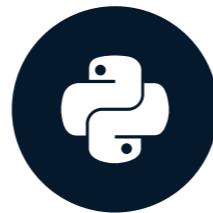


# Hello Python!

INTRODUCTION TO PYTHON



**Hugo Bowne-Anderson**  
Data Scientist at DataCamp

# How you will learn

The screenshot shows a DataCamp exercise page. On the left, the exercise title is "Calculations with variables". The instructions section contains three bullet points: "Create a variable `growth_multiplier` equal to `1.1`.", "Create a variable, `result`, equal to the amount of money you saved after `7` years.", and "Print out the value of `result`". A "Take Hint (-30 XP)" button is visible below the instructions. The main area is a code editor with a dark theme, showing a Python script named `script.py` with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable growth_multiplier
5 growth_multiplier = 1.1
6
7 # Calculate result
8 result = savings *
9
10 # Print out result
11
12
13
```

Below the code editor are three buttons: a refresh button, a "Run Code" button, and a "Submit Answer" button. At the bottom of the interface, there is an "IPython Shell" window with a "Slides" tab and a prompt "In [1]:".

# Python



- General purpose: build anything
- Open source! Free!
- Python packages, also for data science
  - Many applications and fields
- Version 3.x - <https://www.python.org/downloads/>

# IPython Shell

## Execute Python commands

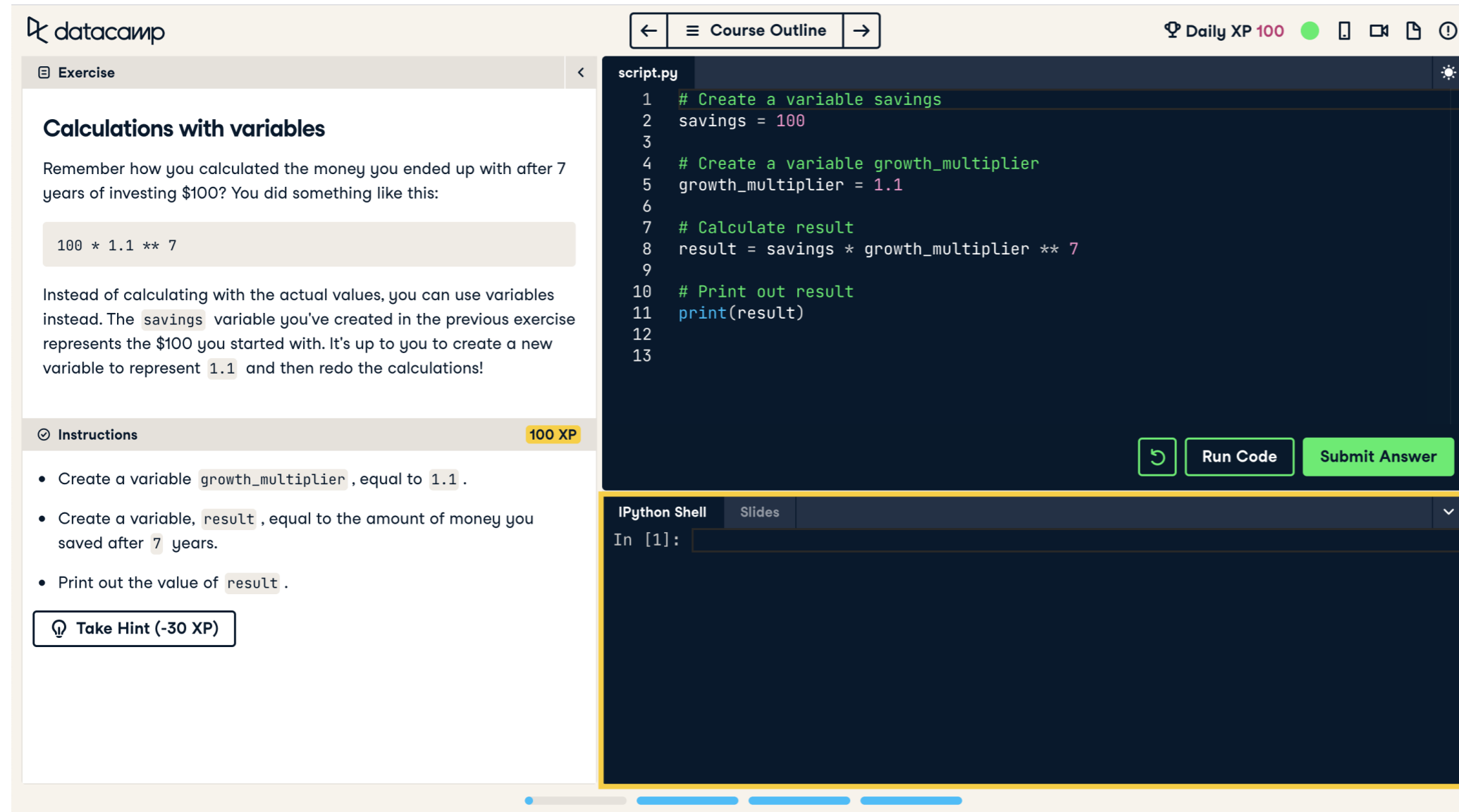
The screenshot shows a DataCamp exercise interface. On the left, the exercise title is "Calculations with variables". The instructions section, worth 100 XP, asks the user to create a variable `growth_multiplier` equal to 1.1, a variable `result` equal to the amount of money saved after 7 years, and to print out the value of `result`. A "Take Hint (-30 XP)" button is visible. The main area contains a code editor for `script.py` with the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable growth_multiplier
5 growth_multiplier = 1.1
6
7 # Calculate result
8 result = savings * growth_multiplier ** 7
9
10 # Print out result
11 print(result)
12
13
```

