

introduction to Datascience course

The screenshot shows a DataCamp Python exercise window. The title bar indicates it's a 'Variable Assignment' exercise. The main area displays a script named 'SCRIPT.PY' with the following code:

```
1 # Create a variable savings
2
3
4 # Print out savings
5
```

Below the code, there's an input field containing the assignment statement `x = 5`. A note explains that you can now use the variable name `x` instead of the value `5`. Another note clarifies that `=` means assignment, not equality. The 'INSTRUCTIONS' section lists two tasks: creating a variable `savings` with value `100` and printing its value. A 'Take Hint (-30 XP)' button is available. On the right, an 'IPYTHON SHELL' tab is open, showing the prompt `In [1]:`. The bottom of the window shows a taskbar with various icons.

The screenshot shows the same DataCamp Python exercise window after the user has completed the assignment. The 'SCRIPT.PY' code now includes the assignment and print statements:

```
1 # Create a variable savings
2 savings = 100
3
4 # Print out savings
5 print(savings)
```

The 'IPYTHON SHELL' tab shows the execution results:

```
In [1]: # Create a variable savings
         savings = 100
         # Print out savings
         print(savings)
100
```

The 'INSTRUCTIONS' section is still present with the same tasks. The bottom of the window shows a taskbar with various icons.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window titled "Calculations with variables". In the "SCRIPT.PY" editor, the following code is displayed:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

In the "INSTRUCTIONS" section, there are three bullet points:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is also present.

The screenshot shows a DataCamp exercise window titled "Calculations with variables". In the "SCRIPT.PY" editor, the following code is displayed:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5 factor=1.1
6
7 # Calculate result
8 result=savings*factor**7
9
10 # Print out result
11 print(result)
```

In the "INSTRUCTIONS" section, there are three bullet points:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is also present.

An orange bar at the bottom indicates an "INCORRECT SUBMISSION".

In the "IPYTHON SHELL" tab, the following output is shown:

```
# Create a variable factor
factor=1.1

# Calculate result
result=savings*factor**7

# Print out result
print(result)
194.87171000000012
```

The "In [5]: |" input field is visible.

● Datacamp datascience course by

The screenshot shows a DataCamp course window titled "Calculations with variables". The "SCRIPT.PY" tab is active, displaying the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains the following text and a list of tasks:

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

The "IPYTHON SHELL" tab is visible at the bottom.

The screenshot shows the same DataCamp course window after the exercise has been completed. The "SCRIPT.PY" tab now includes the line `factor=1.1` and the final line `print(result)`. The "INSTRUCTIONS" section shows the completed code and a "Great!" message.

PRESS ENTER TO

instead of calculating with the actual values you can use variables instead. The `savings` variable you created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.

Become a power user!

SUBMIT ANSWER: **CTRL + SHIFT + ENTER**

See all keyboard shortcuts

The "IPYTHON SHELL" tab is active, showing the output of the script:

```
# calculate result
result=savings*factor**7

# Print out result
print(result)
194.87171000000012

<script.py> output:
194.87171000000012
```

The "IPYTHON SHELL" tab has "Run Code" and "Submit Answer" buttons.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

In [1]: |

fractional part, separated by a point. `factor`, with the value `1.10`, is an example of a float.

Next to numerical data types, there are two other very common data types:

- `str`, or string: a type to represent text. You can use single or double quotes to build a string.
- `bool`, or boolean: a type to represent logical values. Can only be `True` or `False` (the capitalization is important!).

INSTRUCTIONS 100 XP

- Create a new string, `desc`, with the value `"compound interest"`.
- Create a new boolean, `profitable`, with the value `True`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable desc
2
3 str desc="compound interest"
4 # Create a variable profitable
5 bool profitable=true
```

IPYTHON SHELL

In [1]: |

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

In [1]: |

fractional part, separated by a point. `factor`, with the value `1.10`, is an example of a float.

Next to numerical data types, there are two other very common data types:

- `str`, or string: a type to represent text. You can use single or double quotes to build a string.
- `bool`, or boolean: a type to represent logical values. Can only be `True` or `False` (the capitalization is important!).

INSTRUCTIONS 100 XP

- Create a new string, `desc`, with the value `"compound interest"`.
- Create a new boolean, `profitable`, with the value `True`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable desc
2
3 desc = "compound interest"
4 # Create a variable profitable
5 profitable = True
```

IPYTHON SHELL

In [10]: `desc = "compound interest"`

SyntaxError: EOL while scanning string literal

In [11]: # Create a variable desc

```
desc = "compound interest"
# Create a variable profitable
profitable = True
```

In [11]: |

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise instructions state: "Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this: `100 * 1.10 ** 7`". It then says: "Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!". The "INSTRUCTIONS" section contains three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available. The "SCRIPT.PY" code editor shows the following script:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "IPYTHON SHELL" tab is active, showing the command `In [1]:` followed by a blank input field.

The screenshot shows a DataCamp Python course exercise titled "Other variable types". It explains that a float is a number with a fractional part, separated by a point. An example given is `1.10`. It states: "Next to numerical data types, there are two other very common data types: `str`, or string, a type to represent text. You can use single or double quotes to build a str! **Nice!**" and "`bool`, or boolean, a type to represent logical values. Can only be `True` or `False` (with `False` being important).". A "PRESS ENTER TO Continue" button is present. The "INSTRUCTIONS" section lists two tasks:

- Create a new string `desc` with the value "compound interest".
- Create a new boolean `profitable`, with the value `True`.

A "Take Hint (-30 XP)" button is available. A "Become a power user!" sidebar offers to help with keyboard shortcuts. The "SCRIPT.PY" code editor shows the following script:

```
1 # Create a variable desc
2
3 desc = "compound interest"
4 # Create a variable profitable
5 profitable = True
```

The "IPYTHON SHELL" tab is active, showing the command `In [10]: # Create a variable desc` followed by the error message `SyntaxError: EOL while scanning string literal`. The command `desc = "compound interest"` is highlighted with a cursor at the end of the string. The "SLIDES" tab is also visible. A "Ctrl+Shift+Enter" keybinding and a "Submit Answer" button are shown.

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. On the left, there's an 'EXERCISE' panel for 'Calculations with variables'. It contains instructions and a code editor with a script named 'SCRIPT.PY' containing the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the code editor is a 'INSTRUCTIONS' section with 100 XP available. It lists three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

There's also a 'Take Hint (-30 XP)' button.

On the right, there's an 'IPYTHON SHELL' tab with the prompt 'In [1]: |' and an 'SLIDES' tab.

The bottom of the screen shows a Windows taskbar with various icons and a search bar.

The screenshot shows another DataCamp course exercise titled 'Guess the type'. The 'INSTRUCTIONS' section says: '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`. If you forgot the type of this variable, To determine the type of `a`, simply execute `type(a)`'. It has +50 XP available.

The 'IPYTHON SHELL' tab shows the following interactions:

```
In [1]: type(a)
Out[1]: float

In [2]: type(b)
Out[2]: str

In [3]: type(c)
Out[3]: bool

In [4]: type(d)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    type(d)
NameError: name 'd' is not defined

In [5]: |
```

The bottom of the screen shows a Windows taskbar with various icons and a search bar.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

In [1]: |

Operations with other types

Filip mentioned that different types behave differently in Python.

When you sum two strings, for example, you'll get different behavior than when you sum two integers or two booleans.

In the script some variables with different types have already been created. It's up to you to use them.

INSTRUCTIONS 100 XP

- Calculate the product of `savings` and `factor`. Store the result in `year1`.
- What do you think the resulting type will be? Find out by printing out the type of `year1`.
- Calculate the sum of `desc` and `desc` and store the result in a new variable `doubledesc`.
- Print out `doubledesc`. Did you expect this?

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create ...
2 desc = "compound interest"
3
4 # Assign product of factor and savings to year1
5
6 year1=factor*savings
7 # Print the type of year1
8 print(year1)
9
10
11
12
13 # Assign sum of desc and desc to doubledesc
14 doubledesc=desc+desc
15
16 # Print out doubledesc
17 print(doubledesc)
```

IPYTHON SHELL

In [1]: |

Run all/selected code

Run Code **Submit Answer**

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window titled "Calculations with variables". The "SCRIPT.PY" tab is active, displaying the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains the following text and a list of tasks:

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

The window also includes an "IPYTHON SHELL" tab and a "SLIDES" tab. The system tray at the bottom shows a Windows 10 interface with various icons and the date/time.

The screenshot shows a DataCamp exercise window titled "Type conversion | Python". The "SCRIPT.PY" tab is active, displaying the following code:

```
1 # Definition of savings and result
2 savings = 100
3 result = 100 * 1.10 ** 7
4
5 # Fix the printout!
6 print(savings+result)
7 # Definition of pi_string
8 pi_string = "3.1415926"
9
10 # Convert pi_string into float: pi_float
11 float(pi_string)
12
```

The "INSTRUCTIONS" section contains the following text and a list of tasks:

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.
- Fix the code on the right such that the printout runs without errors; use the function `str()` to convert the variables to strings.
- Convert the variable `pi_string` to a float and store this float as a new variable, `pi_float`.

Take Hint (-30 XP)

The window includes an "IPYTHON SHELL" tab and a "SLIDES" tab. The system tray at the bottom shows a Windows 10 interface with various icons and the date/time.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

```
In [1]: |
```

Specified need `str()`, to convert a value into a string. `str(savings)` for example, converts the float `savings` to a string.

Similarly, 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.
- Fix the code on the right such that the printout runs without errors; use the function `str()` to convert the variables to strings.
- Convert the variable `pi_string` to a float and store this float as a new variable, `pi_float`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Definition of savings and result
2 savings = 100
3 str(savings)
4 result = 100 * 1.10 ** 7
5
6 # Fix the printout
7 print("I started with $" + str(savings) + " and now have $" + str(result) + ". Awesome!")
8
9 # Definition of pi_string
10 pi_string = "3.1415926"
11
12 # Convert pi_string into float: pi_float
13 pi_float=float(pi_string)
14
```

IPYTHON SHELL

```
result = 100 * 1.10 ** 7
# Fix the printout
print("I started with $" + str(savings) + " and now have $" + str(result) + ". Awesome!")

