|  |  |
| --- | --- |
| **Variable Assignment**  100xp  In Python, a variable allows you to refer to a value with a name. To create a variable use =, like this example:  x = 5  You can now use the name of this variable, x, instead of the actual value, 5.  Remember, = in Python means *assignment*, not equality!  **Instructions**   * Create a variable savings with the value 100. * Check out this variable by typing print(savings) in the script.   [**Take Hint (-30xp)**](https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-1-python-basics?ex=7) | # Create a variable savings  # Print out savings |

|  |  |
| --- | --- |
| Calculations with variables 100xp  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  * 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. | # Create a variable savings  savings = 100  # Create a variable factor  # Calculate result  # Print out result |

|  |  |
| --- | --- |
| Other variable types 100xp  In the previous exercise, you worked with two Python data types:   * int, or integer: a number without a fractional part. savings, with the value 100, is an example of an integer. * float, or floating point: a number that has both an integer and 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  * Create a new string, desc, with the value "compound interest". * Create a new boolean, profitable, with the value True. | # Create a variable desc  # Create a variable profitable |

|  |  |
| --- | --- |
| Guess the type 50xp  To find out the type of a value or a variable that refers to that value, you can use the [type()](https://docs.python.org/3/library/functions.html#type) function. Suppose you've defined a variable a, but you forgot the type of this variable. To determine the type of a, simply execute:  type(a)  We already went ahead and created three variables: a, b and c. You can use the IPython shell on the right to discover their type. Which of the following options is correct? Possible Answers     a is of type int, b is of type str, c is of type bool      a is of type float, b is of type bool, c is of type str      a is of type float, b is of type str, c is of type bool      a is of type int, b is of type bool, c is of type str |  |
| Operations with other types 100xp  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  * 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? | # Several variables to experiment with  savings = 100  factor = 1.1  desc = "compound interest"  # Assign product of factor and savings to year1  # Print the type of year1  # Assign sum of desc and desc to doubledesc  # Print out doubledesc |

|  |  |
| --- | --- |
| Type conversion 100xp  Using the + operator to paste together two strings can be very useful in building custom messages.  Suppose for example that you've calculated the return of your investment, and want to summarize the results in a string. Assuming the floats savings and result are defined, you can try something like this:  print("I started with $" + savings + " and now have $" + result + ". Awesome!")  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()](https://docs.python.org/3/library/functions.html" \l "func-str" \t "_blank), to convert a value into a string. str(savings), for example, will convert the float savings to a string.  Similar functions such as [int()](https://docs.python.org/3/library/functions.html" \l "int" \t "_blank), [float()](https://docs.python.org/3/library/functions.html#float) and [bool()](https://docs.python.org/3/library/functions.html#bool) will help you convert Python values into any type. Instructions  * Hit Submit Answer 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()](https://docs.python.org/3/library/functions.html" \l "func-str" \t "_blank) to convert the variables to strings. * Convert the variable pi\_string to a float and store this float as a new variable, pi\_float. | # Definition of savings and result  savings = 100  result = 100 \* 1.10 \*\* 7  # Fix the printout  print("I started with $" + savings + " and now have $" + result + ". Awesome!")  # Definition of pi\_string  pi\_string = "3.1415926"  # Convert pi\_string into float: pi\_float |
| Can Python handle everything? 50xp  Now that you know something more about combining different sources of information, have a look at the four Python expressions below. Which one of these will throw an error? You can always copy and paste this code in the IPython Shell to find out! Possible Answers     "I can add integers, like " + str(5) + " to strings."      "I said " + ("Hey " \* 2) + "Hey!"      "The correct answer to this multiple choice exercise is answer number " + 2      True + False |  |