

Introducción a la Programación con Python 2025

Clase 1:

Introducción a la Programación con Python.

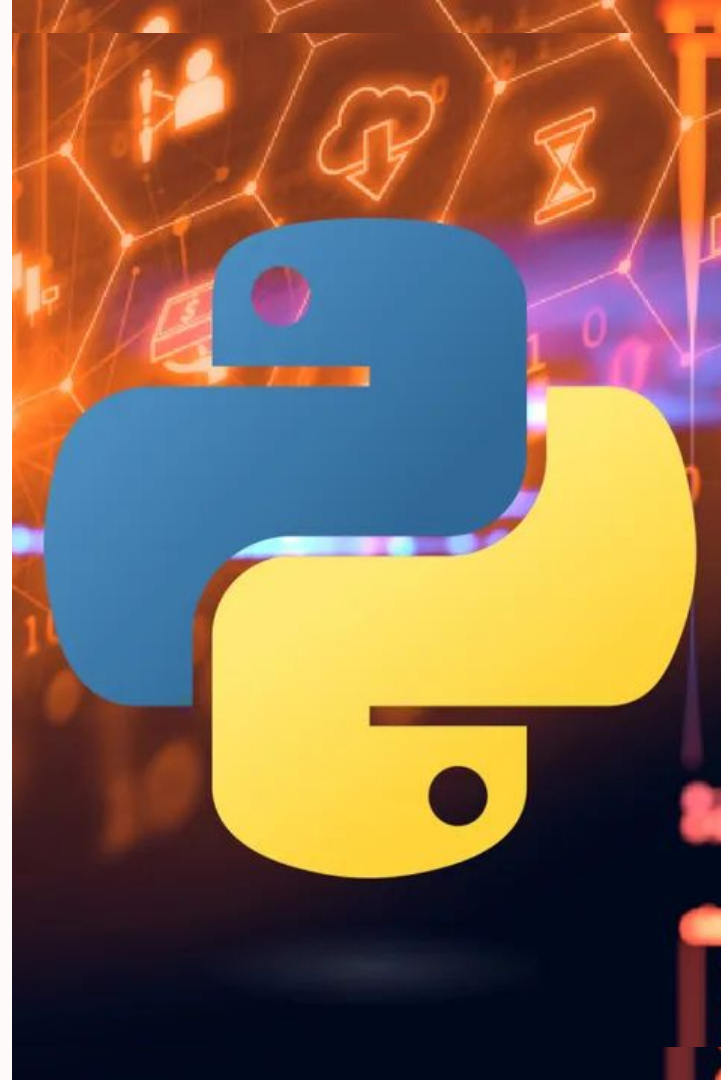
Módulo 1:

Definición de Algoritmos y Resolución de Problemas.

Instructores:

Villar Ailén.

Aguaysol Ernesto.



A conocernos!!!

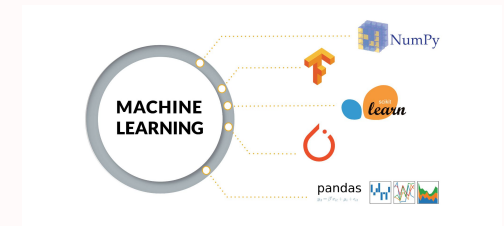
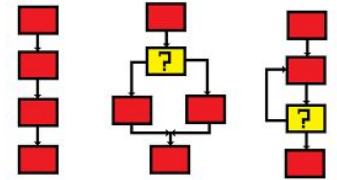
- Nombre Apellido.
- Email.
- ¿Estudias en UNPAZ? ¿Carrera?
- ¿Por qué te anotaste al curso? Ej: por trabajo, por estudio o por otra razón.
- ¿Conoces algo de Python?
- ¿Qué esperás aprender o lograr con este curso?
- ¿Tenés alguna experiencia previa con programación? ¿Con qué lenguajes?
- ¿Tenés compu con internet? ¿Usás computadora regularmente? ¿Para qué actividades?
- ¿Qué sistema operativo usás más seguido (Windows, Linux, Mac, otro)?

¡Bienvenidos al Curso!

- **Filosofía:** Aprender haciendo (Hands-On) desde el primer día.
- **Objetivo general:** Desarrollar habilidades de pensamiento computacional y programación con Python.
- **Modalidad:** Híbrida
 - Clases presenciales con laboratorio: **Martes de 12 a 15 hs.**
 - Clases virtuales: **Jueves de 18 a 19 hs (sincrónica)** 19 a 21 hs (asincrónica).
- **Libros de Referencia:**
 - Python Crash Course (PCC).
 - Fundamentos de Programación: Algoritmos, Estructura de Datos y Objetos (Joyanes Aguilar).
- **Plataforma de comunicación y contenidos:** Por ahora email.