# Definition of pi_string
pi_string = "3.1415926"

# Convert pi_string into float: pi_float
pi_float=float(pi_string)
I started with $100 and now have $194.87171000000012. Awesome!
```

```
In [3]: |
```

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window titled "Calculations with variables". The "SCRIPT.PY" tab contains the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section provides the following task:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available.

The "IPYTHON SHELL" tab shows the command `In [1]: |`.

The screenshot shows a DataCamp exercise window titled "Type conversion | Python". The "SCRIPT.PY" tab contains the following code:

```
1 # Definition of savings and result
2 savings = 100
3 result = 100 * 1.10 ** 7
4
5 # Fix the printout
6 print("I started with $" + str(savings) + " and now have $" + str(result) + ". Awesome!")
7
8 # Definition of pi_string
9 pi_string = "3.1415926"
10
11 # Convert pi_string into float: pi_float
12 pi_float=float(pi_string)
13
```

The "INSTRUCTIONS" section provides the following task:

- Convert the variables `savings` and `result` into strings. Print them as a new variable.

A "PICK ANSWER" button is available.

A "Take Hint (-30 XP)" button is available.

The "IPYTHON SHELL" tab shows the command `In [4]: |` and the output:

```
print("I started with $" + str(savings) + " and now have $" + str(result) + ". Awesome!")

# Definition of pi_string
pi_string = "3.1415926"

# Convert pi_string into float: pi_float
pi_float=float(pi_string)

I started with $100 and now have $194.87171000000012. Awesome!
```

● Datacamp datascience course by

The screenshot shows a DataCamp exercise titled "Calculations with variables". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- An "INSTRUCTIONS" section with 100 XP available, containing three tasks:
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- A "Take Hint (-30 XP)" button.
- An "IPYTHON SHELL" tab showing the command "In [1]: |".

The screenshot shows a DataCamp exercise titled "Can Python handle everything?". The exercise content includes:

- A message stating "You have finished the chapter 'Python Basics'!" with a "Please give a rating:" prompt and a 5-star rating icon.
- A "PICK A RATING" section with options: "1 star", "2 stars", "3 stars", "4 stars", and "5 stars".
- A "Take Hint (-10 XP)" button.
- A "PRESS ENTER TO" button with a "Continue" option.
- An "IPYTHON SHELL" tab showing several commands and their outputs:
 - In [1]: "I can add integers, like " + str(5) + " to strings."
 - Out[1]: "I can add integers, like 5 to strings."
 - In [2]: "I said " + ("Hey " * 2) + "Hey!"
 - Out[2]: "I said Hey Hey Hey!"
 - In [3]: "The correct answer to this multiple choice exercise is answer number " + 2
 - Traceback (most recent call last):
File "<stdin>", line 1, in <module>
"The correct answer to this multiple choice exercise is answer number " + 2
 - TypeError: Can't convert 'int' object to str implicitly
 - In [4]: True + False
 - Out[4]: 1
 - In [5]: |

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

```
In [1]: |
```

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)

SCRIPT.PY

```
1 # area variables (in square meters)
2 hall = 11.25
3 kit = 18.0
4 liv = 28.0
5 bed = 18.75
6 bath = 9.50
7
8 # Create list areas
9
10 areas=["hall","kit","liv","bed","bath"]
11 # Print areas
12 print(areas)
13
```

IPYTHON SHELL

```
kit = 18.0
liv = 28.0
bed = 18.75
bath = 9.50

# Create list areas

areas=['hall','kit','liv','bed','bath']
# Print areas
print(areas)
['hall', 'kit', 'liv', 'bed', 'bath']

In [2]: |
```

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise involves creating a variable `savings` and calculating its value after 7 years of investing at a 10% annual rate. The code provided is:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The exercise includes instructions and a list of tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available.

The screenshot shows a DataCamp Python course exercise titled "Create a list". The exercise involves creating a list of room areas. The code provided is:

```
1 # area variables (in square meters)
2 hall = 11.25
3 kit = 18.0
4 liv = 20.0
5 bed = 10.75
6 bath = 9.50
7
8 # Create list areas
9
10 areas=[hall,kit,liv,bed,bath]
11 # Print areas
12 print(areas)
13
```

The exercise includes instructions and a list of tasks:

- After measuring the below rooms, Nice! A list is way better here, isn't it? Enter information on the areas you're living in. The areas are hall, kitchen, living room, bedroom, and bathroom (hall, kit, liv, bed, bath).
- PRESS ENTER TO
- Continue

A "Take Hint (-30 XP)" button is available.

A "Become a power user!" section provides keyboard shortcuts:

SUBMIT ANSWER: **CTRL + SHIFT + ENTER**

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window titled "Calculations with variables". The "SCRIPT.PY" tab contains the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section provides context: "Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:" followed by the code `100 * 1.10 ** 7`. It then states: "Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!"

The "INSTRUCTIONS" section also lists three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available.

The "IPYTHON SHELL" tab shows the command `In [1]: |` ready for input.

The screenshot shows a DataCamp exercise window titled "Create list with different types". The "SCRIPT.PY" tab contains the following code:

```
1 # area variables (in square meters)
2 hall = 11.25
3 kit = 18.0
4 liv = 20.0
5 bed = 10.75
6 bath = 9.50
7
8 # Adapt list areas
9 areas = ["hallway", hall, "kitchen", kit, "living room", liv, "bedroom", bed, "bathroom", bath]
10
11 # Print areas
12 print(areas)
```

The "INSTRUCTIONS" section provides context: "A list can contain any Python type. Although it's not really common, a list can also contain a mix of Python types including strings, floats, booleans, etc." It then states: "The prompt of the previous exercise is **+100 XP**. Applying it's just a list of numbers representing the areas, but you're not asked what it responds to which part of your house." A note says: "Nice! This list contains both strings and floats, but that's not a problem for Python!" It also notes: "The code on the right is the same as the previous exercise, except the areas, the name of the list responding point already includes the string 'area'." A "Please rate this exercise" button with five stars is shown.

The "PROMPT" section asks: "From the line of code that creates the list `areas`, you know that the list first contains the name of each room as a string and then its areas. More specifically, the strings 'hallway', 'kitchen', and 'bedroom' are the components for `hall`, `kit`, and `bed` respectively." It then asks: "Print `areas` again in the previous step (otherwise this time)." A "Take Hint (-30 XP)" button is available.

The "SUBMIT ANSWER" section shows a keyboard shortcut: **CTRL + SHIFT + ENTER**.

The "IPYTHON SHELL" tab shows the output of the code execution:

```
bath = 9.50
# Adapt list areas
areas = ["hallway", hall, "kitchen", kit, "living room", liv, "bedroom", bed, "bathroom", bath]
# Print areas
print(areas)
['hallway', 11.25, 'kitchen', 18.0, 'living room', 20.0, 'bedroom', 10.75, 'bathroom', 9.5]
```

The "IPYTHON SHELL" tab shows the command `In [10]: |` ready for input.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window. On the left, under 'EXERCISE', there's a section titled 'Calculations with variables'. It contains text about calculating money after 7 years and a code editor with a script named 'SCRIPT.PY' containing the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the code editor is an 'INSTRUCTIONS' section with 100 XP available. It lists three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

There is also a 'Take Hint (-30 XP)' button.

On the right side of the window, there are tabs for 'SCRIPT.PY', 'IPYTHON SHELL', and 'SLIDES'. The 'IPYTHON SHELL' tab is selected, showing the input 'In [1]: |'.

The bottom of the window shows a Windows taskbar with various icons and a system tray indicating the date as 7/16/2018.

The screenshot shows a DataCamp exercise window. On the left, under 'EXERCISE', there's a section titled 'Select the valid list'. It contains text about lists being Python types and a code editor with a script named 'SCRIPT.PY' containing the following code:

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

Below the code editor is an 'INSTRUCTIONS' section with 50 XP available. It asks the user to identify valid ways to build a list from the options A, B, and C.

On the right side of the window, there are tabs for 'SCRIPT.PY', 'IPYTHON SHELL', and 'SLIDES'. The 'IPYTHON SHELL' tab is selected, showing the following interactions:

```
In [1]: [1 + 2, 'a' * 5, 3]
Out[1]: [3, 'aaaaa', 3]

In [2]: [[1, 2, 3], [4, 5, 7]]
Out[2]: [[1, 2, 3], [4, 5, 7]]

In [3]: [1, 3, 4, 2]
Out[3]: [1, 3, 4, 2]

In [4]: |
```

Below the IPython shell, there are four numbered buttons labeled 1, 2, 3, and 4, each with a 'press' label above it. At the bottom right is a 'Submit Answer' button.

The bottom of the window shows a Windows taskbar with various icons and a system tray indicating the date as 7/16/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. On the left, there's an 'EXERCISE' panel for 'Calculations with variables'. It contains instructions and a code editor with a script named 'SCRIPT.PY' containing the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the code editor is an 'INSTRUCTIONS' section with 100 XP available. It lists three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

There's also a 'Take Hint (-30 XP)' button.

On the right, there's an 'IPYTHON SHELL' tab and a 'SLIDES' tab. The IPython shell shows the command 'In [1]: |'.

The taskbar at the bottom shows several open windows, including 'ch1_slides.pdf', 'DataCamp', and 'IPython Shell'.

The screenshot shows another DataCamp course exercise titled 'Select the valid list'. The instructions state: 'A list can contain any Python type. But a list itself is also a Python type. This means that a list can also contain a list.' It includes a note: 'Python is getting funnier by the minute, but fear not, just remember the list syntax.'

The exercise asks: 'Can you tell which ones of the following commands are valid ways to build a Python list?' It lists four options: A, B, C, and D. The correct answer is D, which is '+50 XP'.

Below the exercise, there's a 'PICK ANSWER' section with a 'Continue' button. A 'Become a power user!' overlay is visible, showing keyboard shortcuts for 'SUBMIT ANSWER: CTRL + SHIFT + ENTER'.

The taskbar at the bottom shows several open windows, including 'ch1_slides.pdf', 'DataCamp', and 'IPython Shell'.

● Datacamp datascience course by

The screenshot shows a DataCamp course window. The title bar says "Calculations with variables" and "ch1_slides.pdf". The main area has tabs for "EXERCISE", "SCRIPT.PY", and "IPYTHON SHELL".

EXERCISE: Calculations with variables. Instructions: Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS (100 XP):

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY:1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result

IPYTHON SHELL: In [1]: |

The screenshot shows a DataCamp course window. The title bar says "ch1_slides.pdf", "Careers and Job Opportu...", "List of lists | Python", and "courses-pythonwhat-tut...".

EXERCISE: List of lists. Instructions: As a data scientist, you'll often be dealing with a lot of data, and it will make sense to group some of this data.

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: You didn't assign the correct value to `house`. Have another look at the instructions. Extend the list of lists so it incorporates a list for each pair of room name and room area. Mind the order and types!

SCRIPT.PY:1 # area variables (in square meters)
2 hall = 11.25
3 kit = 18.0
4 liv = 28.0
5 bed = 18.75
6 bath = 9.50
7
8 # house information as list of lists
9 house = [["hallway", hall],
10 ["kitchen", kit],
11 ["living room", liv]]
12
13 # Print out house
14
15 print(house)
16 # Print out the type of house
17 print(type(house))
18

IPYTHON SHELL: In [2]: |

