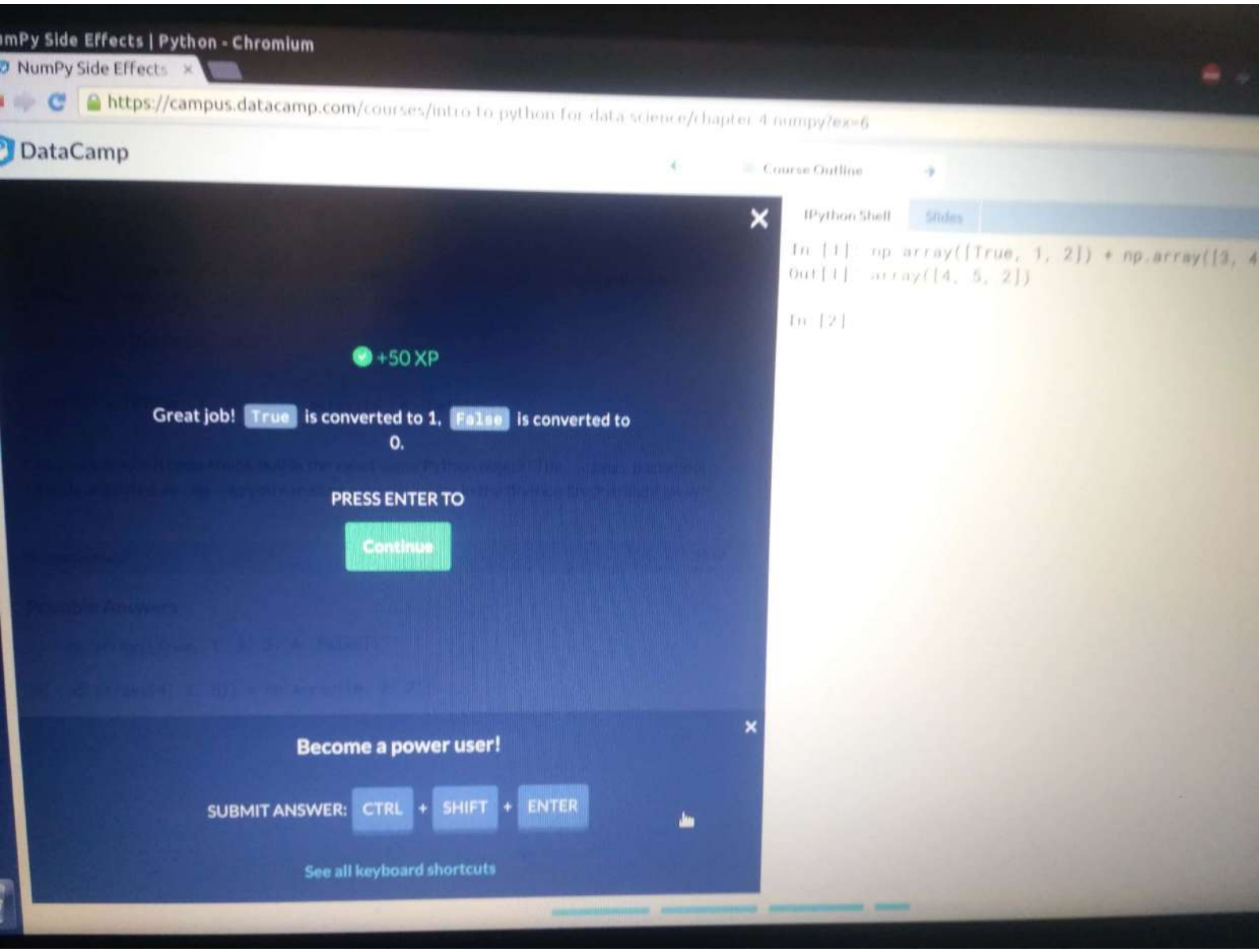
IPython Shell Slides

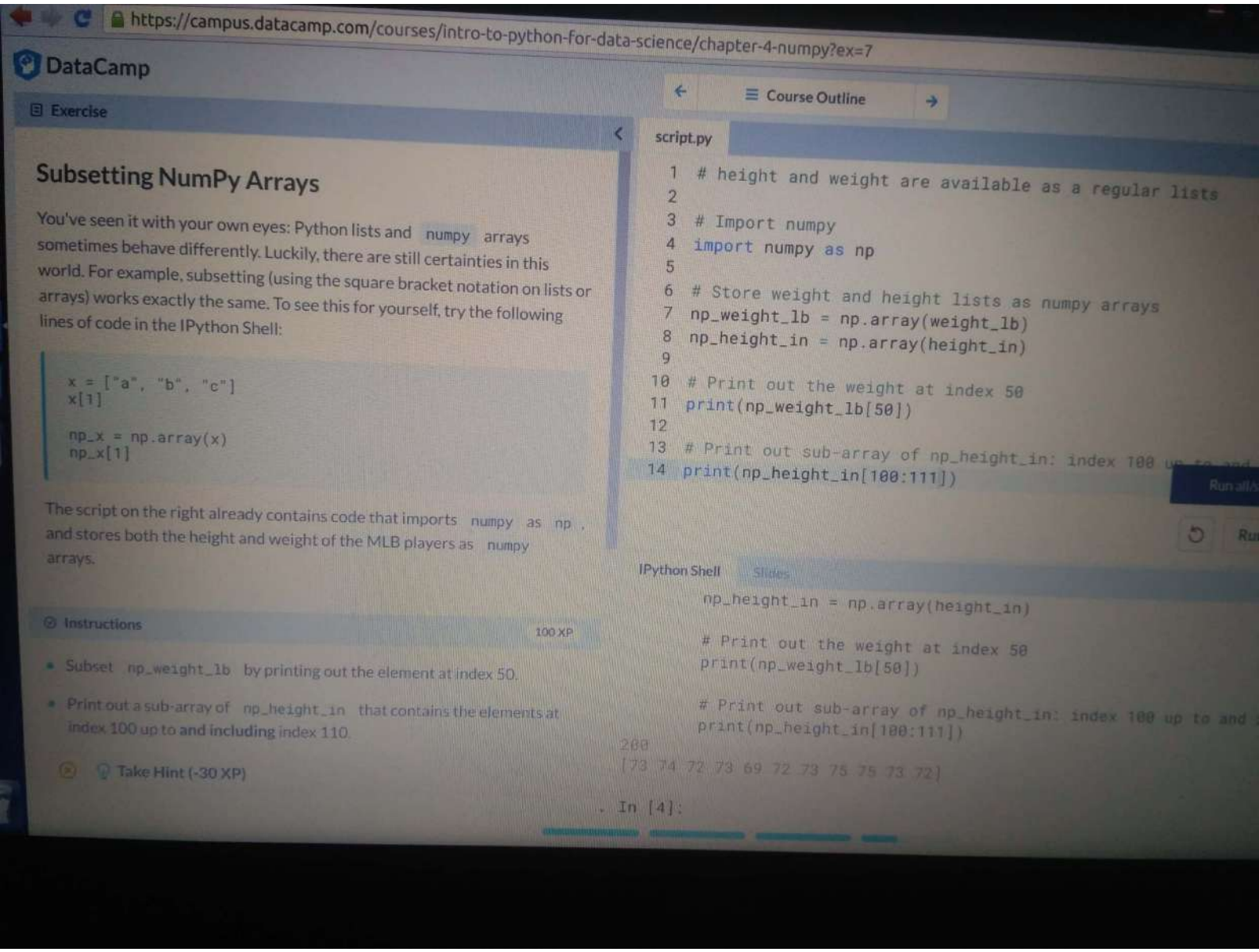
```
# Print out bmi
print(bmi)
[ 23.11037639  27.60406069  28.48088465 ...,  25.62295933  23
  25.72686361]

<script.py> output:
    [ 23.11037639  27.60406069  28.48088465 ...,  25.62295933
      25.72686361]

In [2]:
```