Lo que aprenderemos en este curso

- Variables. Funciones.
- Condicionales.
- Bucles.
- Excepciones.
- Librerías de Python.
- Pruebas unitarias.
- Entrada y salida (E/S) de archivos.
- Expresiones regulares.
- Programación orientada a objetos.
- API Rest con Flask.
- Introducción a Inteligencia Artificial y Machine Learning.

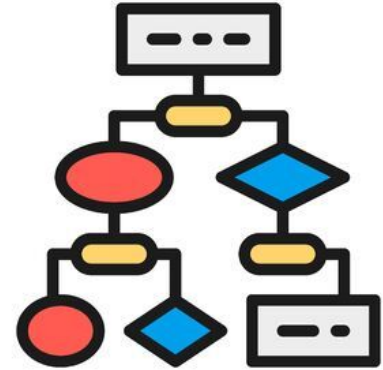


Definición de Algoritmos y Resolución de Problemas



La Base de la Programación

- La programación es el proceso de diseñar, escribir, depurar y mantener el código fuente de programas computacionales.
- Antes de programar, es fundamental **diseñar la solución**.
- Esta solución es lo que conocemos como **algoritmo**.



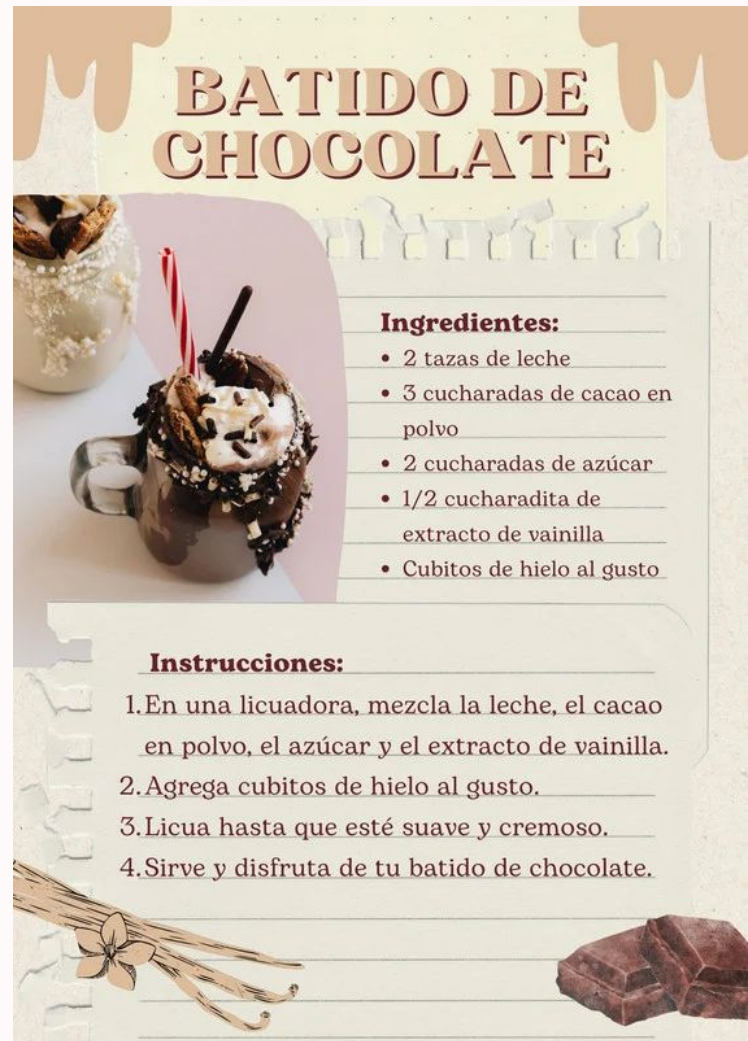
¿Qué es un Algoritmo?

"Un algoritmo es un método para resolver un problema mediante una serie de pasos **precisos, definidos y finitos**."

Características de un algoritmo:

- **Preciso / lógico:** indica el orden de realización en cada paso.
- **Definido:** si se sigue dos veces, obtiene el mismo resultado cada vez.
- **Finito:** tiene fin, un número determinado de pasos.

