

Fundamentos de Programación y Métodos Numéricos

Sistemas de Archivos

2
0
2
6



Introducción

1

Entrada por teclado

2

Salida por pantalla

3

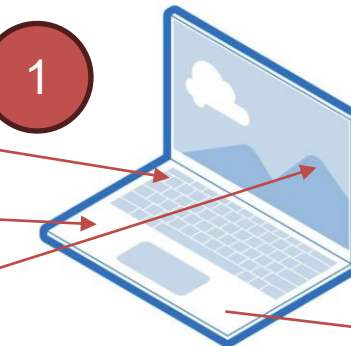
Uso de Memoria Primaria

4

Uso de Memoria Secundaria

Programa en Python

```
1 import math as m
2
3 def cargar_matriz_coeficientes():
4     a = [[0, 0], [0, 0]]
5     for f in range(len(a)):
6         for c in range(len(a[0])):
7             a[f][c] = float(input('Ingrese el coheficiente a' + str(f+1) + str(c+1)))
8     return a
9
10 def cargar_arreglo_constantes():
11     c = [0, 0]
12     for i in range(len(c)):
13         c[i] = float(input('Ingrese la constante c' + str(i+1)))
14     return c
15
16 def error(k1, k1_ant, k2, k2_ant):
17     return m.sqrt(m.pow(k1-k1_ant,2) + m.pow(k2-k2_ant,2)) / m.sqrt(m.pow(k1,2)+m.pow(k2,2))
18
19 a = cargar_matriz_coeficientes()
20 print('Matriz coeficientes: ')
21 print(a)
22 c = cargar_arreglo_constantes()
23 print('Arreglo constantes: ')
24 print(c)
```



Sistema Operativo



Archivo/BD

Necesidades de Almacenar Información

Problema: Contamos con mucha información



.....y en diferentes tipos de almacenamiento:

- Un archivo de texto 
- Varios archivos de texto 
- Base de Datos 



Archivos



- Un **archivo** es un conjunto de datos o información almacenada en un medio digital (como un disco rígido, una memoria USB, la nube, etc.) que se identifica con un **nombre** y generalmente con una **extensión** (por ejemplo: documento.txt, plantilla.xlsx, foto.jpg, programa.py).
- Los archivos sirven para **guardar información permanentemente**.
- Un archivo es la **unidad básica de almacenamiento de información** que maneja un sistema operativo.

Archivos en Python

- Escribir y sobrescribir datos en un archivo:

```
def agregar_un_numero(nombre_archivo, numero):  
    with open(nombre_archivo, "a") as archivo:  
        archivo.write(f"{numero}\n")  
  
def sobrescribir_numeros(nombre_archivo, lista_numeros):  
    with open(nombre_archivo, "w") as archivo:  
        for num in lista_numeros:  
            archivo.write(f"{num}\n")
```

Archivos en Python

- Leer datos desde un archivo:

```
from pathlib import Path

def leer_numeros(nombre_archivo):
    ruta = Path(nombre_archivo)
    if not ruta.exists():
        print(f"El archivo {nombre_archivo} NO existe.")
        return []

    numeros = []
    with open(nombre_archivo, "r") as archivo:
        for linea in archivo:
            try:
                numeros.append(float(linea.strip()))
            except ValueError:
                pass
    return numeros
```

Archivos en Python



- Programa principal:

```
agregar_un_numero("datos.txt", 3.14)
agregar_un_numero("datos.txt", 2.71)

sobrescribir_numeros("datos.txt", [1.0, 2.0, 3.0])
print(Leer_numeros("datos.txt"))
```

Archivos en Python



Problema: Análisis de la resistencia a la tracción de cables de acero

Desarrolla un programa que permita a una empresa de ingeniería analizar qué tan consistente es la calidad de los cables de acero considerando para ello el cálculo de algunas medidas de tendencia central.

La empresa realiza pruebas de resistencia a la tracción en cables de acero utilizados en la construcción de puentes. Cada vez que realiza una prueba de resistencia, se selecciona una muestra de 15 cables y se mide la resistencia a la tracción de cada uno (en kilonewtons, kN).

Archivos en Python



Problema: Análisis de la resistencia a la tracción de cables de acero

Considere que el programa:

- Las mediciones de las resistencias son proporcionadas dentro de un archivo de entrada y se debe almacenar en una estructura adecuada (matriz de N filas por 2 columnas) para su procesamiento.
- El Informe al usuario debe ser almacenado en un archivo de salida con los siguientes resultados:
 - el valor de la media o promedio entre las resistencias medidas,
 - el valor de la moda o valor que más se repite.
 - el valor del rango o diferencia entre el valor máximo y el mínimo.