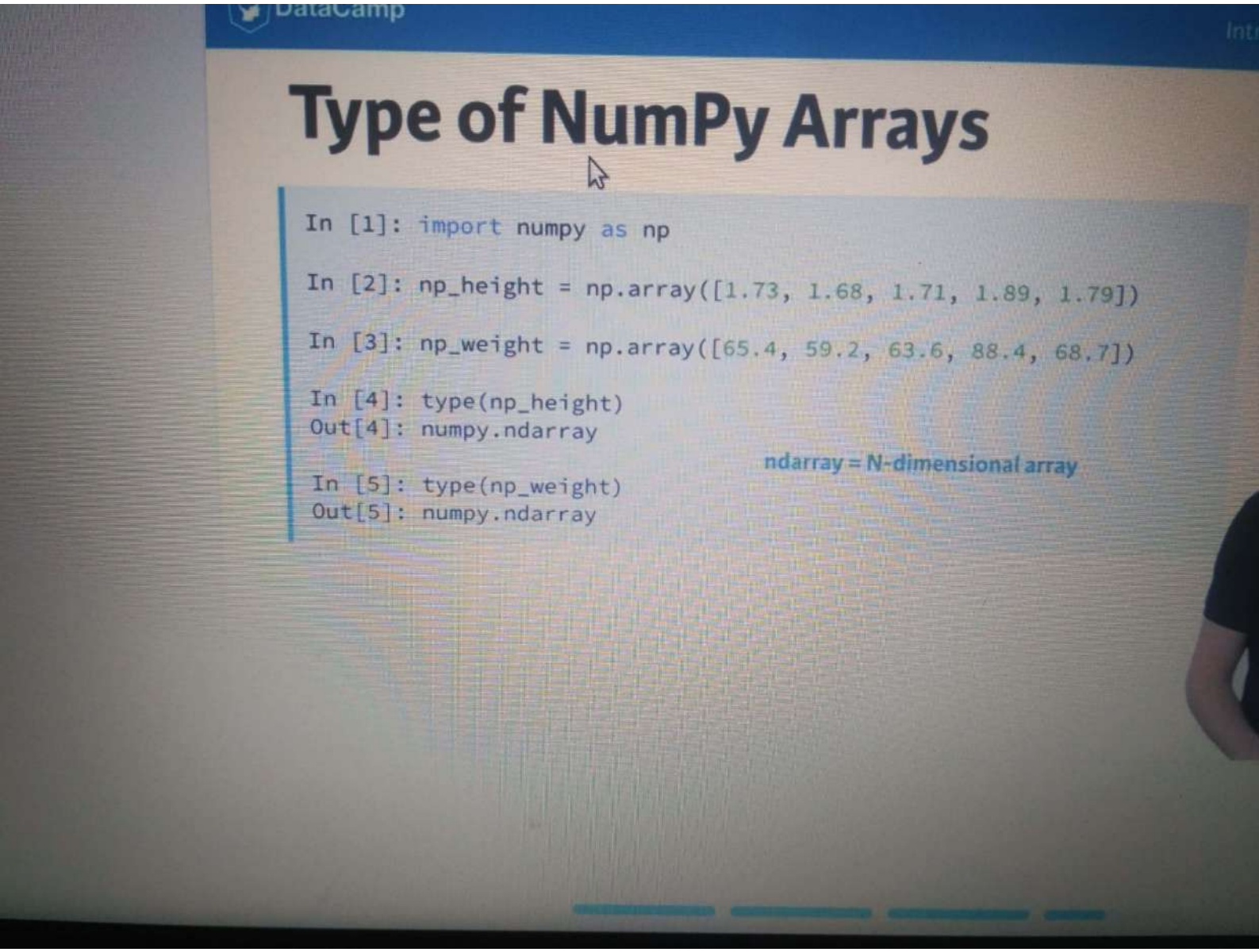




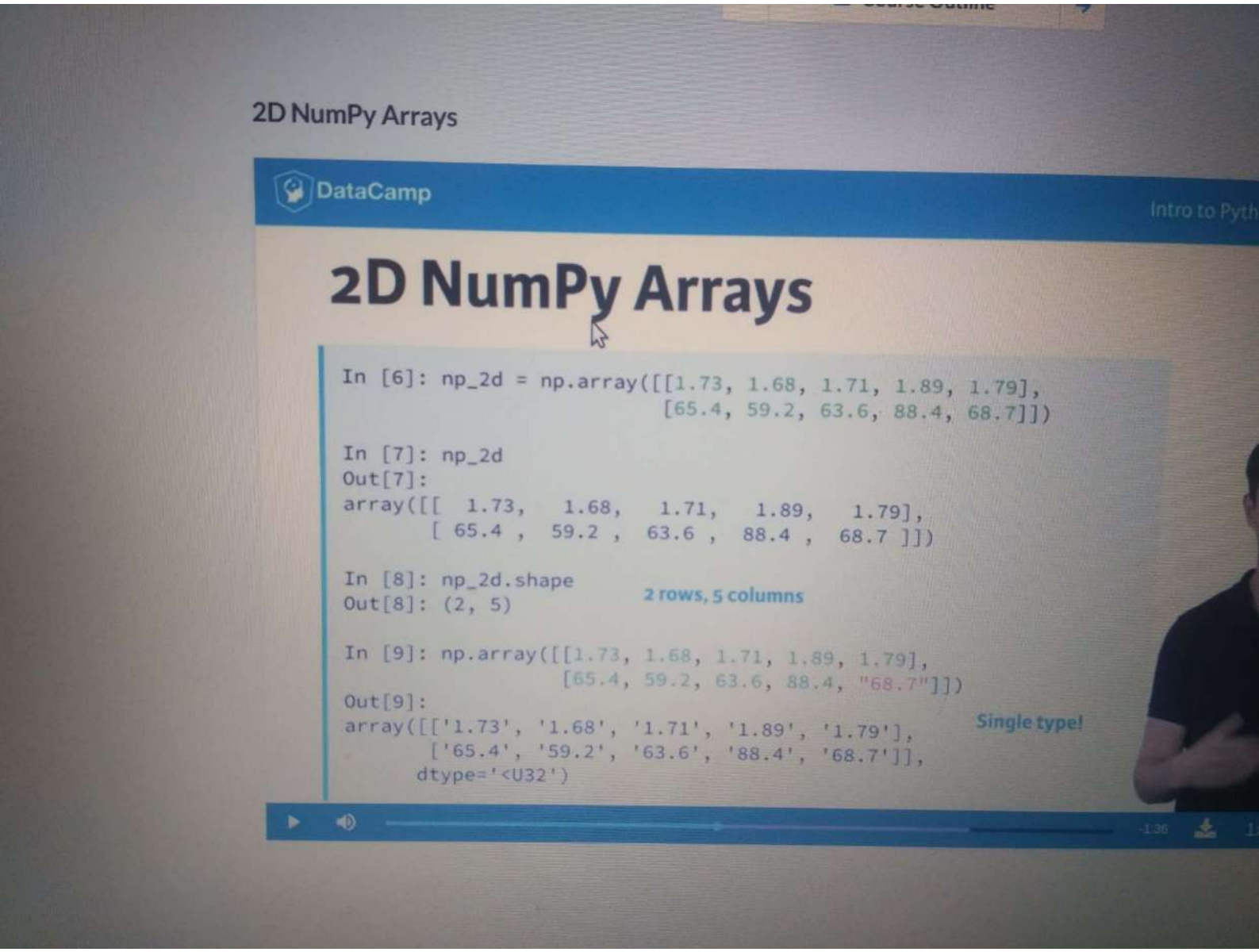


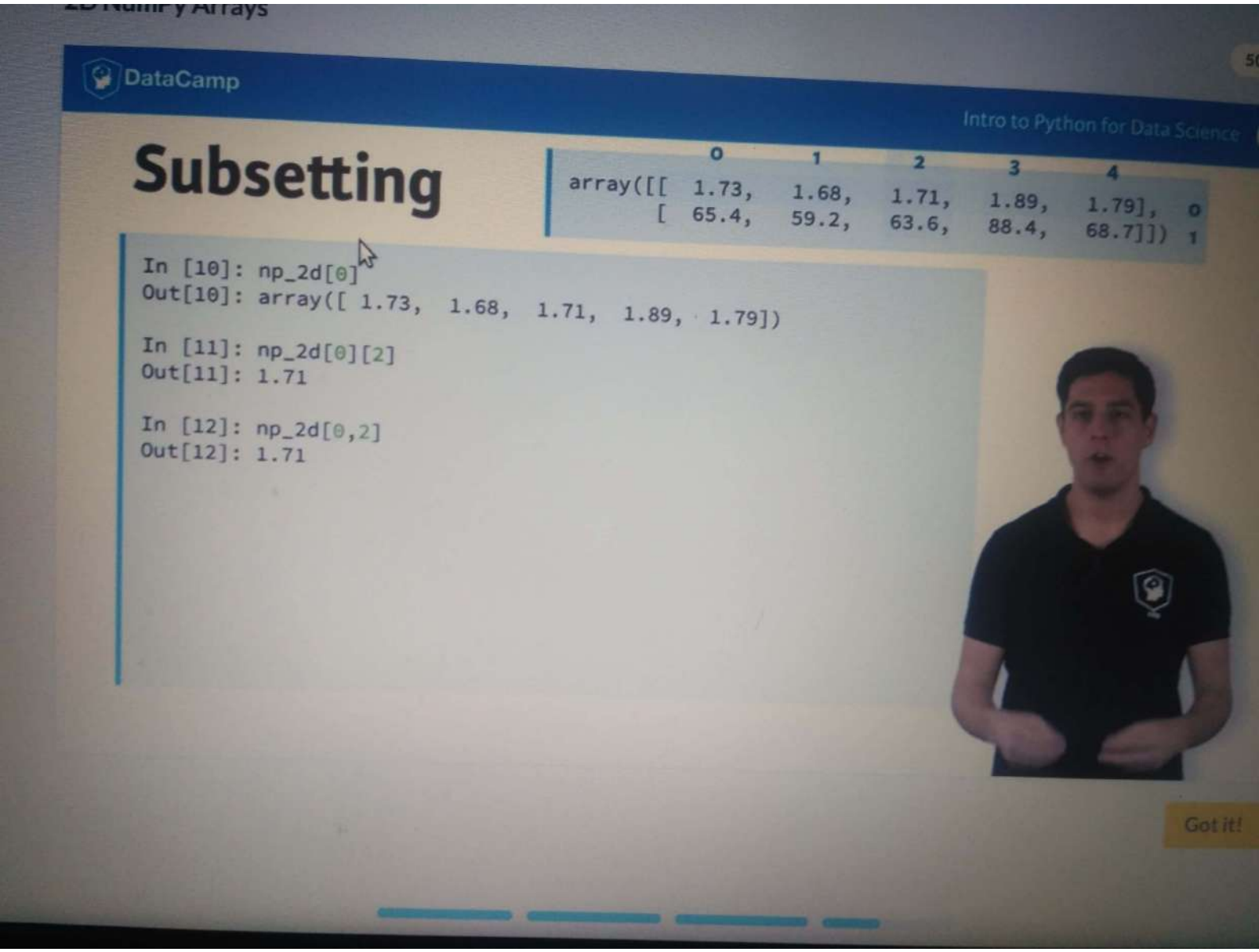






2D NumPy Arrays





2D NumPy Arrays

DataCamp

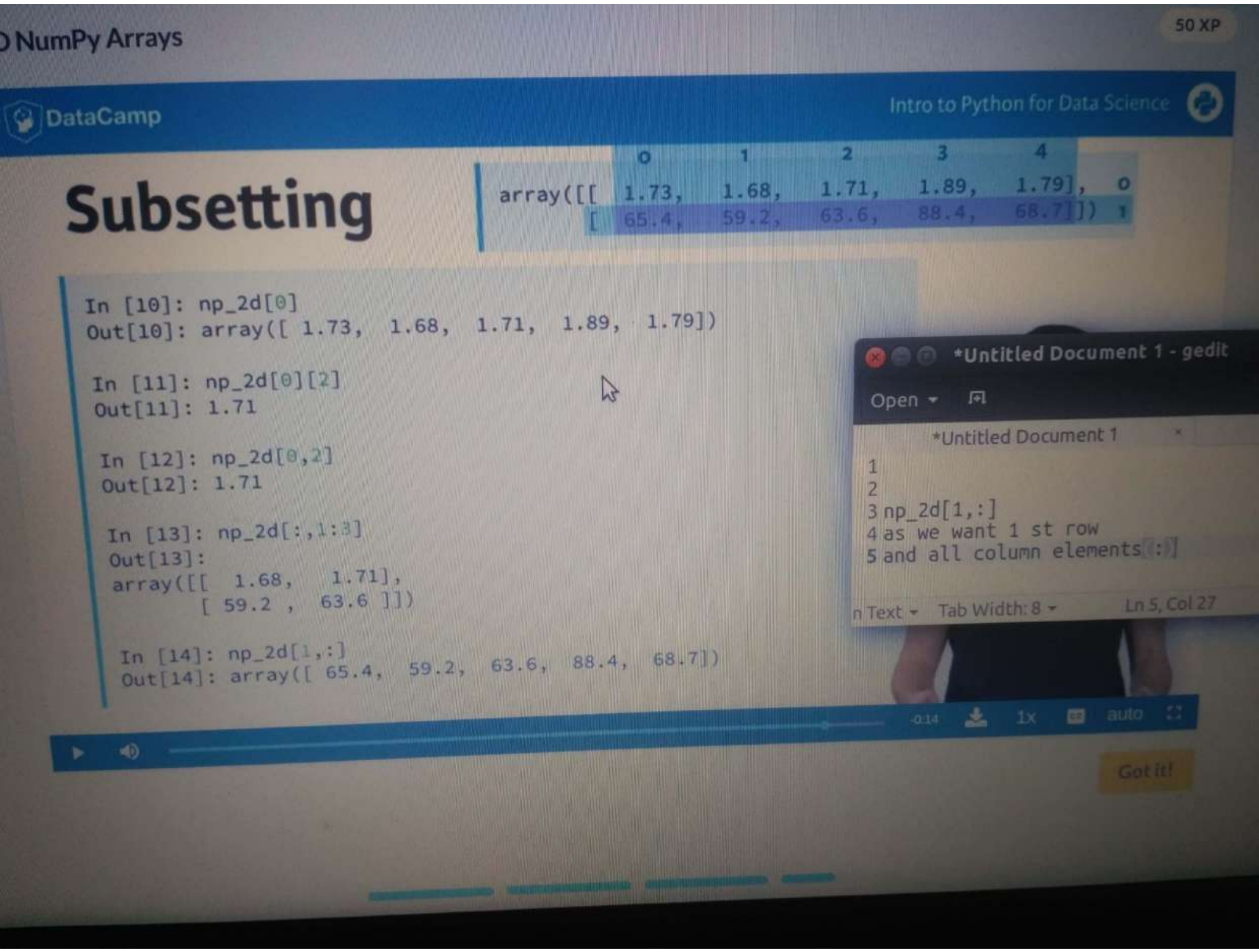
Intro to Python for Data Science

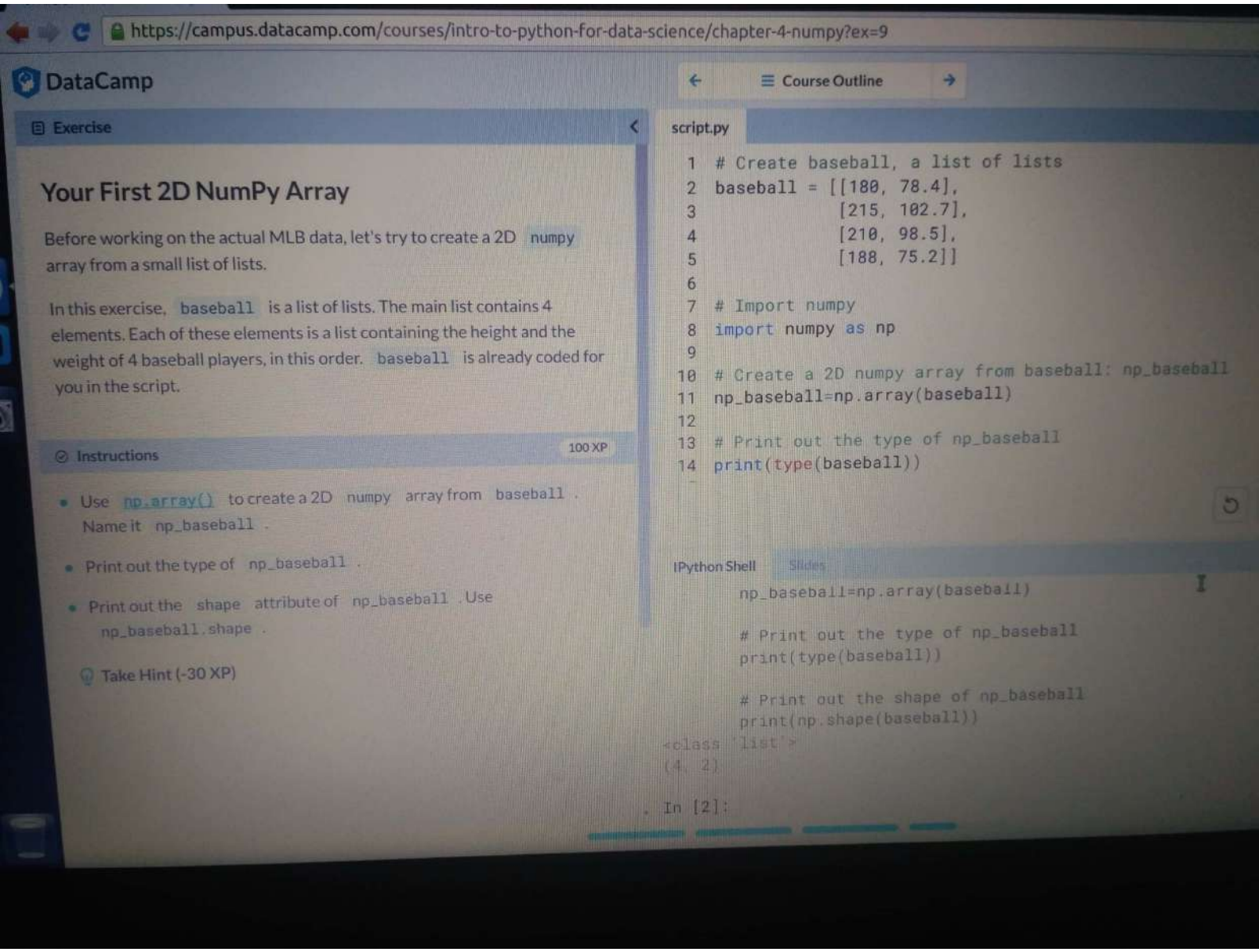
Subsetting

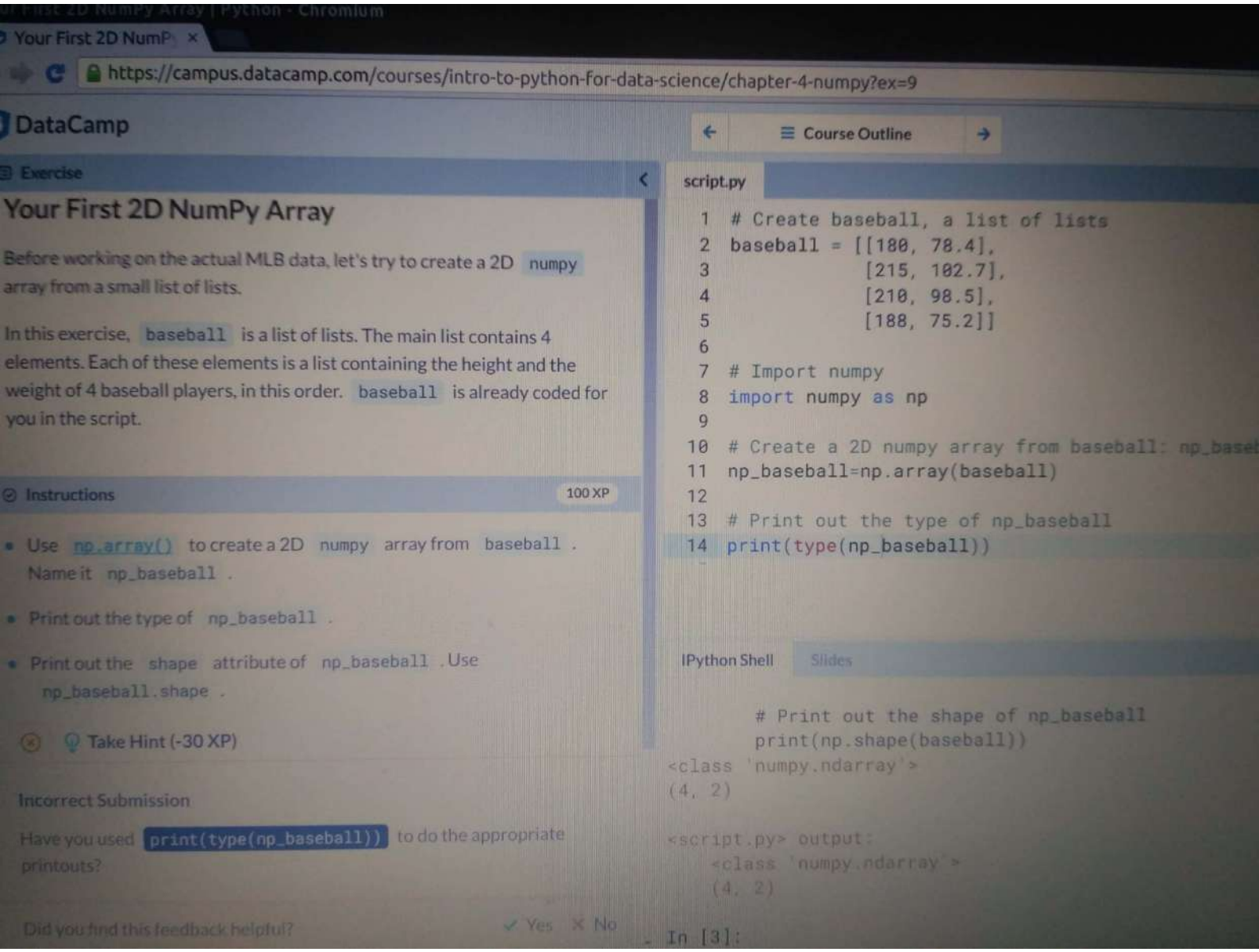
```
          0     1     2     3     4
array([[ 1.73,  1.68,  1.71,  1.89,  1.79],   0
       [ 65.4,  59.2,  63.6,  88.4,  68.7]])  1
```

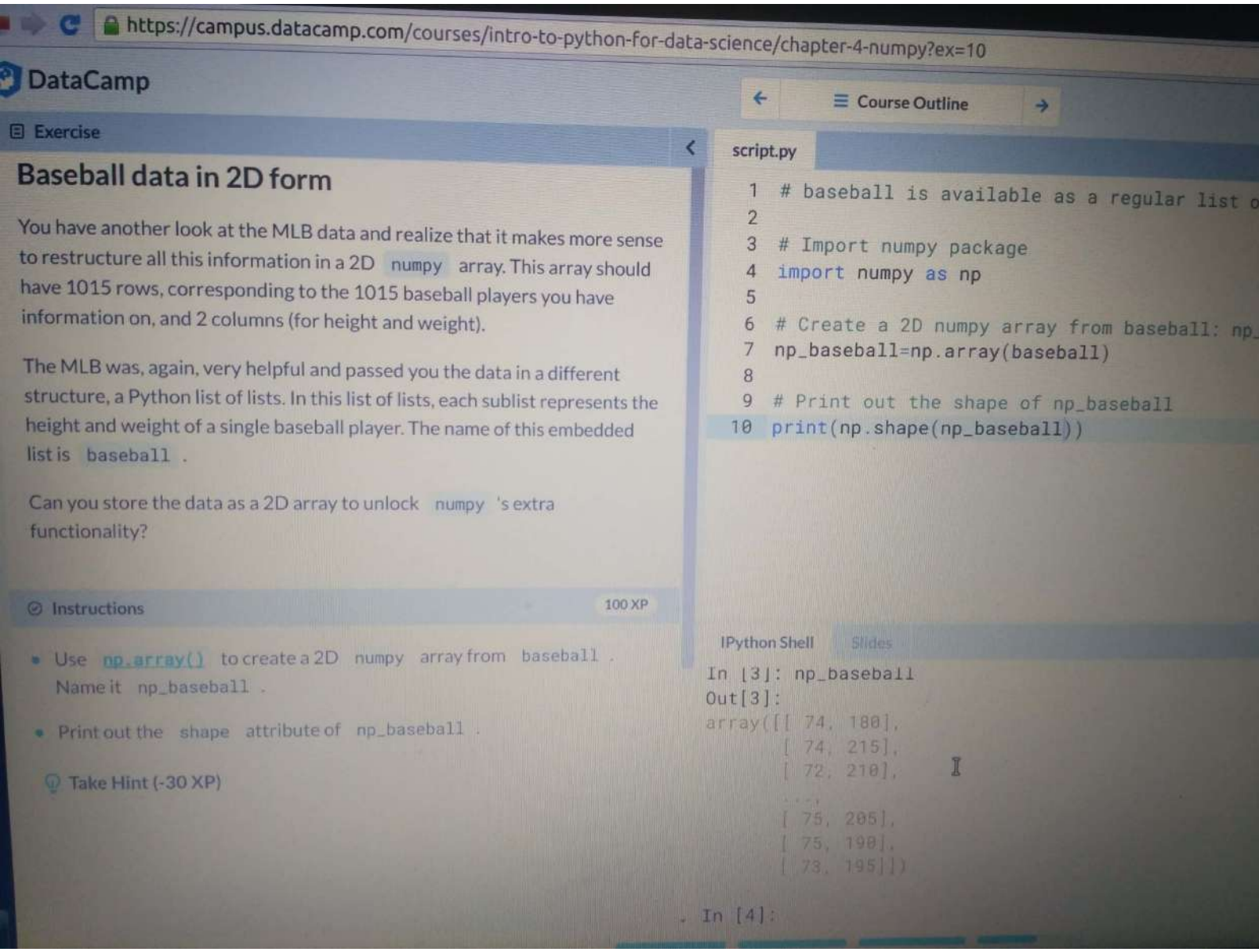
```
In [10]: np_2d[0]
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [11]: np_2d[0][2]
Out[11]: 1.71

In [12]: np_2d[0,2]
Out[12]: 1.71

In [13]: np_2d[:,1:3]
Out[13]:
array([[  1.68,   1.71],
       [ 59.2 ,  63.6 ]])
```

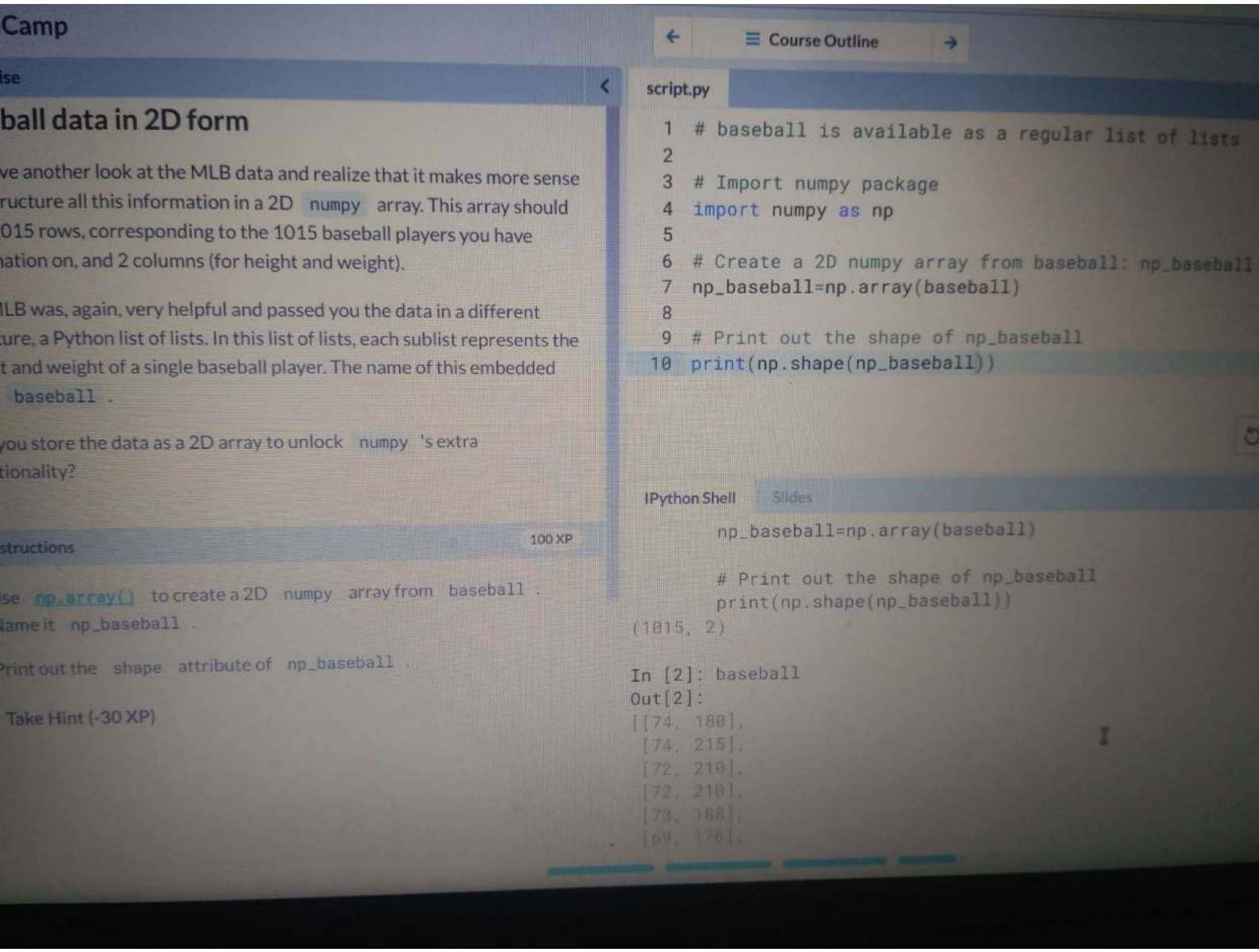
Got it!