Below the code editor is an IPython Shell with a "Slides" tab and a "v" dropdown arrow. The shell prompt is "In [1]:". At the bottom of the interface, there are "Run Code" and "Submit Answer" buttons.

# IPython Shell

## Execute Python commands



The screenshot shows a DataCamp exercise interface. On the left, the exercise title is "Calculations with variables". The instructions ask the user to create a variable `growth_multiplier` equal to `1.1`, a variable `result` equal to the amount of money saved after 7 years, and to print out the value of `result`. A "Take Hint (-30 XP)" button is visible. The main area contains a code editor with the following Python code:

```
script.py
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable growth_multiplier
5 growth_multiplier = 1.1
6
7 # Calculate result
8 result = savings * growth_multiplier ** 7
9
10 # Print out result
11 print(result)
12
13
```

Below the code editor is an IPython Shell with a "Slides" tab and a dropdown arrow. The shell prompt is "In [1]:". At the bottom of the interface, there are three blue progress bars.

# IPython Shell

The screenshot shows a DataCamp exercise page. On the left, the exercise title is "Calculations with variables". Below the title, there is a text block explaining the task: "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 * 1.1 ** 7`. Below this, another text block 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 `1.1` and then redo the calculations!".

Below the text is an "Instructions" section with a "100 XP" badge. It contains three bullet points:

- Create a variable `growth_multiplier`, equal to `1.1`.
- Create a variable, `result`, equal to the amount of money you saved after `7` years.
- Print out the value of `result`.

At the bottom of the instructions is a "Take Hint (-30 XP)" button.

On the right side of the interface is a code editor window titled "script.py" with a dark background. It contains the following Python code:

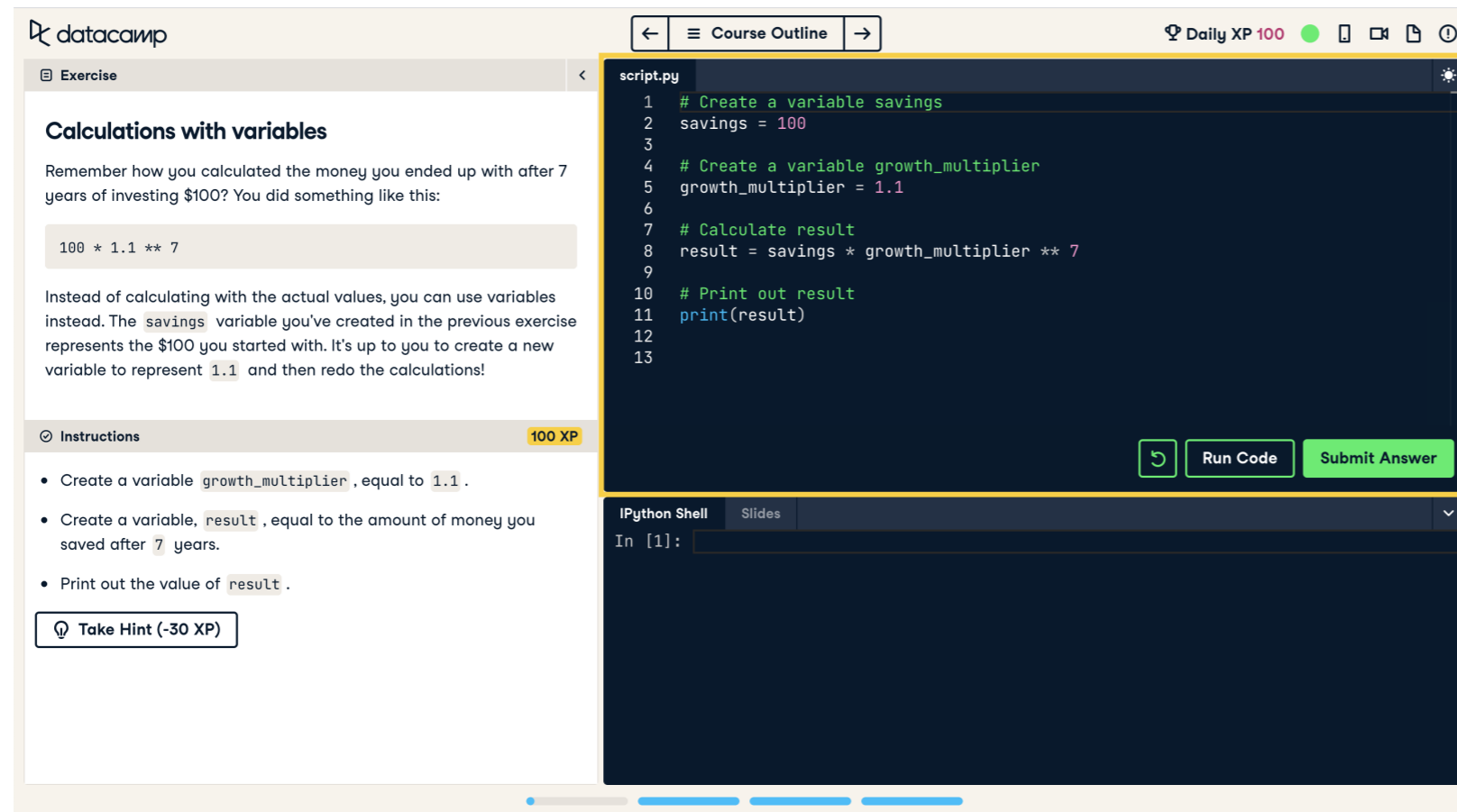
```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable growth_multiplier
5
6
7 # Calculate result
8
9
10 # Print out result
11
```

Below the code editor are three buttons: a refresh button, a "Run Code" button, and a "Submit Answer" button.

At the bottom of the interface is an IPython Shell window with a dark background. It shows the prompt "In [1]:" followed by a cursor.

# Python Script

- Text files - `.py`
- List of Python commands
- Similar to typing in IPython Shell



The screenshot shows a DataCamp exercise interface. On the left, the exercise title is "Calculations with variables". 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 * 1.1 ** 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 `1.1` and then redo the calculations!

**Instructions** 100 XP

- Create a variable `growth_multiplier`, equal to `1.1`.
- Create a variable, `result`, equal to the amount of money you saved after `7` years.
- Print out the value of `result`.