```
# house information as list of lists  
house = [["hallway", hall],  
         ["kitchen", kit],  
         ["living room", liv]]  
  
# Print out house  
  
print(house)  
# Print out the type of house  
[[['hallway', 11.25], ['kitchen', 18.0], ['living room', 28.0]]]
```

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window titled "Calculations with variables". The "SCRIPT.PY" tab is active, displaying the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains the following text and a list of tasks:

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

The window also includes an "IPYTHON SHELL" tab and a "SLIDES" tab. The system tray at the bottom shows the date as 7/16/2018.

The screenshot shows a DataCamp exercise window titled "List of lists". The "SCRIPT.PY" tab is active, displaying the following Python code:

```
1 hall = 11.25
2 kit = 18.0
3 liv = 20.0
4 bed = 10.75
5 bath = 9.50
6
7 # house information as list of lists
8 house = [['hallway', hall],
9                 ['kitchen', kit],
10                ['living room', liv],
11                ['bedroom', bed],
12                ['bathroom', bath]]
```

The "INSTRUCTIONS" section contains the following text and a list of tasks:

As a data scientist, you'll often be dealing with a lot of data, and it will make sense to group some of this data.

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

You didn't assign the correct value to `house`. Have another look at the instructions. Extend the list of lists so it incorporates a list for each pair of room name and room area. Mind the order and types!

The "IPYTHON SHELL" tab shows the output of the script:

```
# Print out house
print(house)

# Print out the type of house
print(type(house))
[[['hallway', 11.25], ['kitchen', 18.0], ['living room', 20.0], ['bedroom', 10.75], ['bathroom', 9.5]]]
<class 'list'>
```

The "SLIDES" tab is also visible. The system tray at the bottom shows the date as 7/16/2018.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

In [1]: |

List of lists

As a data scientist, you'll often be dealing with a lot of data, and it will make sense to group some of this data.

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 do exactly that! +100 XP

Don't get confused now! A list of lists is a list that contains other lists. Each element that represents the float `11.25` is itself a list that represents the float `11.25` and the name `"hallway"`.

PRESS ENTER TO

INSTRUCTIONS 100 XP

Run the list of lists so that it prints the house information and room information. 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)

Become a power user!

SUBMIT ANSWER: CTRL + SHIFT + ENTER

See all keyboard shortcuts

In [9]: |

● Datacamp datascience course by

The screenshot shows a DataCamp course window with the following details:

- Tab Bar:** Calculations with variables, courses-intro-to-python, ch1_slides.pdf.
- Header:** DataCamp, Course Outline.
- Left Sidebar:** EXERCISE, Calculations with variables. Text: Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:
100 * 1.10 ** 7. Instead of calculating with the actual values, you can use variables instead. The savings variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent 1.10 and then redo the calculations!
- Instructions (100 XP):**
 - Create a variable factor, equal to 1.10.
 - Use savings and factor to calculate the amount of money you end up with after 7 years. Store the result in a new variable, result.
 - Print out the value of result.
- SCRIPT.PY:**

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- IPYTHON SHELL:** In [1]: |
- Bottom Bar:** Type here to search, taskbar icons, ENG IN 10:00 PM 7/16/2018.

The screenshot shows a DataCamp course window with the following details:

- Tab Bar:** ch1_slides.pdf, Careers and Job Opportu..., Subset and conquer | Pyt..., courses-intro-to-python, courses-pythonwhat-tut...,
- Header:** DataCamp, Course Outline.
- Left Sidebar:** EXERCISE, Subset and conquer. Text: Subsetting Python lists is a piece of cake. Take the code sample below, which creates a list x, and then selects "b" from it. Remember that this is the second element, so it has index 1. You can also use negative indexing.
x = ["a", "b", "c", "d"]
x[1]
x[-3] # same result!
- Instructions (100 XP):**
 - Print out the second element from the areas list, so 11.25.
 - Subset and print out the last element of areas, being 9.50. Using a negative index makes sense here!
 - Select the number representing the area of the living room and print it out.
- SCRIPT.PY:**

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]
3
4 # Print out second element from areas
5
6
7 # Print out last element from areas
8
9
10 # Print out the area of the living room
11
```
- IPYTHON SHELL:** In [1]: |
- Bottom Bar:** Type here to search, taskbar icons, ENG IN 11:43 PM 7/16/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window. At the top, there are three tabs: 'Calculations with variables', 'courses-intro-to-python', and 'ch1_slides.pdf'. The main area has a 'Course Outline' header with a 'SCRIPT.PY' tab selected. Below it is a code editor with the following script:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the code editor is an 'INSTRUCTIONS' section with 100 XP available. It contains the following tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

There is also a 'Take Hint (-30 XP)' button.

The bottom of the window shows a taskbar with various icons and a system tray indicating the date as 7/16/2018.

The screenshot shows a DataCamp exercise window. At the top, there are five tabs: 'ch1_slides.pdf', 'Careers and Job Opport...', 'Subset and conquer | Pyt...', 'courses-intro-to-python', and 'courses-pythonwhat-tut...'. The main area has a 'Course Outline' header with a 'SCRIPT.PY' tab selected. Below it is a code editor with the following script:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.5]
3
4 # Print out second element from areas
5
6 print(areas)
7 # Print out last element from areas
8
9 areas[4]
10 # Print out the area of the living room
11 print(areas[2])
```

Below the code editor is an 'INSTRUCTIONS' section with +100 XP available. It contains the following text:

Subsetting Python lists is a piece of cake. Take the code sample below, which creates a list `areas`, and then selects 'b' from it. Remember that this is the second element, so its index is 1. You can also use negative indexing.

Good job!

Remember the `areas` list from before? You can still call `len` on it and found its definition already in the script. Can you add the code to do some Python subsetting?

PRESS ENTER TO CONTINUE

Below the instructions is a 'Continue' button. Further down, there are two questions:

- Print out the second element from the `areas` list.
- Select and print out the last element of `areas`. Hint: you can use a negative index.

There is also a 'Take Hint (-30 XP)' button.

The bottom of the window shows a taskbar with various icons and a system tray indicating the date as 7/16/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- An "INSTRUCTIONS" section with 100 XP available, containing three steps:
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- A "HINT" button labeled "Take Hint (-30 XP)".
- An "IPYTHON SHELL" tab showing the command "In [1]: |".
- A "SLIDES" tab.

The operating system taskbar at the bottom shows several open windows, including "ch1_slides.pdf", "DataCamp", and "ipython shell". The system status bar indicates "ENG 10:00 PM IN 7/16/2018".

The screenshot shows a DataCamp Python course exercise titled "Subset and calculate". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.5]
3
4 # Sum of kitchen and bedroom area: eat_sleep_area
5 eat_sleep_area=(areas[1]+areas[3])
6
7 # Print the variable eat_sleep_area
8 print(eat_sleep_area)
```
- An "INSTRUCTIONS" section with 100 XP available, containing two steps:
 - Using a combination of list subsetting and variable assignment, create a new variable, `eat_sleep_area`, that contains the sum of the area of the kitchen and the area of the bedroom.
 - Print the new variable `eat_sleep_area`.
- A "HINT" button labeled "Take Hint (-30 XP)".
- An "IPYTHON SHELL" tab showing the command "In [1]: # Create the areas list" followed by the output "20.25".
- A "SLIDES" tab.

The operating system taskbar at the bottom shows several open windows, including "ch1_slides.pdf", "DataCamp", and "ipython shell". The system status bar indicates "ENG 11:46 PM IN 7/16/2018".

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. The top navigation bar includes tabs for 'Calculus with variable', 'courses-intro-to-python', and 'ch1_slides.pdf'. The main content area has a 'Course Outline' header. On the left, there's an 'EXERCISE' section titled 'Calculations with variables'. It contains instructions: 'Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:' followed by the code `100 * 1.10 ** 7`. Below this, a note says: 'Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!'. The 'INSTRUCTIONS' section provides three tasks: 'Create a variable `factor`, equal to `1.10`', 'Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`', and 'Print out the value of `result`'. A 'Take Hint (-30 XP)' button is available. On the right, there's a 'SCRIPT.PY' editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the editor is an 'IPYTHON SHELL' tab with the prompt 'In [1]: |'. At the bottom of the window is a toolbar with icons for back, forward, search, and other course navigation.

This screenshot shows another DataCamp exercise titled 'Subset and calculate'. The top navigation bar includes tabs for 'ch1_slides.pdf', 'Careers and Job Opport...', 'Subset and calculate | Py', 'courses-intro-to-python', and 'courses-pythonwhat-tut...'. The main content area has a 'Course Outline' header. The 'EXERCISE' section is titled 'Subset and calculate'. It contains instructions: 'After you've extracted values from a list, you can use them to perform additional calculations. Take this example, where the second and fourth elements of a list `areas` are extracted. The strings that result are pasted together using the `+ operator'. A note says: 'After you've extracted values from a list, you can use them to perform additional calculations. Take this example, where the second and fourth elements of a list areas are extracted. The strings that result are pasted together using the + operator'. The 'INSTRUCTIONS' section provides three tasks: 'Using a combination of the len() function and print(), print the sum of the kitchen and the area of the bedroom', 'Print the new variable eat_sleep_area', and 'Take Hint (-30 XP)'. A 'Bellissimo!' badge is shown. On the right, there's a 'SCRIPT.PY' editor with the following code:`

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 18.75, "bathroom", 9.50]
3
4 # Sum of kitchen and bedroom area: eat_sleep_area
5 eat_sleep_area=(areas[-3]+areas[3])
6
7 # Print the variable eat_sleep_area
8 print(eat_sleep_area)
```

Below the editor is an 'IPYTHON SHELL' tab with the prompt 'In [1]: |'. The shell shows the output of the script: '28.75'. At the bottom of the window is a toolbar with icons for back, forward, search, and other course navigation.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes a "SCRIPT.PY" editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the code editor is an "INSTRUCTIONS" section with 100 XP available. It contains the following steps:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is also present.

The bottom of the window shows a Windows taskbar with various icons and the date/time: 7/16/2018, 10:00 PM.

The screenshot shows a DataCamp Python course exercise titled "Slicing and dicing". The exercise interface includes a "SCRIPT.PY" editor with the following code:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]
3
4 # Use slicing to create downstairs
5
6
7 # Use slicing to create upstairs
8
9
10 # Print out downstairs and upstairs
```

Below the code editor is an "INSTRUCTIONS" section with 100 XP available. It contains the following steps:

- Use slicing to create a list, `downstairs`, that contains the first 6 elements of `areas`.
- Do a similar thing to create a new variable, `upstairs`, that contains the last 4 elements of `areas`.
- Print both `downstairs` and `upstairs` using `print()`.

A "Take Hint (-30 XP)" button is also present.