Analogías: Receta de cocina, manual de instrucciones.



A recipe card for 'BATIDO DE CHOCOLATE' with a torn paper edge. It features a photo of a chocolate milkshake in a mug topped with whipped cream, chocolate shavings, and a red and white striped straw. The ingredients list includes milk, cocoa powder, sugar, vanilla extract, and ice cubes. The instructions are numbered 1 to 4, describing the process from blending to serving. Decorative elements include chocolate drips at the top, vanilla beans and a flower at the bottom left, and chocolate bars at the bottom right.

BATIDO DE CHOCOLATE

Ingredientes:

- 2 tazas de leche
- 3 cucharadas de cacao en polvo
- 2 cucharadas de azúcar
- 1/2 cucharadita de extracto de vainilla
- Cubitos de hielo al gusto

Instrucciones:

1. En una licuadora, mezcla la leche, el cacao en polvo, el azúcar y el extracto de vainilla.
2. Agrega cubitos de hielo al gusto.
3. Licua hasta que esté suave y cremoso.
4. Sirve y disfruta de tu batido de chocolate.

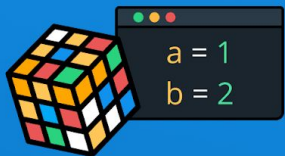
¿QUÉ ES UN ALGORITMO?

Es la **secuencia de pasos** que resuelve un problema y es la base de la programación.

PARTES DE UN ALGORITMO

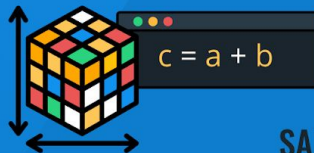
ENTRADA

Son los datos que se le dan al algoritmo.



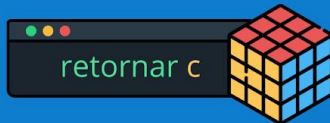
PROCESO

Operaciones que se hacen con los datos.



SALIDA

Resultado final que se obtiene de las operaciones, en este caso será 3.



CARACTERÍSTICAS

PRECISO

Tiene que resolver el problema sin errores.



DEFINIDO

Si ejecutas el algoritmo varias veces, los datos de salida serán iguales en cada repetición.



FINITO

Debe tener un inicio y un final.

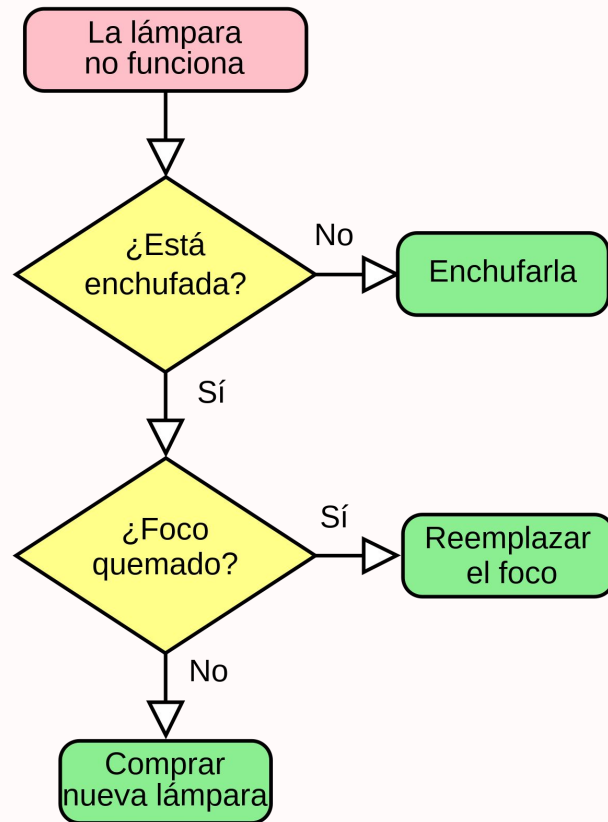


LEGIBLE

Cualquier persona que vea el algoritmo debe ser capaz de comprenderlo.



¿Pasos? Ejemplo



Fases de la Resolución de Problemas Computacionales

1. **Análisis del Problema:** Comprender a fondo qué se necesita resolver.
 - Definir entradas, salidas y requisitos.
2. **Diseño del Algoritmo:** Crear la secuencia de pasos lógicos.
 - Herramientas: Diagramas de flujo, pseudocódigo.
3. **Implementación / Codificación (Programación):** Traducir el algoritmo a un lenguaje de programación (Ej. Python).
4. **Pruebas y Depuración:** Verificar el funcionamiento del programa y corregir errores.
5. **Documentación y Mantenimiento:** Asegurar que el programa sea comprensible y adaptable a futuro.



