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 puede ejecutar para realizar tareas o resolver problemas.
- **Algoritmo:** Secuencia finita de pasos ordenados que permiten resolver un problema de manera sistemática.
- **Programar:** Escribir instrucciones (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:**
  - **Lenguaje binario:** Sistema numérico base 2, que usa 0 y 1, entendido directamente por el hardware.
  - **Lenguaje ensamblador:** Lenguaje de bajo nivel más legible que el binario.
  - **Lenguajes de alto nivel:** Lenguajes más cercanos al lenguaje humano (Python, Java, C++).
- **Entrada / Salida:** Datos que el programa recibe o devuelve.
  - Ejemplo en Python:

```
nombre = input("Introduce tu nombre: ")
print("Hola,", nombre)
```

- **Cambio de estado:** Modificación de los valores de las variables durante la ejecución.
  - Ejemplo:

```
contador = 0
contador += 1  # Cambio de estado de la variable contador
```

- **Lógica:** Reglas que permiten tomar decisiones y controlar el flujo.
- **Datos:** Información que el programa procesa.

2. Evolución y Clasificación de los Lenguajes de Programación

- **Por propósito:** General (cualquier tipo de aplicación) o específico (ej: SQL).
- **Por tipo:** Scripting/Interpretado, Compilado, Marcado.
- **Por uso:** Web, juegos, sistemas, móvil, IA, Big Data, Machine Learning.
- **Por plataforma:** Web (frontend/backend), Desktop, móvil, Embedded.
- **Por paradigma:** Imperativo, Estructurado, Orientado a objetos, Declarativo.

3. Paradigmas de Programación

- **Imperativa:** Describe cómo realizar tareas paso a paso.
  - Procedural/Estructurada: Uso de estructuras de control, evita GOTO.
  - Modular: Divide el programa en subprogramas.
  - Ejemplo:

```
# Programa secuencial
x = 5
y = 3
suma = x + y
print("Suma:", suma)
```

- **Declarativa:** Describe qué resultado se desea.
  - Funcional: Basado en funciones matemáticas.
  - Lógico: Basado en reglas y hechos.
  - Ejemplo en SQL:

```
SELECT nombre, apellido FROM clientes WHERE ciudad='Madrid'
AND edad>30;
```

- **Orientada a objetos:** Agrupa datos y funciones en clases y objetos.
  - Ejemplo en Python:

```
python class Persona:
     def init(self, nombre, edad):
     self.nombre = nombre
     self.edad = edad

persona1 = Persona(“Ana”, 25)

 print(persona1.nombre, persona1.edad)
```

4. Programación Estructurada

```
- Estructuras: Secuencial, Selectiva/Alternativa, Iterativa/Repetitiva,
Modular.
  - Ejemplo secuencial:


print("Inicio")
print("Proceso")
print("Fin")
```

- Ejemplo condicional:

```
edad = 18
if edad >= 18:
    print("Mayor de edad")
else:
    print("Menor de edad")
```

- Ejemplo iterativo:

```
for i in range(1, 6):
    print("Número", i)
```

5. Pseudocódigo y Diagramas de Flujo

- **Pseudocódigo:** Representación de algoritmos sin sintaxis específica.
  - Ejemplo pseudocódigo:

```
Inicio
Leer radio
area = 3.14159 * radio^2
Imprimir area
Fin
```

- **Diagrama de flujo:** Representación gráfica del algoritmo.

6. Estructura de un Programa

- Encabezados/Importaciones, Declaración de variables y constantes.
- Funciones/Métodos.
- Bloque principal.
- Entrada, procesamiento y salida de datos.
- Comentarios para documentación.
  - Ejemplo en Python:

```
# Programa para calcular área de un círculo
pi = 3.14159
radio = float(input("Introduce el radio: "))
area = pi * radio ** 2
print("El área del círculo es:", area)
```

7. Elementos de un Programa

- Variables, Constantes, Literales.
- Operadores: Aritméticos, Comparación, Lógicos, Asignación, Concatenación, Identidad, Pertenencia, Bit a bit.
- Estructuras de control: Condicionales y bucles.
- Expresiones y asignación.
- Dispositivos de entrada/salida.
- Palabras reservadas.

8. Tipos de Datos

- Entero (int), Decimal/Flotante (float), Booleano (bool), Caracter (char), Cadena (str).
    - Ejemplo:

```
entero = 10
decimal = 3.14
booleano = True
caracter = 'A'
cadena = "Hola"
```

9. Estructuras de Control

- Condicionales: `if`, `else`, `elif`.
- Bucles: `while`, `do-while`, `for`.
    - Ejemplo while:

```
contador = 1
while contador <= 5:
  print(contador)
  contador += 1
```

    - Ejemplo for:

```
for i in range(1, 6):
  print(i)
```

10. Modularidad: Procedimientos y Funciones

- Procedimiento: Subprograma que no retorna valor.

```
def saludar():
    print("Hola mundo")
saludar()
```

- Función: Subprograma que retorna valor.

```
def cuadrado(x):
    return x ** 2
resultado = cuadrado(5)
print(resultado)
```

- Funciones predefinidas: `abs()`, `len()`, `round()`, etc.
    - Ejemplo:

```
numero = -7
print(abs(numero))
texto = "Python"
print(len(texto))
```

11. Entornos y Herramientas

- Python: Lenguaje de alto nivel.
- IDLE: Entorno de desarrollo integrado de Python.
- VSCode: Editor de código con soporte Python.
- Git: Control de versiones.
- GitHub: Plataforma para repositorios y colaboración.