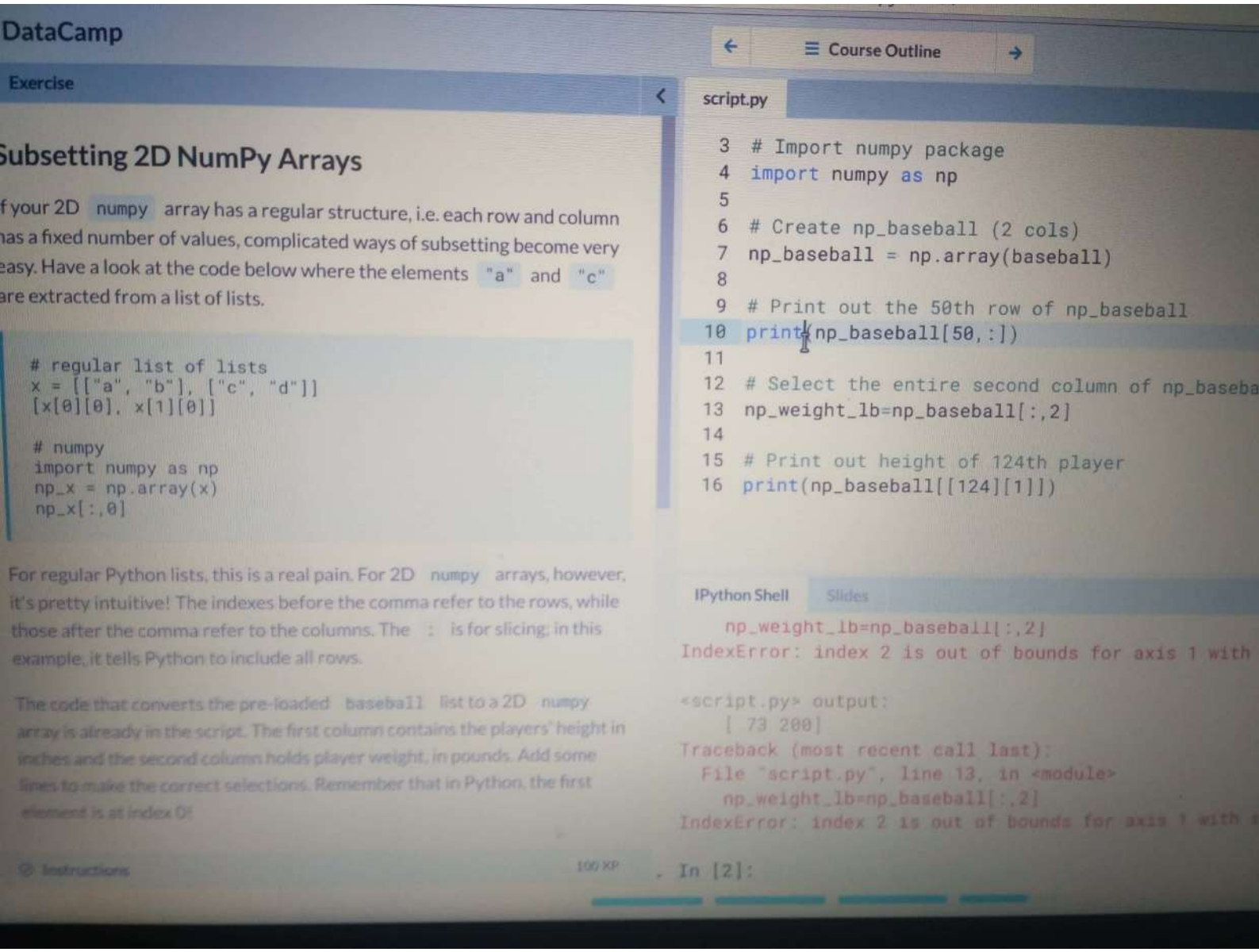


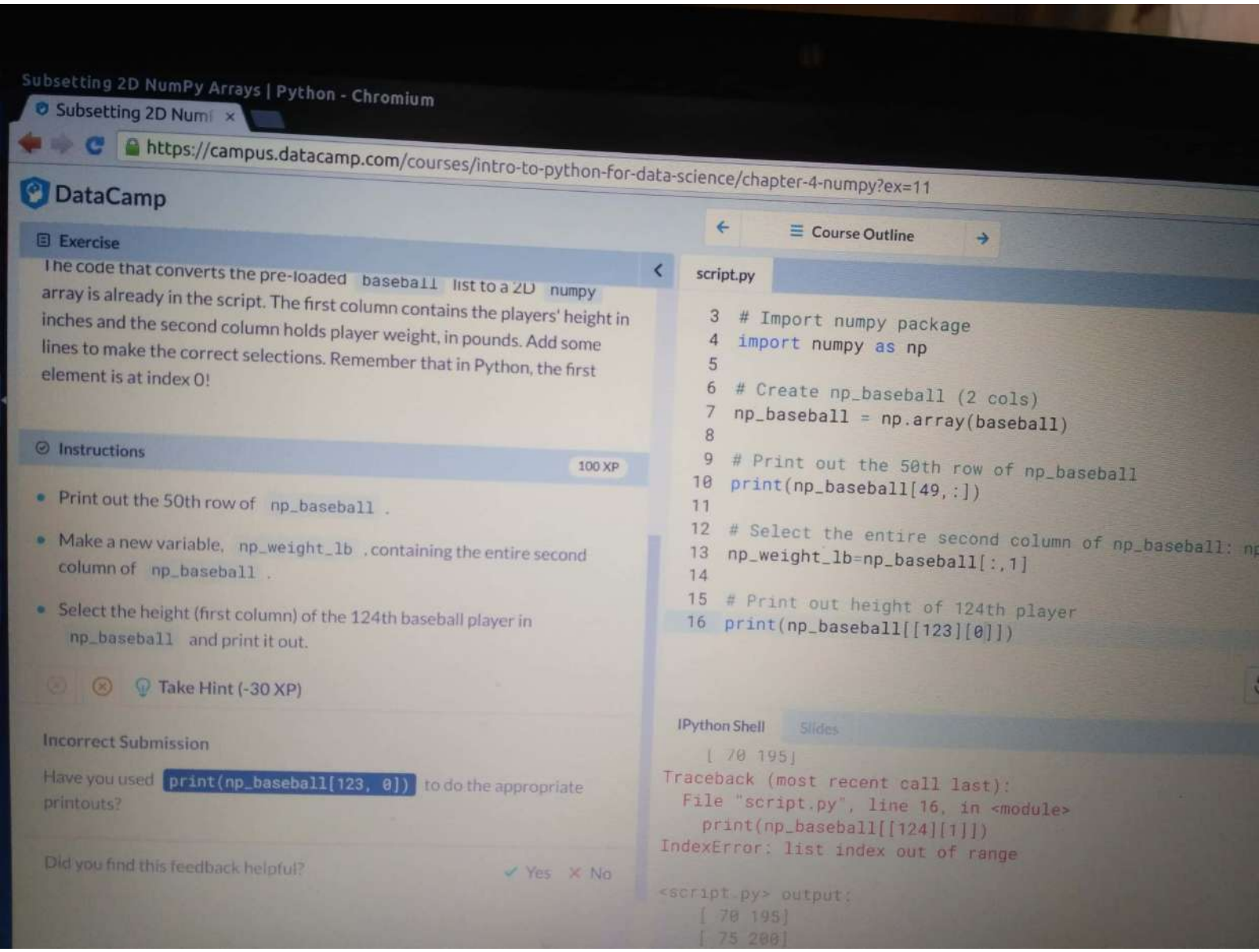


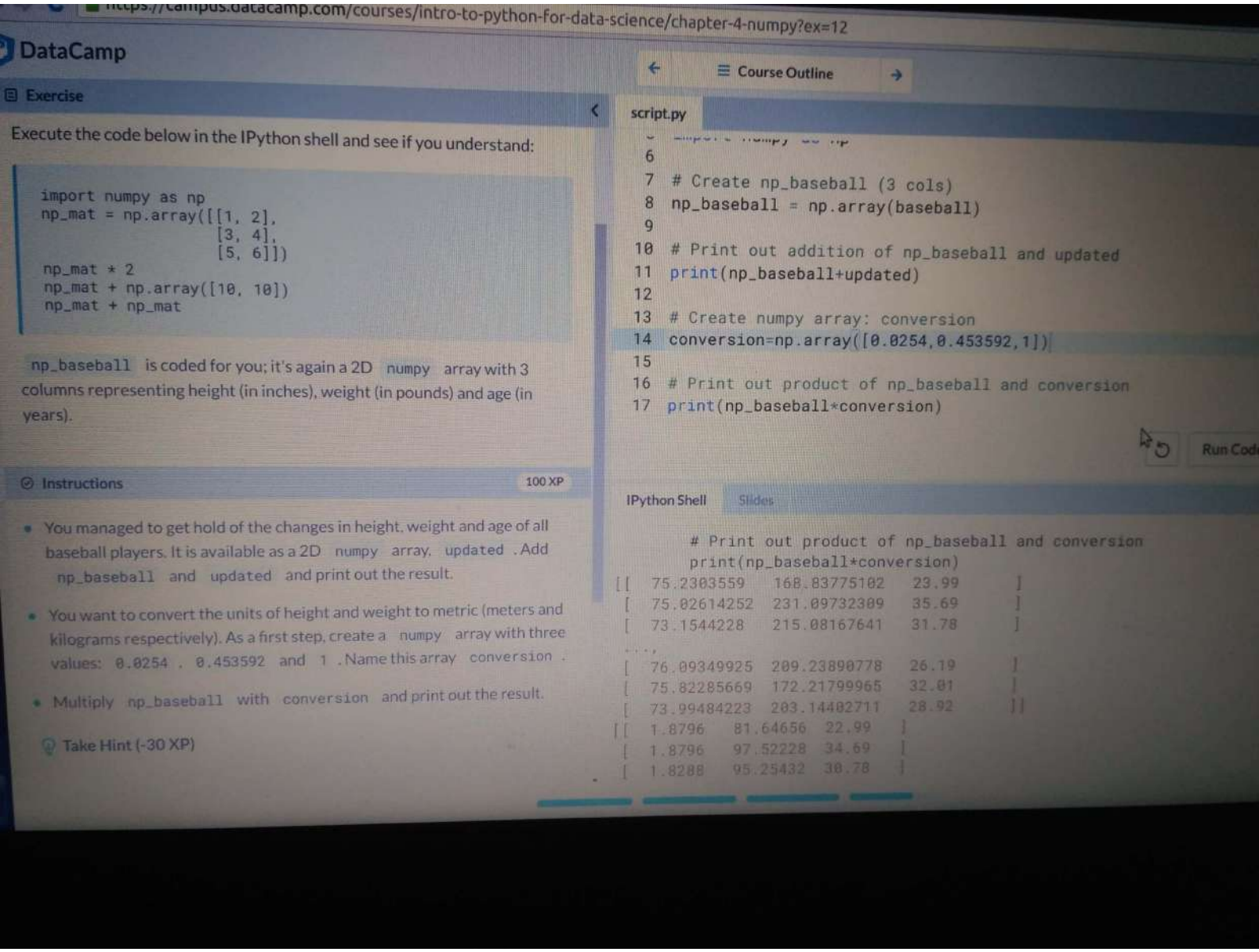


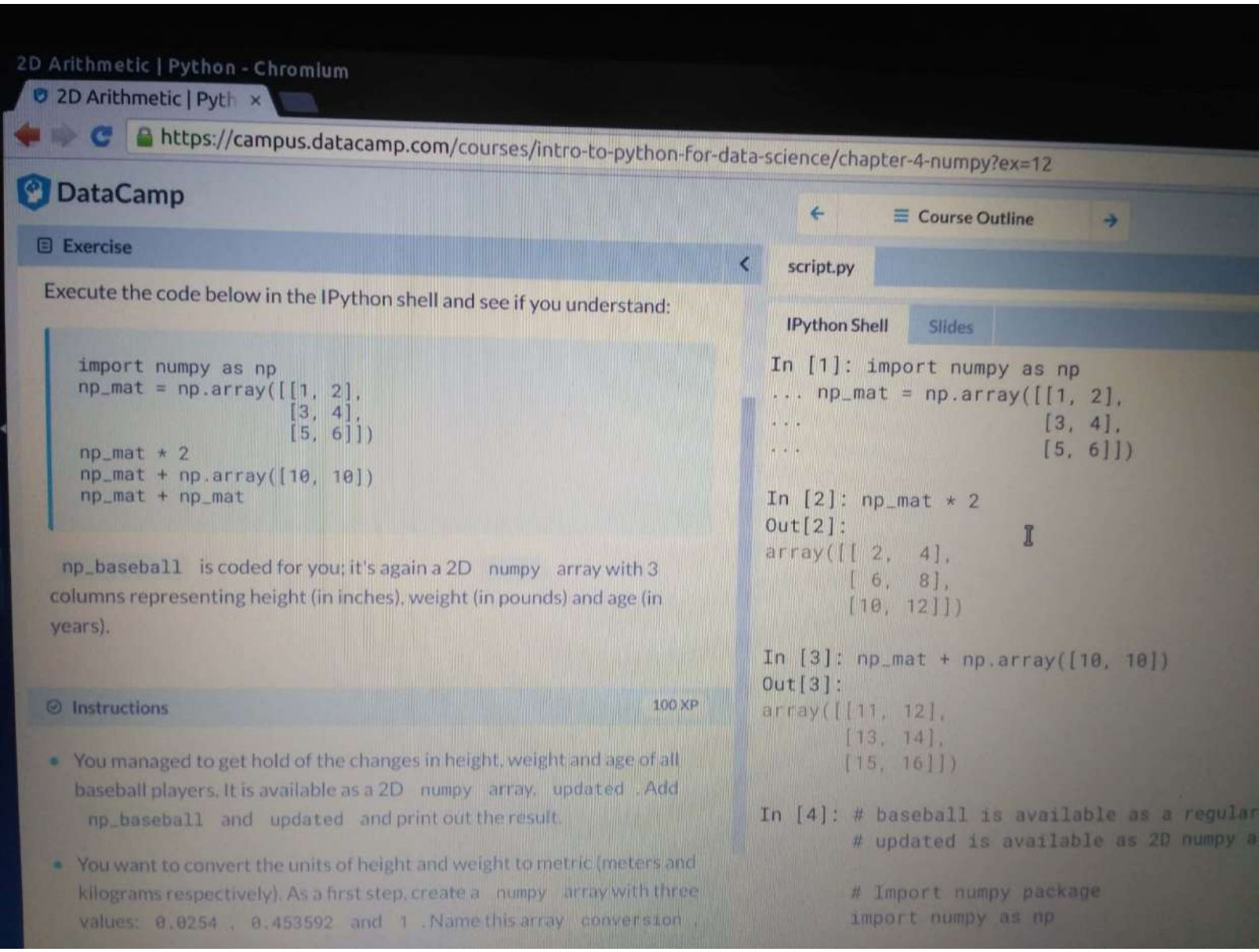


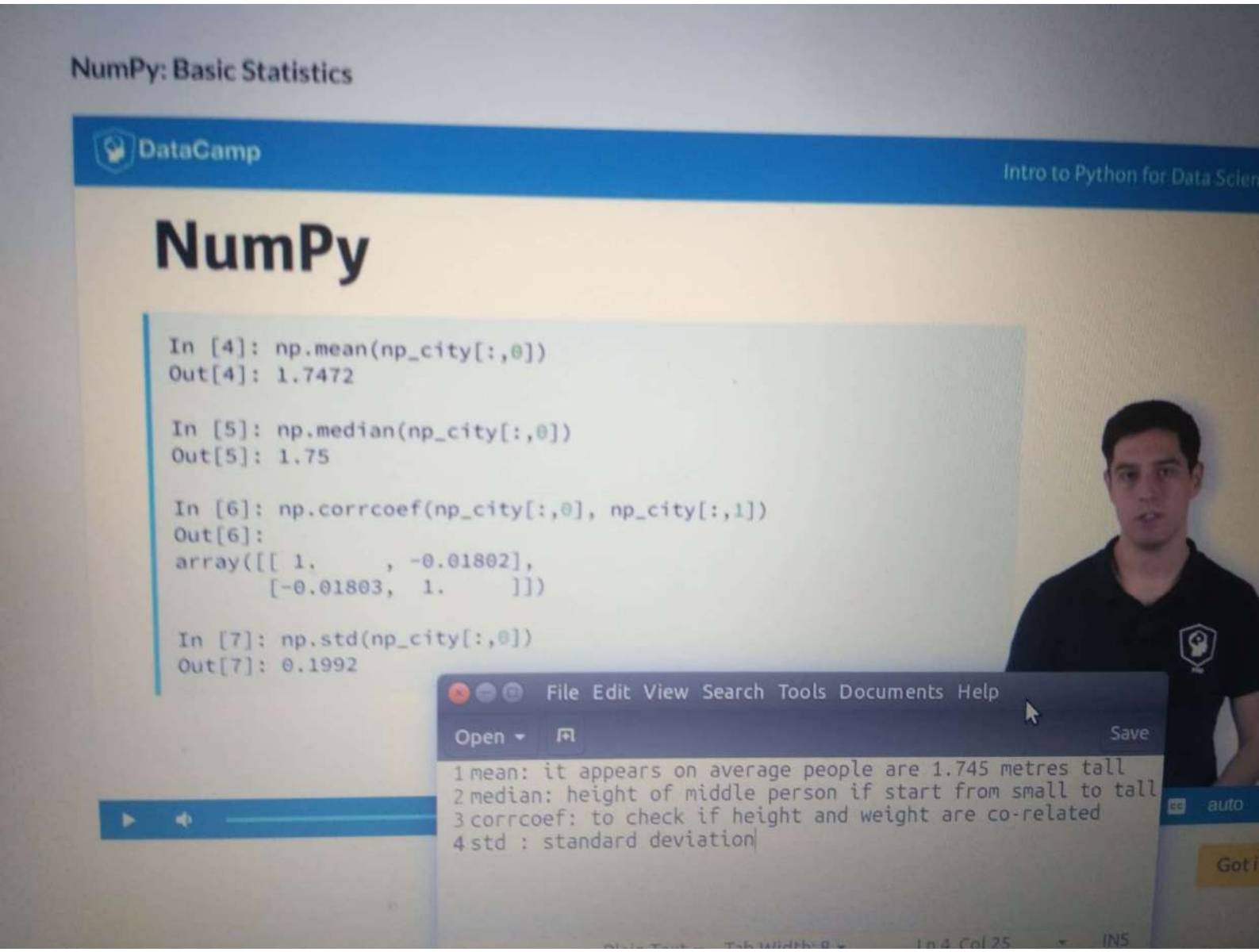


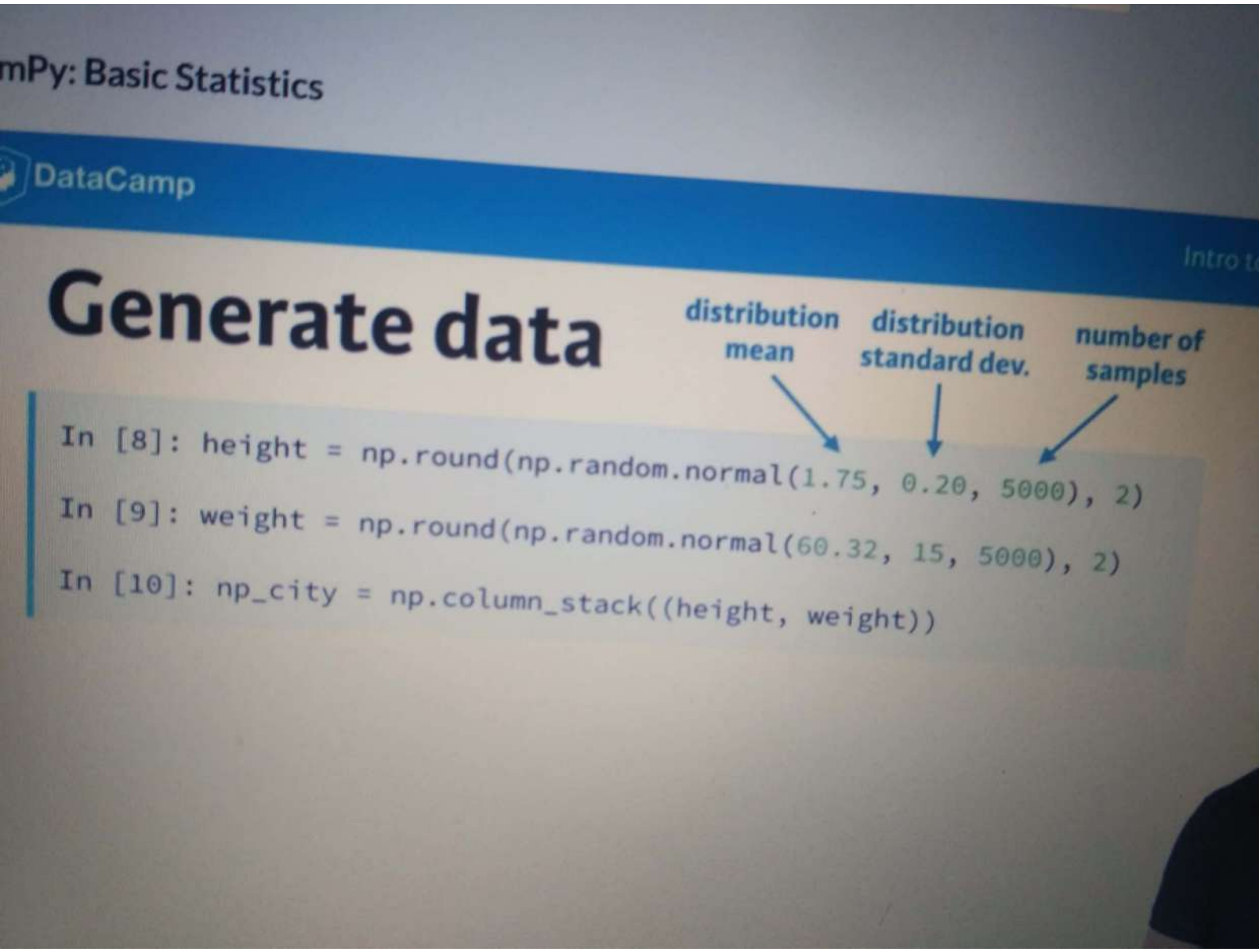


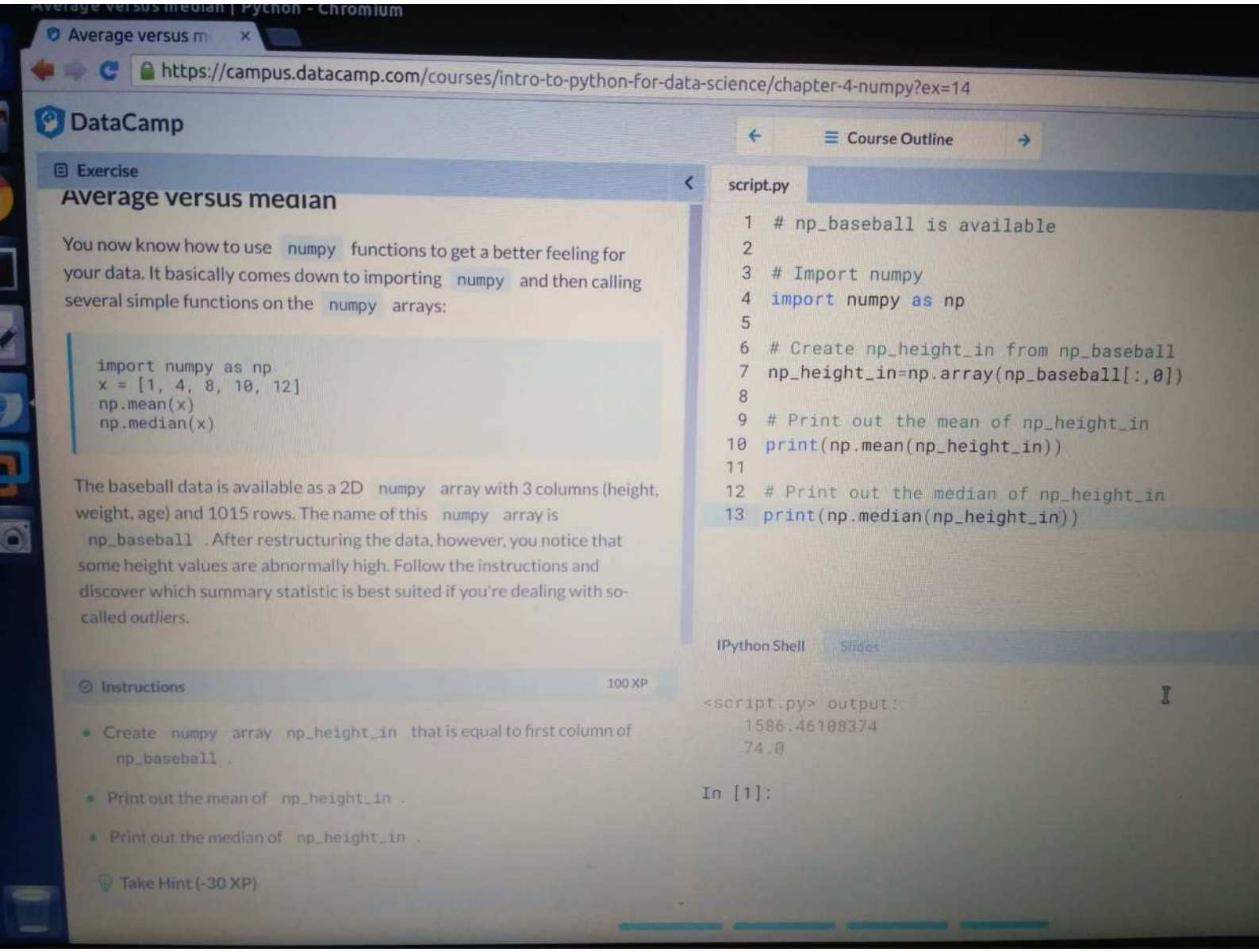


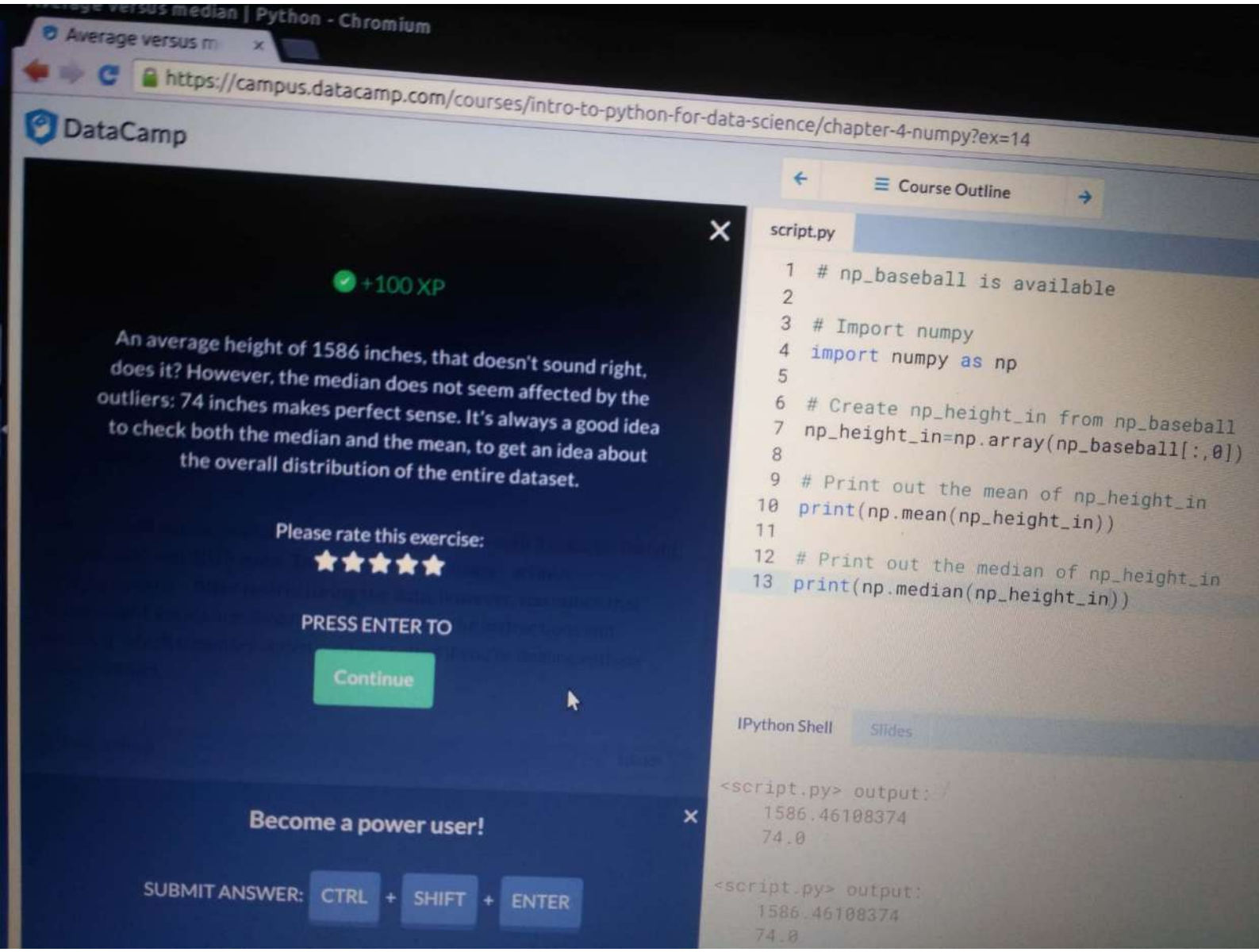


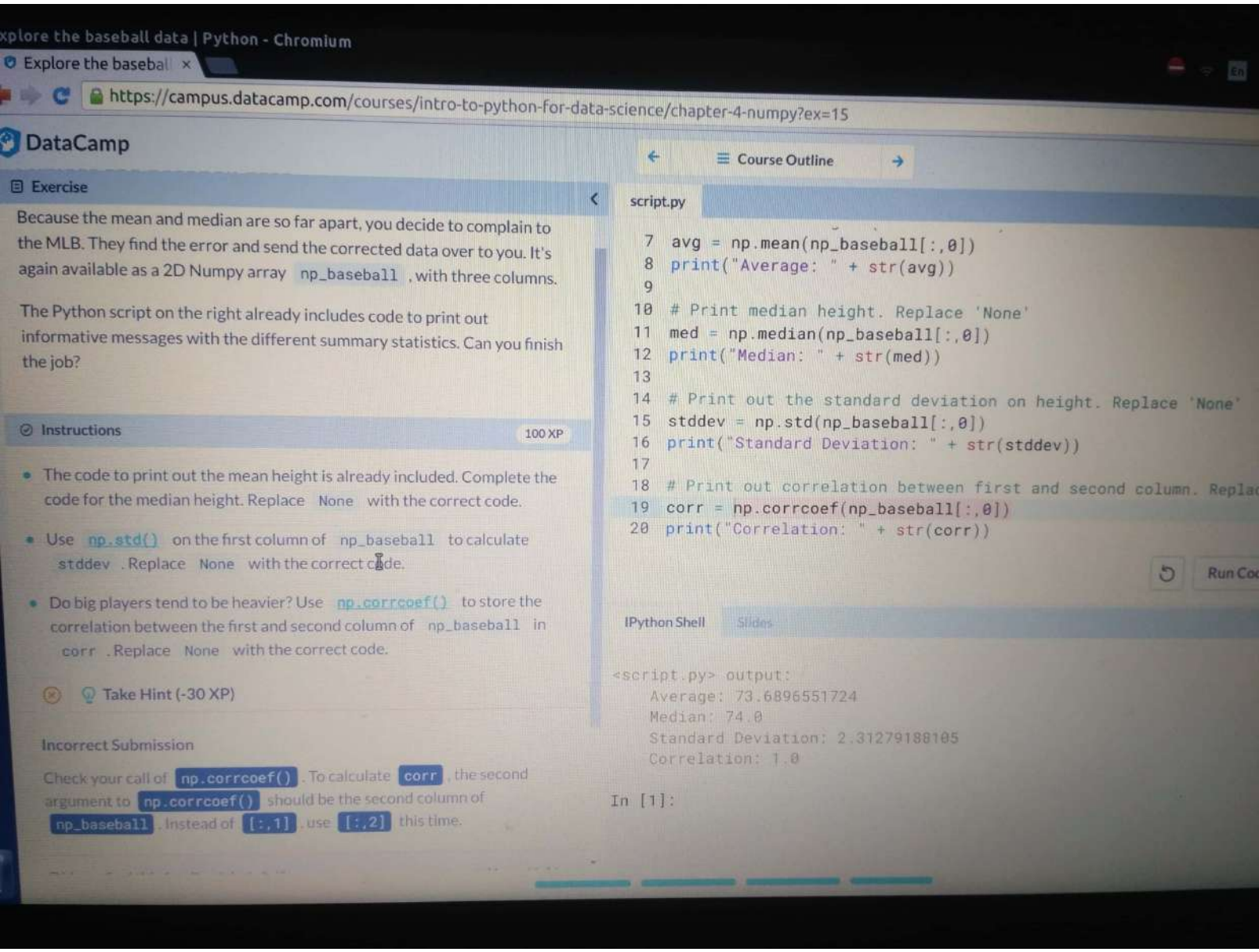


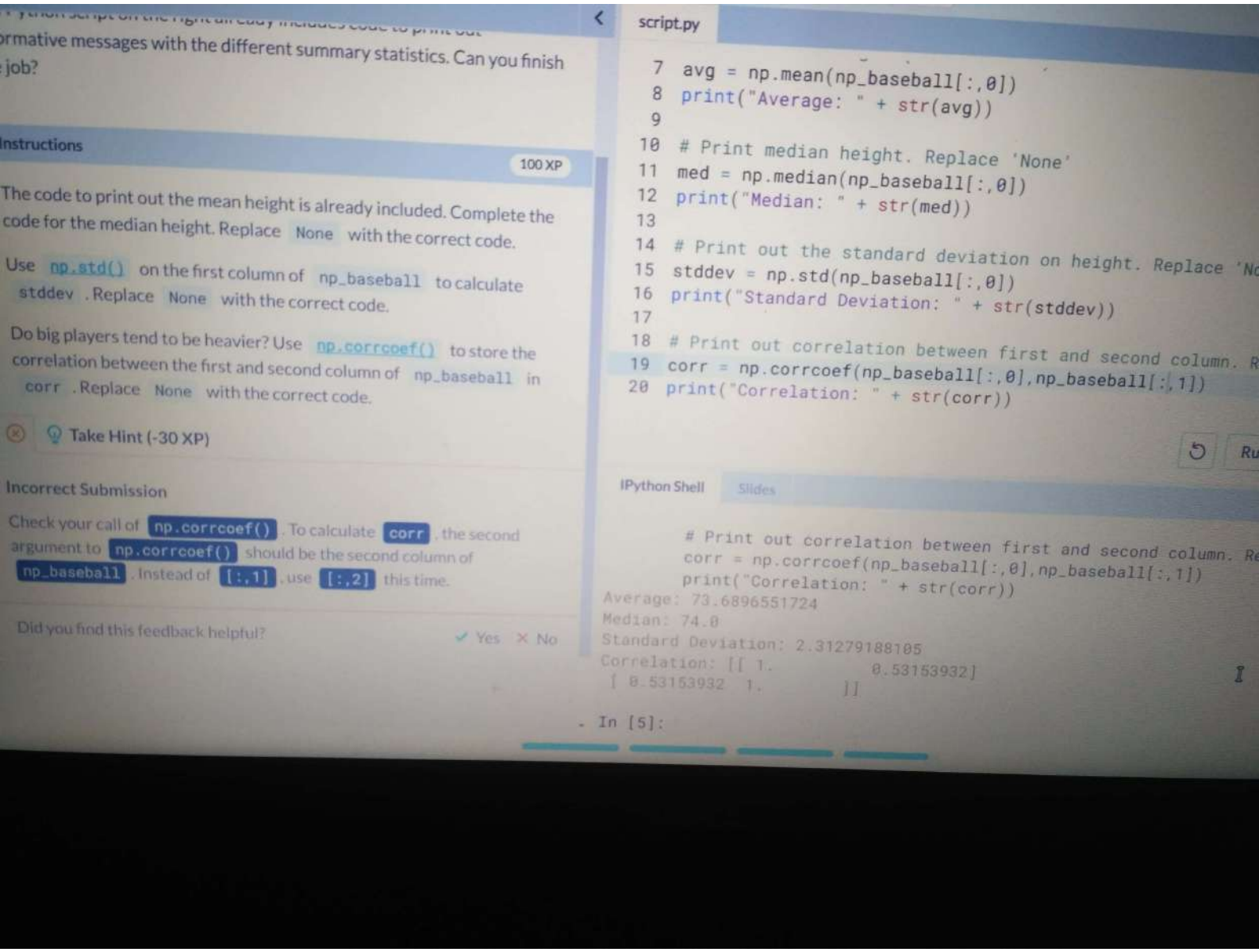


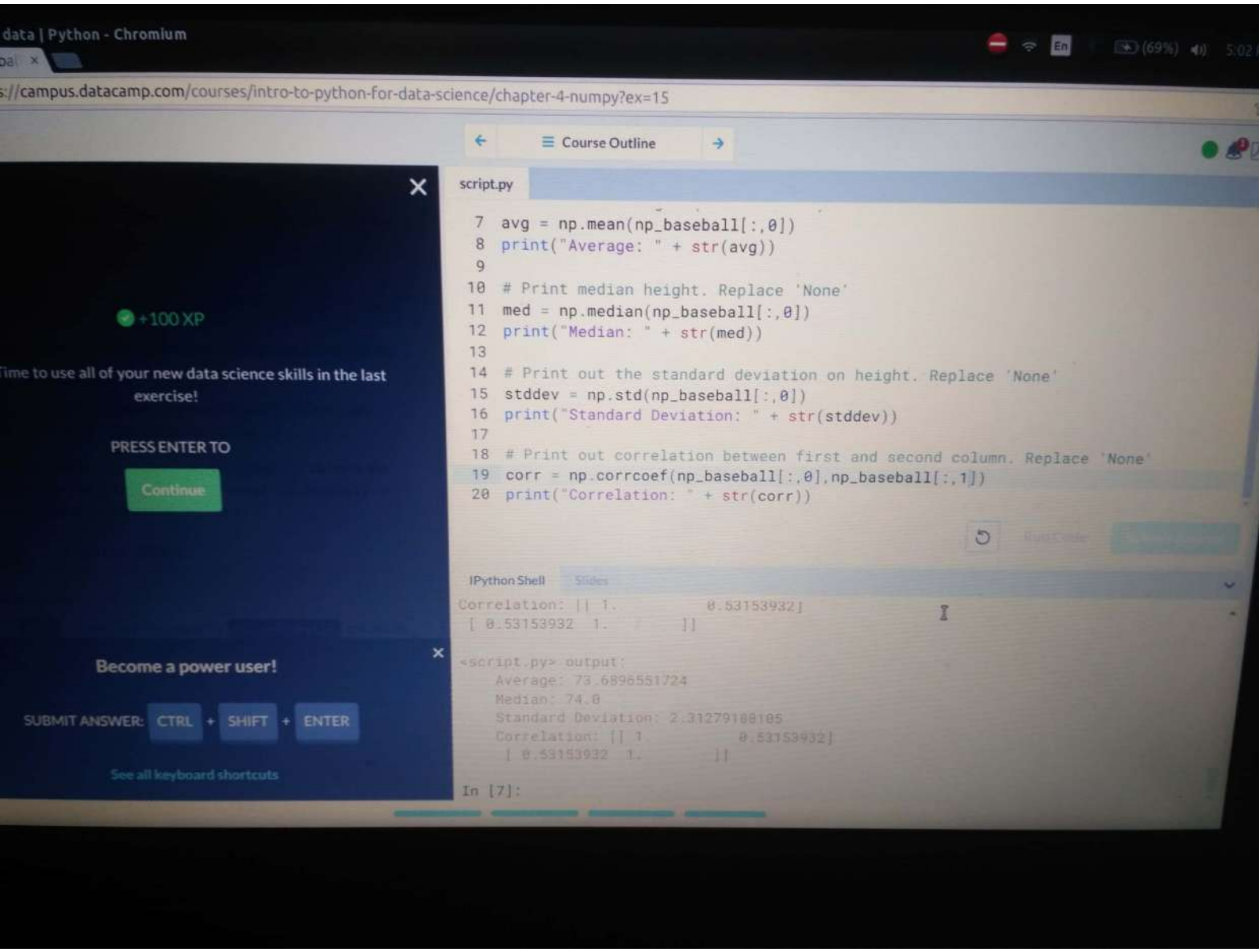


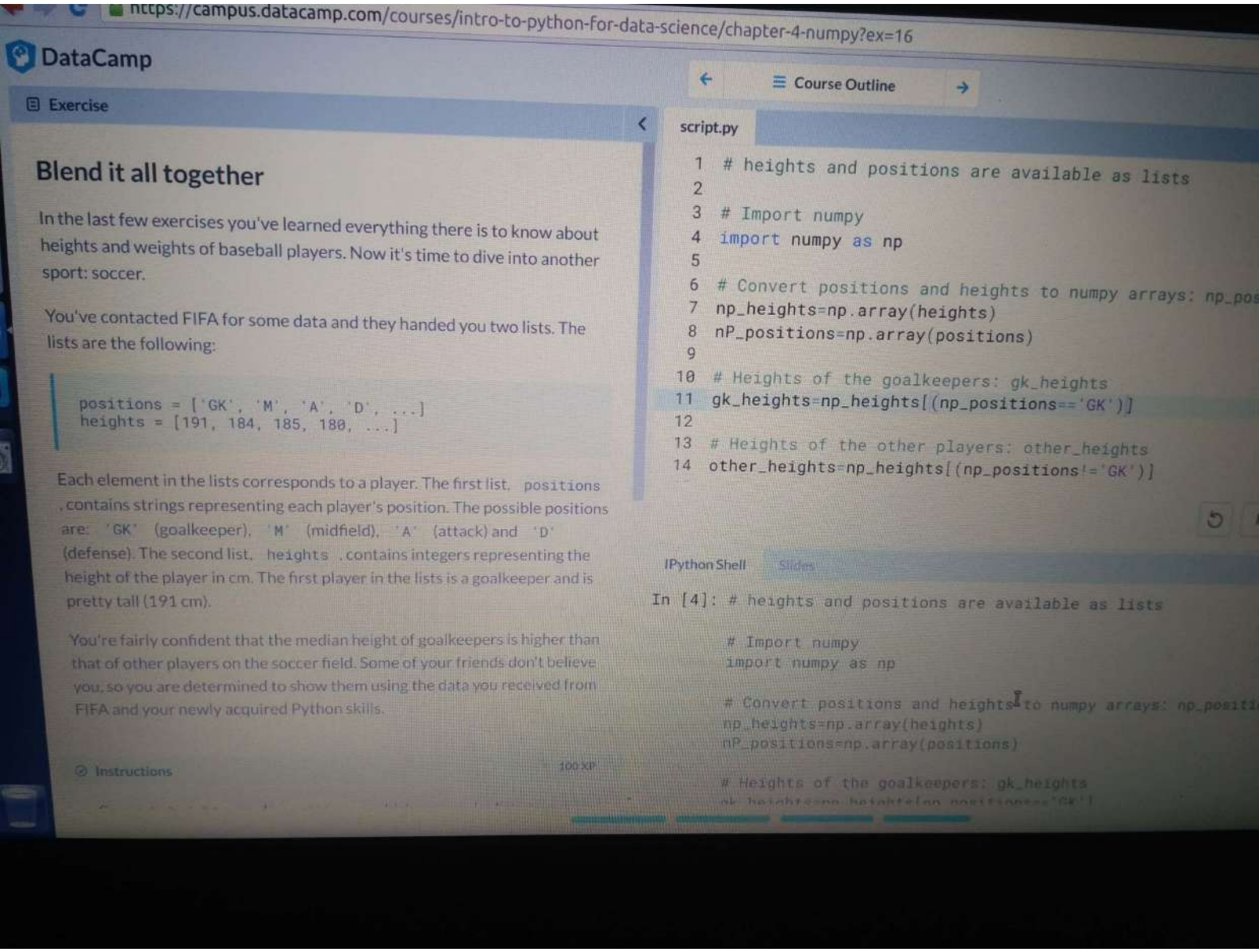


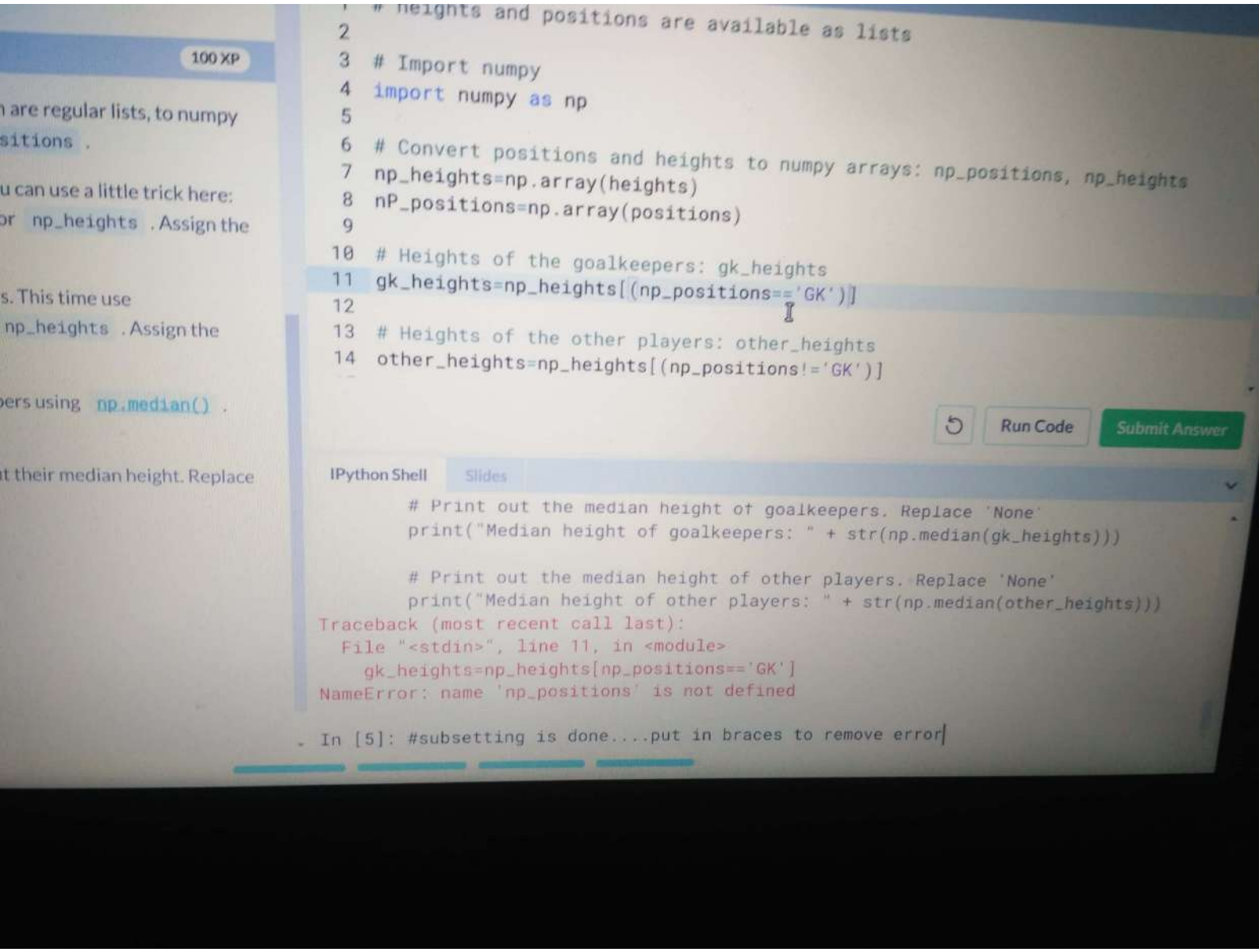












Course Outline

cript.py

```
1  # heights and positions are available as lists
2
3  # Import numpy
4  import numpy as np
5
6  # Convert positions and heights to numpy arrays: np_positions,
7  np_heights=np.array(heights)
8  np_positions=np.array(positions)
9
10 # Heights of the goalkeepers: gk_heights
11 gk_heights=np_heights[(np_positions=='GK')]
12
13 # Heights of the other players: other_heights
14 other_heights=np_heights[(np_positions!='GK')]
```

Run Code

IPython Shell | Slides

```
other_heights=np_heights[(np_positions!='GK')]