Introducción a la Lógica Computacional

Elementos Fundamentales

La **lógica computacional** es la base para diseñar algoritmos y programas, nos permite razonar sobre cómo los pasos del algoritmo se ejecutarán.

Estructuras de Control Básicas:

- **Estructura Secuencial:** Las instrucciones se ejecutan en el orden en que aparecen.
Ejemplo: Leer A, Leer B, Calcular Suma, Imprimir Suma.
- **Estructura Condicional (Decisión):** Ejecución de diferentes bloques de código según el cumplimiento de una condición.

Ejemplo: `SI (condición) ENTONCES (acción 1) SINO (acción 2)`

- **Estructura Repetitiva (Iteración/Bucle):** Un bloque de instrucciones se ejecuta repetidamente mientras se cumpla o no una condición.

Ejemplo: `MIENTRAS (condición) REPETIR (acción)`

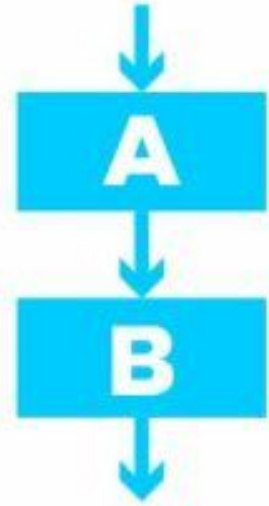
Lógica Computacional:

Elementos Fundamentales

Estructuras de Control Básicas:

- **Estructura Secuencial:** Las instrucciones se ejecutan en el orden en que aparecen. Ejemplo: Leer A, Leer B, Calcular Suma, Imprimir Suma.

Secuencia



Lógica Computacional:

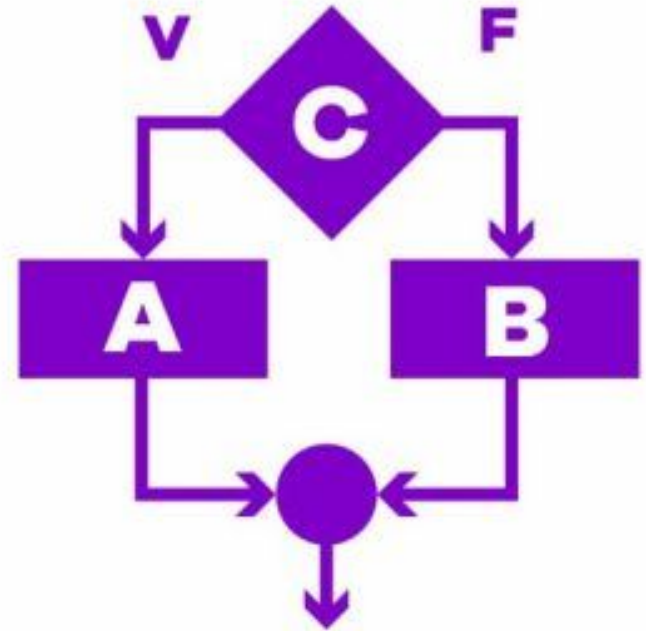
Elementos Fundamentales

Estructuras de Control Básicas:

- **Estructura Condicional (Decisión):** Ejecución de diferentes bloques de código según el cumplimiento de una condición.

Ejemplo: SI (condición) ENTONCES (acción 1)
SINO (acción 2)

Selección o condicional



Lógica Computacional:

Elementos Fundamentales

Estructuras de Control Básicas:

- **Estructura Repetitiva (Iteración/Bucle):** Un bloque de instrucciones se ejecuta repetidamente mientras se cumpla o no una condición.

Ejemplo: `MIENTRAS (condición) REPETIR`
`(acción)`

Iteración (ciclo o bucle)



¿Por Qué Python?



Introducción a Python: ¿Por qué este lenguaje?

- **Sintaxis Clara y Legible:** Se asemeja al inglés, fácil de aprender (ideal para aplicar los conceptos de lógica).
- **Versatilidad:** Web, IA/ML, Ciencia de Datos, Automatización, Desarrollo de Escritorio, etc.
- **Gran Comunidad y Ecosistema:** Abundante documentación, bibliotecas y soporte.
- **Demanda Laboral:** Uno de los lenguajes más solicitados en la industria.

