Programación 1

# Clase 1

## Paradigmas de la programación:

**Programación estructurada**: programación secuencial con ciclos y condicionales.

**Programación funcional**: Tiene estructura secuencial, pero divide el programa en tareas que son ejecutadas por funciones.

**Programación orientada a objetos**: Se agrupan las funciones en entidades llamadas objetos con características y comportamientos específicos. Estos objetos tienen Atributos y tienen Métodos. Los atributos son las características del objeto. Los métodos son las funciones que pueden cumplir esos objetos. Los objetos en si están agrupados en clases, y cada objeto es una instancia de una clase.

**Programación reactiva**: Se orienta a flujos de datos y propagación de cambios. En este enfoque los programas se diseñan para reaccionar automáticamente a eventos o cambios en los datos de forma asíncrona. Deben ser:

* Responsivos: Aseguran la calidad del servicio cumpliendo tiempos de respuesta preestablecidos
* Resilientes: Mantener responsivos incluso en situaciones de error.
* Elásticos: Mantener responsivos incluso en alta carga de trabajo.
* Orientados a mensajes: Minimizar el acoplamiento

Ventajas: Escalabilidad y rendimiento.

Son usadas por Netflix, Github y Microsoft.

## Lenguajes compilados e interpretados:

La funcionalidad principal de un lenguaje de programación es servir como un medio de comunicación entre el ser humano y la máquina, permitiendo a los programadores escribir instrucciones que una computadora puede interpretar y ejecutar para realizar tareas específicas.

Si no fuese por medio de un lenguaje de programación, las instrucciones deberían escribirse en binario, que se la forma en que trabajan las computadoras.

Los lenguajes de programación se pueden clasificar en alto nivel y bajo nivel. Los de alto nivel son mas cercanos al programador, es decir, son más fáciles de leer y de escribir, y nos “ahorran pasos”.

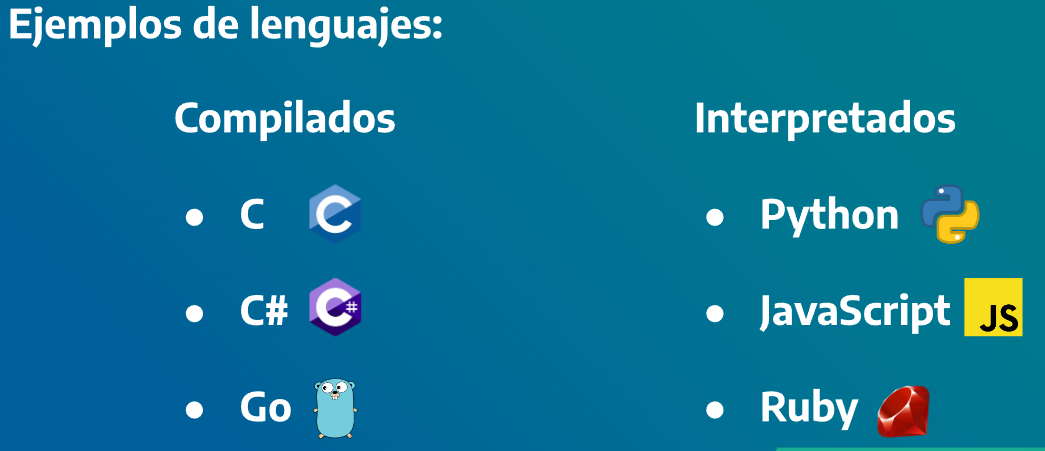
Por otro lado, un lenguaje de bajo nivel es mas cercano al procesador, lo que significa que no se abstraen conceptos, sino que se dan instrucciones detalladas (por ejemplo, permitiendo trabajar con direcciones de memoria).

**Compiladores e intérpretes**

Ya que las computadoras solo entienden ceros y unos, hay dos tipos de programas encargados de transformar nuestro código en binario: los compiladores y los intérpretes.