# Print out the median height of goalkeepers. Replace 'None'
print("Median height of goalkeepers: " + str(np.median(gk_heights

# Print out the median height of other players. Replace 'None'
print("Median height of other players: " + str(np.median(other_hei
Median height of goalkeepers: 188.0
Median height of other players: 181.0

In [7]:
```

c plots with matplotlib

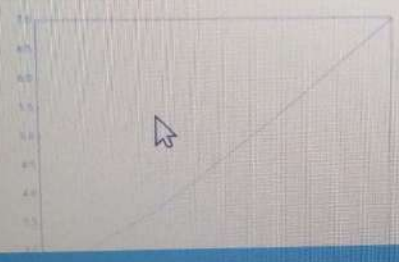
50 XP

DataCamp

Intermediate Python for Data Science

Matplotlib

```
In [1]: import matplotlib.pyplot as plt

In [2]: year = [1950, 1970, 1990, 2010]

In [3]: pop = [2.519, 3.692, 5.263, 6.972]

In [4]: plt.plot(year, pop)
                 x     y

In [5]: plt.show()
```



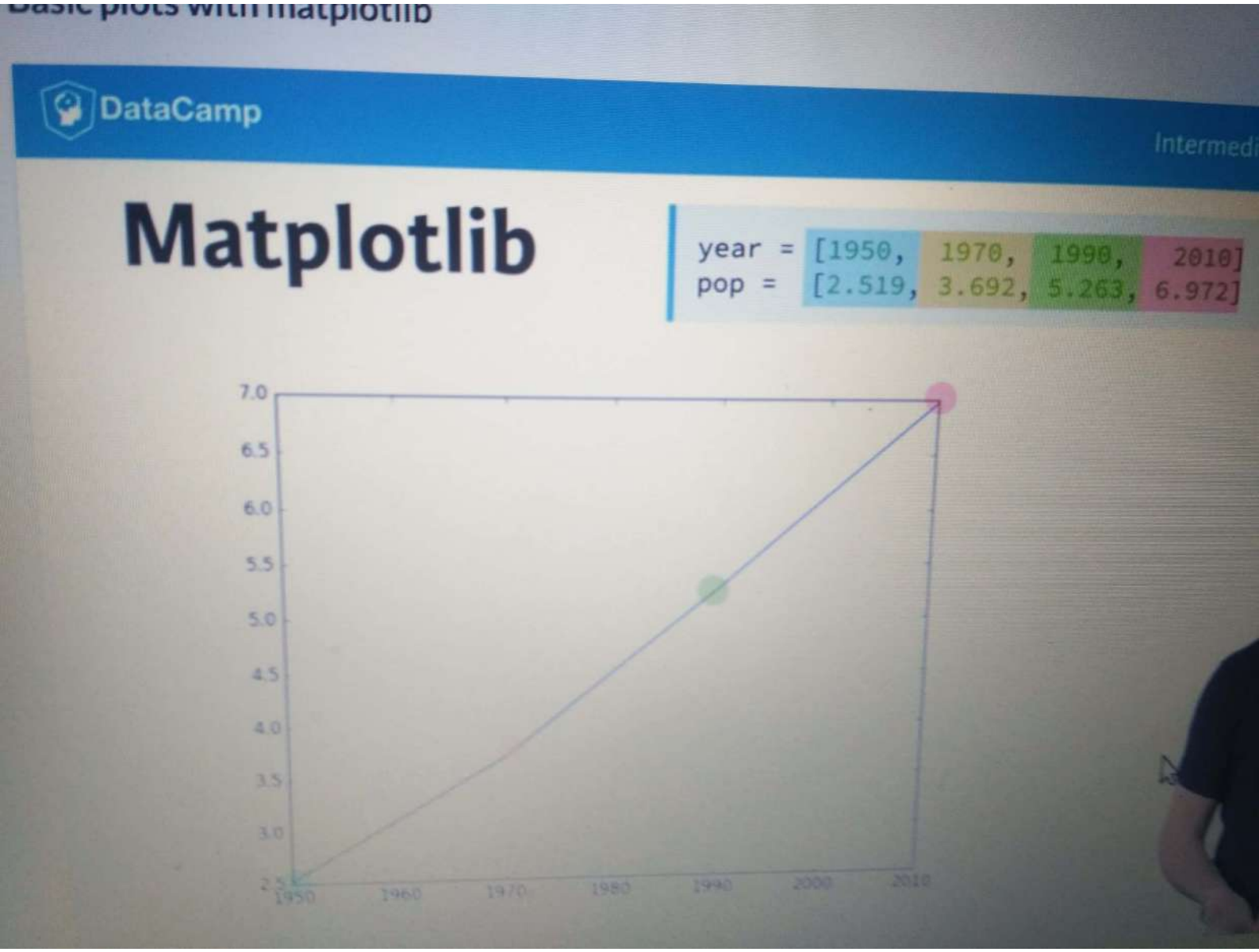
*Untitled Document 1 - gedit

Open

```
1 to plot these data as a line chart:
2 we call: plt.plot(year,pop)
3 and use 2 lists as arguments
4
5 The 1st one corresponds to horizontal axis
6 and the other to vertical axis
7
8 plt.show() - function to display the plot
9
10 pop is population.
```

Plain Text Tab Width: 8 Ln 8, Col 42

Got it!