The bottom of the window shows a Windows taskbar with various icons and the date/time: 7/16/2018, 11:48 PM.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes a "SCRIPT.PY" editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the editor is an "INSTRUCTIONS" section with 100 XP available:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is also present.

The bottom of the window shows a taskbar with various icons and the system clock indicating 10:00 PM on 7/16/2018.

The screenshot shows a DataCamp Python course exercise titled "Slicing and dicing". The exercise interface includes a "SCRIPT.PY" editor with the following code:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]
3
4 # Use slicing to create downstairs
5 downstairs=areas[0:6]
6
7 # Use slicing to create upstairs
8 upstairs=areas[1:4]
9
10 # Print out downstairs and upstairs
11 print(downstairs)
12 print(upstairs)
```

Below the editor is an "INSTRUCTIONS" section with 100 XP available:

- Use slicing to create a list, `downstairs`, that contains the first 6 elements of `areas`.
- Do a similar thing to create a new variable, `upstairs`, that contains the last 4 elements of `areas`.
- Print both `downstairs` and `upstairs` using `print()`.

A "Take Hint (-30 XP)" button is also present.

The bottom of the window shows a taskbar with various icons and the system clock indicating 11:51 PM on 7/16/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window titled "Calculations with variables". The "SCRIPT.PY" tab is active, displaying the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains the following text and a list of tasks:

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

The window also includes tabs for "IPYTHON SHELL" and "SLIDES". The system tray at the bottom shows the date as 7/16/2018.

The screenshot shows a DataCamp exercise window titled "Slicing and dicing | Python". The "SCRIPT.PY" tab is active, displaying the following Python code:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]
3
4 # Use slicing to create downstairs
5 downstairs=areas[0:6]
6
7 # Use slicing to create upstairs
8 upstairs=areas[0:4]
9
10 # Print out downstairs and upstairs
11 print(downstairs)
12 print(upstairs)
```

The "INSTRUCTIONS" section contains the following text and a list of tasks:

The code sample below shows an example. A list with `"b"` and `"c"`, corresponding to indexes 1 and 2, are selected from a list `x`:

```
x = ["a", "b", "c", "d"]
x[1:3]
```

The elements with index 1 and 2 are included, while the element with index 3 is not.

INSTRUCTIONS 100 XP

- Use slicing to create a list, `downstairs`, that contains the first 6 elements of `areas`.
- Do a similar thing to create a new variable, `upstairs`, that contains the last 4 elements of `areas`.
- Print both `downstairs` and `upstairs` using `print()`.

Take Hint (-30 XP)

INCORRECT SUBMISSION

Your definition of `downstairs` is incorrect. Use `areas[...]` and slicing to select the elements you want. You could use `0:6` where the dots are, for example.

The "IPYTHON SHELL" tab is active, showing the following output:

```
# Use slicing to create downstairs
downstairs=areas[0:6]

# Use slicing to create upstairs
upstairs=areas[0:4]

# Print out downstairs and upstairs
print(downstairs)
print(upstairs)
['hallway', 11.25, 'kitchen', 18.0, 'living room', 20.0]
['hallway', 11.25, 'kitchen', 18.0]
```

The system tray at the bottom shows the date as 7/16/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise titled 'Calculations with variables'. On the left, there's an 'INSTRUCTIONS' section with 100 XP available. It contains three bullet points: 'Create a variable `factor`, equal to `1.10`', 'Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`', and 'Print out the value of `result`'. Below this is a 'Take Hint (-30 XP)' button. In the center, there's a 'SCRIPT.PY' code editor with the following content:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

On the right, there's an 'IPYTHON SHELL' tab where the user can run code. At the bottom, a Windows taskbar shows various open windows and the date/time.

The screenshot shows a DataCamp exercise titled 'Slicing and dicing'. On the left, there's an 'INSTRUCTIONS' section with 100 XP available. It contains three bullet points: 'Use slicing to create a list, `downstairs`, that contains the first 6 elements of `areas`', 'Do a similar thing to create a new variable, `upstairs`, that contains the last 4 elements of `areas`', and 'Print both `downstairs` and `upstairs` using `print()`'. Below this is a 'Take Hint (-30 XP)' button. In the center, there's a 'SCRIPT.PY' code editor with the following content:

```
1 # Create the areas list
2 areas = ['hallway', 11.25, 'kitchen', 18.0, 'living room', 20.0, 'bedroom', 10.75, 'bathroom', 9.5]
3
4 # Use slicing to create downstairs
5 downstairs=areas[0:6]
6
7 # Use slicing to create upstairs
8 upstairs=areas[6:10]
9
10 # Print out downstairs and upstairs
11 print(downstairs)
12 print(upstairs)
```

On the right, there's an 'IPYTHON SHELL' tab where the user can run code. At the bottom, a Windows taskbar shows various open windows and the date/time.

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. The top navigation bar includes tabs for 'Calculus with variable', 'courses-intro-to-python', and 'ch1_slides.pdf'. The main content area has a 'Course Outline' header. On the left, there's an 'EXERCISE' section with a title 'Calculations with variables'. It contains instructions: 'Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:' followed by a code snippet: `100 * 1.10 ** 7`. Below this, a note says: 'Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!'. An 'INSTRUCTIONS' bar indicates '100 XP'. A list of tasks: 'Create a variable `factor`, equal to `1.10`', 'Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`', and 'Print out the value of `result`'. A 'Take Hint (-30 XP)' button is present. On the right, there's a 'SCRIPT.PY' editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the editor is an 'IPYTHON SHELL' tab and a 'SLIDES' tab. The shell shows 'In [1]: |'. At the bottom, a Windows taskbar shows various open windows and the date/time as 7/16/2018.

This screenshot shows another DataCamp exercise titled 'Slicing and dicing'. The top navigation bar includes tabs for 'ch1_slides.pdf', 'Careers and Job Opport...', 'Slicing and dicing | Python', 'courses-intro-to-python', and 'courses-pythonwhat-tut...'. The main content area has a 'Course Outline' header. The 'EXERCISE' section contains a note: 'The code below shows an example of list slicing. The elements corresponding to indexes 1 and 2, are selected from a list.' Below this, a note says: 'The elements with index 1 and 2 are +100 XP. The element with index 3 is not'. A 'Great!' message is displayed. A 'PRESS ENTER TO' button with 'Continue' is shown. The 'INSTRUCTIONS' bar indicates '+100 XP'. A note says: 'Use slicing to create a list containing the first 6 elements of areas'. Another note says: 'Do a similar thing to create a list containing the last 4 elements of areas, that contain the last 4 elements of areas'. A 'Take Hint (-30 XP)' button is present. A 'Become a power user!' section at the bottom includes a 'SUBMIT ANSWER' button with 'CTRL + SHIFT + ENTER' and a link to 'See all keyboard shortcuts'. On the right, there's a 'SCRIPT.PY' editor with the following code:

```
1 # Create the areas list
2 areas = ['hallway', 11.25, 'kitchen', 18.0, 'living room', 20.0, 'bedroom', 10.75, 'bathroom', 9.5]
3
4 # Use slicing to create downstairs
5 downstairs=areas[0:6]
6
7 # Use slicing to create upstairs
8 upstairs=areas[6:10]
9
10 # Print out downstairs and upstairs
11 print(downstairs)
12 print(upstairs)
```

Below the editor is an 'IPYTHON SHELL' tab and a 'SLIDES' tab. The shell shows 'In [7]: |'. The output of the code execution is visible in the shell. At the bottom, a Windows taskbar shows various open windows and the date/time as 7/16/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- An "INSTRUCTIONS" section with 100 XP available, containing three tasks:
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- A "Take Hint (-30 XP)" button.
- An "IPYTHON SHELL" tab showing the command "In [1]: |".
- A "SLIDES" tab.

The operating system taskbar at the bottom shows other open windows like "ch1_slides.pdf" and "Slicing and dicing (2) | Py".

The screenshot shows a DataCamp Python course exercise titled "Slicing and dicing (2)". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.5]
3
4 # Alternative slicing to create downstairs
5
6
7 # Alternative slicing to create upstairs
8
```
- An "INSTRUCTIONS" section with 100 XP available, containing a note about list slicing and several commands to run in the IPython Shell:

```
x = ["a", "b", "c", "d"]
x[:2]
x[2:]
x[:]
```

Use slicing to create the lists `downstairs` and `upstairs` again, but this time without using indexes if it's not necessary. Remember `downstairs` is the first 6 elements of `areas` and `upstairs` is the last 4 elements of `areas`.
- A "Take Hint (-30 XP)" button.
- An "IPYTHON SHELL" tab showing the following session:

```
In [1]: x = ["a", "b", "c", "d"]
In [2]: x[:2]
Out[2]: ['a', 'b']
In [3]: x[2:]
Out[3]: ['c', 'd']
In [4]: x[:]
Out[4]: ['a', 'b', 'c', 'd']
```

In [5]:
- A "SLIDES" tab.

The operating system taskbar at the bottom shows other open windows like "ch1_slides.pdf" and "Slicing and dicing (2) | Py".

● Datacamp datascience course by

The screenshot shows a DataCamp course exercise window titled "Calculations with variables". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- An "INSTRUCTIONS" section with 100 XP available, listing three tasks:
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- A "Take Hint (-30 XP)" button.

The screenshot shows the same DataCamp course exercise window after completion. The exercise content includes:

- A message indicating the task was completed successfully: "+100 XP" and "Wonderful!"
- A rating section: "Please rate this exercise: ★★★★★"
- A "PRESS ENTER TO" instruction followed by a "Continue" button.
- A note about slicing lists: "If your list, if you don't specify the `start` index, the slice will go all the way to the last element of your list. To experiment with this, try the following command in the Python Shell."
- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.5]
3
4 # Alternative slicing to create downstairs
5 downstairs=areas[:6]
6
7 # Alternative slicing to create upstairs
8 upstairs=areas[6:]
```
- An "IPYTHON SHELL" window showing the output of the code:

```
In [4]: ['hallway', 11.25, 'kitchen', 18.0, 'living room', 20.0, 'bedroom', 10.75, 'bathroom', 9.5]
Out[4]: ['a', 'b', 'c', 'd']

In [5]: # Create the areas list
areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.5]

# Alternative slicing to create downstairs
downstairs=areas[:6]

# Alternative slicing to create upstairs
upstairs=areas[6:]
```

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. The top navigation bar includes tabs for 'Calculations with variables', 'courses-intro-to-python', and 'ch1_slides.pdf'. The main content area has a 'Course Outline' header. On the left, there's an 'EXERCISE' section titled 'Calculations with variables'. It contains instructions: 'Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:' followed by the code `100 * 1.10 ** 7`. Below this, a note explains that instead of calculating with actual values, variables can be used. The 'INSTRUCTIONS' section (100 XP) lists three tasks: creating a variable `factor`, using `savings` and `factor` to calculate a result, and printing the result. A 'Take Hint (-30 XP)' button is available. On the right, the 'SCRIPT.PY' editor shows the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The bottom right corner shows a preview of the 'IPYTHON SHELL' tab.

