G1962 - Programación

Grado en Ingeniería Civil

Examen Fundamentos

Instrucciones

- Crear un único fichero mediante JupyterLab con la extensión '. ipynb '.
- Cada ejercicio debe realizarse en una celda por separado dentro del Notebook de Jupyter a excepción de los ejercicios enlazados. Cada ejercicio deberá ser documentado de manera precisa mediante comentarios y documentación.
- Al finalizar, subir el fichero al apartado Examen Fundamentos alojado en Moodle.
- Se debe entregar esta hoja de examen como acreditación de asistencia al examen con las respuestas teóricas.
- Se dispone de un total de 2 horas de examen, tras las cuales la entrega en Moodle se cerrará, por lo que se aconseja entregar unos minutos antes ya que cualquier entrega fuera de ese periodo será descartada .

Nombre y apellidos:

Ejercicio 1 (1p):

1.1) (0.5p) Transforma, mostrando explícitamente el proceso utilizado, el siguiente número binario a decimal donde la parte entera es **1110110** y parte decimal es **1101101**.

1.2) (0.5p) Transforma, mostrando explícitamente el proceso utilizado, el número decimal **67.3** a binario utilizando a lo sumo 16 bits y representando por un lado su parte entera y por otro su parte decimal.

Ejercicio 2 (0.75p):

Indica de los siguientes códigos de Python, cuales son correctos y cuales están mal implementados, indicando en este ultimo caso que errores se cometen en la implementación y una breve idea de como se solucionarían, sin necesidad de corregirlos.

```
def exponenciacion(base, exponente):
    resultado = 1
    for i in range(str(exponente)):
    resultado = resultado * base
    return resultado
```

```
numerador = 100
denominador = '0'
division = numerador/denominador
print(float(division))
```

```
float = 10.0
entero = 20
v = float(input("Número decimal: "))
print(float + entero + v)
```

Ejercicio 3 (1.25p):

Completa el diagrama de flujo presentado en la Figura 1, de manera que represente fielmente el comportamiento del siguiente bloque de código Python:

```
i = 0
L = []
while (i < 10):
    j = 0
    while (j < 5):
        L.append(j * i)
        j = j + 1
    i = i + 1
    print("-----> " + str(i))
```

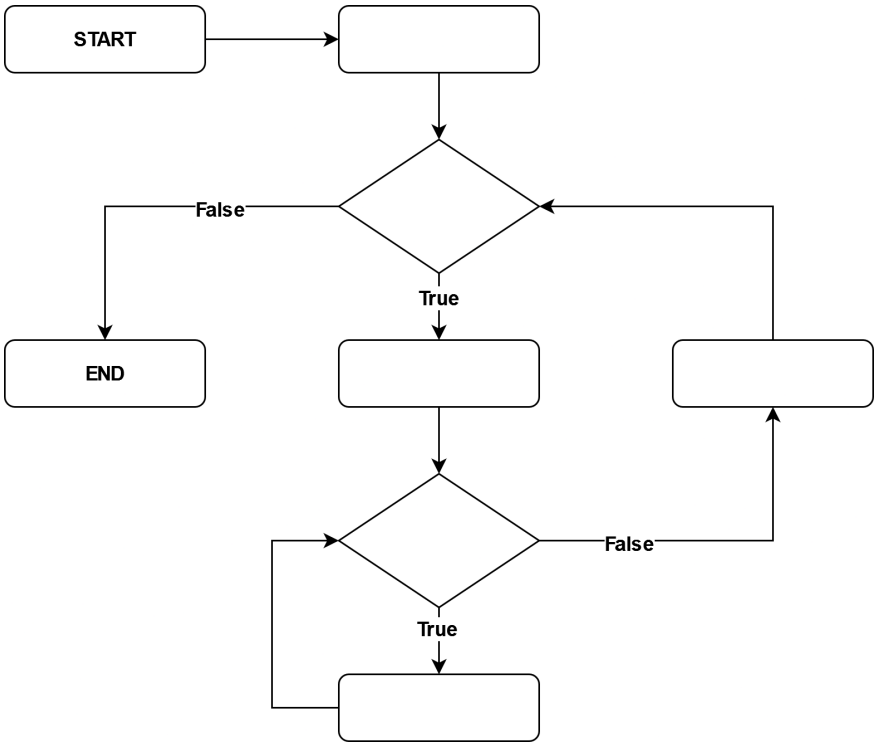

Figura 1: Diagrama de flujo.

Ejercicio 4 (1.5p):

Implementa un código en Python que solicite al usuario que escriba por teclado los coeficientes y el termino independiente de una parábola ($f(x) = ax^2 + bx + c\ ,\ a \neq 0$). Tras esto calcular el vértice de la parábola mediante la formula:

$$V = \left(\frac{-b}{2a}, f\left(\frac{-b}{2a}\right)\right) \tag{1}$$

Posteriormente con esos resultados mostrar por pantalla el siguiente mensaje de manera idéntica sustituyendo los valores entrecomillados por los calculados previamente.

```
El vértice de la parabola se encuentra en el punto "V".
```

Ejercicio 5 (1.5p):

Implementa la función **numerosPell(n)**, con las instrucciones que prefieras, que retorne una lista con los primeros n valores de la serie números de Pell. Como aclaración se tiene que la serie de Pell sigue la siguiente secuencia:

$$0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, ... \tag{2}$$

donde la formula explicita para el cálculo de los diferentes valores es la siguiente:

$$P_n = \frac{(1+\sqrt{2})^n - (1-\sqrt{2})^n}{2\sqrt{2}} \tag{3}$$

Ejemplo de utilización: **numerosPell(8)** retornaría la lista [0, 1, 2, 5, 12, 29, 70, 169].