Python en 2025: ¿Qué sigue para el lenguaje de programación favorito del mundo?

Su papel en la inteligencia artificial y el aprendizaje automático (ML) es indiscutible, con bibliotecas como TensorFlow y PyTorch impulsando la innovación. Las empresas utilizan IA basada en Python para modelos predictivos y automatización.

Trends Shaping Python Development In 2025

1. Artificial Intelligence And Python: Driving Innovation Together
2. Machine Learning With Python: Simplifying Complexities
3. Python's Role In Cybersecurity: Enhancing Digital Safety
4. Advancements In Python Automation Tools
5. Data Science And Big Data: Python's Continued Dominance
6. The Future Of Python In Cloud Computing
7. Python For IoT Applications: Connecting The World



Python

The World's Most
Popular Programming
Language

Top programming languages on GitHub

RANKED BY COUNT OF DISTINCT USERS CONTRIBUTING TO PROJECTS OF
EACH LANGUAGE.



Web Developer



Game Developer



Data Analysis



Desktop Developer



Embedded System program



Mobile Apps Development



Preparando Nuestro Entorno: Python y VS Code

Python:

- Es el "intérprete" que ejecuta nuestro código.
- Descarga desde: python.org/downloads (PCC Capítulo 1).
- ¡Importante! Marcar "Add Python to PATH" durante la instalación.



VS Code (Visual Studio Code):

- Editor de código ligero y potente.
- Descarga desde: code.visualstudio.com (PCC Capítulo 1)
- Extensiones recomendadas: "Python" de Microsoft.



> A instalar...

> Pero... Nosotros/as usaremos GitHub Codespaces

Git y GitHub

Git

Es un **sistema de control de versiones**. Permite registrar los cambios que hacemos en archivos (como código) a lo largo del tiempo. Ideal para trabajar en equipo y evitar perder trabajo o sobrescribir cambios.



GitHub

Es una **plataforma online** que permite guardar repositorios de Git en la nube. Facilita la colaboración entre programadores, ya que permite compartir, revisar y fusionar cambios fácilmente. Es como una red social para el código.

<https://github.com/>



Versiones de Python

Download the latest source release

Download Python 3.13.4

Looking for Python with a different OS? Python for [Windows](#),
[Linux/Unix](#), [macOS](#), [other](#)

Want to help test development versions of Python 3.14? [Pre-releases](#),
[Docker images](#)

Todos los ejemplos de este curso deberían funcionar con Python 3.9 o posterior.

Verificar si Python está instalado en tu sistema con el comando

```
python --version
```

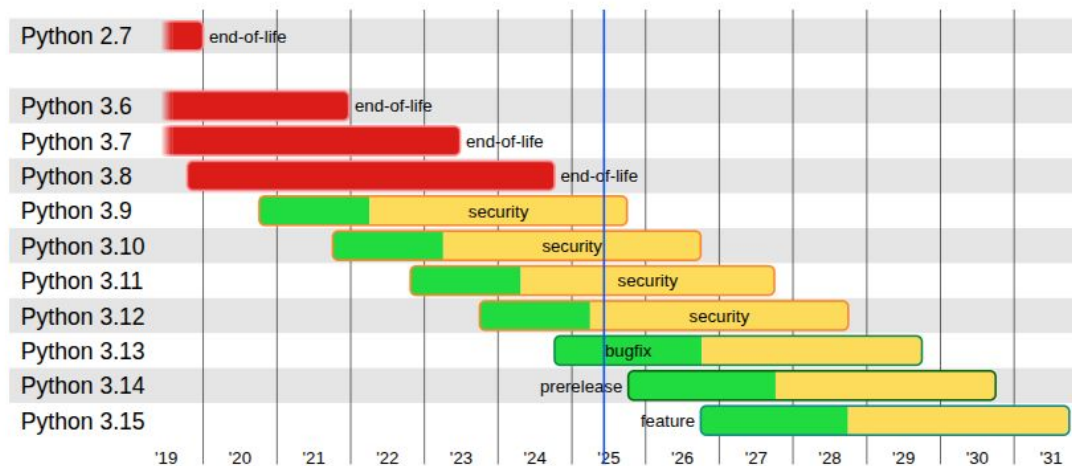
Simplemente `python`

```
>>>
```

NOTA: en linux es `python3`

Estado de las versiones de Python

La `main` rama actual es la futura versión Python 3.14 y es la única que acepta nuevas características. La última versión de cada versión de Python se puede encontrar en la [página de descargas](#).



