G1962 - Programación

Grado en Ingeniería Civil
Problemas 4

Javier González Villa
(19 de diciembre de 2025)

Excepciones, Validación y Depuración

Ejercicio 1:

Dado el siguiente código mal implementado, se pide depurarlo con la técnica que se considere oportuna (depurador, print, flujo, etc.) para que pueda ser ejecutado sin dar error, independientemente de que el resultado que proporcione sea o no correcto.

```
def mediaNotasEstudiantes(ex1, ex2, ex3):
    '''
    Argumentos: ex1 diccionario con las notas del primer parcial
                ex2 diccionario con las notas del segundo parcial
                ex3 diccionario con las notas del tercer parcial
    Retorna: un diccionario con las notas medias {estudiante: media, ...}
    '''
    medias = []
    for est in ex1.values():
        medias[est] = (ex1[est]+ex2[ex1]+ex3[esl])/3
    return medidas

ex1 = {'Pedro': 4, 'María': 7,'Carlos': 6}
ex2 = {'Pedro': 3, 'María': 4,'Carla': 5}
ex3 = {'Pedro': 9, 'Luis': 8,'Carlos': 8}
mediaNotasEstudiantes(ex1,ex2,ex3)
```

Ejercicio 2:

Dada la siguiente función que calcula la media aritmética y la desviación típica de una lista de valores numéricos, se pide realizar un proceso de validación mediante caja negra acompañado de uno de depuración. Se deberá identificar cuando la función no opera adecuadamente y corregir, tratar o manejar los errores mediante las estructuras que se consideren oportunas (excepciones, condiciones, afirmaciones, etc.), de manera que la función nunca produzca ni propague errores.

```
import math

def analisisEstadistico(L):
    '''
    Argumento: una lista de valores numéricos L
    Retorna: media aritmética y desviación tipica
    '''
    media = sum(L)/len(L)
    sdev = math.sqrt(sum([(x-media)**2) for x in L])/len(L))
    return media, sdev
```

Ejercicio 3:

Se desea elaborar un programa completo capaz de resolver ecuaciones matriciales. Para ello, se pide elaborar de manera teórica la estructura de funciones necesarias a implementar y sus interacciones antes de proceder a la resolución de la ecuación matricial. ¿Qué enfoque de validación y depuración se consideraría el más oportuno para cada una de las funciones planteadas?. ¿En qué casos sería inevitable el uso de excepciones? y en caso de tener dichas excepciones ¿cómo se tratarían de manera que el código completo no propague errores?.

Ejercicio 4:

Dada la siguiente función que calcula el factorial de un número dado de manera iterativa, se pide realizar un proceso de validación y depuración completo. ¿Qué enfoque es mejor para la validación en este caso? y ¿cuál es la mejor manera de garantizar que nuestro código no producirá errores?. Modificar el programa para garantizar que en ningún caso nuestra implementación será vulnerable frente a errores.

```
def factorial(n):
    '''
    Argumento: número n del cual se desea hacer el factorial.
    Retorna: factorial de n, n!
    '''
    i = n
    factorial = 1
    while i > 0:
        factorial = factorial * i
        i = i - 1
    return factorial
```

Ejercicio 5:

Dada la siguiente función recursiva, mal implementada, que permite jugar al juego ¿Piedra, Papel o Tijera? con la máquina hasta que se gane, se pide modificar el código para validarlo por los métodos de pruebas aleatorias y caja blanca. En el de pruebas aleatorias la clave será contar la cantidad de veces que lo hace bien frente a las que lo hace mal mientras que en el enfoque de caja blanca la idea es recorrer todas las posibilidades de flujo de ejecución del programa. Una vez encontrado los errores en la validación, se pide depurar y modificar el código encontrando y reparando los errores.

```
import random

def ppt():
    ju = input('¿Piedra, Papel o Tijera?: ')
    jm = random.choice(['Piedra','Papel','Tijera'])
    if ( ((ju == 'Papel') and (jm == 'Tijera')) or
         ((ju == 'Piedra') and (jm == 'Papel')) or
         ((ju == 'Tijera') and (jm == 'Papel')) ):

        return 'Ganaste!'
    else:
        print('Ganó la maquina: Tú: '+str(ju)+', Máquina: '+str(jm))
        return  ppt()
```

Ejercicio 6:

Realizar las acciones de validación, depuración y tratamiento de excepciones que se consideren oportunas para que la siguiente función, correctamente implementada, produzca el resultado esperado o retorne mensajes de error en cualquiera de los escenarios en los que se pueda utilizar, considerando las condiciones que hacen que sea imposible su utilización.

```
def fuerzaGravitacion(m1, m2, r):
    '''
    Argumentos: m1 masa del primer cuerpo
                m2 masa del segundo cuerpo
                r distancia a la que se encuentran
    Retorna: fuerza de atracción gravitacional entre dos cuerpos
    '''
    G = 6.67e-11
    F = G * (m1 * m2) / (r**2)
    return F

masaTierra = 5.972e24
masaYo = 80
radioTierra = 6371000
acelGravedad = 9.8
F = fuerzaGravitacion(masaTierra,masaYo,radioTierra) # Newtons = m*kg*s^-2
print(F)
print(masaYo * acelGravedad) # F = m * g
```