**Lenguajes compilados**: Primero se compila el código generando un archivo con binarios, luego se ejecuta.

**Lenguajes interpretados**: El código se traduce al mismo tiempo que se ejecuta. El intérprete lee línea por línea el código para ejecutarlo.



## Introducción a Python.

Python es un lenguaje de programación popular. Fue creado por Guido van Rossum en el año 1989 y lanzado en el año 1991.

Con el se pueden hacer los siguientes tipos de software:

* Software multiplataforma
* Videojuegos
* Aplicaciones web.
* Aplicaciones de Big Data.
* Inteligencia artificial

Se suele usar por las siguientes razones:

* Funciona en diferentes plataformas
* Sintaxis simple
* Se ejecuta en un sistema de interpretación
* Es procedural y también orientado a objetos

Sintaxis: Tiene similitudes con el idioma inglés. Usa salto de línea para completar un comando. Se basa en sangría para definir alcance.

# Operadores

Operadores aritméticos:

Los operadores aritméticos se utilizan con valores numéricos para realizar operaciones matemáticas comunes.

* + | Realiza adición entre los operandos
* - | Realiza sustracción entre los operandos
* \* | Realiza multiplicación entre los operandos
* / | Realiza división entre los operandos
* %| Realiza modulo entre los operandos
* // | Realiza una división entera entre los operandos
* \*\* | Realiza la potencia de los operandos

Operadores relacionales:

Un operador relacional se emplea para comparar y establecer la relación entre ellos. Devuelve un valor booleano (true o false) basado en la condición.

* > | Devuelve True si el operador de la izquierda es mayor que el operador de la derecha
* < | Devuelve True si el operador de la derecha es mayor que el operador de la izquierda
* == | Devuelve True si ambos operandos son iguales
* >= | Devuelve True si el operador de la izquierda es mayor o igual al de la derecha.
* <= | Devuelve True si el operador de la derecha es mayor o igual al de la izquierda.
* ¡= | Devuelve True si ambos operandos NO son iguales.

Operadores lógicos:

Se utiliza un operador lógico para tomar una decisión basada en múltiples condiciones.

* And | Devuelve True si ambos operandos son True
* Or | Devuelve True si alguno de los operandos es True
* Not | Devuelve True si el operando es False y viceversa

Operadores de asignación:

Se utiliza un operador de asignación para asignar valores a una variable. Esto generalmente se combina con otros operadores (como aritmética, bit a bit) donde la operación se realiza entre los operandos y el resultado se asigna al operando izquierdo.

* = | El valor puesto a la derecha se asignará a la variable que está a la izquierda
* += | Se sumará el valor de la derecha a la variable de la izquierda.
* -= | Se restará el valor de la derecha a la variable de la izquierda.
* \*= | Se multiplicará el valor de la variable por el numero que se ponga a la derecha.
* / = | Se dividirá el valor de la variable por el numero que se ponga a la derecha
* %= | Se hará modulo del valor de la variable por el numero de la derecha.

## Tipos de datos

Variable: En algunos lenguajes una variable se puede entender como una especie de contenedor en el cual guardar un valor de un tipo en particular, por ejemplo, un valor numérico.

En Python las variables son “etiquetas” que permiten hacer referencia a los datos (que se guardan en unas “cajas” llamadas objetos).

Para cada dato que aparece en un programa, Python crea un objeto que lo contiene. Cada objeto tiene:

* Un identificador único (un número entero, distinto para cada objeto). El identificador permite a Python referirse al objeto sin ambigüedades.
* Un tipo de dato (Entero, decimal, cadena de caracteres, etc.) El tipo de dato permite saber a Python que operaciones pueden hacerse con el dato.
* Un valor (el propio dato)

Todos los valores que aparecen en un programa tienen un tipo.

Cada tipo de información se almacena de forma distinta, por lo que existen diferentes tipos de variables para cada tipo de información.

Algunos lenguajes de programación (C, C++, Java, etc) exigen que antes de utilizar una variable se defina el tipo de información que se va a guardar en esa variable.