This screenshot shows another DataCamp exercise titled 'Subsetting lists of lists'. The top navigation bar includes tabs for 'ch1_slides.pdf', 'Careers and Job Opport...', 'Subsetting lists of lists', 'courses-intro-to-python', and 'courses-pythonwhat-tut...'. The main content area has a 'Course Outline' header. The 'EXERCISE' section contains sample code in the IPython Shell:

```
x = [['a', 'b', 'c'],
      ['d', 'e', 'f'],
      ['g', 'h', 'i']]
x[2][0]
x[2][:2]
```

A note states: '(x[2]) results in a list, that you can subset again by adding additional square brackets.' It asks what `house[-1][1]` returns. The 'INSTRUCTIONS' section (50 XP) lists possible answers: 'A float: the kitchen area', 'A string: "kitchen"', 'A float: the bathroom area', and 'A string: "bathroom"'. Radio buttons allow selecting one. Buttons for 'Enter' and 'Submit Answer' are present. A 'Take Hint (-15 XP)' button is available. An 'INCORRECT SUBMISSION' message says: 'Incorrect. house[-1] indeed selects the list that represents the bathroom information, but [1] selects the second element of the sublist, not the first. Python uses zero-based indexing.' The bottom right corner shows a preview of the 'SLIDES' tab.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise titled "Calculations with variables". On the left, there's an "INSTRUCTIONS" section with 100 XP available. It contains three steps: creating a variable `factor` equal to `1.10`, using `savings` and `factor` to calculate a result, and printing the result. Below the instructions is a "Take Hint (-30 XP)" button. On the right, there's a "SCRIPT.PY" editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the editor is an "IPYTHON SHELL" tab with the prompt "In [1]: |".

The screenshot shows a DataCamp exercise titled "Subsetting lists of lists". On the left, there's a "PICK A CHALLENGE" section with a "Manipulations" challenge selected. It includes a "Possible Answers" section with options like "A list of the kitchen", "A string", "A tuple", and "A list of the bathroom". Below that is a "+50 XP" badge and a "Correct! The last piece of the list puzzle is manipulation." message. There are "Take Hint (-15 XP)" and "Continue" buttons. On the right, there's an "IPYTHON SHELL" tab with the following interaction:

```
In [1]: x = [["a", "b", "c"],  
...           ["d", "e", "f"],  
...           ["g", "h", "i"]]  
In [2]: x[2][0]  
Out[2]: 'g'  
In [3]: x[2][1:2]  
Out[3]: ['g', 'h']  
In [4]: x[2][1]  
Out[4]: ['g', 'h', 'i']  
In [5]: house[-1]  
File "<stdin>", line 1  
    house[-1]  
          ^  
SyntaxError: invalid syntax  
In [6]: house  
Out[6]:  
[['hallway', 11.25],  
['kitchen', 18.0],  
['living room', 20.0],  
['bedroom', 10.75],  
['bathroom', 9.5]]  
In [7]: house[-1][1]  
Out[7]: 9.5  
In [8]: |
```

At the bottom left, there's a "Become a power user!" section with a "SUBMIT ANSWER: CTRL + SHIFT + ENTER" button and a "See all keyboard shortcuts" link.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes a "SCRIPT.PY" editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the code editor is an "INSTRUCTIONS" section with 100 XP available. It contains the following steps:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is also present.

The bottom of the window shows a Windows taskbar with various icons and the date 7/16/2018.

The screenshot shows a DataCamp Python course exercise titled "Replace list elements". The exercise interface includes a "SCRIPT.PY" editor with the following code:

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.5]
3
4 # Correct the bathroom area
5 areas[-1]=10.5
6
7
8 # Change "living room" to "chill zone"
9 areas[4]="chill zone"
```

Below the code editor is an "INSTRUCTIONS" section with 100 XP available. It contains the following steps:

- You did a miscalculation when determining the area of the bathroom; it's 10.50 square meters instead of 9.50. Can you make the changes?
- Make the `areas` list more trendy! Change "living room" to "chill zone".

A "Take Hint (-30 XP)" button is also present.

An orange "INCORRECT SUBMISSION" box states: "Your changes to `areas` did not result in the correct list. Are you sure you used the correct subset operations? When in doubt, you can use a hint!"

The bottom of the window shows a Windows taskbar with various icons and the date 7/17/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes:

- SCRIPT.PY:** A code editor window containing the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- INSTRUCTIONS (100 XP):**
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- ipython shell:** An empty command-line interface window labeled "In [1]: |".

The screenshot shows a DataCamp Python course exercise titled "Extend a list". The exercise interface includes:

- SCRIPT.PY:** A code editor window containing the following Python code:

```
1 # Create the areas list and make some changes
2 areas = ["hallway", 11.25, "kitchen", 18.0, "chill zone", 20.0,
3          "bedroom", 10.75, "bathroom", 10.50]
4
5 # Add poolhouse data to areas, new list is areas_1
6 areas_1=areas+["poolhouse",2.5]
7
8 # Add garage data to areas_1, new list is areas_2
9 areas_2=areas_1+["garage",15.45]
```
- INSTRUCTIONS (100 XP):**
 - Use the `+` operator to paste the list `("poolhouse", 24.5)` to the end of the `areas` list. Store the resulting list as `areas_1`.
 - Further extend `areas_1` by adding data on your garage. Add the string `"garage"` and float `15.45`. Name the resulting list `areas_2`.
- ipython shell:** An empty command-line interface window labeled "In [5]: |".

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

IPYTHON SHELL

```
In [1]: |
```

Extend a list

If you can change elements in a list, you sure want to be able to add elements to it, right? You can use the `+` operator.

Cool! The list is shaping up nicely!

You just won the lottery! You're getting a poolhouse and a garage. Can you add the information to the areas list?

PRESS ENTER TO
Continue

INSTRUCTIONS 100 XP

Use the `+` operator to paste the list `areas = [11.25, ...]` to the end of the `areas` list. Store the resulting list as `areas_1`.

Further extend `areas_1` by adding data on your garage. Add the string `"garage": 15.45`. Name the resulting list `areas_2`.

Take Hint (-30 XP)

Become a power user!

SUBMIT ANSWER: **CTRL + SHIFT + ENTER**

See all keyboard shortcuts

IPYTHON SHELL

```
In [6]: # Create the areas list and make some changes
areas = ["hallway", 11.25, "kitchen", 18.0, "chill zone", 20.0,
         "bedroom", 10.75, "bathroom", 10.50]
# Add poolhouse data to areas, new list is areas_1
areas_1=areas+["poolhouse",24.5]

In [7]: # Create the areas list and make some changes
areas = ["hallway", 11.25, "kitchen", 18.0, "chill zone", 20.0,
         "bedroom", 10.75, "bathroom", 10.50]
# Add poolhouse data to areas, new list is areas_1
areas_1=areas+["poolhouse",24.5]
# Add garage data to areas_1, new list is areas_2
areas_2=areas_1+["garage",15.45]

In [7]: |
```

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. On the left, there's an 'EXERCISE' panel for 'Calculations with variables'. It contains instructions and a code editor with the following script:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the script, there are three tabs: 'INSTRUCTIONS' (100 XP), 'HINT' (Take Hint (-30 XP)), and 'ANSWER'.

The main workspace has tabs for 'SCRIPT.PY', 'IPYTHON SHELL', and 'SLIDES'. The 'IPYTHON SHELL' tab is active, showing the command 'In [1]: |'.

The taskbar at the bottom shows several open windows, including 'ch1_slides.pdf', 'DataCamp', and browser tabs for DataCamp courses.

This screenshot shows another DataCamp exercise. The left panel displays a question about deleting elements from a list and provides a 'Correct!' feedback message with '+50 XP'.

The right panel shows the 'IPYTHON SHELL' tab with the following interaction:

```
In [3]: del(areas[10:11])
      ^          ^
SyntaxError: invalid syntax

In [4]: delete(areas[-4:-2])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    delete(areas[-4:-2])
NameError: name 'delete' is not defined

In [5]: del(areas[-4:-2])

In [6]: areas
Out[6]:
['hallway',
 11.25,
 'kitchen',
 18.0,
 'chill zone',
 28.0,
 'bedroom',
 10.75,
 'bathroom',
 10.5,
 'garage',
 15.45]
```

The taskbar at the bottom shows multiple windows, including 'ch1_slides.pdf', 'DataCamp', and browser tabs for DataCamp courses.

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. The main window displays an exercise titled "Calculations with variables". The exercise instructions ask the user to calculate the value of a variable after 7 years of investing \$100 at a 10% annual rate. The code provided in the "SCRIPT.PY" editor is:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains three steps:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available.

The bottom of the screen shows a Windows taskbar with various icons and a search bar.

The screenshot shows a DataCamp course interface. The main window displays an exercise titled "Inner workings of lists". The exercise instructions ask the user to change the first element of a list and then print the list to verify the change. The code provided in the "SCRIPT.PY" editor is:

```
1 # Create list areas
2 areas = [11.25, 18.0, 20.0, 10.75, 9.5]
3
4 # Create areas_copy
5 areas_copy = list(areas)
6
7 # Change areas_copy
8 areas_copy[0] = 5.0
9
10 # Print areas
11 print(areas)
```

The "INSTRUCTIONS" section contains two steps:

- Change the second command to make `areas_copy` a reference copy, such that changes made to `areas_copy` shouldn't affect `areas`. Hit Submit Answer to check this.
- A "Take Hint (-40 XP)" button is available.

The bottom of the screen shows a Windows taskbar with various icons and a search bar.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

In [1]: |

The general recipe for calling functions and saving the result to a variable is thus:

```
output = function_name(input)
```

INSTRUCTIONS 100 XP

- Use `print()` in combination with `type()` to print out the type of `var1`.
- Use `len()` to get the length of the list `var1`. Wrap it in a `print()` call to directly print it out.
- Use `int()` to convert `var2` to an integer. Store the output as `out2`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create variables var1 and var2
2 var1 = [1, 2, 3, 4]
3 var2 = True
4
5 # Print out type of var1
6
7 print(type(var1))
8 # Print out length of var1
9
10 print(len(var1))
11 # Convert var2 to an integer: out2
12 out2=int(var2)
```

IPYTHON SHELL

In [3]: `print(var1)`
[1, 2, 3, 4]

In [4]: `print(var2)`
True

In [5]:

● Datacamp datascience course by

The screenshot shows a DataCamp exercise titled "Calculations with variables". The exercise instructions state: "Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:" followed by the code `100 * 1.10 ** 7`. A note explains: "Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!" Below the instructions is a list of tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available. The interface includes tabs for "SCRIPT.PY", "IPYTHON SHELL", and "SLIDES". The system tray at the bottom shows a Windows 10 taskbar with various icons.