Comandos Linux

ls	ls – Lista los archivos y carpetas del directorio actual.
cp	cp – Copia archivos o carpetas. Ej: cp archivo.txt copia.txt
mv	mv – Mueve o renombra archivos o carpetas. Ej: mv archivo.txt nueva_carpeta/
rm	rm – Elimina archivos o carpetas. Ej: rm archivo.txt
mkdir	mkdir – Crea una nueva carpeta. Ej: mkdir nueva_carpeta
cd	cd – Cambia de carpeta (navega entre directorios). Ej: cd Documentos
rmdir	rmdir – Elimina una carpeta vacía.
clear	clear – Limpia la pantalla de la terminal.

COMANDO EXTRA:

code – Abre el archivo específico o lo crea en VS Code. Ej: code archivo.py

Nuestro Primer Programa: "¡Hola Mundo!"

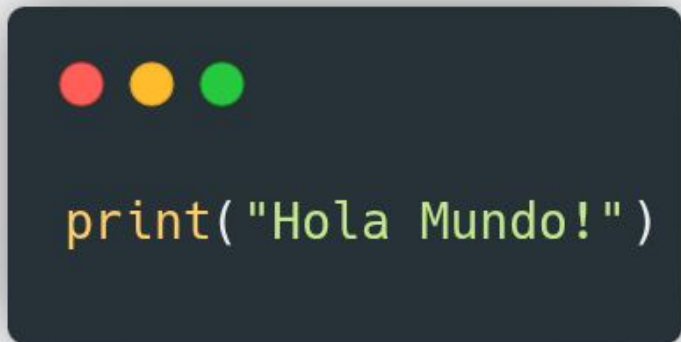
1. Abrir VS Code.
2. Crear una nueva carpeta **curso_python** y abrirla en VS Code.
3. Dentro crear otra carpeta **unidad_01**
4. Crear un nuevo archivo: **hello_world.py**
5. Escribir el código:
5. Abrir la terminal integrada en VS Code (Ctrl+Ñ, View > Terminal, Ctrl+` en linux),
6. Ejecutar el programa:

```
python hello_world.py
```

NOTA: “La función `print()` es una función que le dice a Python que muestre algo en la pantalla (consola)”.

<https://docs.python.org/3/library/functions.html>

1



```
print("Hola Mundo!")
```



```
> Hola Mundo!
```

Variables en Python: Guardando Información

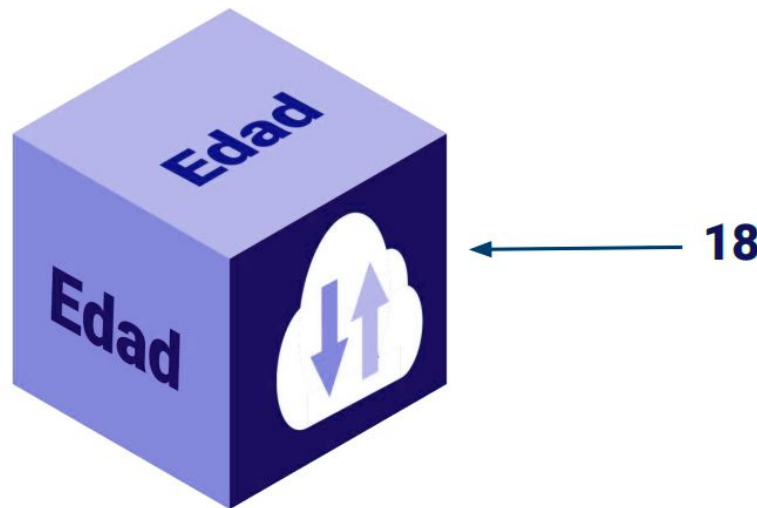
¿Qué son las variables?

Son "contenedores" o "cajas" para almacenar datos en la memoria de la computadora.

Propósito: Nos permiten usar datos de forma dinámica y reutilizable en nuestros programas.

Reglas para nombrar variables (PCC Capítulo 2):

- Pueden contener letras, números y guiones bajos (_).
- No pueden empezar con un número.
- Son sensibles a mayúsculas y minúsculas (**nombre** es diferente de **Nombre**).
- No pueden ser palabras reservadas de Python (ej. `print`, `if`, `for`).



