# Trabajo Práctico 5: Manejo de excepciones

1. Desarrollar una función para ingresar a través del teclado un número. La función rechazará cualquier ingreso inválido de datos utilizando excepciones y mostrará la razón exacta del error. Devolver el valor ingresado cuando éste sea correcto. Escribir también un programa que permita probar el correcto funcionamiento de la misma.
2. Realizar una función que reciba como parámetros dos cadenas de caracteres conteniendo números reales, sume ambos valores y devuelva el resultado como un número real. Devolver *None* si alguna de las cadenas no contiene un número válido, utilizando manejo de excepciones para detectar el error.
3. Desarrollar una función que devuelva una cadena de caracteres con el nombre del mes cuyo número se recibe como parámetro. Los nombres de los meses deberán obtenerse de una tupla de cadenas de caracteres inicializada dentro de la función. Devolver una cadena vacía si el número de mes es inválido. La detección de meses inválidos deberá realizarse a través de excepciones.
4. El método index permite buscar un elemento dentro de una lista, devolviendo la posición que éste ocupa. Sin embargo, si el elemento no pertenece a la lista se produce una excepción de tipo *ValueError*. Desarrollar un programa que cargue una lista con números enteros ingresados a través del teclado (terminando con -1) y permita que el usuario ingrese el valor de algunos elementos para visualizar la posición que ocupan, utilizando el método index. Si el número no pertenece a la lista se imprimirá un mensaje de error y se solicitará otro para buscar. Abortar el proceso al tercer error detectado. No utilizar el operador in durante la búsqueda.
5. Escribir un programa que juegue con el usuario a adivinar un número. El programa debe generar un número al azar entre 1 y 500 y el usuario debe adivinarlo. Para eso, cada vez que se introduce un valor se muestra un mensaje indicando si el número que tiene que adivinar es mayor o menor que el ingresado. Cuando consiga adivinarlo, se debe imprimir en pantalla la cantidad de intentos que le tomó hallar el número. Si el usuario introduce algo que no sea un número se mostrará un mensaje en pantalla y se lo contará como un intento más.
6. Convertir una cadena en un número: Escribe un programa que solicite al usuario una cadena y luego intente convertirla en un número entero. Si la conversión falla, muestra un mensaje de error.
7. Escribe una función llamada calcularRaizCuadrada que reciba un número como argumento y calcule su raíz cuadrada. Si el número es negativo, la función debe generar una excepción *ValueError* con un mensaje indicando que no se puede calcular la raíz cuadrada de un número negativo.
8. Escribe un programa que pida al usuario que ingrese dos números enteros y muestre el resultado de la división del primer número por el segundo. Si el segundo número es cero, muestra un mensaje de error y vuelve a solicitar el segundo número hasta que sea diferente de cero. Lo mismo si el tipo de dato ingresado no es correcto. Con cada intento de realizar la operación se debe mostrar en pantalla la leyenda: «Se ha intentado realizar una división».