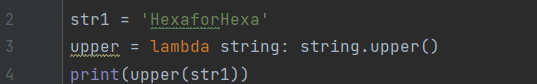
**Python Lambda Function**

In this example, we made a little function called 'upper' using a lambda function. What this 'upper' function does is take a word and make it all uppercase using the 'upper()' method. We put this function to work by using it on the word 'HexaforHexa' and displaying the output.

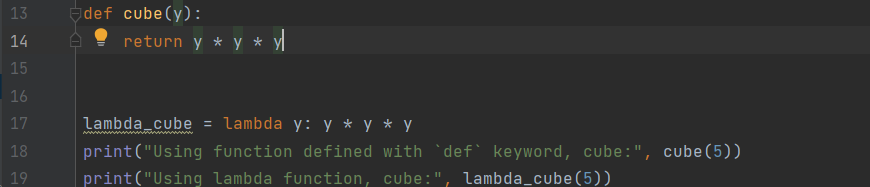
****

**Output:**

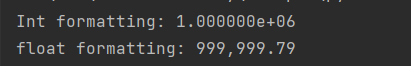
****

**Difference Between Lambda functions and def defined function**

The provided code introduces a cube function through both the 'def' keyword and a lambda function. It computes the cube of a specified number, which is set to 5 in this instance, using both approaches and subsequently displays the results. The output, 125, remains consistent for both the 'def' and lambda functions, underscoring their equivalency in performing the cube calculation.

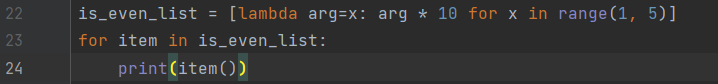
****

**Output:**

****

**Python Lambda Function with List Comprehension**

Within each iteration in the list comprehension, a new lambda function is generated, featuring a default argument of x (where x corresponds to the current item in the iteration). Subsequently, in the for loop, this identical function object with the default argument is invoked using `item()` to obtain the intended value. Consequently, the variable `is\_even\_list` accumulates a list of lambda function objects.

****

**Output:**

****

**Python Lambda Function with if-else**

Here we are using the Max lambda function to find the maximum of two integers.

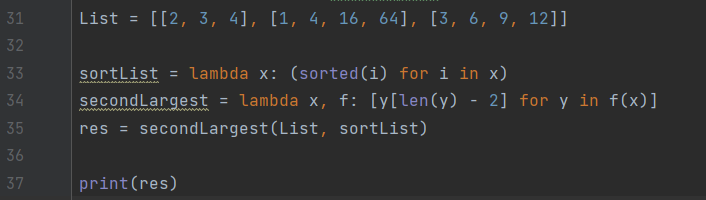
****

**Output:**

****

**Python Lambda with Multiple Statements**

The provided code establishes a list named 'List' comprising sublists. Lambda functions are employed to both sort each sublist and identify their second-largest elements. The outcome is a list containing these second-largest elements, which is subsequently displayed. The printed output showcases the second-largest element extracted from each original sublist within the list.

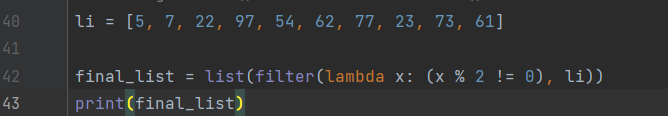
****

**Output:**

****

**Using lambda() Function with filter()**

In this code, the lambda function `lambda x: (x % 2 != 0)` evaluates to either True or False based on whether x is an odd number. As the filter() function selectively retains elements for which the lambda function yields True, it effectively eliminates all odd numbers that result in a False output.

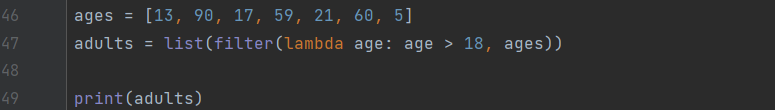
****

**Output:**

****

**Filter all people having age more than 18, using lambda and filter() function**

The code employs a lambda function and the 'filter' function to sift through a list of ages, isolating those belonging to adults (ages exceeding 18). Subsequently, it prints the compiled list of adult ages. The output showcases the ages of individuals who are 18 years or older.

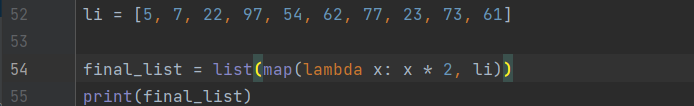
****

**Output:**

****

**Using lambda() Function with map()**

The code utilizes a lambda function and the 'map' function to double each element within a list. Following this operation, it prints the modified list containing the doubled elements. The output presents each element from the original list, now multiplied by 2.

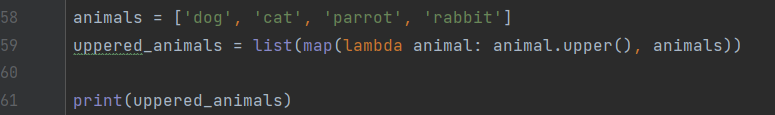
****

**Output:**

****

**Transform all elements of a list to upper case using lambda and map() function**

The code employs a lambda function and the 'map' function to transform a list of animal names into uppercase. Subsequently, it prints the modified list, now featuring animal names in uppercase. The output showcases the animal names presented in uppercase letters.

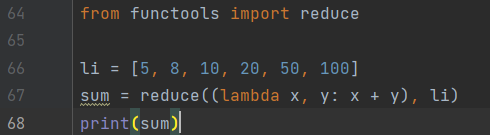
****

**Output:**

****

**Using lambda() Function with reduce()**

The code utilizes the 'reduce' function from the 'functools' module to compute the sum of elements within a list. After importing 'reduce', it defines a list and applies a lambda function that incrementally adds two elements at a time. Finally, it prints the cumulative sum of all elements in the list. The output presents the calculated sum.

****

**Output:**

****