G1962 - Programación

Grado en Ingeniería Civil
Problemas 1

Javier González Villa
(19 de diciembre de 2025)

Programación y Tipos Básicos

Ejercicio 1:

Codificación: transforma los siguientes números decimales en binarios usando a lo sumo 16 bits para su representación: 25.875, 435 y 36.2.

Decodificación: transforma los siguientes números binarios en decimales:

Cuadro 1: Números en binario.

Parte entera	Parte decimal
10011011	0011001
00101101	1011010
00110100	0000011

Ejercicio 2:

Siguiendo la representación de punto flotante definida en el estándar IEEE 754 calcular el valor numérico en representación decimal de las siguientes secuencias de 32 bits:

Cuadro 2: Números en punto flotante.

Signo	Exponente	Mantisa
0	01111111	10000000000000000000000
1	10000100	01000101000000000000000
0	10000010	01001000000000000000000
1	01111110	00000000000000000000000
0	10001001	00000000001000000000000

Ejercicio 3:

Extrae del siguiente código los operadores matemáticos y lógicos que encuentres e indica en que orden se ejecutarán acorde al sistema de prioridades que implementa Python y la disposición del código. También indica el resultado del programa que será impreso por pantalla y si ese es el esperado por el programador. De cada una de los operadores identificados indica de que tipo especifico son y que operación realizarán.

```
num1 = 5
num2 = 3
num_res = (num1+num2)**2 + num2*5 - num1 / 5
resultado_esperado = 78
print(num_res)
print(resultado_esperado)
print(num_res == resultado_esperado)
```

Ejercicio 4:

Implementar un código sencillo que permita resolver una ecuación de primer grado sencilla: $y = ax + b$. Los datos de a y b se le solicitaran al usuario por teclado y se le retornara por pantalla el resultado de resolver la ecuación y.

Ejercicio 5:

Implementar un código que permita representar el movimiento rectilíneo uniformemente acelerado de un tren. Se conoce que el tren tiene una aceleración $a = 0{,}5m/s^2$ la cual desea ser almacenada en una variable para ser usada posteriormente. También se conoce que el tren parte de reposo por lo que su velocidad inicial es $v_0 = 0m/s$ y su posición inicial $e_0 = 0m$. Se pide implementar un programa que almacene esas variables y solicite al usuario por teclado un instante temporal t en segundos y retorne por pantalla la velocidad y la posición en dicho instante.

$$v(t) = v_0 + a \cdot t \tag{1}$$

$$e(t) = \frac{1}{2} \cdot a \cdot t^2 + v_0 \cdot t + e_0 \tag{2}$$

Cadenas, Listas, Tuplas y Diccionarios

Ejercicio 1:

Implementa un código que solicite por pantalla al usuario su nombre, su primer y su segundo apellido. Concatena esos valores independiente en una sola cadena junto con cadenas adicionales y posteriormente píntala por pantalla mostrando el nombre completo siguiendo el formato: *Apellido1 Apellido2, Nombre*.

Ejercicio 2:

Guarda el texto que se indica a continuación en una cadena para ser utilizado posteriormente:

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lantejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda. El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con

sus pantuflos de lo mesmo, y los días de entresemana se honraba con su vellorí de lo más fino. Tenía en su casa una ama que pasaba de los cuarenta y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza que así ensillaba el rocín como tomaba la podadera. Frisaba la edad de nuestro hidalgo con los cincuenta años. Era de complexión recia, seco de carnes, enjuto de rostro, gran madrugador y amigo de la caza.

Con ese texto guardado, realiza una búsqueda de los índices de la primera aparición de los términos *calzas* y *fiestas*. Con esos índices realiza una extracción y pinta por pantalla la frase que se encuentra entre ambos términos con estos contenidos.

Ejercicio 3:

Definir la siguiente lista $lista = 15, 12, 65, 74, 21, 35, 6, 8, 45, 12, 74, 96, 25, 66, 3$ y ordenarla de mayor a menor. Posteriormente añadir al final de la lista un elemento cuyo valor es 1 menos que el mínimo de la lista y retornar su longitud. Con otra lista idéntica ordénala en este caso de menor a mayor y tras esto concaténala con la primera quedando el valor mínimo en el centro de la nueva lista generada.

Ejercicio 4:

Razona en que casos es útil utilizar tuplas en vez de listas y explica tres ejemplos de aplicaciones donde tenga más sentido su utilización.

Ejercicio 5:

Crea un diccionario (*dnombres*) con los datos de la siguiente tabla, donde las claves serán los IDs de los alumnos y los valores serán listas con el resto de datos personales exceptuando el campo *Nota*. Crea otro diccionario (*dnotas*) donde en este caso las claves sean las mismas y los valores sean numéricos y representen las notas reflejadas en la tabla.

Cuadro 3: Datos de estudiantes.

ID	Nombre	Apellido1	Apellido2	Nota
11	Antonio	García	Pérez	8.4
8	Claudia	Alvarez	Moreno	6.3
20	Manuel	González	Díaz	4
3	Carmen	López	Villa	7.9
5	Francisco	Martín	Sanchez	5.2

Mediante el uso de estos diccionarios se pide extraer la nota y los apellidos del alumnos con $ID = 3$. Posteriormente se desea automatizar más el proceso y por tanto primeramente se le mostrará al usuario por pantalla la lista de *IDs* disponibles y se le pedirá al usuario por teclado un número de *ID* de los de la lista. A partir de este *ID* se extraerán los datos del alumno seleccionado y se retornaran los datos del alumno por pantalla siguiendo el formato: *El alumno* ***Nombre Apellido1 Apellido2*** *tiene una nota media de* ***Nota***.