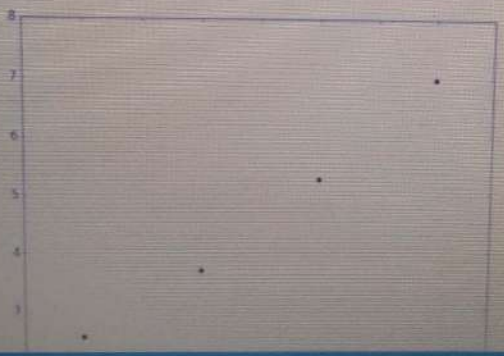
Basic plots with matplotlib

DataCamp

Intermediate Python for Data Science

Scatter plot

```
In [1]: import matplotlib.pyplot as plt
In [2]: year = [1950, 1970, 1990, 2010]
In [3]: pop = [2.519, 3.692, 5.263, 6.972]
In [4]: plt.scatter(year, pop)
In [5]: plt.show()
```



*Untitled Document 1 - gedit

Open Save

```
1 plt.scatter() - function python doesn't
  connect lines with plots
2 sometimes,its better than plt.plot
```

Text Tab Width: 8 Ln 2, Col 35 INS

Got it!

ine plot (1) | Python - Chromium

Line plot (1) | Pytho ×

https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=2

DataCamp

Exercise

Course Outline

With matplotlib, you can create a bunch of different plots in Python. The most basic plot is the line plot. A general recipe is given here.

```
import matplotlib.pyplot as plt
plt.plot(x,y)
plt.show()
```

In the video, you already saw how much the world population has grown over the past years. Will it continue to do so? The world bank has estimates of the world population for the years 1950 up to 2100. The years are loaded in your workspace as a list called `year`, and the corresponding populations as a list called `pop`.

Instructions 100 XP

- `print()` the last item from both the `year` and the `pop` list to see what the predicted population for the year 2100 is. Use two `print()` functions.
- Before you can start, you should import `matplotlib.pyplot as plt`. `pyplot` is a sub-package of `matplotlib`, hence the dot.
- Use `plt.plot()` to build a line plot. `year` should be mapped on the horizontal axis, `pop` on the vertical axis. Don't forget to finish off with the `show()` function to actually display the plot.

script.py

```
1  # Print the last item from year and pop
2  print(year[-1])
3  print(pop[-1])
4
5
6  # Import matplotlib.pyplot as plt
7  import matplotlib.pyplot as plt
8
9  # Make a line plot: year on the x-axis, pop on the y-axis
10 plt.plot(year,pop)
11
12 # Display the plot with plt.show()
13
```

Run

IPython Shell Slides

```
In [3]: # Display the plot with plt.show()
2100
10.85
Out[3]: [<matplotlib.lines.Line2D at 0x7fbe2fccae48>]

In [4]:
```

Intro to pytho....pdf

Line plot (1) | Pytho

https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=2

DataCamp

Course Outline

Exercise

With matplotlib, you can create a bunch of different plots in Python. The most basic plot is the line plot. A general recipe is given here.

```
import matplotlib.pyplot as plt
plt.plot(x,y)
plt.show()
```

In the video, you already saw how much the world population has grown over the past years. Will it continue to do so? The world bank has estimates of the world population for the years 1950 up to 2100. The years are loaded in your workspace as a list called `year`, and the corresponding populations as a list called `pop`.

Instructions 100 XP

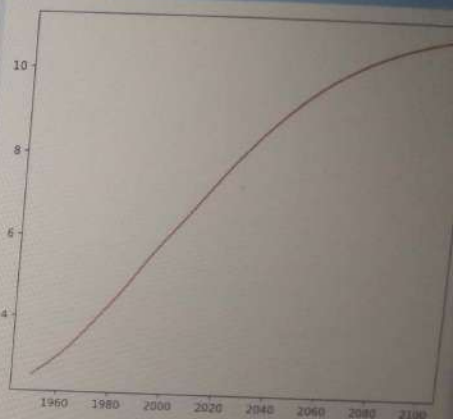
- `print()` the last item from both the `year` and the `pop` list to see what the predicted population for the year 2100 is. Use two `print()` functions.
- Before you can start, you should import `matplotlib.pyplot as plt`. `pyplot` is a sub-package of `matplotlib`, hence the dot.
- Use `plt.plot()` to build a line plot. `year` should be mapped on the horizontal axis, `pop` on the vertical axis. Don't forget to finish off with the `show()` function to actually display the plot.

script.py

```
# Print the last item from year and pop
print(year[-1])
print(pop[-1])


# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# Make a line plot: year on the x-axis, pop on the y-axis
plt.plot(year,pop)

# Display the plot with plt.show()
plt.show()
```

Run Code Submit Answer

Plots



1/1

IPython Shell

```
# Display the plot with plt.show()
plt.show()
2100
10.85

In [9]:
```

Intro to pytho....pdf

Show all downloads...

https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=5

DataCamp

Exercise

Course Outline

Scatter Plot (1)

When you have a time scale along the horizontal axis, the line plot is your friend. But in many other cases, when you're trying to assess if there's a correlation between two variables, for example, the scatter plot is the better choice. Below is an example of how to build a scatter plot.

```
import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.show()
```

Let's continue with the `gdp_cap` versus `life_exp` plot, the GDP and life expectancy data for different countries in 2007. Maybe a scatter plot will be a better alternative?

Again, the `matplotlib.pyplot` package is available as `plt`.

Instructions 100 XP

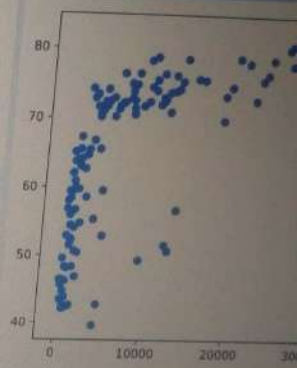
- Change the line plot that's coded in the script to a scatter plot.
- A correlation will become clear when you display the GDP per capita on a logarithmic scale. Add the line `plt.xscale('log')`.
- Finish off your script with `plt.show()` to display the plot.

script.py

```
# Change the line plot below to a scatter plot
plt.scatter(gdp_cap, life_exp)

# Put the x-axis on a logarithmic scale


# Show plot
plt.show()
```

Run Code | Submit Answer

Plots



1/1

IPython Shell | Slides

```
In [1]: # Change the line plot below to a scatter plot
        plt.scatter(gdp_cap, life_exp)

        # Put the x-axis on a logarithmic scale


        # Show plot
        plt.show()

In [2]:
```

Intro to pytho....pdf

https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=5

DataCamp

Exercise

Course Outline

Scatter Plot (1)

When you have a time scale along the horizontal axis, the line plot is your friend. But in many other cases, when you're trying to assess if there's a correlation between two variables, for example, the scatter plot is the better choice. Below is an example of how to build a scatter plot.

```
import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.show()
```

Let's continue with the `gdp_cap` versus `life_exp` plot, the GDP and life expectancy data for different countries in 2007. Maybe a scatter plot will be a better alternative?

Again, the `matplotlib.pyplot` package is available as `plt`.

Instructions 100 XP

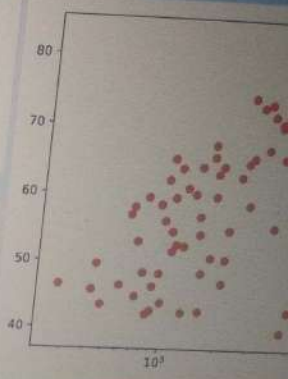
- Change the line plot that's coded in the script to a scatter plot.
- A correlation will become clear when you display the GDP per capita on a logarithmic scale. Add the line `plt.xscale('log')`.
- Finish off your script with `plt.show()` to display the plot.

script.py

```
# Change the line plot below to a scatter plot
plt.scatter(gdp_cap, life_exp)

# Put the x-axis on a logarithmic scale
plt.xscale('log')

# Show plot
plt.show()
```

Run Code | Submit Answer

Plots



Previous Plot 2/2

IPython Shell | Slides

```
In [2]: # Change the line plot below to a scatter plot
        plt.scatter(gdp_cap, life_exp)

        # Put the x-axis on a logarithmic scale
        plt.xscale('log')

        # Show plot
        plt.show()

In [3]:
```

Intro to pytho....pdf

Show all

rams

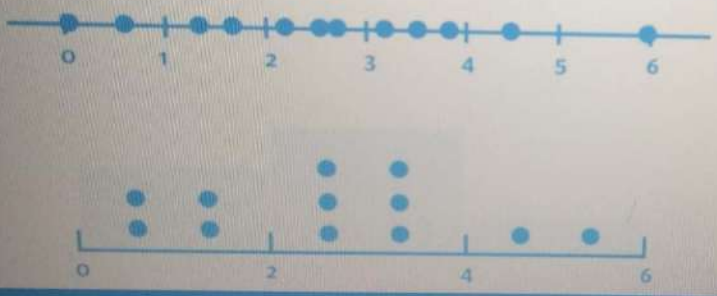
50 XP

DataCamp

Intermediate Python for Data Science

Histogram

- Explore dataset
- Get idea about distribution



*Untitled Document 1 - gedit

Open

Save

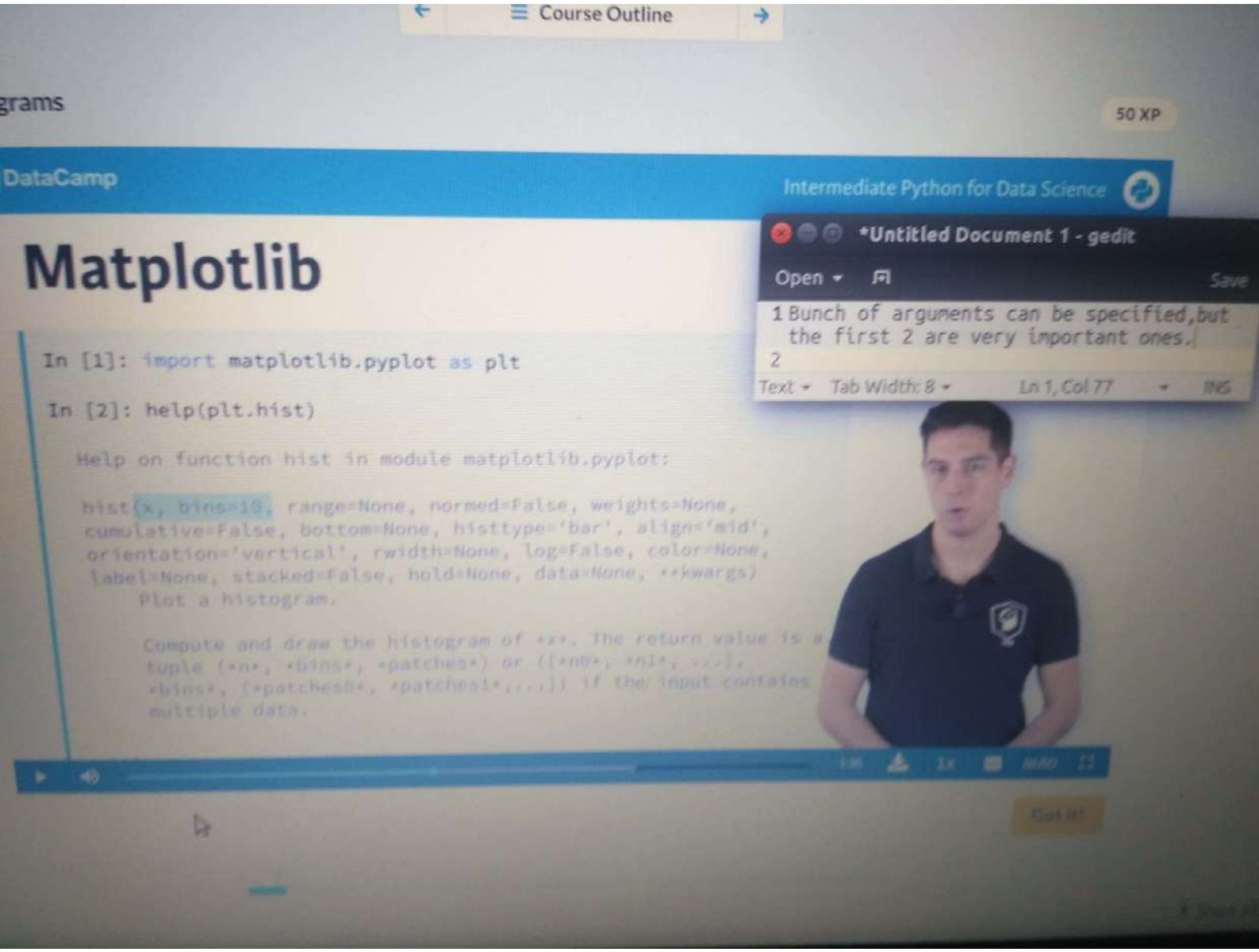
```
1 ) Dividing each line into equal parts
  called bins
2 ) Accordingly, distribute no. of
  datapoints in each bin
3 ) Draw a box for it.The height of each
  bin corresponds to no.of datapoints that
  fall in each bin.
4 ) Result is a Histogram.
5 result: there is more values below 2
  than above 4.
```

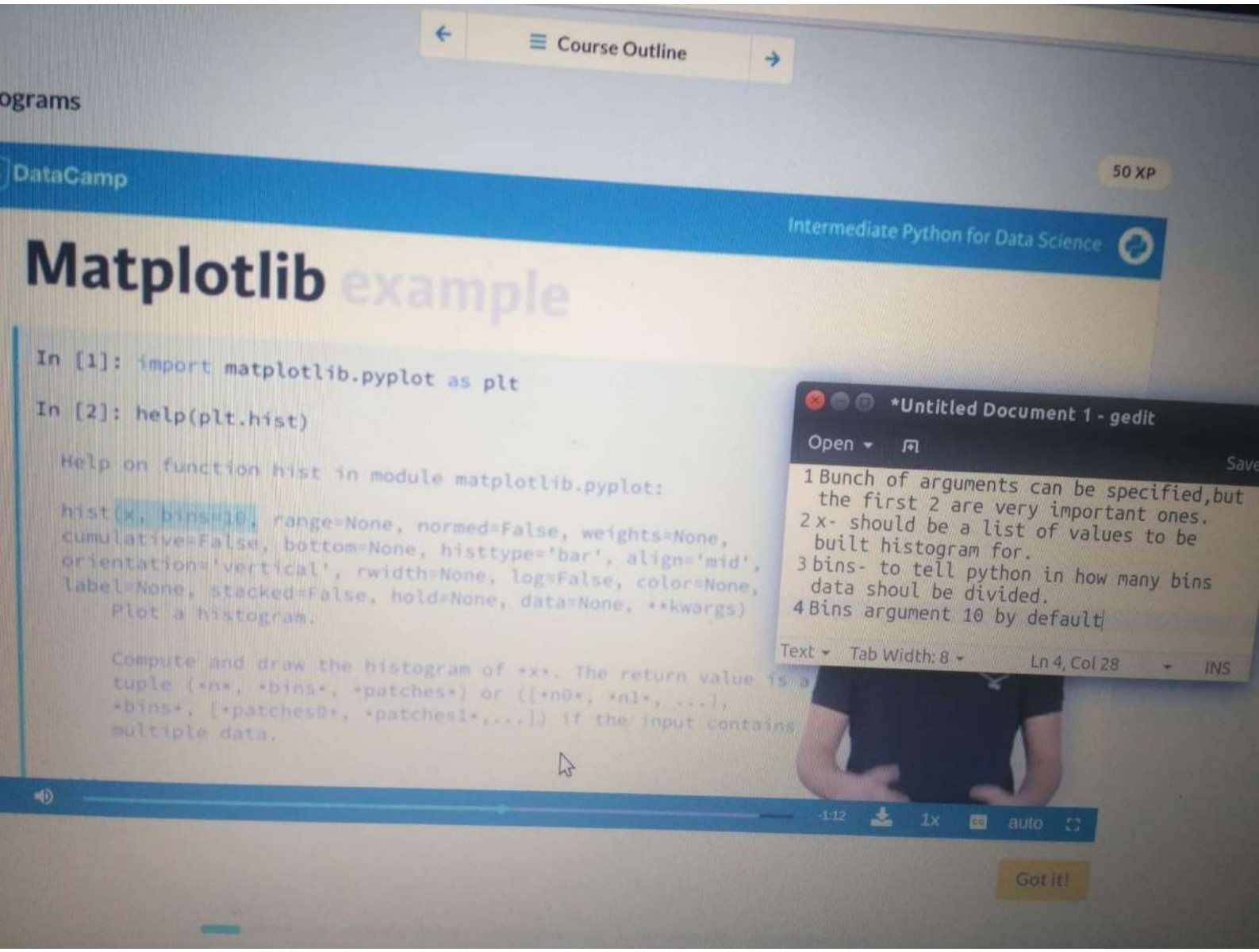
Text Tab Width: 8 Ln 5, Col 51 INS

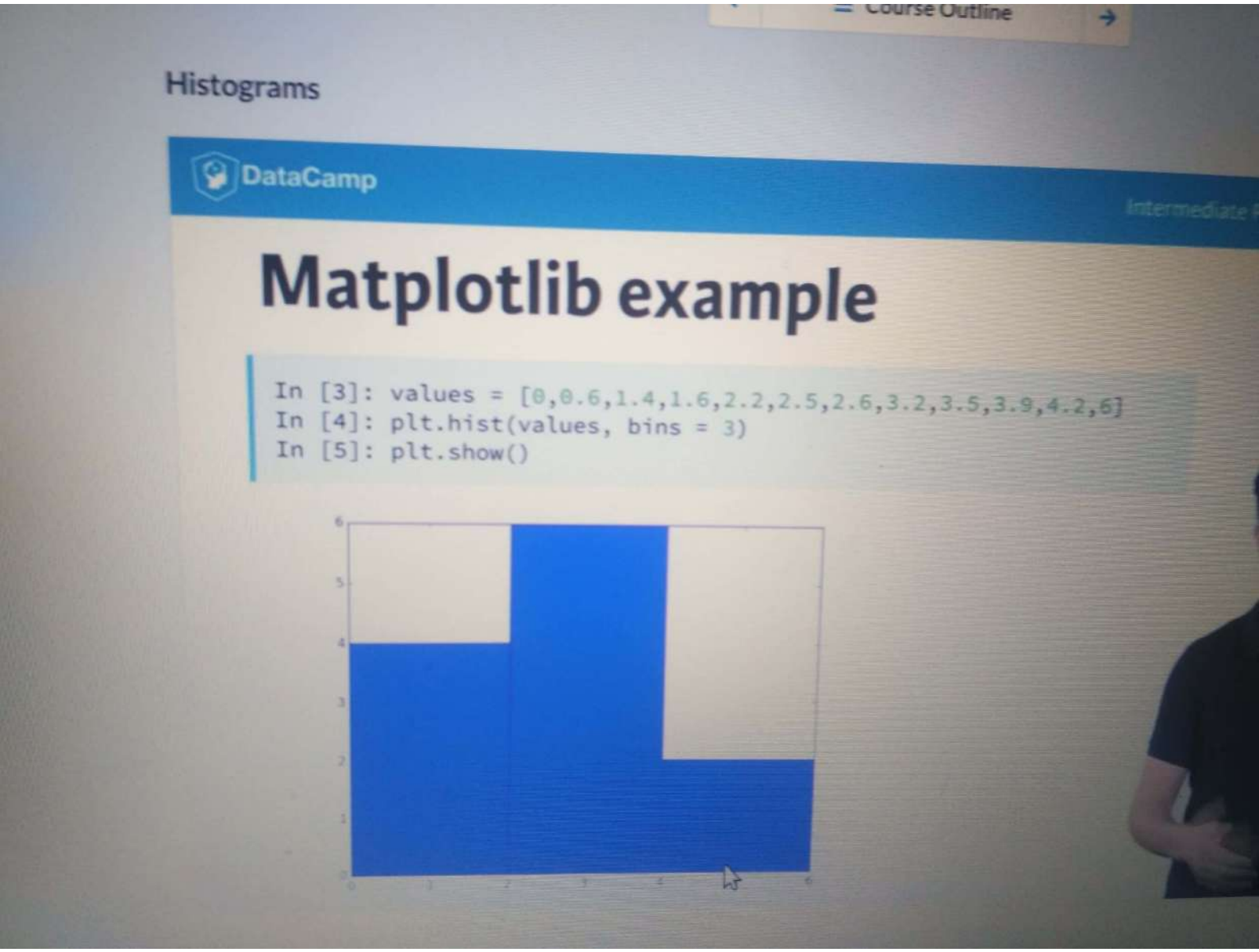
-1:50 1x auto

Got it!