The screenshot shows a DataCamp exercise titled "Familiar functions | Python". The instructions explain: "The goal of this exercise is creating functions and saving the result to a variable in this function. You will learn how to use the `len()` function to count the number of characters in a string." A note says: "Great job! The `len()` function is extremely useful; it also works on strings to count the number of characters!" Below the instructions is a list of tasks:

- Use `len()` to get the length of the string `var1`.
- Use `int()` to convert `var2` to an integer: `out2`.

A "PRESS ENTER TO" button is present, along with a "Continue" button and a "Take Hint (-30 XP)" button. The interface includes tabs for "SCRIPT.PY", "IPYTHON SHELL", and "SLIDES". The system tray at the bottom shows a Windows 10 taskbar with various icons.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise titled "Calculations with variables". The exercise instructions ask the user to calculate the value of a variable after 7 years of investing \$100 at a 10% annual rate. The code provided in the script is:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The IPython Shell tab is active, showing the input "In [1]: |". The status bar at the bottom indicates it's 10:00 PM on 7/16/2018.

The screenshot shows a DataCamp exercise titled "complex()". The exercise asks the user to determine which statement about the `complex()` function is true. The possible answers are:

- `complex()` takes exactly two arguments: `real` and `[, imag]`. Press 1
- `complex()` takes two arguments: `real` and `imag`. Both these arguments are required. Press 2
- `complex()` takes two arguments: `real` and `imag`. `real` is a required argument, `imag` is an optional argument. Press 3
- `complex()` takes two arguments: `real` and `imag`. If you don't specify `imag`, it is set to 1 by Python. Press 4

The IPython Shell tab is active, showing the input "In [1]: ,my_function /home/me # becomes my_function('/home/me')". The status bar at the bottom indicates it's 10:00 PM on 7/16/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise instructions state: "Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:" followed by the code `100 * 1.10 ** 7`. The exercise provides 100 XP and includes three steps:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available. The right side of the screen shows an IPython Shell interface with the command `In [1]:` and an empty output area.

The screenshot shows a DataCamp Python course exercise titled "Use the Shell on the right to open up the documentation on `complex()`. Which of the following statements is true?". The exercise instructions provide 50 XP and list four possible answers:

- `complex()` takes exactly two arguments: `real` and `[, imag]`. Press 1.
- `complex()` takes two arguments: `real` and `imag`. Both these arguments are required. Press 2.
- `complex()` takes two arguments: `real` and `imag`. `real` is a required argument, `imag` is an optional argument. Press 3.
- `complex()` takes two arguments: `real` and `imag`. If you don't specify `imag`, it is set to 1 by Python. Press 4.

A "Submit Answer" button is present. An orange box at the bottom indicates an incorrect submission: "Incorrect. `[, imag]` shows that `imag` is an optional argument." The right side of the screen shows an IPython Shell interface with several commands entered and their outputs, including a syntax error message for `In [4]: x = ,my_function /home/me`.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

In [1]: |

In this exercise, you'll only have to specify `iterable` and `reverse`, not `key`. The first input you pass to `sorted()` will be matched to the `iterable` argument, but what about the second input? To tell Python you want to specify `reverse` without changing anything about `key`, you can use `*`:

```
sorted(..., reverse = ...)
```

Two lists have been created for you on the right. Can you paste them together and sort them in descending order?

Note: For now, we can understand an `iterable` as being any collection of objects, e.g. a List.

INSTRUCTIONS 100XP

- Use `+` to merge the contents of `first` and `second` into a new list: `full`.
- Call `sorted()` on `full` and specify the `reverse` argument to be `True`. Save the sorted list as `full_sorted`.
- Finish off by printing out `full_sorted`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create lists first and second
2 first = [11.25, 18.0, 20.0]
3 second = [10.75, 9.50]
4
5 # Paste together first and second: full
6 full=first+second
7
8 # Sort full in descending order: full_sorted
9 full_sorted=sorted(full,reverse=True)
10
11 # Print out full_sorted
12 print(full_sorted)
```

IPYTHON SHELL

```
# Paste together first and second: full
full=first+second

# Sort full in descending order: full_sorted
full_sorted=sorted(full,reverse=True)

# Print out full_sorted
print(full_sorted)
[20.0, 18.0, 11.25, 10.75, 9.5]
```

In [6]: |

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

In this exercise, you'll only have to specify `first` and `second` in one line. The first input you pass to `sorted()` will be matched to the `first` argument, but what about the second input? To tell Python you want to specify `reverse` without changing anything about `first`, you can use `full=first+second`.

+100 XP

Two lists have been created: `first` and `second`. Merge them together and sort them in descending order.

PRESS ENTER TO
Continue

NOTE: For now, we can understand `list` as any collection of objects e.g. `[1, 2, 3]`.

Become a power user!

SUBMIT ANSWER: **CTRL + SHIFT + ENTER**

See all keyboard shortcuts

SCRIPT.PY

```
1 # Create lists first and second
2 first = [11.25, 18.0, 20.0]
3 second = [10.75, 9.50]
4
5 # Paste together first and second: full
6 full=first+second
7
8 # Sort full in descending order: full_sorted
9 full_sorted=sorted(full,reverse=True)
10
11 # Print out full_sorted
12 print(full_sorted)
```

IPYTHON SHELL

```
# Sort full in descending order: full_sorted
full_sorted=sorted(full,reverse=True)

# Print out full_sorted
print(full_sorted)
[20.0, 18.0, 11.25, 10.75, 9.5]

<script.py> output:
[20.0, 18.0, 11.25, 10.75, 9.5]
```

In [6]: |

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- An "INSTRUCTIONS" section with 100 XP available, containing three tasks:
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- A "Take Hint (-30 XP)" button.

The interface also includes a "Course Outline" tab, an "IPYTHON SHELL" tab, and a "SLIDES" tab. The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating the date and time as 7/16/2018.

The screenshot shows a DataCamp Python course exercise titled "String Methods". The exercise content includes:

- A code editor window titled "SCRIPT.PY" containing the following Python code:

```
1 # string to experiment with: room
2 room = "poolhouse"
3
4 # Use upper() on room: room_up
5 room_up=room.upper()
6 # Print out room and room_up
7 print(room)
8 print(room_up)
9
10 # Print out the number of o's in room
11 room.count("o")
```
- An "INSTRUCTIONS" section with 100 XP available, containing three tasks:
 - Use the `upper()` method on `room` and store the result in `room_up`. Use the syntax for calling methods that you learned in the previous video.
 - Print out `room` and `room_up`. Did both change?
 - Print out the number of o's on the variable `room` by calling `count()` on `room` and passing the letter "`"o"`" as an input to the method. We're talking about the variable `room`, not the word "`"room"`!
- A "Take Hint (-30 XP)" button.
- An "INCORRECT SUBMISSION" message in an orange box: "For the second instruction, print out `room` using `print()`".

The interface also includes a "Course Outline" tab, an "IPYTHON SHELL" tab, and a "SLIDES" tab. The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating the date and time as 7/17/2018.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

[Take Hint \(-30 XP\)](#)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

```
In [1]: |
```

String Methods

Strings come with a bunch of methods. Follow the instructions closely to discover some of them. If you want to discover them in more detail, you can always type `help(str)` in the IPython Shell.

A string `room` has already been created for you to experiment with.

INSTRUCTIONS 100 XP

- Use the `upper()` method on `room` and store the result in `room_up`. Use the syntax for calling methods that you learned in the previous video.
- Print out `room` and `room_up`. Did both change?
- Print out the number of o's on the variable `room` by calling `count()` on `room` and passing the letter "`o`" as an input to the method. We're talking about the variable `room`, not the word "`room`"!

[Take Hint \(-30 XP\)](#)

INCORRECT SUBMISSION

Don't forget to print out the number of o's in `room`.

SCRIPT.PY

```
1 # string to experiment with: room
2 room = "poolhouse"
3
4 # Use upper() on room: room_up
5 room_up=room.upper()
6 # Print out room and room_up
7 print(room)
8 print(room_up)
9
10 # Print out the number of o's in room
11 print(room.count("o"))
12
```

IPYTHON SHELL

```
room_up=room.upper()
# Print out room and room_up
print(room)
print(room_up)

# Print out the number of o's in room
print(room.count("o"))

poolhouse
POOLHOUSE
3

In [9]: |
```

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes:

- SCRIPT.PY:** A code editor window containing the following Python script:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- INSTRUCTIONS (100 XP):**
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- ipython shell:** An empty command-line interface window labeled "In [1]: |".

The screenshot shows a DataCamp Python course exercise titled "String Methods". The exercise interface includes:

- SCRIPT.PY:** A code editor window containing the following Python script:

```
1 # string to experiment with: room
2 room = "poolhouse"
3
4 # Use upper() on room: room_up
5 room_up=room.upper()
6 # Print out room and room_up
7 print(room)
8 print(room_up)
9
10 # Print out the number of o's in room
11 print(room.count("o"))
12
```
- ipython shell:** An empty command-line interface window labeled "In [9]: |".

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise instructions ask the user to calculate the amount of money after 7 years of investing \$100 at a 10% annual rate. The code provided in the "SCRIPT.PY" editor is:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains the following steps:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available.

The "IPYTHON SHELL" tab is active, showing the input "In [1]: |".

The Windows taskbar at the bottom shows the following open windows: ch1_slides.pdf, www.rapidvaluesolutions.com, List Methods | Python, courses-intro-to-python, and a File Explorer window.

The screenshot shows a DataCamp Python course exercise titled "List Methods | Python". The exercise instructions explain that lists, floats, integers and booleans are types that have methods associated with them. It asks the user to experiment with the `index()` and `count()` methods on a list of areas.

The code provided in the "SCRIPT.PY" editor is:

```
1 # Create list areas
2 areas = [11.25, 18.0, 20.0, 10.75, 9.50]
3
4 # Print out the index of the element 20.0
5 print(areas.index(20.0))
6
7 # Print out how often 14.5 appears in areas
8 print(areas.count(14.5))
9
```

The "INSTRUCTIONS" section contains the following steps:

- Use the `index()` method to get the index of the element in `areas` that is equal to `20.0`. Print out this index.
- Call `count()` on `areas` to find out how many times `14.5` appears in the list. Again, simply print out this number.

A "Take Hint (-30 XP)" button is available.

The "IPYTHON SHELL" tab is active, showing the input "In [5]: # Create list areas areas = [11.25, 18.0, 20.0, 10.75, 9.50]". Below it, the output "2" is shown for the previous command.

The Windows taskbar at the bottom shows the following open windows: ch1_slides.pdf, www.rapidvaluesolutions.com, List Methods | Python, courses-intro-to-python, and a File Explorer window.

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes a "SCRIPT.PY" code editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

A text box contains instructions: "Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:" followed by the calculation `100 * 1.10 ** 7`. The "INSTRUCTIONS" section lists the following tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is available.

The screenshot shows a DataCamp Python course exercise titled "List Methods (2)". The exercise interface includes a "SCRIPT.PY" code editor with the following code:

```
1 # Create list areas
2 areas = [11.25, 18.0, 20.0, 10.75, 9.50]
3
4 # Use append twice to add poolhouse and garage size
5 areas.append("poolhouse")
6 areas.append("garage size")
7
8
9 # Print out areas
10 print(areas)
11
12 # Reverse the orders of the elements in areas
13 areas.reverse()
14
15 # Print out areas
16 print(areas)
```

The "INSTRUCTIONS" section lists the following tasks:

- Use `append()` twice to add the size of the poolhouse and the garage again: `24.5` and `15.45`, respectively. Make sure to add them in this order.
- Print out `areas`.
- Use the `reverse()` method to reverse the order of the elements in `areas`.
- Print out `areas` once more.

A "Take Hint (-30 XP)" button is available.

The "IPYTHON SHELL" tab shows the command `# Print out areas` and its output: `print(areas)` and `[11.25, 18.0, 20.0, 10.75, 9.5, 'poolhouse', 'garage size']`.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

[Take Hint \(-30 XP\)](#)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

```
In [1]: |
```

List Methods (2)

Most list methods will change the list they're called on. Examples are:

- `append()`, that adds an element to the list it is called on,
- `remove()`, that removes the first element of a list that matches the input, and
- `reverse()`, that reverses the order of the elements in the list it is called on.

You'll be working on the list with the area of different parts of the house: `areas`.

INSTRUCTIONS 100 XP

- Use `append()` twice to add the size of the poolhouse and the garage again: `24.5` and `15.45`, respectively. Make sure to add them in this order.
- Print out `areas`.
- Use the `reverse()` method to reverse the order of the elements in `areas`.
- Print out `areas` once more.

[Take Hint \(-30 XP\)](#)

INCORRECT SUBMISSION

Use the `append()` method on `areas` to expand with `24.5` the first time.

SCRIPT.PY

```
1 # Create list areas
2 areas = [11.25, 18.0, 20.0, 10.75, 9.5]
3
4 # Use append twice to add poolhouse and garage size
5 areas.append(24.5)
6 areas.append(15.45)
7
8
9 # Print out areas
10 print(areas)
11
12 # Reverse the orders of the elements in areas
13 areas.reverse()
14
15 # Print out areas
16 print(areas)
```

IPYTHON SHELL

```
# Print out areas
print(areas)

# Reverse the orders of the elements in areas
areas.reverse()

# Print out areas
print(areas)
[11.25, 18.0, 20.0, 10.75, 9.5, 24.5, 15.45]
[15.45, 24.5, 9.5, 10.75, 20.0, 18.0, 11.25]
```

In [7]: |

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise window. The title bar indicates the page is secure and shows the URL: https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-1-python-basics?ex=8. The main area is titled "SCRIPT.PY" and contains the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

To the left of the code editor is a sidebar with "INSTRUCTIONS 100 XP" containing the following tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Below the instructions is a "Take Hint (-30 XP)" button.

At the bottom of the exercise window is an "IPYTHON SHELL" tab with the prompt "In [1]: |".

The taskbar at the bottom of the screen shows several open windows, including "ch1_slides.pdf", "www.rapidvaluesolutions.com", "List Methods (2) | Python", and "courses-intro-to-python". The system tray shows the date as 7/16/2018.

The screenshot shows a DataCamp Python course exercise window. The title bar indicates the page is secure and shows the URL: https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-3-functions-and-packages?ex=8. The main area is titled "SCRIPT.PY" and contains the following Python code:

```
1 # Create list areas
2 areas = [11.25, 18.0, 20.0, 10.75, 9.5]
3
4 # Use append twice to add poolhouse and garage size
5 areas.append(24.5)
6 areas.append(15.45)
7
8
9 # Print out areas
10 print(areas)
11
12 # Reverse the orders of the elements in areas
13 areas.reverse()
14
15 # Print out areas
16 print(areas)
```

To the left of the code editor is a sidebar with "INSTRUCTIONS 100 XP" containing the following tasks:

- `append()` that adds an element to the list it's called on.
- `remove()` that removes the first element of a list that matches the input, and **+100 XP**.
- `reverse()`, that reverses the order of the elements in the list it's called on.

Below the instructions is a "Great!" message and a "PRESS ENTER TO Continue" button.

At the bottom of the exercise window is an "IPYTHON SHELL" tab with the prompt "In [7]: |". The shell output shows the execution of `areas.reverse()` and the resulting list: `[11.25, 18.0, 20.0, 10.75, 9.5, 24.5, 15.45]` followed by `[15.45, 24.5, 9.5, 10.75, 20.0, 18.0, 11.25]`.

The taskbar at the bottom of the screen shows several open windows, including "ch1_slides.pdf", "www.rapidvaluesolutions.com", "List Methods (2) | Python", and "courses-intro-to-python". The system tray shows the date as 7/17/2018.

● Datacamp datascience course by

The screenshot shows a DataCamp exercise window. On the left, there's an 'EXERCISE' panel titled 'Calculations with variables'. It contains instructions and a code editor with the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the code editor is an 'INSTRUCTIONS' section with 100 XP available. It lists three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

There's also a 'Take Hint (-30 XP)' button. On the right side of the window, there's an 'IPYTHON SHELL' tab and a 'SLIDES' tab. The IPython Shell tab has the text 'In [1]: |'.

The screenshot shows a DataCamp video player window. The title of the video is 'Import package'. The video frame shows a man in a black polo shirt with the DataCamp logo on it. To the left of the video frame is a code editor window showing the following Python code:

```
In [1]: import numpy
In [2]: array([1, 2, 3])
NameError: name 'array' is not defined
In [3]: numpy.array([1, 2, 3])
Out[3]: array([1, 2, 3])
In [4]: import numpy as np
In [5]: np.array([1, 2, 3])
Out[5]: array([1, 2, 3])
```

At the bottom of the video player, there are playback controls (play/pause, volume, etc.) and a progress bar. A yellow 'Got it!' button is located at the bottom right of the video frame. The video player is part of a larger browser window with multiple tabs open, including 'ch1_slides.pdf', 'www.rapidvaluesolutions.com', 'Packages | Python', and 'courses/intro-to-python'. The browser taskbar at the bottom shows various pinned icons and the date/time as 7/17/2018 1:32 AM.

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

```
In [1]: |
```

Import package

As a data scientist, some notions of geometry never hurt. Let's refresh some of the basics.

For a fancy clustering algorithm, you want to find the circumference, C , and area, A , of a circle. When the radius of the circle is `r`, you can calculate C and A as:

$$C = 2\pi r$$
$$A = \pi r^2$$

To use the constant `pi`, you'll need the `math` package. A variable `r` is already coded in the script. Fill in the code to calculate `C` and `A` and see how the `print()` functions create some nice printouts.

INSTRUCTIONS 100 XP

- Import the `math` package. Now you can access the constant `pi` with `math.pi`.
- Calculate the circumference of the circle and store it in `C`.
- Calculate the area of the circle and store it in `A`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Definition of radius
2 r = 0.43
3
4 # Import the math package
5 import math
6
7 # Calculate C
8 C = 0
9 C=2*r*math.pi
10
11 # Calculate A
12 A = 0
13 A=r**r*math.pi
14
15 # Build printout
16 print("Circumference: " + str(C))
17 print("Area: " + str(A))
```

IPYTHON SHELL

```
# Calculate A
A = 0
A=r**r*math.pi

# Build printout
print("Circumference: " + str(C))
print("Area: " + str(A))
Circumference: 2.70176962087222
Area: 0.588804816487527
```

Run Code **Submit Answer**

IN [6]: |

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes a sidebar with "EXERCISE" and "INSTRUCTIONS" sections, a main content area with a "SCRIPT.PY" code editor, and an "IPYTHON SHELL" tab.

INSTRUCTIONS (100 XP)

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

```
In [1]: |
```

The screenshot shows a DataCamp Python course exercise titled "Selective import". The exercise interface includes a sidebar with "EXERCISE" and "INSTRUCTIONS" sections, a main content area with a "SCRIPT.PY" code editor, and an "IPYTHON SHELL" tab.

INSTRUCTIONS (100 XP)

- Perform a selective import from the `math` package where you only import the `radians` function.
- Calculate the distance travelled by the Moon over 12 degrees of its orbit. Assign the result to `dist`. You can calculate this as `r * phi`, where `r` is the radius and `phi` is the angle in radians. To convert an angle in degrees to an angle in radians, use the `radians()` function, which you just imported.
- Print out `dist`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Definition of radius
2 r = 192500
3
4 # Import radians function of math package
5
6 from math import radians
7 # Travel distance of Moon over 12 degrees. Store in dist.
8 dist=r*radians(12)
9
10 # Print out dist
11 print(dist)
```

IPYTHON SHELL

```
# Import radians function of math package
from math import radians
# Travel distance of Moon over 12 degrees. Store in dist.
dist=r*radians(12)

# Print out dist
print(dist)
40317.18572186981
```

In [4]: |

● Datacamp datascience course by

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

```
In [1]: |
```

Selective import

General imports, like `from math import *`, make all functionality from the `math` package available to you. However, if you don't want to use a specific part of a package, you can always make your own! **+100 XP** active.

Nice! Head over to the next exercise.

Please rate this exercise:
★★★★★

PRESS ENTER TO

INSTRUCTIONS 100 XP

Perform a selective import from the `math` module where you only import the `radians` function.

Calculate the distance travelled by the Moon over 12 degrees of its orbit. Assign the result to `dist`. You can calculate this as $\pi \times \text{radius} \times \theta$, where `radius` is the radius and `theta` is the angle in radians. To convert an angle in degrees to an angle in radians, multiply it by $\frac{\pi}{180}$.

Become a power user!

SUBMIT ANSWER: **CTRL + SHIFT + ENTER**

See all keyboard shortcuts

SCRIPT.PY

```
1 # Definition of radius
2 r = 192500
3
4 # Import radians function of math package
5
6 from math import radians
7 # Travel distance of Moon over 12 degrees. Store in dist.
8 dist=r*radians(12)
9
10 # Print out dist
11 print(dist)
```

IPYTHON SHELL

```
from math import radians
# Travel distance of Moon over 12 degrees. Store in dist.
dist=r*radians(12)