"¡Hola Mundo!" con variable

1. Carpeta: **unidad_01**
2. Archivo: **01_variable.py**
3. Ejecutar:

```
python 01_variable.py
```



```
mensaje = "Hola Mundo!"  
print(mensaje)
```

1. Carpeta: **unidad_01**
2. Archivo: **02_variables.py**
3. Ejecutar:

```
python 02_variables.py
```



```
mensaje = "Hola Mundo!"  
print(mensaje)  
  
mensaje = "Curso de programación en Python"  
print(mensaje)
```

Tipos de Datos Simples: Strings (Cadenas de Texto)

- Representan texto.
- Se encierran entre comillas simples (' ') o dobles (" ").

Carpeta: **unidad_01**

Archivo: **03_strings.py**

```
nombre = "Ana"
saludo = 'Hola, ¿cómo estás?'
print(nombre)
print(saludo)
```

- Esta flexibilidad nos permite usar comillas y apóstrofes dentro de nuestras cadenas.

Carpeta: **unidad_01**

Archivo: **04_apostrofe.py**

```
mensaje = 'Este curso está "GENIAL"'
print(mensaje)
```

Tipos de Datos Simples: Strings (Cadenas de Texto)

Operación básica: Concatenación:

En algunas situaciones, nos interesará usar un valor de variable dentro de una cadena.

Forma 1

Carpeta: **unidad_01**

Archivo: **05_concatenacion_1.py**

```
nombre = "Juan"
apellido = "Pérez"
nombre_completo = nombre + " " + apellido
print(nombre_completo)
```

Forma 2

Carpeta: **unidad_01**

Archivo: **06_concatenacion_2.py**

```
nombre = "Juan"
apellido = "Pérez"
nombre_completo = f"{nombre} {apellido}"
print(nombre_completo)
```

Tipos de Datos Simples: Números

Integers (Enteros): Números sin punto decimal.

Carpeta: **unidad_01**

Archivo: **07_numeros_enteros.py**



```
edad = 25  
cantidad = 100
```

Floats (Flotantes/Decimales): Números con punto decimal. NO usar coma (,).

Carpeta: **unidad_01**

Archivo: **08_numeros_decimales.py**



```
precio = 99.99  
temperatura = 23.5
```

Operaciones Aritméticas Básicas

- Suma: +
- Resta: -
- Multiplicación: *
- División: / (siempre devuelve un float)
- División: // (devuelve un integer)
- Módulo (resto de la división): %
- Potencia: **

Carpeta: **unidad_01**

Archivo: **09_aritmetica.py**

A screenshot of a terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The terminal displays a Python script that defines two variables, 'mun1' and 'mun2', and then prints the results of various arithmetic operations using f-strings. The operations include addition, subtraction, multiplication, division, integer division, modulo, and exponentiation.


```
mun1 = 10
mun2 = 3
print(f"Suma: {mun1 + mun2}")
print(f"Resta: {mun1 - mun2}")
print(f"Multiplicación: {mun1 * mun2}")
print(f"División: {mun1 / mun2}")
print(f"División entera: {mun1 // mun2}")
print(f"Módulo: {mun1 % mun2}")
print(f"Potencia: {mun1 ** mun2}")
```


Operaciones Aritméticas Básicas

- Python respeta el **orden de las operaciones**, así que se pueden usar varias en una expresión. También podemos usar paréntesis para modificar el orden de las operaciones y que Python evalúe las expresiones en el orden que especifiquemos.

Carpeta: **unidad_01**

Archivo: **10_operaciones.py**



```
print(2 + 3 * 4)
print((2 + 3) * 4)
# print(0.2 + 0.1)
```

Ejercicios de Laboratorio

Títulos

Todos los ejercicios van en la carpeta **unidad_01**.

Ejercicio 1: Cambiar mayúsculas y minúsculas en una cadena con métodos (un método es una acción que Python puede realizar con datos). El método `title()` cambia cada palabra a formato de título, con inicial mayúscula en todas las palabras.

Archivo **ejercicio_01.py**




```
nombre = "ada"  
apellido = "lovelace"  
nombre_completo = f"{nombre} {apellido}"  
print(nombre_completo.title())
```

Ejercicios de Laboratorio

Mayúsculas y Minúsculas

Ejercicio 2: Hay otros muchos métodos útiles para tratar con mayúsculas y minúsculas. Por ejemplo, podemos cambiar una cadena a todo mayúsculas `upper()` o todo minúsculas `lower()`. El método `lower()` es especialmente útil para almacenar datos.