Archivos en Python



Problema: Análisis de la resistencia a la tracción de cables de acero

EJEMPLO DE UNA PRUEBA DE RESISTENCIA:

Cable	Resistencia (kN)
1	450
2	470
3	465
4	455
5	460
6	475
7	480
8	460
9	455
10	470
11	465
12	460
13	475
14	470
15	460

*Ejemplo de lo que el programa debe
mostrar como datos de salida:*

La media es: 464,67 kN

La moda es: 460 kN

El rango es: 30 kN

Archivos en Python



Objetivo:

El objetivo es desarrollar una herramienta que permita a una empresa de ingeniería evaluar la consistencia en la calidad de cables de acero a partir de ensayos de resistencia a la tracción.

Archivos en Python



Análisis del Problema:

Se necesita procesar datos provenientes de un archivo de entrada que contiene las mediciones de resistencia de múltiples muestras, organizándolos en una estructura matricial de dos columnas.

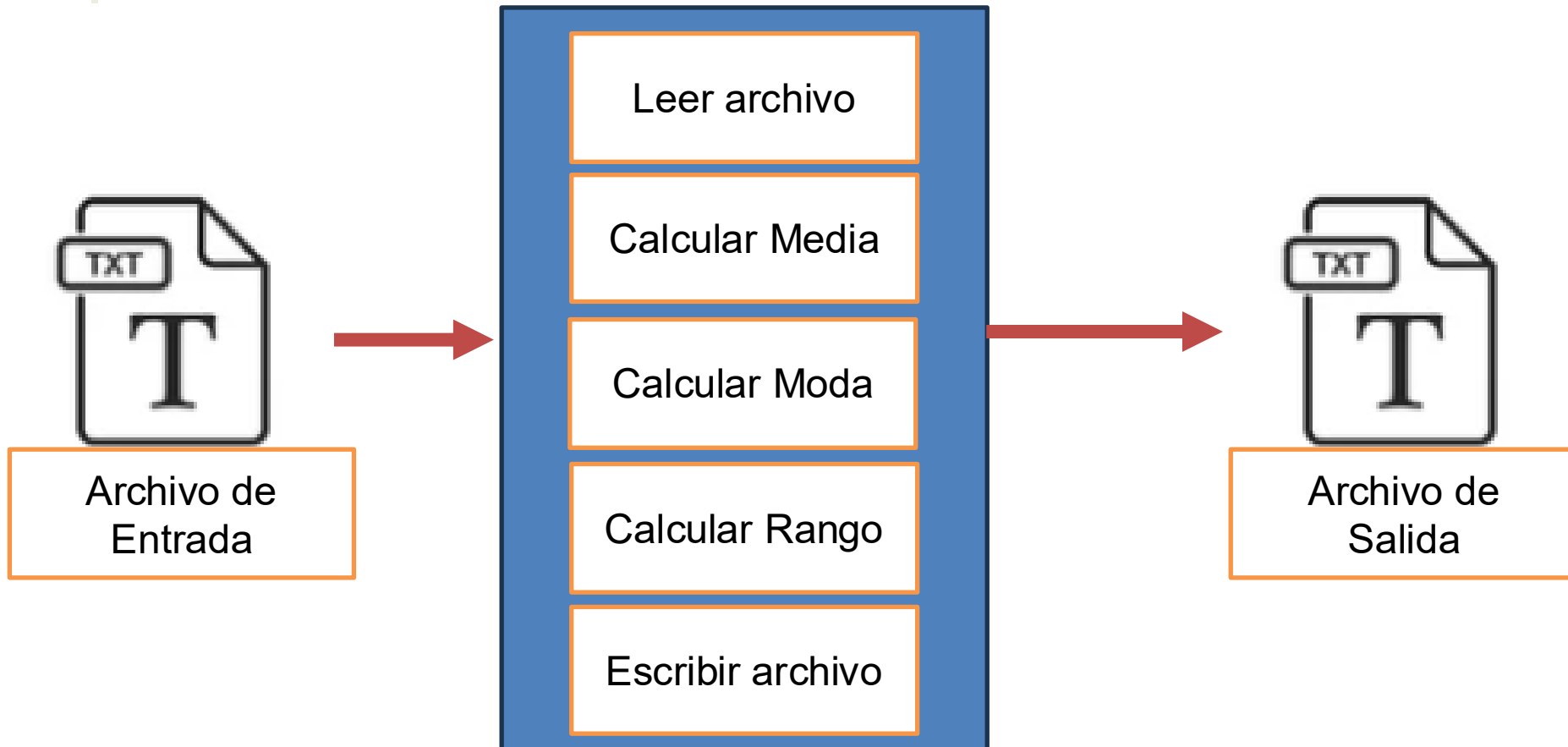
Con estos datos, se necesita calcular las métricas de media, moda y rango.