Otros lenguajes de programación (Python, PHP, etc.) no lo exigen y es el interprete del lenguaje el que decide el tipo de variable a utilizar en el momento que se guarda la información.

**Definir una variable**

Las variables en Python se crean cuando se definen por primera vez, es decir, cuando se les asigna un valor por primera vez. Para asignar un valor a una variable se utiliza el operador de igualdad (=). A la izquierda de la igualdad se escribe el nombre de la variable y a la derecha el valor que se quiere dar a la variable.

**TIPOS:**

* Entero | int | Representa un número positivo o negativo sin decimales.
* Flotantes | float | Representa un número positivo o negativo con decimales.
* Booleano | bool | Es un tipo de dato que puede guardar True o False.
* Cadena de caracteres | str | Tipo de dato que almacena secuencias de caracteres.

**Casting**: Hacer un cast significa convertir un tipo de dato a otro. En Python es posible convertir a string con str(), a entero con int() y a flotante con float().

**Mutabilidad de variables:**

**Objetos inmutables:** No se pueden modificar despues de ser creados -> int, str, bool

Al crear un objeto inmutable y luego pisarle otro valor, Python ELIMINA el objeto previo y le aplica su nuevo valor.

**Objetos mutables:** Son modificables. Listas, diccionarios y tuplas. Al modificarlo no requiere eliminar el objeto y recrearlo.

## Estructuras de control

Un programa se ejecuta mediante una serie de instrucciones, que por norma general son ejecutadas una tras otra de manera secuencial

En muchas ocasiones no basta con seguir una estructura secuencial, puede ser que ciertas instrucciones se tengan que ejecutar sí y solo si se cumple una determinada condición.

También se puede necesitar repetir un determinado bloque de código más de una vez.

**Condicional if-elif-else:**

Permiten cambiar el flujo de ejecución de un programa, haciendo que ciertos bloques de código se ejecuten si y solo si se dan determinadas condiciones.

**Bucle while:**

El uso del while permite ejecutar una sección de código mientras una condición determinada se cumpla.

Cuando se deje de cumplir se saldrá del bucle y se continuara la ejecución normal.

**Selección múltiple match:**

La sentencia match permite organizar bloques de códigos, de forma que se ejecuten cuando se cumple cierta condición o caso.

Match status:

Case 400:

Return “Error de request”

Y asi…

### Reglas de estilo:

Según guido van rossum el código es leído mas veces que es escrito, por lo que resulta importante escribir código que no solo funcione, sino que además pueda ser leído con facilidad.

PEP es un documento que proporciona información a la comunidad de Python, o que describe una nueva característica.

Es un conjunto de recomendaciones cuyo objetivo es ayudar a escribir código mas legible y abarca desde como nombrar a las variables, al máximo numero de caracteres que debe tener una línea.

Algunas reglas son:

INDENTADO: Python usa bloques de indentacion para indicar el contexto al que pertenece el bloque de código.

TAMAÑO MAXIMO DE LINEA: Se recomienda el máximo de cada línea a 79 caracteres.

ESPACIADO: Nos indica en que casos usas espacios en blanco y en cuáles no.

## Estructura for:

El uso del for permite ejecutar una sección de código repetidas veces, pero a diferencia de while el número de iteraciones está definido de antemano.

Una vez se cumpla el numero de iteraciones, se saldrá del bucle y se continuará la ejecución normal.

El range() genera una secuencia de números que van desde cero por defecto hasta el numero que se pasa como parámetros menos uno.

Tambien se le pueden pasar hasta tres parámetros separados por coma (inicio, fin y salto)

SENTENCIA BREAK:

El break es una palabra reservada que puede ser usada dentro de una estructura repetitiva para detener la ejecución de la misma y continuar con el resto del código.

SENTENCIA CONTINUE:

Es una palabra reservada que puede ser utilizada dentro de una estructura repetitiva para saltear una iteración y luego seguir ejecutando el bucle con las iteraciones restantes.