Archivo `ejercicio_02.py`



```
nombre = "Ada"
apellido = "Lovelace"
nombre_completo = f"{nombre} {apellido}"
print(nombre_completo)
print(nombre_completo.upper())
print(nombre_completo.lower())
```

Ejercicios de Laboratorio

Espacio en blanco

Ejercicio 3: En programación, un "espacio en blanco" es cualquier carácter que no se imprima, como un espacio, una tabulación o un símbolo de fin de línea. Para añadir una tabulación a un texto se utiliza la combinación de caracteres `\t`, para añadir una nueva línea en una cadena `\n`.

Archivo `ejercicio_03.py`



```
print( '\tPython' )  
print( "Lenguajes:\n\tPython\n\tC\n\tJavaScript" )
```

Ejercicios de Laboratorio

Eliminar espacios en blanco y Eliminar prefijos

- **Ejercicio 4:** Que no haya espacios en blanco a la derecha (al final), usaremos el método `rstrip()`. Archivo **ejercicio_04.py**
- **Ejercicio 5:** Que no haya espacios en blanco a la izquierda (al inicio), usaremos el método `lstrip()`. Archivo **ejercicio_05.py**
- **Ejercicio 6:** Que no haya espacios en blanco de ambos lados, usaremos el método `strip()`. Archivo **ejercicio_06.py**
- **Ejercicio 7:** Con el método `removeprefix()` escribir entre paréntesis el prefijo que deseamos eliminar de la cadena original. Archivo **ejercicio_07.py**

NOTA: A menudo es necesario cambiar el valor de una variable en programación. Es así como se podemos **actualizar el valor de una variable** cuando se ejecuta un programa o en **respuesta** a la entrada del usuario.

Ejercicios de Laboratorio

Asignación múltiple

Ejercicio 8: Podemos asignar valores a más de una variable usando solo una línea. Esto ayuda a acortar los programas y hacerlos más fáciles de leer; usará esta técnica con mayor frecuencia cuando inicialice un conjunto de números. Por ejemplo, así inicializamos las variables **x**, **y** y **z** a cero, de la siguiente forma: `x, y, z = 0, 0, 0`.

Archivo `ejercicio_08.py`

A dark-themed code editor window with three colored window control buttons (red, yellow, green) at the top left. The code is written in a light blue font. The first line is `x, y, z = 0, 0, 0` and the second line is `print(x, y, z)`.

```
x, y, z = 0, 0, 0
print(x, y, z)
```

Ejercicios de Laboratorio

Constantes en Python

Ejercicio 9: Una constante es como una variable cuyo valor permanece invariable a lo largo del programa. Python **no tiene tipos de constantes integrados**, pero los programadores de Python utilizan mayúsculas para indicar que una variable debería tratarse como una constante y no alterarse nunca. Por ejemplo, así inicializamos una constante:

```
MAX_CONNECTIONS = 5000.
```

Archivo **ejercicio_09.py**



Ejercicios de Laboratorio

Comentarios en Python

Ejercicio 10: Un comentario permite escribir notas del usuario dentro de un programa. En Python, la almohadilla (#) indica un comentario. El intérprete de Python ignorará cualquier cosa que vaya detrás de una almohadilla.

Archivo **ejercicio_10.py**




```
# Esto es un comentario  
print("Hola Mundo!")
```

Elegir dos de los programas que ha escrito y escriba al menos un comentario en cada uno de ellos.

Recapitulación de la Clase 1

- **Algoritmos:** Secuencias lógicas/precisas, definidas y finitas para resolver problemas.
- **Lógica Computacional:** Secuencia, Decisión, Repetición.
- **Entorno de Desarrollo:** Python y VS Code. Con GitHub Codespaces
- **Primer Programa:** `print(";Hola Mundo!")`.
- **Variables:** Contenedores de datos.
- **Tipos de Datos Simples:** `strings` (texto), `integers` (enteros), `floats` (decimales).
- **Operaciones Básicas:** Aritméticas.



```
clase_actual = 1
proxima_clase = clase_actual + 1

print("Terminamos la clase", clase_actual)
print("Gracias por tu energía y participación.")
print(f"¡Nos vemos en la clase {proxima_clase}! ")
```

```
"""
```

```
Información de Contacto:
ernestoaguaysol.docente@gmail.com
ailenvillar4058@gmail.com
```

```
"""
```