Finalmente, los resultados deberán ser guardados en un archivo de salida.

Archivos en Python



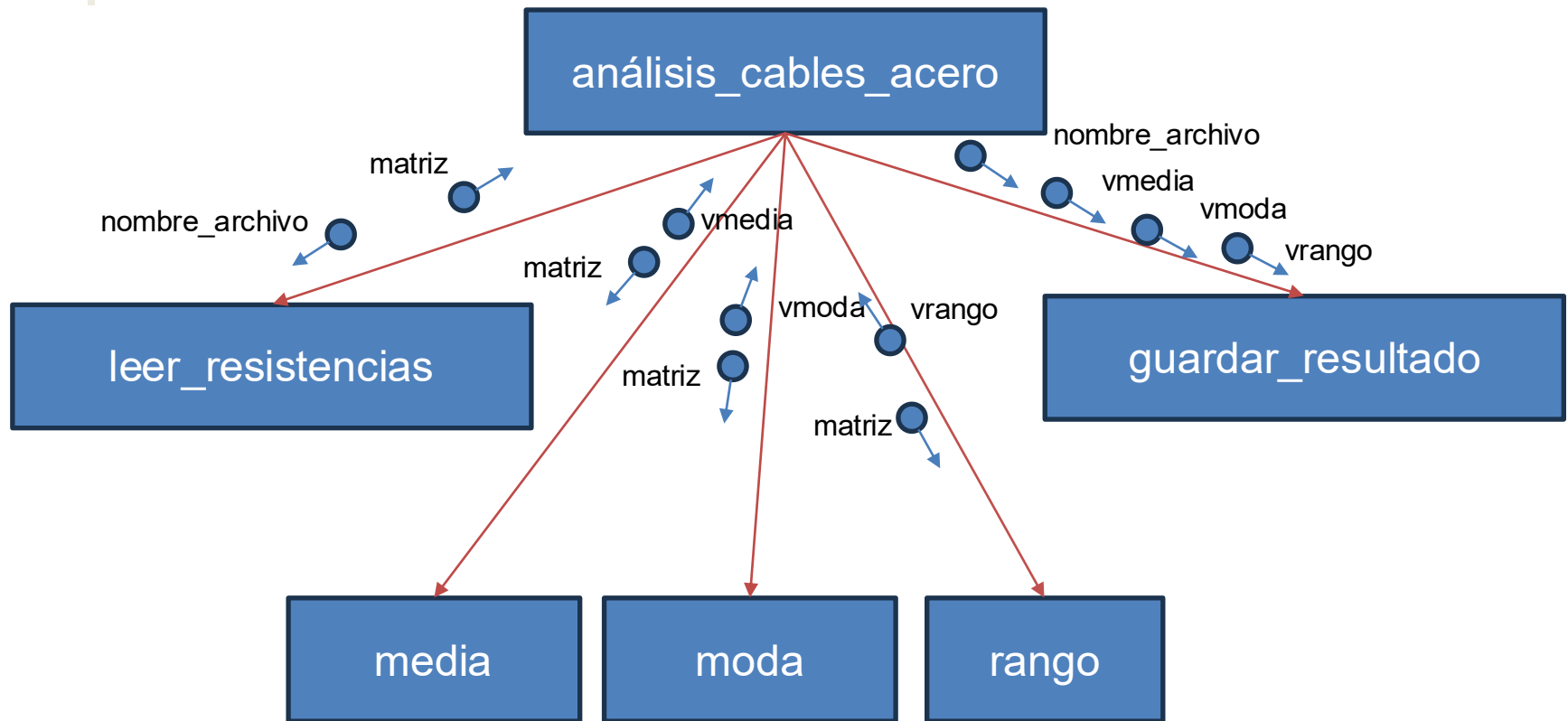
Análisis del Problema:



Archivos en Python



Diseño Problema: Diagrama de Estructura





Archivos en Python

Algoritmo Programa Principal:

Programa AnalisisCables

Definir matriz **Como Real**

Definir vmedia, vmoda, vrango **Como Real**

Dimension matriz[15,2] // tamaño máximo estimado

Definir n **Como Entero**

```
matriz = leer_resistencias(" analisis-entrada.txt")
```

```
vmedia = media(matriz)
```

```
vmoda = moda(matriz)
```

```
vrango = rango(matriz)
```

```
guardar_resultado(" analisis-salida.txt", vmedia, vmoda, vrango)
```

FinPrograma

Archivos en Python



Algoritmo Leer Archivo:

Funcion leer_resistencias(nombre) : **Entero**

Definir archivo, linea **Como Cadena**

Definir i **Como Entero**

Definir valor1 **Como Entero**

Definir valor2 **Como Real**

i = 0

// Saltar encabezado

Leer archivo, linea

Mientras No FinDeArchivo(archivo)