Ejercicio 6 (1.5p):

Se desea escribir un programa en Python que simule un sistema de autentificación para una aplicación. El programa debe solicitar al usuario que ingrese su nombre de usuario y contraseña mediante teclado. Luego, debe verificar si las credenciales son correctas según las siguientes reglas:

- (0.25p) Si el nombre de usuario es **admin** y la contraseña es **admin123**, pintara por pantalla el mensaje *Inicio de sesión exitoso como administrador*.
- (0.5p) Si el nombre de usuario es **distinto de admin**, pero la contraseña contiene **al menos 8 caracteres y al menos un número**, pintará por pantalla el mensaje *Inicio de sesión exitoso como usuario regular*.
- (0.25p) Si las credenciales no cumplen ninguna de las condiciones anteriores, pintará por pantalla el mensaje *Error: Credenciales incorrectas*.

Por ultimo se desea modificar ese código introduciendo la posibilidad de dar tres intentos al usuario de autentificarse en caso de que introduzca la contraseña de manera incorrecta (0.5p). En dicho caso si tras tres intentos se accede al caso de error de las creadenciales se deberá mostrar el mensaje *Acceso denegado. Ha excedido el número máximo de intentos permitidos. Por favor, espere 5 minutos antes de intentar acceder nuevamente.*

Ejercicio 7 (2.5p):

Se requiere desarrollar un módulo estadístico para la gestión y análisis de datos hidrográficos con el fin de detectar posibles riadas. Los datos a analizar incluyen mediciones de caudal de agua en puntos específicos de varios ríos a lo largo de un periodo determinado. El objetivo es identificar mediciones atípicas que puedan estar relacionadas con eventos de crecida. Dentro de este objetivo principal se pueden disgregar los siguientes objetivos específicos los cuales nos permitirán gestionar y comprender de manera efectiva los datos hidrográficos extraídos, mejorando la capacidad predictiva y preventiva frente a eventos extremos como las riadas.

- (0.5p) Almacenar los siguientes datos, recogidos en la Tabla 1, en un **diccionario** donde las claves corresponderán con los ríos donde se llevaron a cabo las mediciones y los valores son listas de las diversas mediciones realizadas, las cuales figuran en la misma tabla.

Tabla 1: Datos recopilados de mediciones de caudal.

Río	**Medidas de Caudal (m^3/s)**
Nansa	1.10, 1.15, 1.20, 0.95, 0.71, 1.22, 1.08, 1.25, 1.30, 1.12
Besaya	2.15, 2.20, 2.25, 2.10, 2.22, 2.18, 2.30, 2.12, 2.28, 2.23
Pisuerga	0.40, 0.45, 0.42, 0.48, 0.60, 0.38, 0.46, 0.43, 0.28, 0.41

- (0.75p) Implementar la función **calculosEstadisticos(mediciones)** la cual recibe como argumento una **lista de mediciones** y retorna cuatro datos indicando los valores del primer cuartil **Q1 (25 % de los datos por debajo de dicho valor)**, segundo cuartil o mediana **Q2 (50 % de los datos por debajo de dicho valor)**, tercer cuartil **Q3 (75 % de los datos por debajo de dicho valor)** y rango intercuartílico **IQR = Q3 - Q1**. Para más clarificación consultar la Figura 2.

- (0.75p) Implementar la función, **sin hacer uso de ninguna librería externa**, **buscarValoresAtipicos(mediciones)** la cual recibe como argumento una **lista de mediciones** y retorna una **lista con los valores atípicos (outliers)** encontrados (ver Figura 2). Para este cálculo ha que tener en cuenta que un valor atípico u outlier es considerado como tal cuando el valor del dato o medición es superior al límite superior $L_u = Q3 + 1{,}5 \cdot IQR$ o cuando es inferior al límite inferior $L_l = Q1 - 1{,}5 \cdot IQR$.

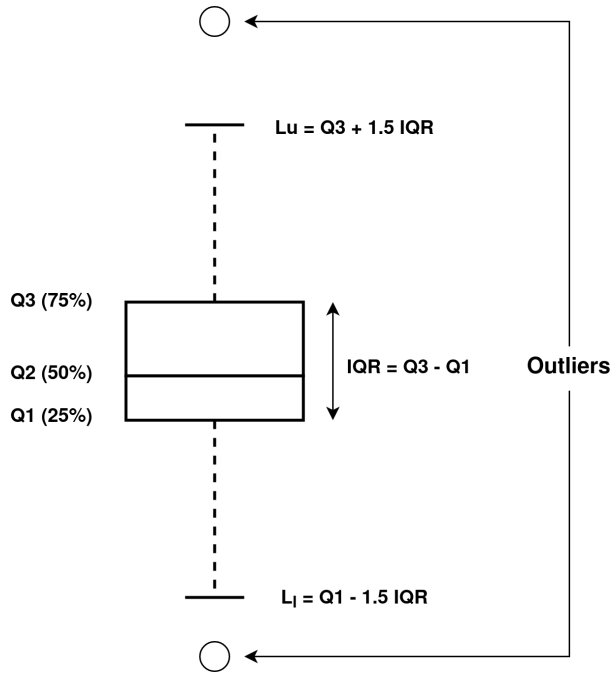

Figura 2: Diagrama de caja enfatizando la detección de valores atípicos (outliers).

- (0.5p) Haciendo uso exclusivamente de las funciones antes implementadas y de los diccionarios antes definidos, identificar si existen o no mediciones atípicas en los datos extraídos, permitiendo anticipar posibles anomalías en el caudal de los mismos.