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

On the right, a code editor shows a Python script named `script.py` with the following code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable growth_multiplier
5 growth_multiplier = 1.1
6
7 # Calculate result
8 result = savings * growth_multiplier ** 7
9
10 # Print out result
11 print(result)
12
13
```

Below the code editor is an IPython Shell with the prompt `In [1]:`. At the bottom right of the code editor are buttons for `Run Code` and `Submit Answer`.

# Python Script

The screenshot shows a DataCamp exercise interface. On the left, the exercise title is "Calculations with variables". The instructions section, worth 100 XP, contains three bullet points: "Create a variable `growth_multiplier` equal to `1.1`.", "Create a variable, `result`, equal to the amount of money you saved after `7` years.", and "Print out the value of `result`.". A "Take Hint (-30 XP)" button is located below the instructions. On the right, a code editor window titled "script.py" is open, showing a single line of code: `1`. Below the code editor are three buttons: a refresh button, a "Run Code" button, and a "Submit Answer" button. At the bottom of the interface, a "Python Shell" window is visible, showing the prompt "In [1]:".

# Python Script

The screenshot shows a DataCamp exercise interface. On the left, the exercise title is "Calculations with variables". Below the title, there is a paragraph of text: "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 block containing the expression `100 * 1.1 ** 7`. Below this, another paragraph explains that instead of using actual values, variables can be used. It mentions a variable `savings` and asks the user to create a new variable to represent `1.1` and redo the calculations. Underneath, there is an "Instructions" section with a "100 XP" badge and three bullet points: "Create a variable `growth_multiplier` equal to `1.1`.", "Create a variable, `result`, equal to the amount of money you saved after `7` years.", and "Print out the value of `result`.". A "Take Hint (-30 XP)" button is located below the instructions. On the right side of the interface, there is a code editor window titled "script.py" with a line number "1" and a cursor. Below the code editor are three buttons: a refresh button, a "Run Code" button, and a "Submit Answer" button. At the bottom of the interface, there is a "Python Shell" window with a "Slides" tab and a prompt "In [1]:".

- Use `print()` to generate output from script

# DataCamp Interface

The screenshot displays the DataCamp interface for an exercise. On the left, the exercise title is "Calculations with variables". The instructions section, worth 100 XP, asks the user to create a variable `growth_multiplier` equal to 1.1, a variable `result` equal to the amount of money saved after 7 years, and to print out the value of `result`. A "Take Hint (-30 XP)" button is visible. The main area is a code editor for a file named `script.py`. The code contains the following Python code:

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable growth_multiplier
5
6
7 # Calculate result
8
9
10 # Print out result
11
```

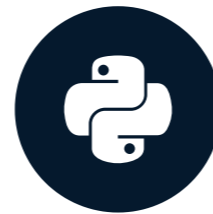
Below the code editor is an IPython Shell with a "Slides" tab. The shell prompt is "In [1]:". At the bottom of the interface, there are "Run Code" and "Submit Answer" buttons, along with a refresh icon.

# Let's practice!

INTRODUCTION TO PYTHON

# Variables and Types

INTRODUCTION TO PYTHON



**Hugo Bowne-Anderson**  
Data Scientist at DataCamp

# Variable

- Specific, case-sensitive name
- Call up value through variable name
- 1.79 m - 68.7 kg

```
height = 1.79  
weight = 68.7  
height
```

```
1.79
```

# Calculate BMI

```
height = 1.79  
weight = 68.7  
height
```

```
1.79
```

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2}$$

```
68.7 / 1.79 ** 2
```

```
21.4413
```

```
weight / height ** 2
```

```
21.4413
```

```
bmi = weight / height ** 2  
bmi
```

```
21.4413
```

# Reproducibility

```
height = 1.79  
weight = 68.7  
bmi = weight / height ** 2  
print(bmi)
```

```
21.4413
```

# Reproducibility

```
height = 1.79
weight = 74.2 # <-
bmi = weight / height ** 2
print(bmi)
```

```
23.1578
```

# Python Types

```
type(bmi)
```

```
float
```

```
day_of_week = 5  
type(day_of_week)
```

```
int
```

# Python Types (2)

```
x = "body mass index"  
y = 'this works too'  
type(y)
```

str

```
z = True  
type(z)
```

bool

# Python Types (3)

```
2 + 3
```

```
5
```

```
'ab' + 'cd'
```

```
'abcd'
```

- Different type = different behavior!

# Let's practice!

INTRODUCTION TO PYTHON