Show all do







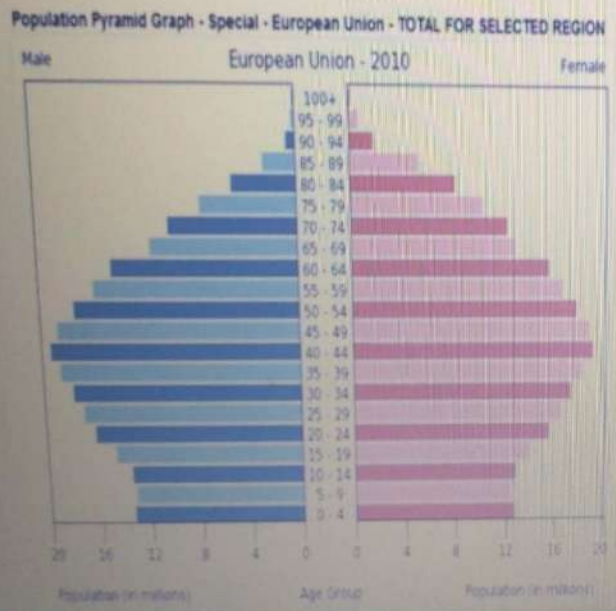
Course Outline

Histograms

DataCamp

Intermediate Python for Data Science

Population Pyramid



*Untitled Document 1 - gedit

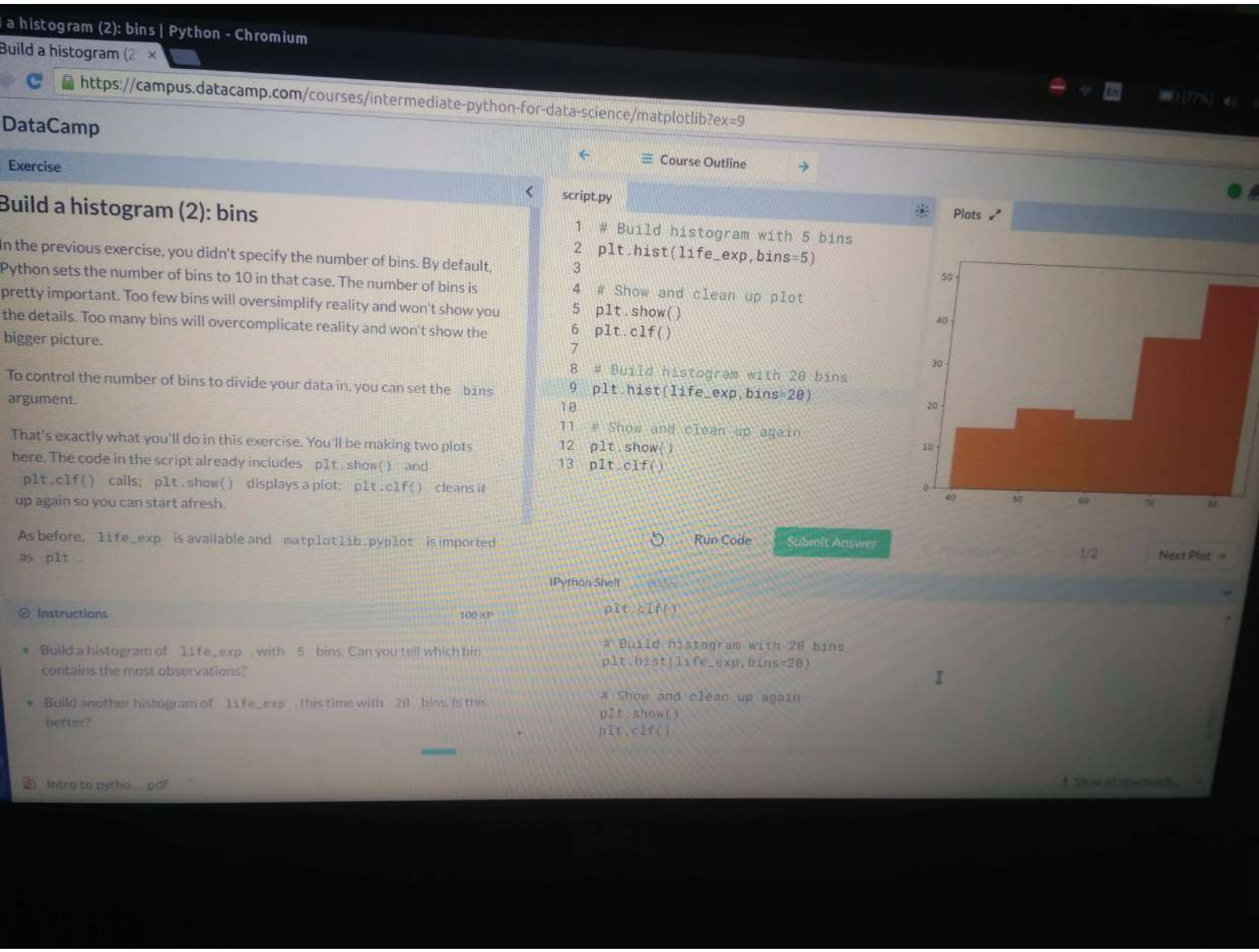
Open Save

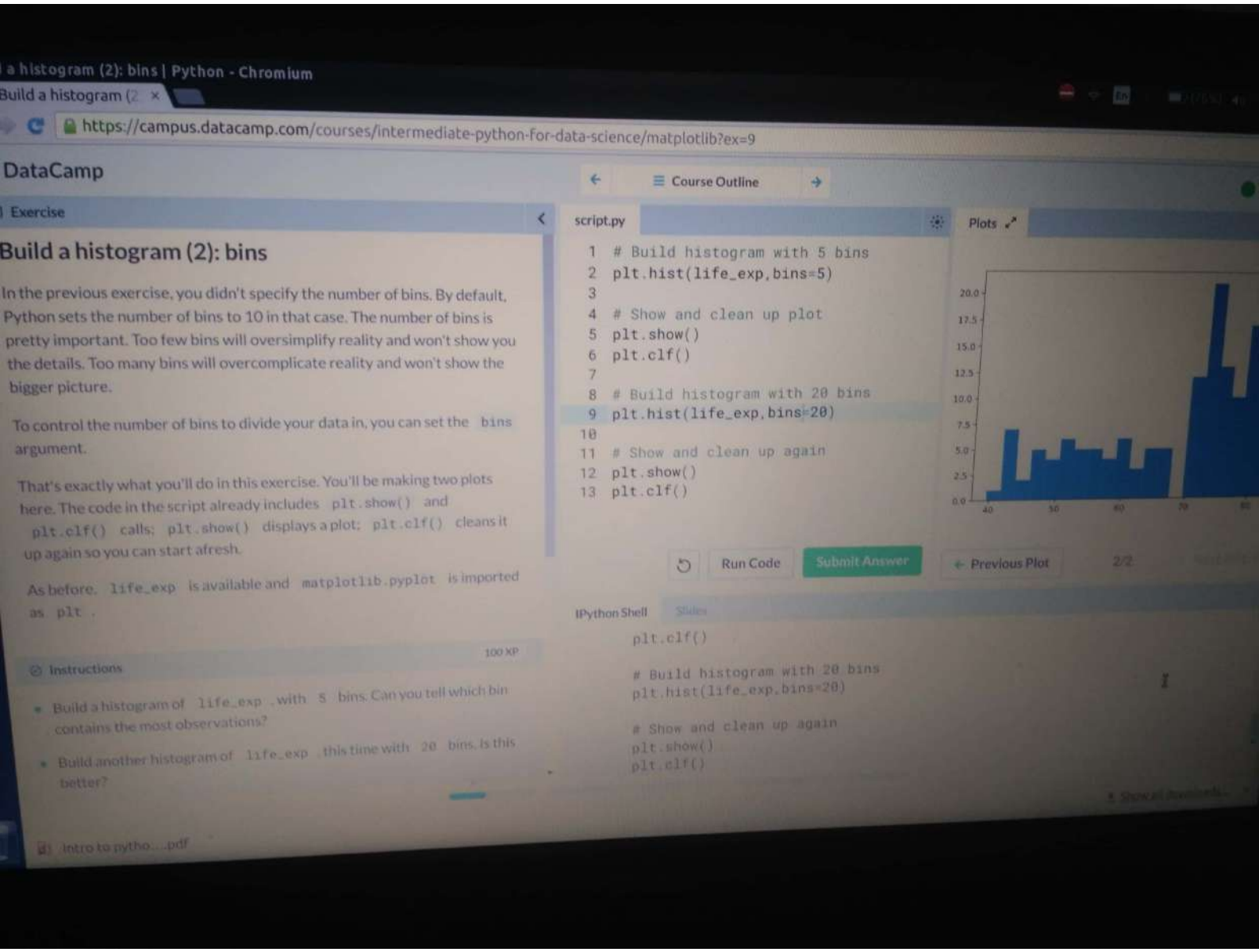
1 The population is highest for age group 40-44, i.e. 20 million population for both male and female.These are called baby-boomers
2 The histogram is flipped by 90(degree)

Text Tab Width: 8 Ln 1, Col 2 INS

-0:23 1x auto

Got it!





Choose the right plot (1) | Python - Chromium

Choose the right plc ×

https://campus.datacamp.com/courses/intermediate-python-for-data-science/

DataCamp

Exercise

Choose the right plot (1)

You're a professor teaching Data Science with Python, and you want to visually assess if the grades on your exam follow a particular distribution. Which plot do you use?

Instructions

50 XP

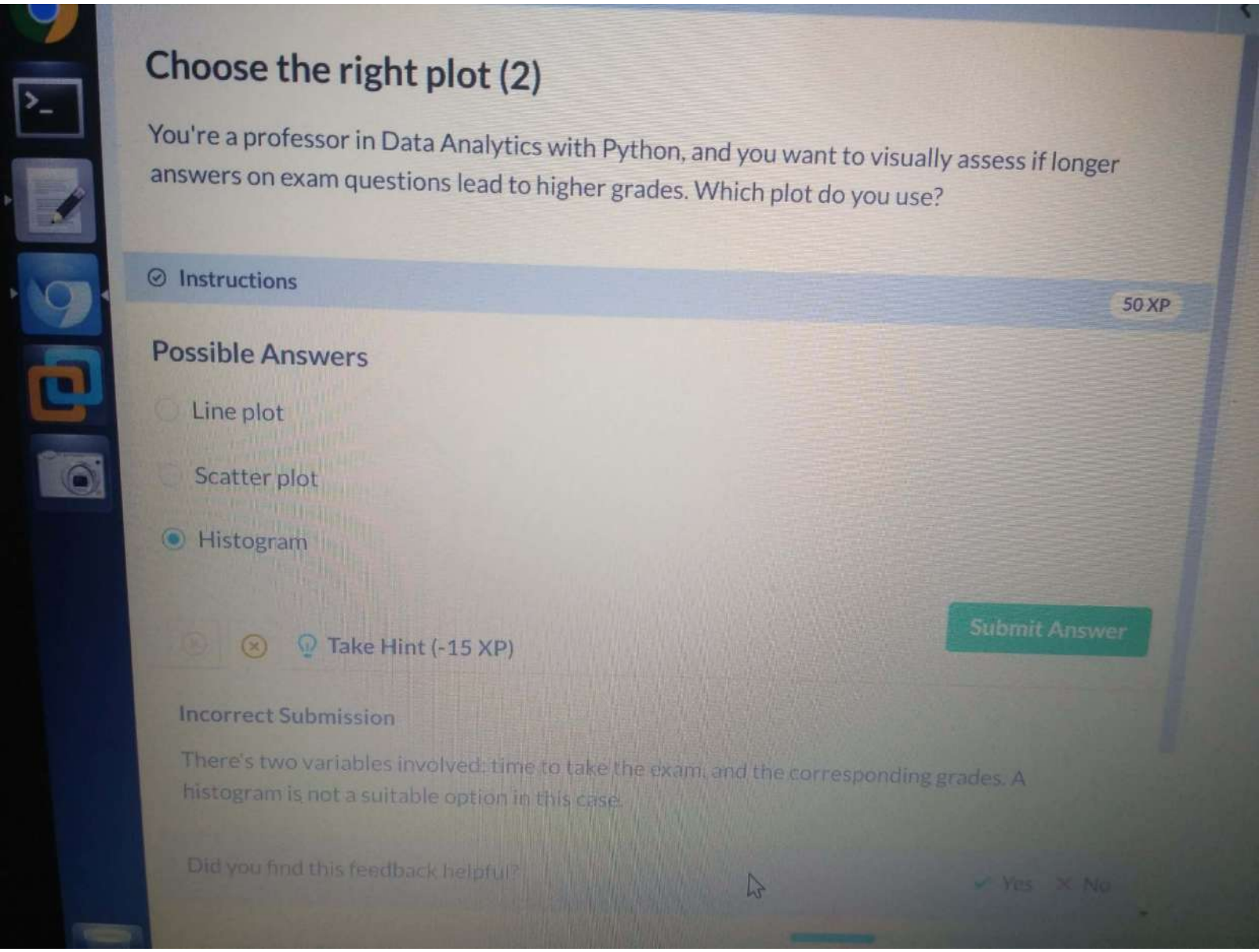
Possible Answers

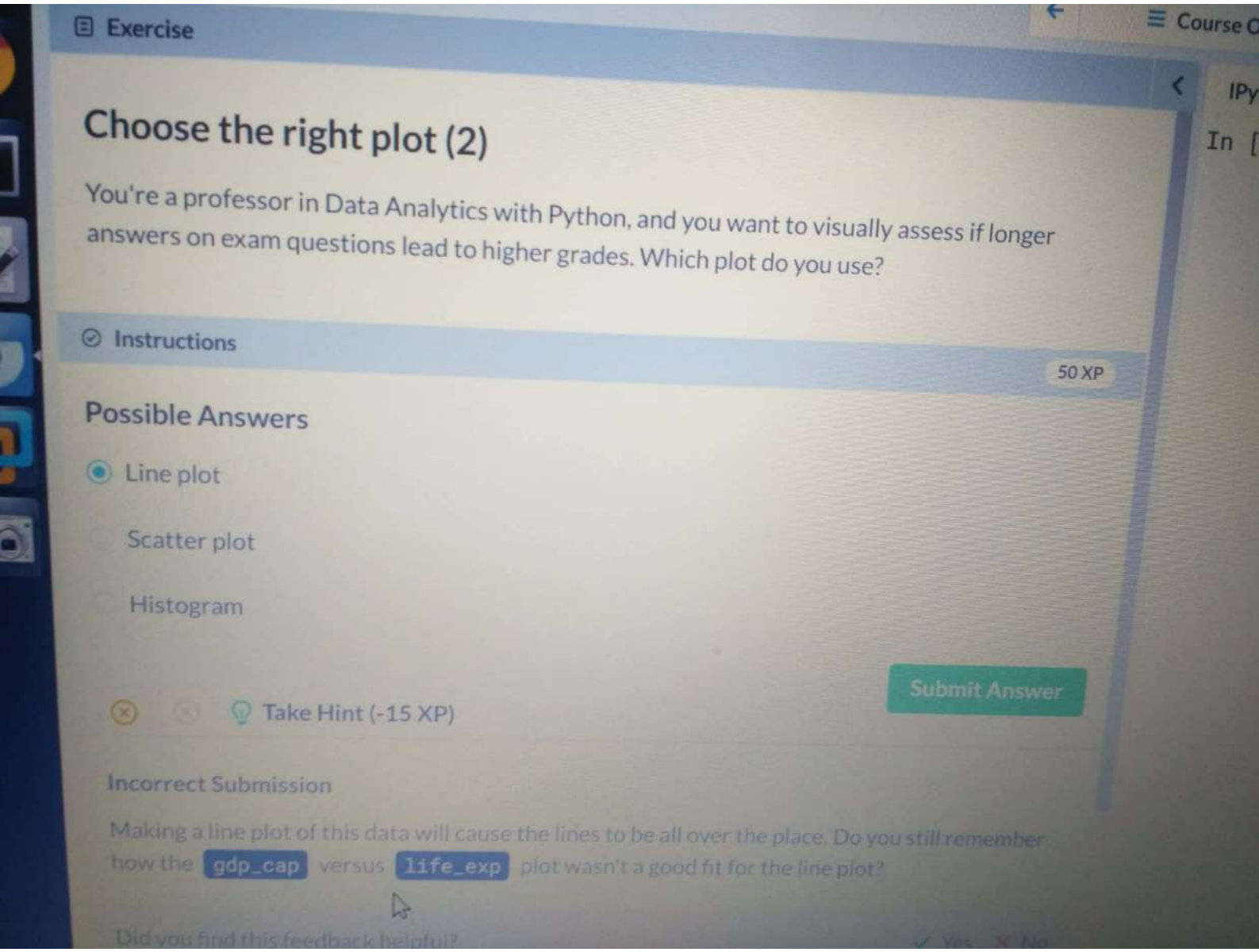
- Line plot
- Scatter plot
- Histogram

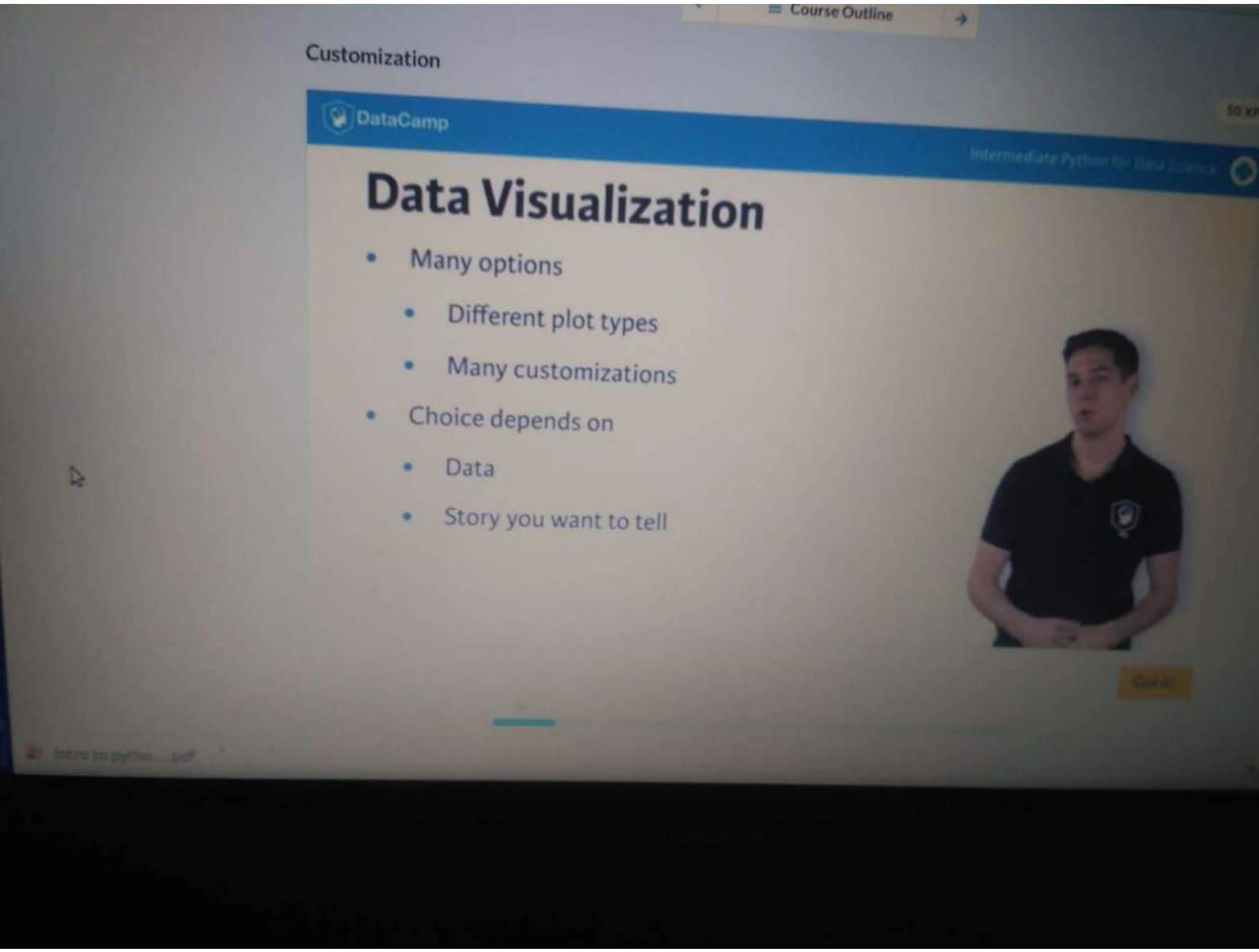
Take Hint (-15 XP)

Submit Answer

intro to pytho....pdf







Customization

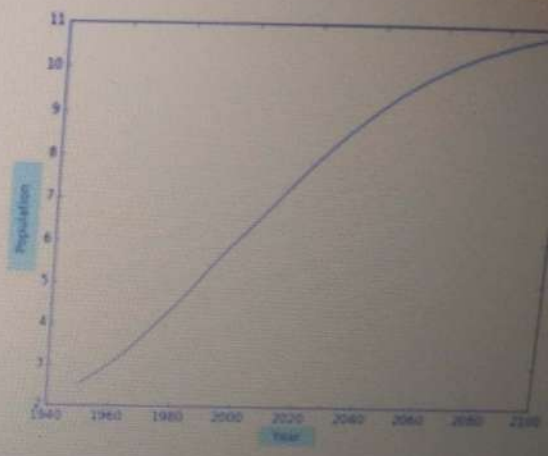
DataCamp

Intermediate Python for Data Scien

Axis labels

population.py

```
import matplotlib.pyplot as plt
year = [1950, 1951, 1952, ..., 2100]
pop = [2.538, 2.57, 2.62, ..., 10.85]

plt.plot(year, pop)

plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



-1:44 1x 360p

Got it!

Course Outline

Customization

50 XP

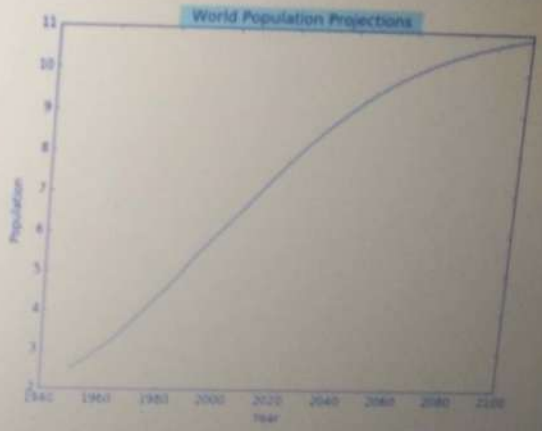
DataCamp

Intermediate Python for Data Science

Title

population.py