Leer archivo, valor1, valor2

matriz[i,0] = valor1

matriz[i,1] = valor2

i = i + 1

FinMientras

Retornar matriz

FinFuncion

Archivos en Python



Algoritmo Calcular Media:

Funcion media(matriz) : Real

Definir suma **Como** Real

Definir i **Como** Entero

suma = 0

n = 0

Para i = 0 **Hasta** 15

 suma = suma + matriz[i,1]

 n=n+1

FinPara

Retornar suma / n

FinFuncion

Archivos en Python



Algoritmo Calcular Moda:

Funcion moda(matriz) : **Real**

Definir i, j, contador **Como Entero**

Definir max_frecuencia **Como Entero**

Definir valor_moda **Como Real**

max_frecuencia = 0

Para i = 0 **Hasta** 15

 contador = 0

Para j = 0 **Hasta** 15

Si matriz[i,1] = matriz[j,1] **Entonces**

 contador = contador + 1

FinSi

FinPara

Si contador > max_frecuencia **Entonces**

 max_frecuencia = contador

 valor_moda = matriz[i,1]

FinSi

FinPara

Retornar valor_moda

FinFuncion



Archivos en Python

Algoritmo Calcular Rango:

Funcion rango(matriz) : Real

Definir i **Como Entero**

Definir maximo, minimo **Como Real**

maximo = matriz[0,1]

minimo = matriz[0,1]

Para i = 1 **Hasta** 15

Si matriz[i,1] > maximo **Entonces**

maximo = matriz[i,1]

FinSi

Si matriz[i,1] < minimo **Entonces**

minimo = matriz[i,1]

FinSi

FinPara

Retornar maximo - minimo

FinFuncion

Archivos en Python



Algoritmo Escribir Archivo:

SubProceso guardar_resultado(nombre, vmedia, vmoda, vrango)

Definir archivo **Como Cadena**

Abrir archivo nombre Para Escritura Como archivo

Escribir archivo, "Media: ", vmedia

Escribir archivo, "Moda: ", vmoda

Escribir archivo, "Rango (max - min): ", vrango

FinSubProceso

Archivos en Python



Programa Programa Principal:

```
matriz = leer_resistencias(" analisis-entrada.txt")

vmedia = media(matriz)
vmoda = moda(matriz)
vrango = rango(matriz)

guardar_resultado(" analisis-salida.txt", vmedia, vmoda, vrango)
```

Archivos en Python



Programa Leer Archivo:

```
from pathlib import Path

def leer_resistencias(nombre_archivo):
    ruta = Path(nombre_archivo)

    if not ruta.exists():
        print(f"El archivo {nombre_archivo} NO existe.")
        return []

    matriz = []

    with open(nombre_archivo, "r") as archivo:
        encabezado = True
        for linea in archivo:
            if encabezado:
                encabezado = False
            else:
                partes = linea.strip().split()

                try:
                    matriz.append([int(partes[0]), float(partes[1])])
                except ValueError:
                    pass

    return matriz
```

Archivos en Python



Programa Calcular Media:

```
def media(matriz):  
    suma = 0  
    cant = 0  
    for i in range(len(matriz)):  
        suma += matriz[i][1]  
        cant += 1  
    vmedia = suma / cant  
    return vmedia
```

Archivos en Python



Programa Calcular Moda:

```
def moda(matriz):  
    max_frecuencia = 0  
    valor_moda = 0  
  
    for i in range(len(matriz)):  
        contador = 0  
  
        for j in range(len(matriz)):  
            if matriz[i][1] == matriz[j][1]:  
                contador += 1  
  
        if contador > max_frecuencia:  
            max_frecuencia = contador  
            vmoda = matriz[i][1]  
  
    return vmoda
```

Archivos en Python



Programa Calcular Rango:

```
def rango(matriz):  
    max = 0  
    min = 0  
    for i in range(len(matriz)):  
        if matriz[i][1] > max:  
            max = matriz[i][1]  
        if matriz[i][1] < min:  
            min = matriz[i][1]  
    vrango = max - min  
    return vrango
```

Archivos en Python



Programa Escribir Archivo:

```
def guardar_resultado(nombre_archivo, vmedia, vmoda, vrango):  
    with open(nombre_archivo, "w") as archivo:  
        archivo.write(f"Media: {vmedia}\n")  
        archivo.write(f"Moda: {vmoda}\n")  
        archivo.write(f"Rango (max - min): {vrango}\n")
```

Archivos en Python



Resultado:

- **Media:** 464.6666666666667
- **Moda:** 460.0
- **Rango** (max - min): 480.0

Contextualizados:

- **Media:** 464.6 KN
- **Moda:** 460.0 KN
- **Rango** (max - min): 480.0 KN