Ramificación e Iteración

Ejercicio 1:

Los siguientes programas están mal escritos y contienen errores de diversos tipos. Se pide identificar el error, explicar en cada caso de donde proviene y aportar una posible solución.

```
def sumaValores(a, b):
    suma = a + b
return suma
```

```
def iteraCalculos(n):
    lista = []
    for i in range(n):
        lista.append(n/i)
    return lista
```

```
divisionCaja(10, 2)
def divisionCaja(dvdn, dvsr):
    return dvdn // dvsr, dvdn % dvsr
```

```
nombre = "Sira"
especie = "Canis familiaris"
def = "Mamífero de cuatro patas."
raza = "Pastor Alemán"
```

Ejercicio 2:

Escribir el siguiente código mediante la utilización de la instrucción **while** en lugar de **for**, de manera que el resultado retornado sea idéntico.

```
listaNumeros = range(10)
listaLetras = ['A', 'B', 'C', 'D', 'E']
for numero in listaNumeros:
    for letra in listaLetras:
        print(letra)
```

Ejercicio 3:

¿Cuál de los siguientes tres diagramas representa el siguiente fragmento de código?. Explica tu respuesta o porque has descartado el resto de diagramas.

```
if n < 0:
    r = 1
if m < 0:
    s = 2
else:
    t = 3
```

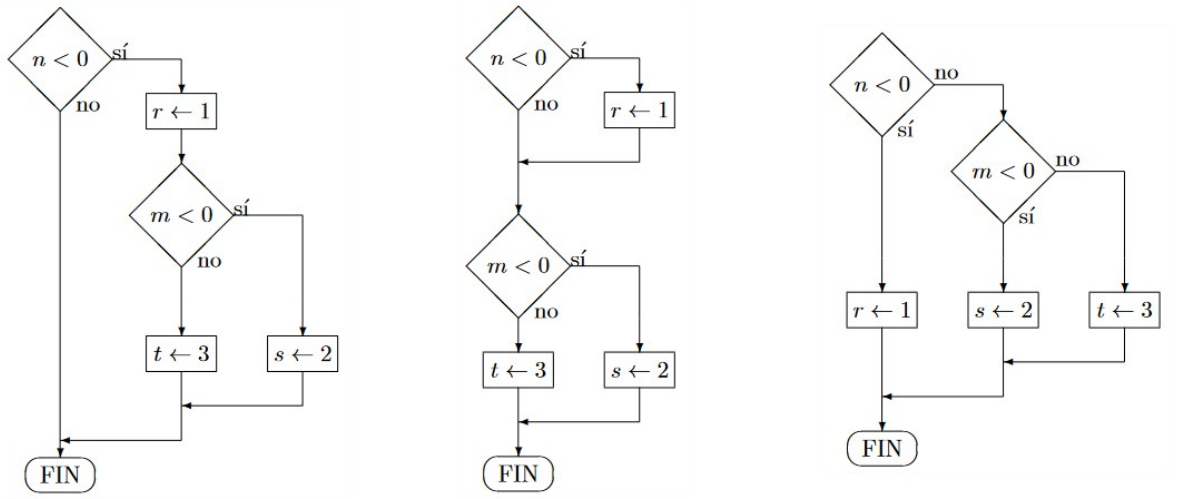

Figura 1: Diagramas de flujo.

Ejercicio 4:

Se desea diseñar un código que permita calcular de manera automática el interés bancario compuesto. Para ello se le solicitara al usuario los datos de capital inicial C, rédito anual r, periodo de capitalización (mensual ($r/12, n \cdot 12$), trimestral ($r/4, n \cdot 4$), semestral ($r/2, n \cdot 2$) o anual) y el periodo de tiempo en años que se desea analizar.

$$C_f = C \cdot \left(1 + \frac{r}{100}\right)^n \tag{3}$$

El programa pintará por pantalla el capital final en cada año, indicando el beneficio obtenido, hasta alcanzar el tiempo de análisis. El programa deberá ejecutarse de manera continua hasta que el usuario introduzca un 0 en el campo de capital inicial.

Ejercicio 5:

Escribe un programa que le pida al usuario por teclado los valores de los coeficientes de una ecuación de segundo grado a, b así como el termino independiente c y calcule las raíces de la ecuación $ax^2 + bx + c = 0$, teniendo en cuenta los casos especiales $a = 0$ (ecuación de primer grado) o raíz única.

Ejercicio 6:

Diseñar un programa que permita transformar un número escrito en numero romanos a partir del siguiente diccionario: *numeros_romanos* = {*'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D':*

500, 'M': 1000}. Para ello es necesario leer del usuario por teclado el número a procesar y posteriormente acceder al diccionario para comprobar su equivalencia decimal. Teniendo en cuenta el orden de las letras en la secuencia recibida realizar las operaciones de suma o resta pertinentes y posteriormente se presentara el resultado del número en decimal por pantalla. Como trabajo adicional se desea comprobar si la secuencia esta bien escrita por parte del usuario y es valida para nuestro programa, es decir si cumple con:

- Los caracteres utilizados están contenidos en las claves de nuestro diccionario.
- Los múltiplos de cinco como por ejemplo V siempre suman.
- Se permite como máximo tres repeticiones consecutivas de un mismo símbolo (III).
- Solo se puede restar múltiplos de tipo 1 (I,X,...) sobre el inmediato mayor de tipo 1 (X,C,...)
- Exceptuando los símbolos que están restando, los demás deben ir en orden de izquierda a derecha ordenados de mayor a menor valor.