```
import matplotlib.pyplot as plt
year = [1950, 1951, 1952, ..., 2100]
pop = [2.538, 2.57, 2.62, ..., 10.85]

plt.plot(year, pop)

plt.xlabel('Year')
plt.ylabel('Population')
plt.title('World Population Projections')

plt.show()
```



-1:39 1x 360p

Got it!

Course Outline

Customization

50 XP

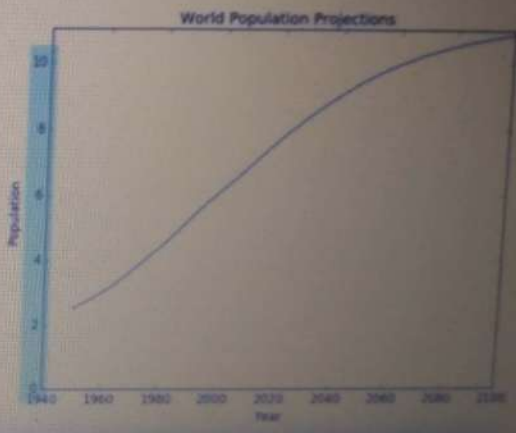
DataCamp

Intermediate Python for Data Science

Ticks

population.py

```
import matplotlib.pyplot as plt
year = [1950, 1951, 1952, ..., 2100]
pop = [2.538, 2.57, 2.62, ..., 10.85]

plt.plot(year, pop)

plt.xlabel('Year')
plt.ylabel('Population')
plt.title('World Population Projections')
plt.yticks([0, 2, 4, 6, 8, 10])

plt.show()
```



*Untitled Document 1 - gedit

Open Save

```
1 to have y-axis start from 0
2 use: yticks() function
3 The plot changes accordingly with
  intervals of 2 upto 10.
4 the curve shifts up
```

Text Tab Width: 8 Ln 4, Col 20 INS

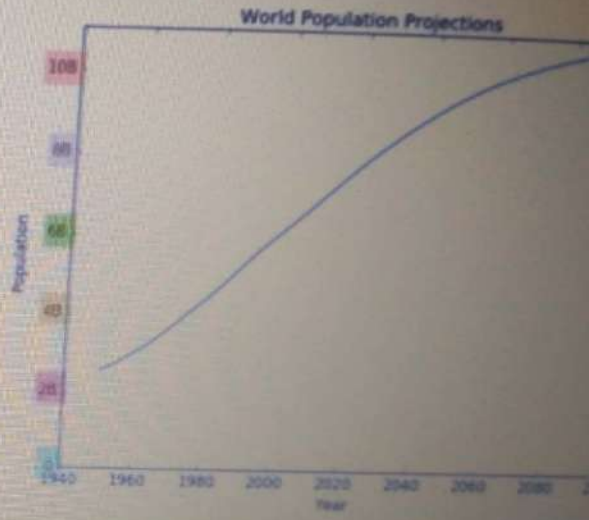
DataCamp

Intermediate Python for Data Sci

Ticks (2)

population.py

```
import matplotlib.pyplot as plt
year = [1950, 1951, 1952, ..., 2100]
pop = [2.538, 2.57, 2.62, ..., 10.85]

plt.plot(year, pop)

plt.xlabel('Year')
plt.ylabel('Population')
plt.title('World Population Projections')
plt.yticks([0, 2, 4, 6, 8, 10],
           ['0', '2B', '4B', '6B', '8B', '10B'])

plt.show()
```



*Untitled Document 1 - gedit

Open

1 now, to make clear population in billions.
2 we specify another list with B.
3 0 as 0B , 1 as 1B..
4 The labels change accordingly.
5

Text Tab Width: 8 Ln 4, Col 31

omization

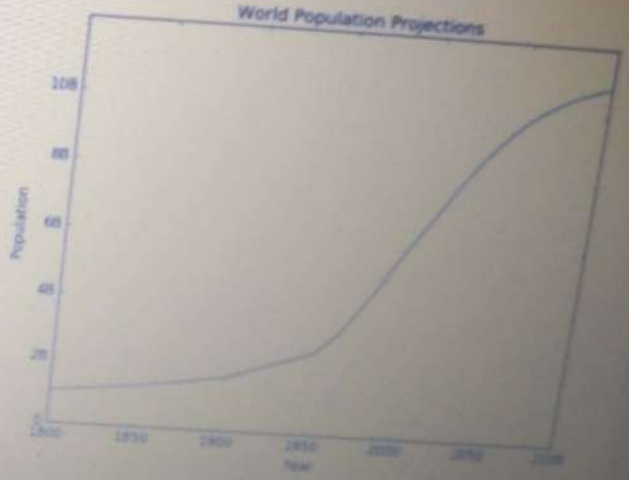
DataCamp

Intermediate Python for Data Science

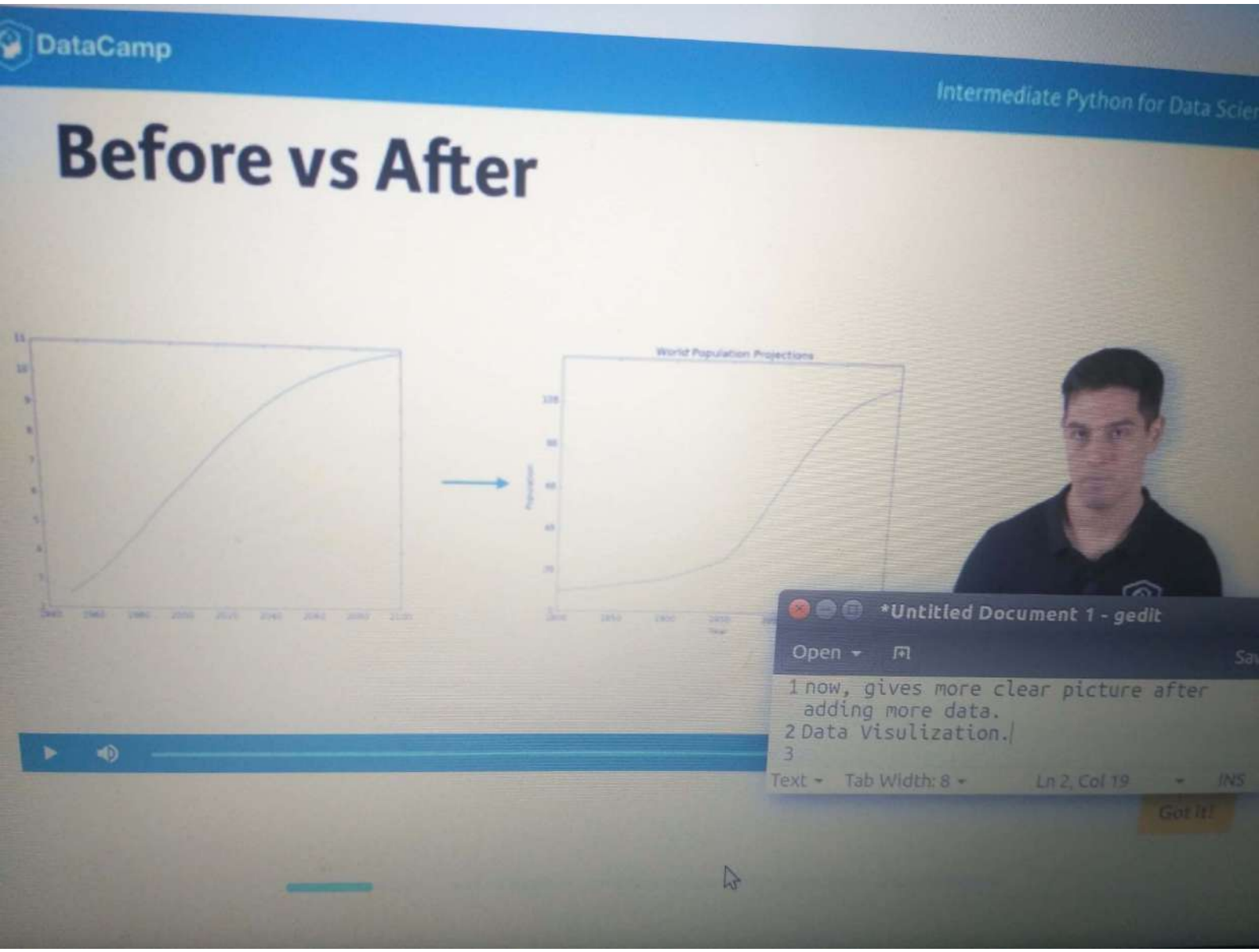
Add historical data

population.py

```
import matplotlib.pyplot as plt
year = [1950, 1951, 1952, ..., 2100]
pop = [2.538, 2.57, 2.62, ..., 10.85]

# Add more data
year = [1800, 1850, 1900] + year
pop = [1.0, 1.262, 1.650] + pop

plt.plot(year, pop)

plt.xlabel('Year')
plt.ylabel('Population')
plt.title('World Population Projections')
plt.yticks([0, 2, 4, 6, 8, 10],
           ['0', '2B', '4B', '6B', '8B', '10B'])

plt.show()
```



Got it!



https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=14

DataCamp

Exercise

Course Outline

Labels

It's time to customize your own plot. This is the fun part, you will see your plot come to life!

You're going to work on the scatter plot with world development data: GDP per capita on the x-axis (logarithmic scale), life expectancy on the y-axis. The code for this plot is available in the script.

As a first step, let's add axis labels and a title to the plot. You can do this with the `xlabel()`, `ylabel()` and `title()` functions, available in `matplotlib.pyplot`. This sub-package is already imported as `plt`.

Instructions

100 XP

- The strings `xlab` and `ylab` are already set for you. Use these variables to set the label of the x- and y-axis.
- The string `title` is also coded for you. Use it to add a title to the plot.
- After these customizations, finish the script with `plt.show()` to actually display the plot.

Take Hint (-30 XP)

Incorrect Submission

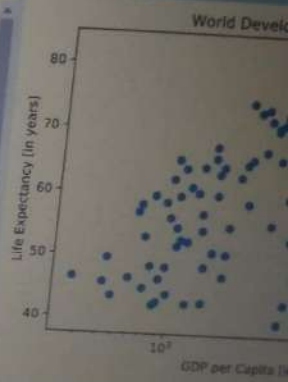
Did you call `plt.title()`?

script.py

```
# Basic scatter plot, log scale
plt.scatter(gdp_cap, life_exp)
plt.xscale('log')

# Strings
xlab = 'GDP per Capita [in USD]'
ylab = 'Life Expectancy [in years]'
title = 'World Development in 2007'

# Add axis labels
plt.xlabel(xlab)
plt.ylabel(ylab)
```

Run Code | Submit Answer

Plots



Previous Plot 4/4

IPython Shell | Slides

```
plt.xlabel=xlab
plt.ylabel=ylab

# Add title
plt.title=title

# After customizing, display the plot
plt.show()
```

In [3]:

Intro to pytho...pdf

Show all

Ticks | Python - Chromium

Ticks | Python

https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=15

DataCamp

Course Outline

Exercise

Ticks

The customizations you've coded up to now are available in the script, in a more concise form.

In the video, Filip has demonstrated how you could control the y-ticks by specifying two arguments:

```
plt.yticks([0,1,2], ["one","two","three"])
```

In this example, the ticks corresponding to the numbers 0, 1 and 2 will be replaced by one, two and three, respectively.

Let's do a similar thing for the x-axis of your world development chart, with the `xticks()` function. The tick values `1000` , `10000` and `100000` should be replaced by `1k` , `10k` and `100k` . To this end, two lists have already been created for you: `tick_val` and `tick_lab` .

Instructions

100 XP

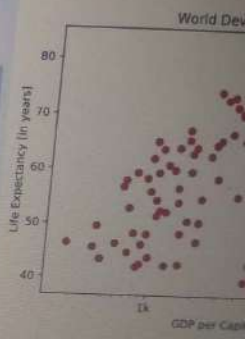
- Use `tick_val` and `tick_lab` as inputs to the `xticks()` function to make the the plot more readable.
- As usual, display the plot with `plt.show()` after you've added the customizations.

script.py

```
4  plt.xscale('log')
5  plt.xlabel('GDP per Capita [in USD]')
6  plt.ylabel('Life Expectancy [in years]'
   )
7  plt.title('World Development in 2007')
8
9  # Definition of tick_val and tick_lab
10 tick_val = [1000, 10000, 100000]
11 tick_lab = ['1k', '10k', '100k']
12 # Adapt the ticks on the x-axis
13 plt.xticks(tick_val, tick_lab)
14 plt.show()
15
```

Run Code | Submit Answer

Plots

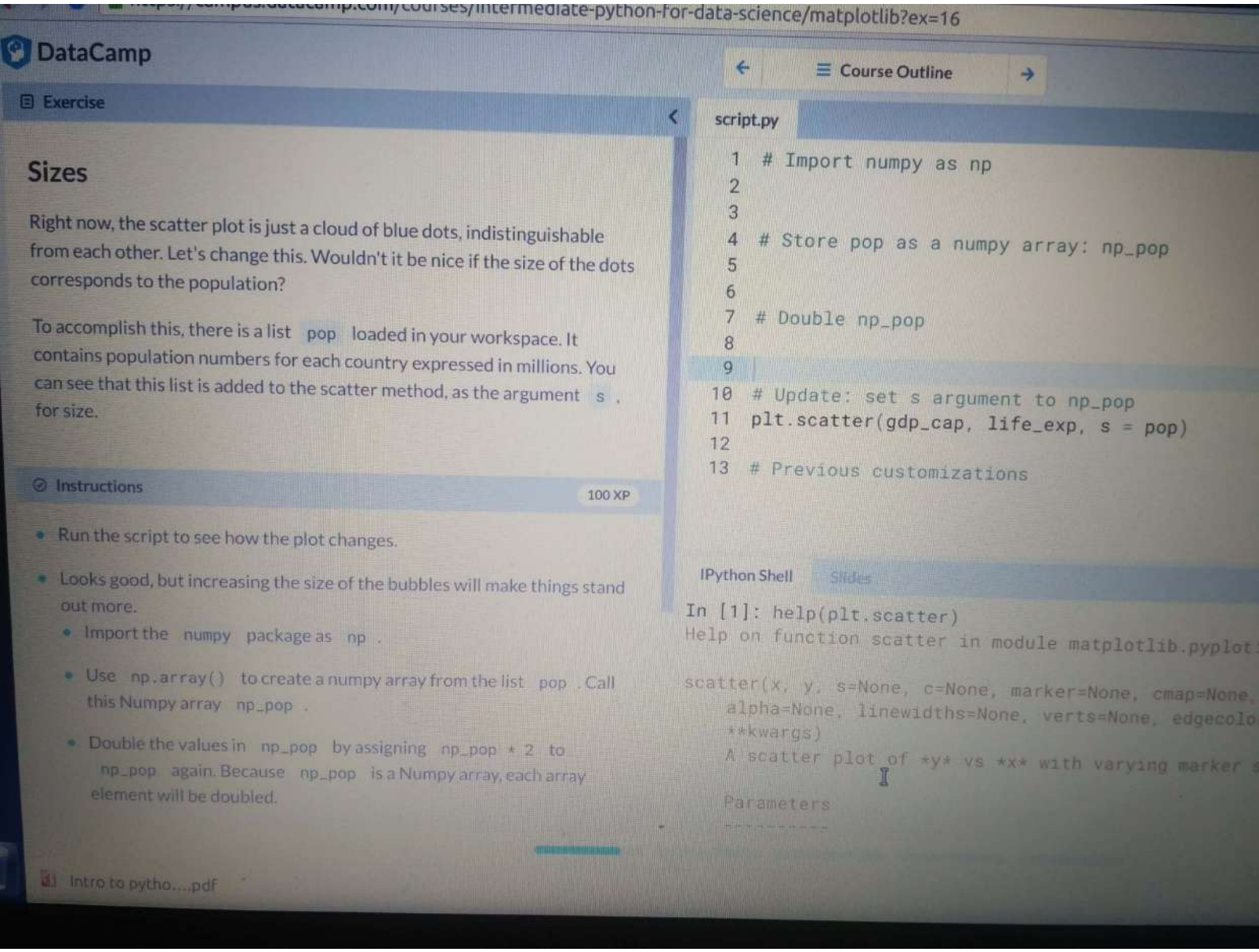


IPython Shell | Slides

```
plt.title('World Development in 2007')

# Definition of tick_val and tick_lab
tick_val = [1000, 10000, 100000]
tick_lab = ['1k', '10k', '100k']
# Adapt the ticks on the x-axis
plt.xticks(tick_val, tick_lab)
plt.show()
```

In [5]:

Intro to pytho....pdf



s | Python - Chromium

Sizes | Python ×

https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=16

DataCamp

Course Outline

Exercise

rom each other. Let's change this. Wouldn't it be nice if the size of the dots corresponds to the population?

To accomplish this, there is a list `pop` loaded in your workspace. It contains population numbers for each country expressed in millions. You can see that this list is added to the scatter method, as the argument `s`, for size.

Instructions 100 XP

- Run the script to see how the plot changes.
- Looks good, but increasing the size of the bubbles will make things stand out more.
  - Import the `numpy` package as `np`.
  - Use `np.array()` to create a numpy array from the list `pop`. Call this Numpy array `np_pop`.
  - Double the values in `np_pop` by assigning `np_pop * 2` to `np_pop` again. Because `np_pop` is a Numpy array, each array element will be doubled.
  - Change the `s` argument inside `plt.scatter()` to be `np_pop` instead of `pop`.

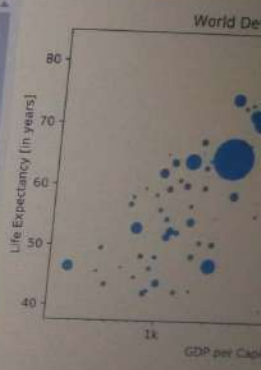
Take Hint (-30 XP)

script.py

