|  |  |
| --- | --- |
| **Subset and conquer**  100xp  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 = list["a", "b", "c", "d"]  x[1]  x[-3] # same result!  Remember the areas list from before, containing both strings and floats? Its definition is already in the script. Can you add the correct code to do some Python subsetting?  **Instructions**   * 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. | # Create the areas list  areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]  # Print out second element from areas  # Print out last element from areas  # Print out the area of the living room |

|  |  |
| --- | --- |
| **Subset and calculate**  100xp  After you've extracted values from a list, you can use them to perform additional calculations. Take this example, where the second and fourth element of a list x are extracted. The strings that result are pasted together using the + operator:  x = ["a", "b", "c", "d"]  print(x[1] + x[3])  **Instructions**   * 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 this new variable eat\_sleep\_area. | # Create the areas list  areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]  # Sum of kitchen and bedroom area: eat\_sleep\_area  # Print the variable eat\_sleep\_area |

|  |  |
| --- | --- |
| **Slicing and dicing**  100xp  Selecting single values from a list is just one part of the story. It's also possible to *slice* your list, which means selecting multiple elements from your list. Use the following syntax:  my\_list[start:end]  The start index will be included, while the end index is *not*.  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**   * 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()](https://docs.python.org/3/library/functions.html#print). | # Create the areas list  areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]  # Use slicing to create downstairs  # Use slicing to create upstairs  # Print out downstairs and upstairs |

|  |  |
| --- | --- |
| Slicing and dicing (2) 100xp  In the video, Filip only discussed the syntax where you specify both where to begin and end the slice of your list:  my\_list[begin:end]  However, it's also possible not to specify these indexes. If you don't specify the begin index, Python figures out that you want to start your slice at the beginning of your list. If you don't specify the end index, the slice will go all the way to the last element of your list. To experiment with this, try the following commands in the IPython Shell:  x = ["a", "b", "c", "d"]  x[:2]  x[2:]  x[:] Instructions 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. | # Create the areas list  areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]  # Alternative slicing to create downstairs  # Alternative slicing to create upstairs |

|  |  |
| --- | --- |
| Subsetting lists of lists 50xp  You saw before that a Python list can contain practically anything; even other lists! To subset lists of lists, you can use the same technique as before: square brackets. Try out the commands in the following code sample in the IPython Shell:  x = [["a", "b", "c"],  ["d", "e", "f"],  ["g", "h", "i"]]  x[2][0]  x[2][:2]  x[2] results in a list, that you can subset again by adding additional square brackets.  What will house[-1][1] return? house, the list of lists that you created before, is already defined for you in the workspace. You can experiment with it in the IPython Shell. | Possible Answers     A float: the kitchen area      A string: "kitchen"      A float: the bathroom area      A string: "bathroom" |