# Print out dist
print(dist)
40317.10572106901

<script.py> output:
40317.10572106901
```

In [4]: |

● Datacamp datascience course by

The screenshot shows a DataCamp Python course exercise titled "Calculations with variables". The exercise interface includes:

- SCRIPT.PY** tab: Displays the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- INSTRUCTIONS 100 XP** tab: Contains the following steps:
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- I PYTHON SHELL** tab: Shows the command `In [1]: |` followed by a blank input field.
- SLIDES** tab: Shows a thumbnail of the slide content.

The screenshot shows a DataCamp Python course exercise titled "Different ways of importing". The exercise interface includes:

- INSTRUCTIONS 50 XP** tab: Contains the following steps:
 - Possible Answers:
 - `import scipy`
 - `import scipy.linalg`
 - `from scipy.linalg import my_inv`
 - `from scipy.linalg import inv as my_inv`
 - Enter** button: A large blue button with white text.
 - Press** buttons: Four numbered buttons (1, 2, 3, 4) corresponding to the answer options.
 - Take Hint (-15 XP)** button: A small button with a question mark icon.- I PYTHON SHELL** tab: Shows the following interaction:

```
In [1]: my_inv([[1,2], [3,4]])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    my_inv([[1,2], [3,4]])
NameError: name 'my_inv' is not defined
```

```
In [2]: import scipy.linalg
In [3]: my_inv([[1,2], [3,4]])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    my_inv([[1,2], [3,4]])
NameError: name 'my_inv' is not defined
```

```
In [4]: from scipy.linalg import my_inv
In [5]: my_inv([[1,2], [3,4]])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    my_inv([[1,2], [3,4]])
NameError: name 'my_inv' is not defined
```

```
In [6]: from scipy.linalg import my_inv
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    from scipy.linalg import my_inv
ImportError: No module named 'scipy.linalg'
```

```
In [7]: from scipy.linalg import import inv as my_inv
File "<stdin>", line 1
  from scipy.linalg import import inv as my_inv
^
SyntaxError: invalid syntax
```

```
In [8]: from scipy.linalg import inv as my_inv
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    from scipy.linalg import inv as my_inv
ImportError: No module named 'scipy.linalg'
```

```
In [9]: my_inv([[1,2], [3,4]])
Out[9]:
array([[-2.,  1.],
       [ 1.5, -0.5]])
```

```
In [10]: |
```
- SLIDES** tab: Shows a thumbnail of the slide content.

● Datacamp datascience course by

A screenshot of a DataCamp Python course exercise window. The title bar shows "Calculations with variables" and "courses-intro-to-python". The main area has tabs for "EXERCISE", "SCRIPT.PY", and "IPYTHON SHELL". The "EXERCISE" tab contains instructions: "Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:" followed by the code "100 * 1.10 ** 7". Below this, it says "Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!". The "INSTRUCTIONS" section lists three tasks: "Create a variable `factor`, equal to `1.10`.", "Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.", and "Print out the value of `result`.". A "Take Hint (-30 XP)" button is present. The "SCRIPT.PY" tab shows the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "IPYTHON SHELL" tab shows "In [1]: |". The taskbar at the bottom shows multiple open windows including "ch1_slides.pdf", "www.rapidvaluesolutions.com", "NumPy | Python", and "courses-intro-to-python". The system tray shows the date and time as 7/16/2018.

A screenshot of a DataCamp NumPy course illustration window. The title bar shows "ch1_slides.pdf", "www.rapidvaluesolutions.com", "NumPy | Python", and "courses-intro-to-python". The main area has tabs for "Course Outline" and "Slides". The "Slides" tab is active and displays a slide titled "NumPy Illustration". The slide content includes a code block showing "In [1]: height = [1.73, 1.68, 1.71, 1.89, 1.79]" and "Out[2]: [1.73, 1.68, 1.71, 1.89, 1.79]". To the right of the code block is a video player showing a man in a black polo shirt with a DataCamp logo. A yellow "Got it!" button is at the bottom right of the slide. The taskbar at the bottom shows multiple open windows including "ch1_slides.pdf", "www.rapidvaluesolutions.com", "NumPy | Python", and "courses-intro-to-python". The system tray shows the date and time as 7/17/2018.

● Datacamp datascience course by

A screenshot of a DataCamp Python course exercise window. The window title is "Calculations with variables". The main area shows a script named "SCRIPT.PY" with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains the following tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Below the instructions is a "Take Hint (-30 XP)" button. The "IPYTHON SHELL" tab is active, showing the input "In [1]: |". The status bar at the bottom shows "Type here to search" and system information like "ENG 10:00 PM IN 7/16/2018".

A screenshot of a DataCamp NumPy course illustration window. The title is "NumPy". The main area shows a section titled "Illustration" with the following Python code in the IPython shell:

```
In [1]: height = [1.73, 1.68, 1.71, 1.89, 1.79]
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]
Out[4]: [65.4, 59.2, 63.6, 88.4, 68.7]
```

To the right of the code is a video player showing a man in a black polo shirt. A yellow "Got it!" button is visible at the bottom right of the video player. The status bar at the bottom shows "Type here to search" and system information like "ENG 1:50 AM IN 7/17/2018".

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. At the top, there are three tabs: 'Calculations with variables', 'courses-intro-to-python', and 'ch1_slides.pdf'. The main area has a 'Course Outline' header. On the left, an 'EXERCISE' panel titled 'Calculations with variables' contains instructions and a code editor with the following script:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the script is an 'INSTRUCTIONS' section with 100 XP available. It lists three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

There is also a 'Take Hint (-30 XP)' button.

At the bottom, there are tabs for 'IPYTHON SHELL' and 'SLIDES'. The IPython Shell tab shows 'In [1]: |'.

The screenshot shows a DataCamp course interface. At the top, there are three tabs: 'ch1_slides.pdf', 'www.rapidvaluesolutions.com', and 'NumPy | Python'. The main area has a 'Course Outline' header. On the left, an 'EXERCISE' panel titled 'NumPy' contains an 'Illustration' section. The illustration features a man in a black polo shirt with a DataCamp logo. To his right is a yellow 'Got it!' button. The illustration includes a code editor with the following Python code:

```
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
```

At the bottom, there are tabs for 'IPYTHON SHELL' and 'SLIDES'. The IPython Shell tab shows 'In [1]: |'.

● Datacamp datascience course by

The screenshot shows a DataCamp course interface. On the left, there's an 'EXERCISE' panel titled 'Calculations with variables'. It contains instructions: 'Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:' followed by the code `100 * 1.10 ** 7`. Below this, a note says: 'Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!'. An 'INSTRUCTIONS 100 XP' section lists three tasks: 'Create a variable `factor`, equal to `1.10`', 'Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`', and 'Print out the value of `result`'. A 'Take Hint (-30 XP)' button is present. On the right, there's a 'SCRIPT.PY' editor with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

Below the editor is an 'IPYTHON SHELL' tab labeled 'SLIDES' and an 'In [1]: |' input field. The bottom of the window shows a Windows taskbar with various icons.

The screenshot shows a DataCamp video player window. The title bar says 'ch1_slides.pdf', 'www.rapidvaluesolutions.com', 'NumPy | Python', and 'courses/intro-to-python'. The main content area is titled 'NumPy' and features a video player with a thumbnail of a man in a black shirt. The video title is 'Solution: NumPy' and the subtitle is '• Numeric Python'. A 'Got it!' button is at the bottom right. The bottom of the window shows a Windows taskbar with various icons.

● Datacamp datascience course by

A screenshot of a DataCamp course window titled "Calculations with variables". The window is divided into several sections: "EXERCISE" on the left containing instructions and code snippets; "SCRIPT.PY" in the center showing a Python script with code for calculating compound interest; "IPYTHON SHELL" at the bottom where users can run their code; and "SLIDES" on the right. The operating system taskbar at the bottom shows other open windows like "ch1_slides.pdf" and "NumPy | Python".

EXERCISE

Calculations with variables

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

INSTRUCTIONS 100 XP

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

Take Hint (-30 XP)

SCRIPT.PY

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

IPYTHON SHELL

In [1]: |

A screenshot of a DataCamp video player titled "Solution: NumPy". The video features a man in a black polo shirt speaking. To the right of the video frame is a progress bar showing "50XP". The operating system taskbar at the bottom shows other open windows like "ch1_slides.pdf" and "NumPy | Python".

NumPy

Solution: NumPy

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

Got it!

● Datacamp datascience course by

A screenshot of a DataCamp course window titled "Calculations with variables". The window is divided into several sections: "EXERCISE" (containing instructions and a code editor), "SCRIPT.PY" (containing a script with code), "IPYTHON SHELL" (containing an input field "In [1]:"), and "SLIDES" (which is currently inactive). The code in the "SCRIPT.PY" section is:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```

The "INSTRUCTIONS" section contains the following text:

Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:

```
100 * 1.10 ** 7
```

Instead of calculating with the actual values, you can use variables instead. The `savings` variable you've created in the previous exercise represents the \$100 you started with. It's up to you to create a new variable to represent `1.10` and then redo the calculations!

The "INSTRUCTIONS" section also lists three tasks:

- Create a variable `factor`, equal to `1.10`.
- Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
- Print out the value of `result`.

A "Take Hint (-30 XP)" button is also present.

A screenshot of a DataCamp video lesson titled "NumPy". The video player interface shows a progress bar, volume control, and other standard video controls. The video content features a DataCamp instructor speaking. The video title is "NumPy" and it is part of the "Intro to Python for Data Science" series. The video has a rating of 50XP.

The video player displays the following Python code in the "IPYTHON SHELL" section:

```
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]
```

● Datacamp datascience course by

The screenshot shows a DataCamp course interface for "Introduction to Python for Data Science". The main window title is "Calculations with variables". The URL in the browser bar is <https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-1-python-basics?ex=8>. The interface includes:

- EXERCISE** tab: Shows the title "Calculations with variables". Below it, a note says: "Remember how you calculated the money you ended up with after 7 years of investing \$100? You did something like this:" followed by the code `100 * 1.10 ** 7`.
- SCRIPT.PY** tab: Displays the following Python script:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable factor
5
6
7 # Calculate result
8
9
10 # Print out result
```
- INSTRUCTIONS 100 XP**:
 - Create a variable `factor`, equal to `1.10`.
 - Use `savings` and `factor` to calculate the amount of money you end up with after 7 years. Store the result in a new variable, `result`.
 - Print out the value of `result`.
- IPYTHON SHELL** tab: Shows the prompt "In [1]: |".
- SLIDES** tab: Shows a small thumbnail of a slide.
- System Taskbar**: Shows the Windows Start button, a search bar with "Type here to search", pinned icons for File Explorer, Edge, and others, and system status indicators including battery level, signal strength, and date/time (7/16/2018, 10:00 PM).