```
1  # Import numpy as np
2  import numpy as np
3
4  # Store pop as a numpy array: np_pop
5  np_pop=np.array(pop)
6
7  # Double np_pop
8  #np_pop=np_pop*2
9
10 # Update: set s argument to np_pop
11 plt.scatter(gdp_cap, life_exp, s = pop
   )
12
```

Run Code Submit Answer

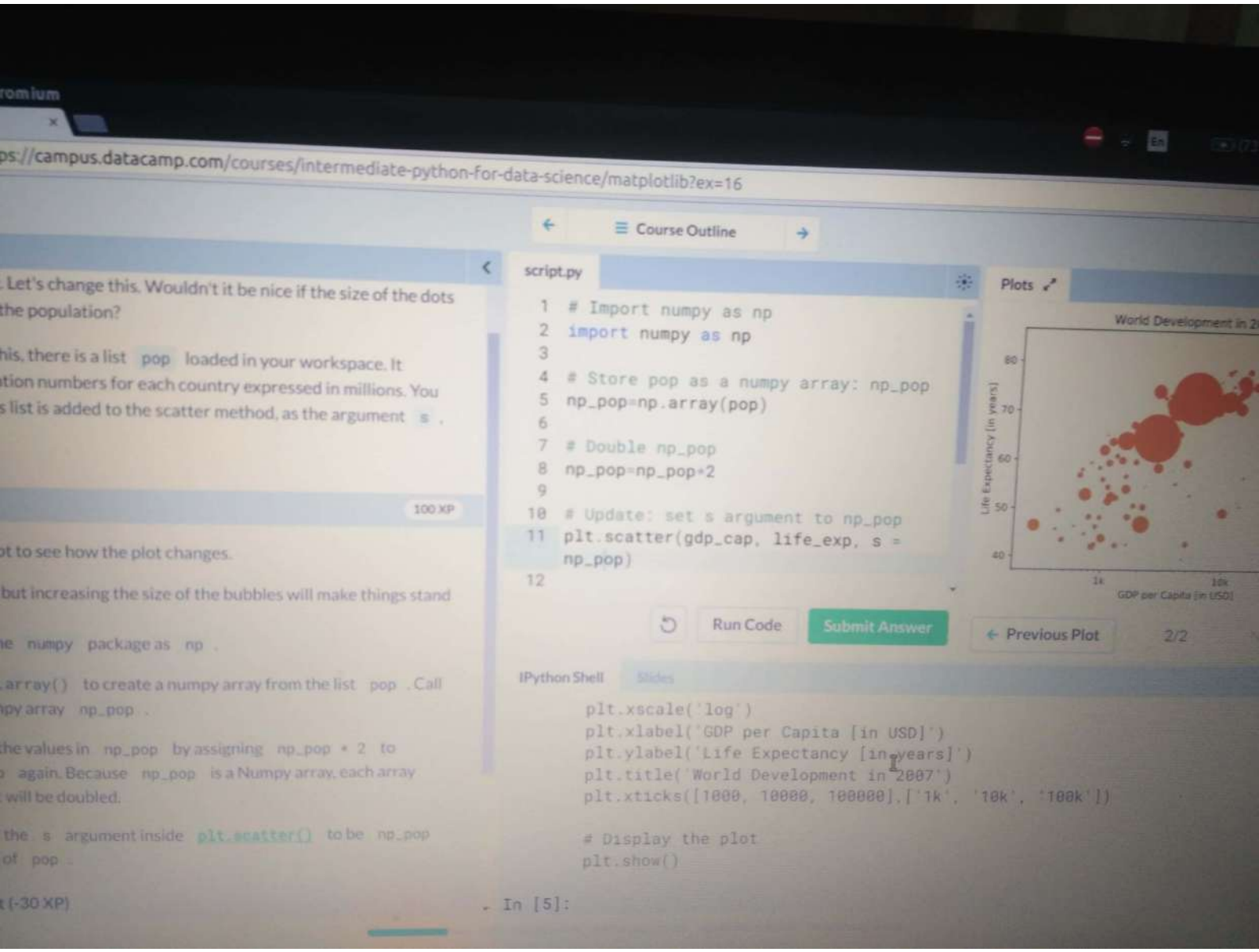
Plots

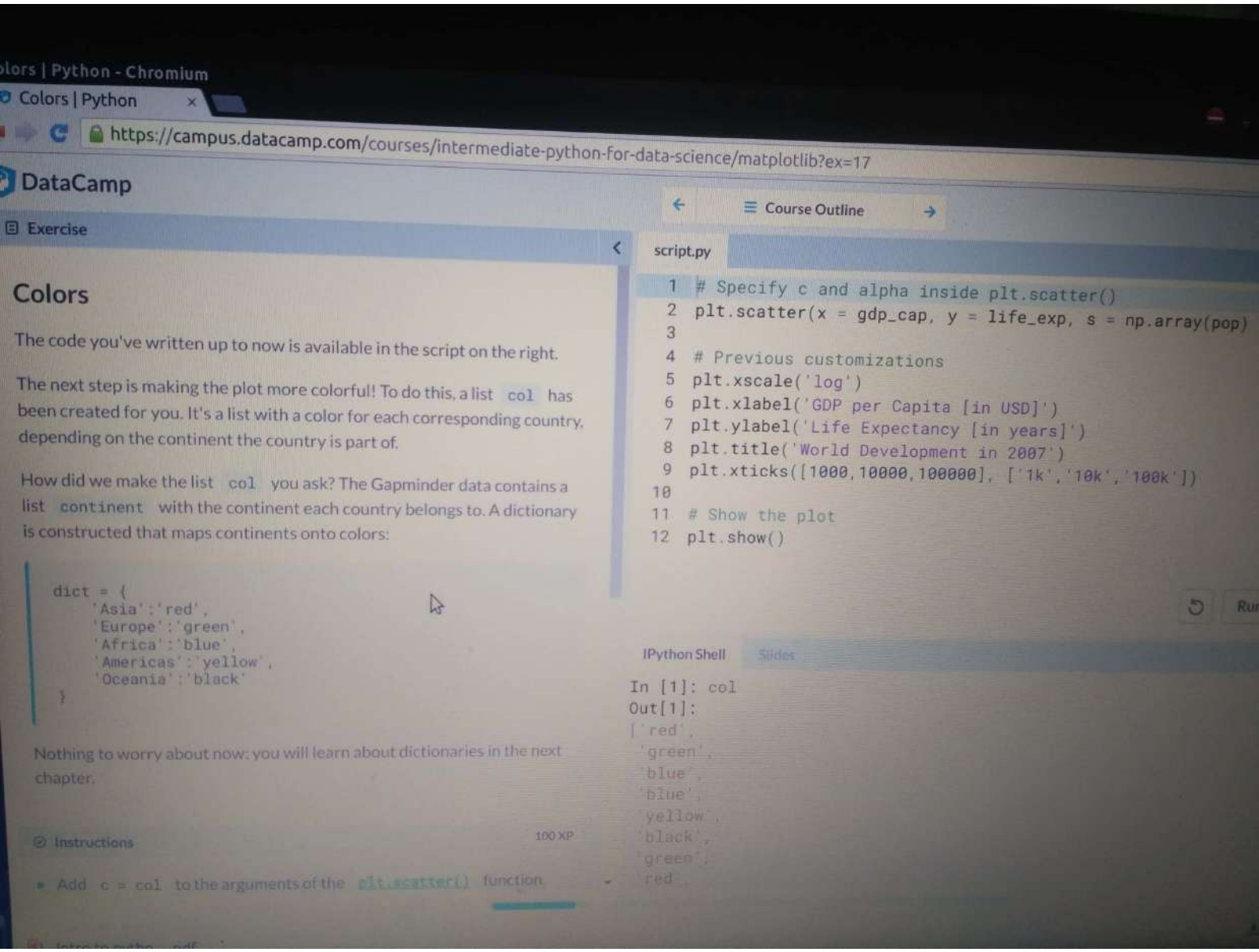


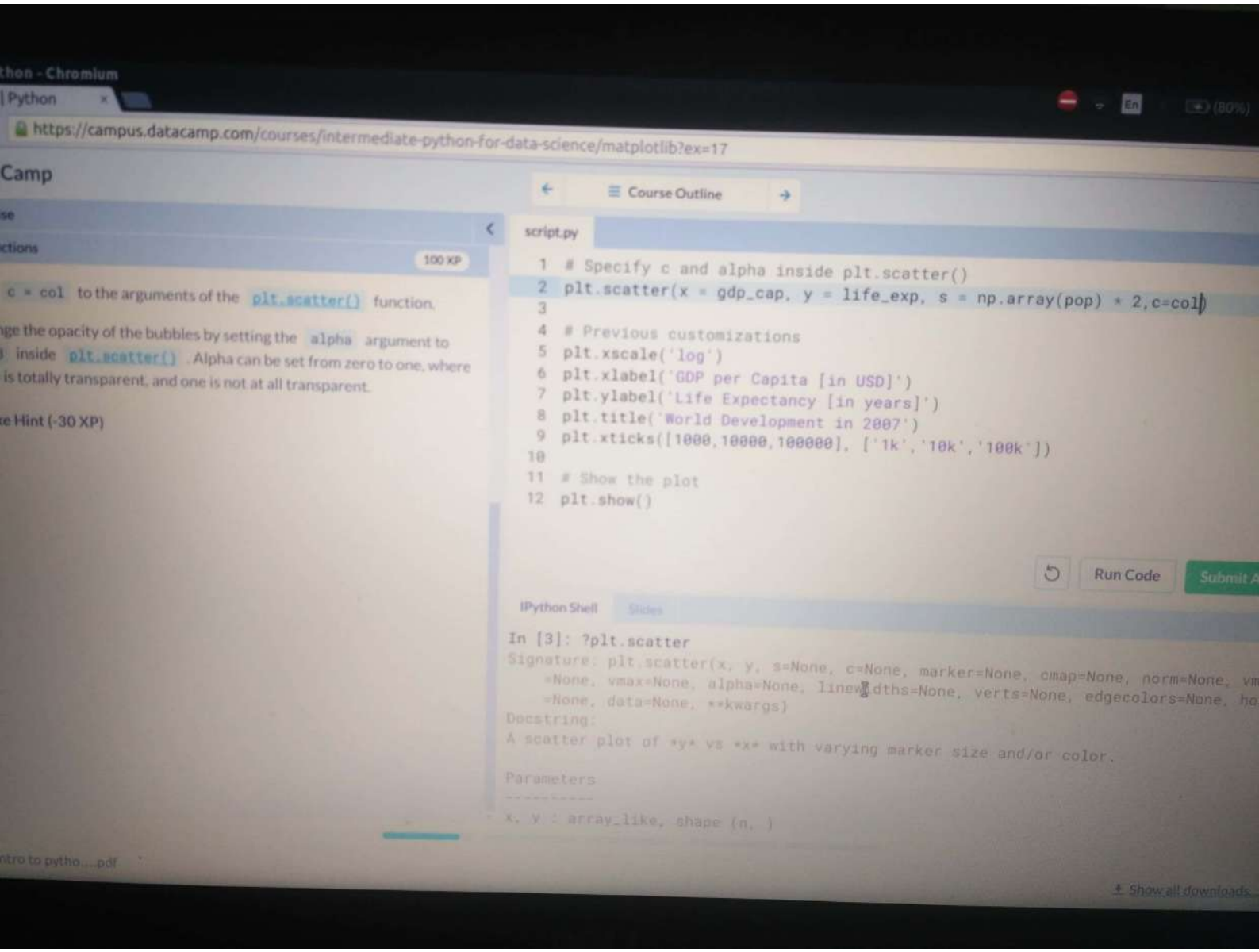
IPython Shell Slides

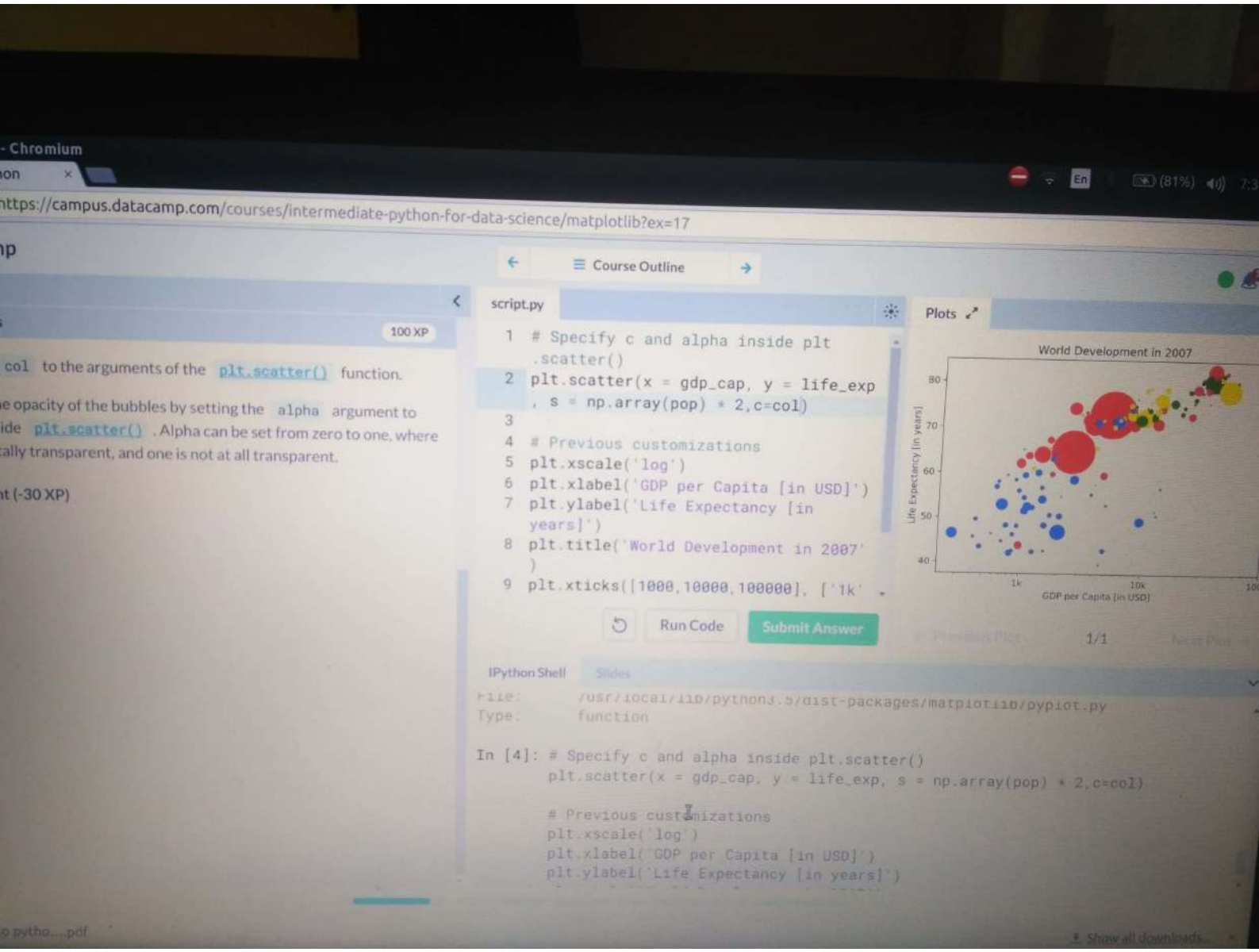
```
* All arguments with the following names: 'c', 'color', 'edgeco
  'facecolor', 'facecolors', 'linewidths', 's', 'x', 'y'.

In [2]: pop
Out[2]:
[31.889923,
 3.600523,
 33.333216,
 12.420476
```

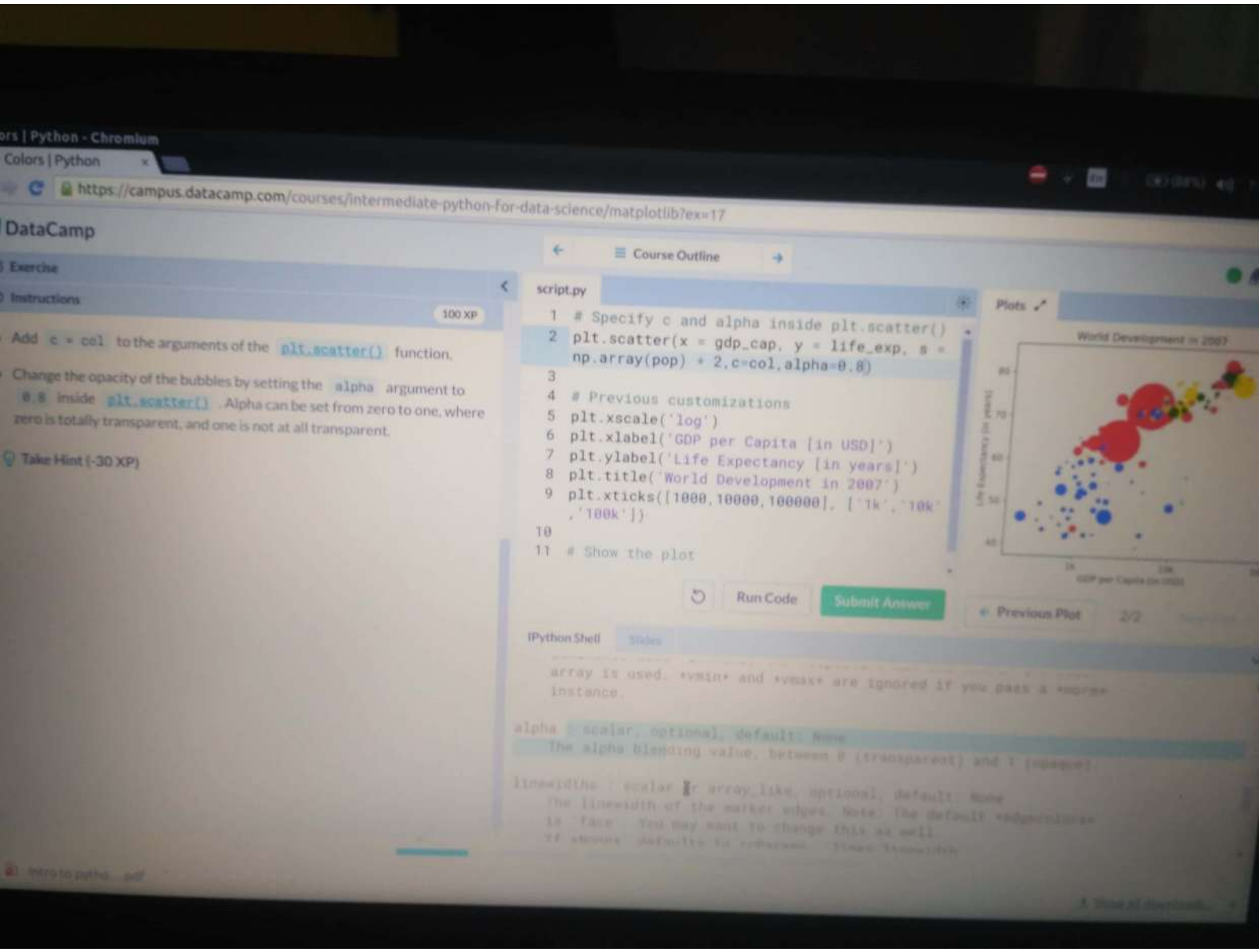
Intro to pytho....pdf

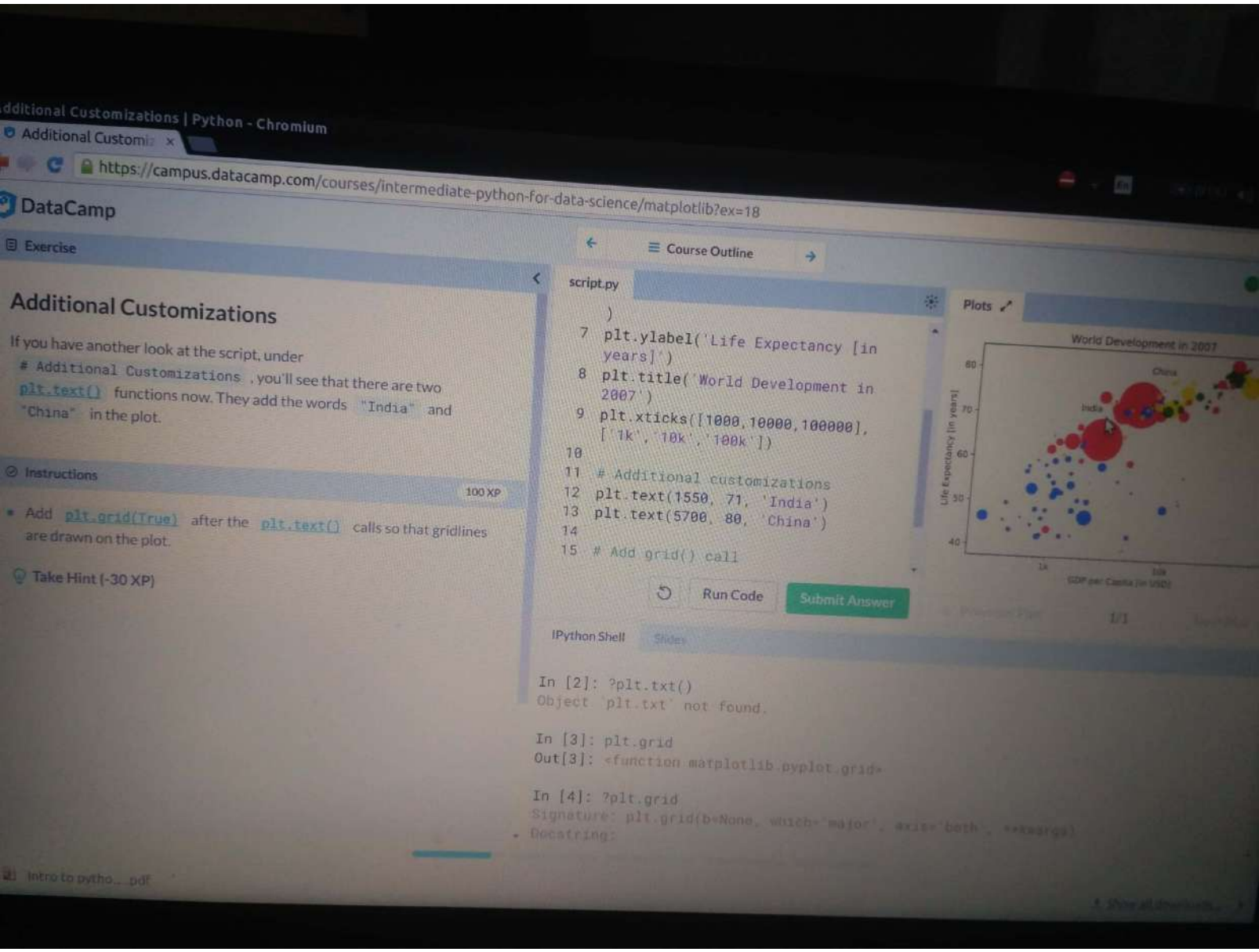


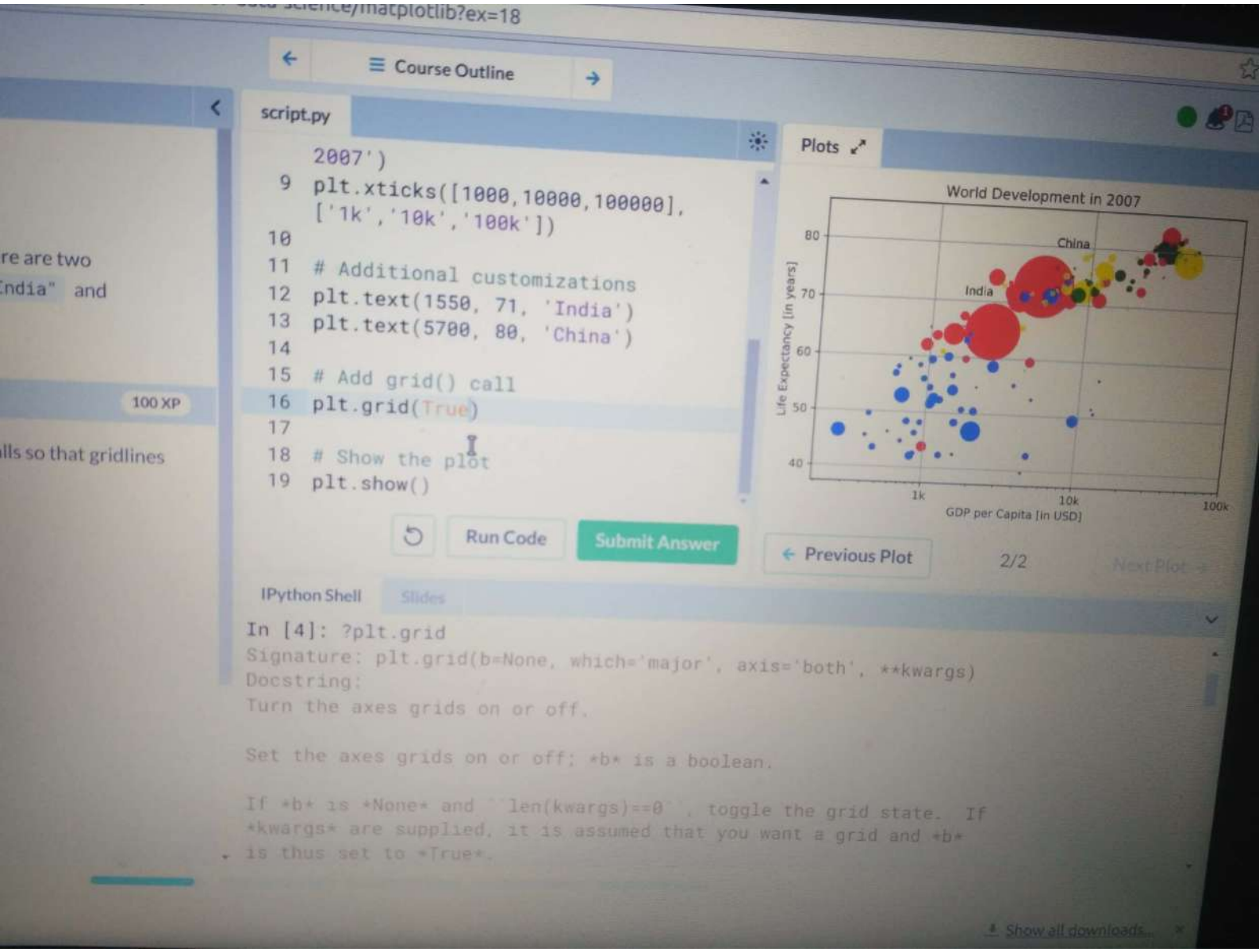












https://campus.datacamp.com/courses/intermediate-python-for-data-science/matplotlib?ex=19

DataCamp

Course Outline

Exercise

Interpretation

If you have a look at your colorful plot, it's clear that people live longer in countries with a higher GDP per capita. No high income countries have really short life expectancy, and no low income countries have very long life expectancy. Still, there is a huge difference in life expectancy between countries on the same income level. Most people live in middle income countries where difference in lifespan is huge between countries; depending on how income is distributed and how it is used.

What can you say about the plot?

Instructions

50 XP

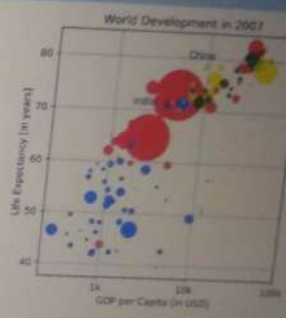
Possible Answers

- The countries in blue, corresponding to Africa, have both low life expectancy and a low GDP per capita.
- There is a negative correlation between GDP per capita and life expectancy.
- China has both a lower GDP per capita and low expectancy compared to India.

Take Hint (-15 XP)

Submit Answer

Plots



Previous Plot

1/1

IPython Shell

Slides

```
In [1]:
```

Intro to pytho....pdf