|  |
| --- |
| 1. Write a function that returns the sum of two numbers.   Example  For param1 = 1 and param2 = 2, the output should be add(param1, param2) = 3.  Input/Output   * **[execution time limit] 4 seconds (py3)** * **[input] integer param1**   Guaranteed constraints: -1000 ≤ param1 ≤ 1000.   * **[input] integer param2**   Guaranteed constraints: -1000 ≤ param2 ≤ 1000.   * **[output] integer**   The sum of the two inputs.  **[Python 3] Syntax Tips**  # Prints help message to the console  # Returns a string  def helloWorld(name):  print "This prints to the console when you Run Tests"  return "Hello, " + name |
| 2. Given a year, return the century it is in. The first century spans from the year 1 up to and including the year 100, the second - from the year 101 up to and including the year 200, etc.  Example   * For year = 1905, the output should be centuryFromYear(year) = 20; * For year = 1700, the output should be centuryFromYear(year) = 17.   Input/Output   * **[execution time limit] 4 seconds (py3)** * **[input] integer year**   A positive integer, designating the year.  *Guaranteed constraints:* 1 ≤ year ≤ 2005.   * **[output] integer**   The number of the century the year is in.  **[Python 3] Syntax Tips**  # Prints help message to the console  # Returns a string  def helloWorld(name):  print "This prints to the console when you Run Tests"  return "Hello, " + name |
| * 1. Given the string, check if it is a [palindrome](keyword://palindrome).   Example   * For inputString = "aabaa", the output should be checkPalindrome(inputString) = true; * For inputString = "abac", the output should be checkPalindrome(inputString) = false; * For inputString = "a", the output should be checkPalindrome(inputString) = true.   Input/Output   * **[execution time limit] 4 seconds (py3)** * **[input] string inputString**   A non-empty string consisting of lowercase characters.  *Guaranteed constraints:* 1 ≤ inputString.length ≤ 105.   * **[output] boolean**   true if inputString is a palindrome, false otherwise.  **[Python 3] Syntax Tips**  # Prints help message to the console  # Returns a string  def helloWorld(name):  print "This prints to the console when you Run Tests"  return "Hello, " + name |
| * 1. Given an array of integers, find the pair of adjacent elements that has the largest product and return that product.   Example  For inputArray = [3, 6, -2, -5, 7, 3], the output should be adjacentElementsProduct(inputArray) = 21.  7 and 3 produce the largest product.  Input/Output   * **[execution time limit] 4 seconds (py3)** * **[input] array.integer inputArray**   An array of integers containing at least two elements.  *Guaranteed constraints:* 2 ≤ inputArray.length ≤ 10, -1000 ≤ inputArray[i] ≤ 1000.   * **[output] integer**   The largest product of adjacent elements.  **[Python 3] Syntax Tips**  # Prints help message to the console  # Returns a string  def helloWorld(name):  print "This prints to the console when you Run Tests"  return "Hello, " + name |
| 5 Below we will define an n-interesting polygon. Your task is to find the area of a polygon for a given n.  A 1-interesting polygon is just a square with a side of length 1. An n-interesting polygon is obtained by taking the n - 1-interesting polygon and appending 1-interesting polygons to its rim, side by side. You can see the 1-, 2-, 3- and 4-interesting polygons in the picture below.    Example   * For n = 2, the output should be shapeArea(n) = 5; * For n = 3, the output should be shapeArea(n) = 13.   Input/Output   * **[execution time limit] 3 seconds (java)** * **[input] integer n**   *Guaranteed constraints:* 1 ≤ n < 104.   * **[output] integer**   The area of the n-interesting polygon.  **[Java] Syntax Tips**  // Prints help message to the console  // Returns a string  //  // Globals declared here will cause a compilation error,  // declare variables inside the function instead!  String helloWorld(String name) {  System.out.println("This prints to the console when you Run Tests");  return "Hello, " + name;  } |

**Numpy:**

1. Cree un arreglo que almacene todos los enteros en el rango 10 a 49 luego invierta todos los valores del vector.
2. Genere un arreglo unidimensional con una longitud de 12, rellénelo con ceros. Luego haga que el séptimo elemento sea igual a 1.
3. Genere un arreglo unidimensional y asegúrese de que los valores de las celdas estén en secuencia del 21 al 59
4. Diseñe e implemente una matriz bidimensional de 6\*6 e imprima el tipo de dato al que corresponde la matriz.
5. Diseñe e implemente una matriz, de 64 elementos, en 4 matrices de 4\*4 y utilice una función para generar la transposición de todas las coordenadas.
6. Utilice listas y diccionarios para realizar los ejercicios anteriores.
7. Cree una función email\_list que reciba un diccionario, que contenga nombres de dominio como claves y una lista de usuarios como valores. Desarrolle el código para generar una lista que contenga direcciones de correo electrónico completas (por ejemplo, [diana.prince@gmail.com](mailto:diana.prince@gmail.com)). Obtenga los datos de un json y retorne un xlsx (csv) con una nueva tabla.