Introducción: Este portfolio resume los conceptos fundamentales de la Introducción a la Programación, correspondiente al módulo de Programación 1º DAW. Su objetivo es ofrecer un compendio de definiciones, paradigmas, estructuras y herramientas esenciales para el aprendizaje de la programación, facilitando la comprensión y organización de los contenidos teóricos.

Portfolio: Introducción a la programación - 1º DAW

---

1. Conceptos básicos

- **Programación:** Creación de instrucciones que una computadora ejecuta para realizar tareas o resolver problemas.
- **Algoritmo:** Secuencia finita de pasos ordenados que permiten resolver un problema de manera sistemática.
- **Programar:** Escribir instruciones (código) usando un lenguaje de programación para que la computadora ejecute un algoritmo.
- **Código:** Conjunto de instrucciones escritas en un lenguaje de programación.
- **Lenguajes de programación:**
    - Linea cronológica
    - **Lenguaje binario:** Sistema numerico base 2, que usa o y 1, entendido directamente por el hardware.
    - **Lenguaje ensamblador:** Lenguaje de bajo nivel mas legible que el binario.
    - **Lenguaje de alto nivel:** Lenguajes más cercanos al lenguaje humano (Python, java, C++)
- **Clasificación de los lenguajes:**

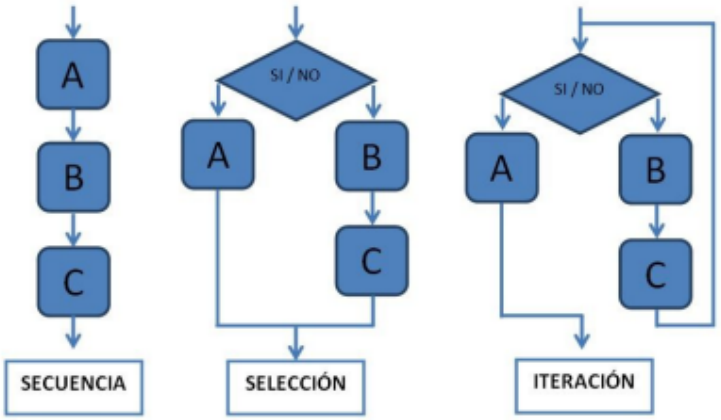
2. Paradigma de programación

- **Imperativa** → Método que permite desarrollar programas a través de procedimientos (instrucciones paso a paso).
    - Ejemplo:

```
int main() {
        printf("tomando una rebanada de pan\n");
        printf("untando mantequilla\n");
        printf("Colocando queso\n");
        printf("Tomando otra rebanada\n");
        printf("Cerrando el sándwich\n");
        return 0;
}
```

    - **Estructurada** o procedural → Organización y la claridad del código a través de un uso sistemático de estructuras de control, como secuencias, selecciones (condicionales) y bucles (iteraciones) (Instrucciones siguiendo estructura).
        - Ejemplo:

- Secuencial
- Alternativa o seleccion
- Interativa o iteracion



    - **Modular** → Organiza un programa en módulos o componentes independientes y reutilizables (Divide el programa en partes).

- **Declarativa** → Se centra en describir qué se desea lograr en lugar de cómo lograrlo (Resultado deseado).
    - Ejemplo: En SQL

```
SELECT nombre, apellido
FROM clientes
WHERE ciudad = "madrid" AND edad > 30;
```

    - **Funcional** → Funciones matemáticas.

    - **Lógico** → reglas lógicas.

- **Orientado a objetos** → Organiza el software en torno a "objetos" (Agrupa Datos y Lógica).

3. Elementos y estructura de un programa

Todo programa incluye:

- **Entrada:** datos proporcionados por el usuario.
- **Procesamiento:** operaciones y cálculos.
- **Salida:** resultados o información generada.

- **Otros elementos:**
- Variables y constantes.
- Operadores aritméticos, lógicos, relacionales, etc.
- Comentarios y palabras reservadas.

- **Estructura típica:**
  Encabezado → Declaraciones → Funciones/Módulos → Bloque principal (inicio y fin).

4. Pseudocódigo y diagramas de flujo

Antes de programar, se recomienda representar la lógica mediante:

- **Pseudocódigo:** descripción en lenguaje natural de los pasos del programa.
- **Diagramas de flujo:** representación gráfica del flujo de ejecución.

Ambos ayudan a entender y documentar los algoritmos.

5. Tipos de datos y operadores

- **Datos:** numéricos, booleanos, cadenas, etc.

- **Operadores:** aritméticos (+, -, *), relacionales (==, !=), lógicos (and, or, not), de asignación (=), pertenencia (in), identidad (is), entre otros.

La precedencia determina el orden en que se evalúan.

6. Estructuras de control

Permiten modificar el flujo de ejecución:

- **Secuenciales:** ejecución lineal.
- **Selectivas:** condicionales simples o múltiples (if, elif, else, switch).
- **Iterativas:** bucles (while, for, do-while).
- **Modulares:** funciones y procedimientos.

7. Funciones, procedimientos y librerías

- **Funciones:** devuelven un valor.
- **Procedimientos:** no devuelven valor.
- Promueven la reutilización y organización del código.
- Se mencionan funciones predefinidas de Python y el uso de módulos como math o datetime.

8. Ejercicios prácticos

incluyen desarrollo de algoritmos, pseudocódigos, diagramas y codigos en Python

- Cálculo del área de un circulo
- Login con usuario y contraseña
- Determinar si un numero es par, primo, o bisiesto.
- Promedios, sumas, factorial, tablas de multiplicar.
- Programas modulares y funciones con parámetros.