



Python aplicado + Flask

{ Python Backend > Clase 3 }

Mentoras: Brisa Sandoval, Matías Villagrán y
Eva Durán



Agenda

LatinasInCloud

1

Tipos de Datos y Estructuras Python

2

Resumen Git + comandos

3

¿Qué es un framework?

4

Flask





En la escala de Los Vengadores ¿Cómo te sientes hoy?



1



2



3



4



5



6



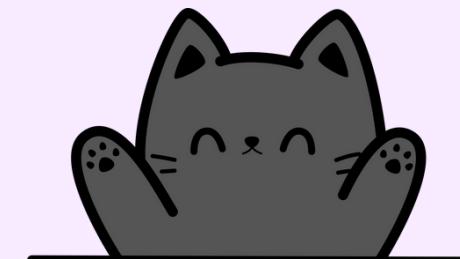
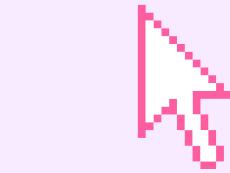
7



8



9



LatinasInCloud

Quizz 2

Wayground

Tipos de Datos

En Python **trabajamos con distintos tipos de datos**, y la elección depende de la **situación o del tipo de información** que queremos manejar. Cada una tiene sus propias características y ventajas:

- **Enteros** (int) → cuando necesitamos trabajar con números enteros (ej. contar objetos)
- **Flotantes** (float) → cuando se requiere precisión con decimales (ej. cálculos científicos)

```
1 # 1. Tipos de Datos en Python
2 # -----
3
4 # Enteros (int): números enteros, con positivos o negativos
5 y = 5
6 y = -3
7
8 # Flotantes (float): números con decimales
9 pi = 3.14
10 neg = -2.7
11
12 # Cadenas de texto (str): texto encerrado entre comillas
13 saludo = "Hola"
14 lenguaje = ' Python'
15
16 # Booleanos (bool): solo True o False
17 es_mayor = True
18 es_menor = False
19
20 # Listas (list): conjunto ordenado y modificable
21 numeros = [1, 2, 3, 4]
22
23 # Tuplas (tuple): conjunto ordenado pero inmutable
24 coordenadas = (10, 20)
25
26 # Conjuntos (set): no tienen orden y no permiten duplicados
27 # No se pueden modificar Los creados
28 unicos = {1, 2, 3, 3} # {1, 2, 3}
29
30 # Diccionarios (dict): pares clave: valor
31 persona = {"nombre": "Ana", "edad": 25}
```

Los str pueden estar entre comillas dobles "" o simples ''

Ordenado pq tiene un índice empiezan de 0,1,2,etc (es como la posición del valor en la lista)



LatinasInCloud

Estructuras de Control y Funciones

Los **condicionales** nos permiten **tomar decisiones** en el código. Sirven para **ejecutar distintas acciones** dependiendo de si una condición es True o False.

Los **bucles** se utilizan para **repetir un bloque de código varias veces** sin tener que escribirlo manualmente.

Las **funciones** permiten **agrupar instrucciones en un bloque con un nombre, para poder reutilizarlo** en diferentes partes del programa.

```
● ● ●  
1 # 2. Condicionales  
2 # -----  
3  
4 x = 0  
5  
6 if x > 0:  
7     print("Número positivo")  
8 elif x == 0:  
9     print("Es cero")  
10 else:  
11     print("Número negativo")  
12  
13 # 3. Bucles  
14 # -----  
15  
16 # Bucle for: recorre una secuencia  
17 for i in range(5):  
18     print("For:", i)  
19  
20 # Bucle while: se ejecuta mientras la condición sea True  
21 x = 0  
22 while x < 5:  
23     print("While:", x)  
24     x += 1  
25  
26 # 4. Funciones  
27 # -----  
28  
29 # Definición de una función  
30 def saludar(nombre):  
31     return f"Hello, {nombre}"  
32  
33 # Uso de La función  
34 print(saludar("Ana"))
```



Git

Permite llevar un **historial de cambios en los archivos** de un proyecto. Con él puedes:

- Volver a **versiones anteriores**
- Colaborar con otras personas sin sobreescribir el trabajo
- Probar nuevas ideas **sin dañar el código estable**

Una rama es como una **línea paralela de desarrollo** dentro del proyecto. Sirven para:

- Desarrollar **nuevas funciones** sin afectar el código principal
- **Corregir errores** en una copia aislada
- **Experimentar sin romper** la versión estable

```
1 # Configuración Inicial
2 # -----
3
4 # Se usa una sola vez en un equipo nuevo (o para cambiar de usuario)
5 # Configura tu nombre y correo para Los commits
6 git config --global user.name "tu nombre"
7 git config --global user.email "tuemail@example.com"
8
9 # Comandos Principales
10 # -----
11
12 # Inicializa un nuevo repositorio en La carpeta actual
13 git init
14
15 # Clona un repositorio remoto en tu máquina Local
16 git clone url_repositorio      en git hub sale en la parte de <code> el url
17
18 # Muestra el estado actual del repositorio
19 git status
20
21 # Prepara archivos para el commit (se añaden al staging area)
22 git add nombre_archivo
23 git add . # agrega todos Los cambios
24
25 # Guarda Los cambios en el historial
26 git commit -m "mensaje"
27
28 # Muestra el historial de commits
29 git log
30
31 # Descarga y fusiona Los cambios desde el remoto al Local
32 git pull      trae los cambios de la nube al local
33
34 # Descarga cambios del remoto, pero NO Los fusiona todavía
35 git fetch
```

```
1 # Ramas (branches)
2 # -----
3
4 # Lista Las ramas existentes
5 git branch
6
7 # Crea una nueva rama
8 git branch nombre_rama
9
10 # Cambia a La rama indicada
11 git checkout nombre_rama
12
13 # Crea y cambia a La nueva rama directamente
14 git checkout -b nombre_rama
15
16 # Fusiona La rama indicada con La rama actual
17 git merge nombre_rama      fusiona los cambios a la rama
18
19 # Elimina una rama Local
20 git branch -d nombre_rama
21
22 # Sube La rama al remoto
23 git push origin nombre_rama
```

d: delete



LatinasInCloud

Workflow Git

Local

working
directory

staging
area

local repo

Remote

La Nube que queda en Github

remote
repo

git add

git commit

git push

git fetch

git checkout

git merge



LatinasInCloud

¿Qué es un framework?



Un framework es un **conjunto de herramientas, librerías y estructuras** que facilitan el desarrollo de aplicaciones.

- Te da una **base predefinida** para no empezar desde cero
- Permite trabajar **más rápido, organizado y seguro**
- En programación web, los frameworks ayudan a **manejar rutas, peticiones, respuestas, plantillas HTML, bases de datos, etc**



LatinasInCloud

FRAMEWORK LIBRERÍA

Conjunto de herramientas que trabajan en un proyecto completo bajo ciertas reglas.



Herramienta con una sola utilidad específica.



Tiene funcionalidades integradas para que no necesites librerías externas.



Eres libre de usar las librerías que deseas en la estructura que quieras.



La compatibilidad de sus funcionalidades está asegurada.



Debes controlar la compatibilidad de cada librería con las demás.



El framework define la forma en que debes desarrollar el proyecto.



Puedes usar varias librerías según tus necesidades.



Laravel



EDgrid

Usualmente los framework ya vienen con algunas librerías instaladas para trabajar

Aprende a programar con las librerías y frameworks más populares en:

👉 ed.team/cursos

EDteam



LatinasInCloud



15 minutos



LatinasInCloud

¿Qué es Flask?



Flask es un **framework minimalista** de Python para **desarrollar aplicaciones web**. Es muy ligero y flexible: te da lo esencial para crear una aplicación, pero **puedes agregar extensiones según lo necesites** (bases de datos, autenticación, etc). Es **ideal para proyectos pequeños**, APIs y prototipos rápidos.

[extensiones \(dependencias, librerías, etc\)](#)

- Puedes crear una aplicación web desde cero
- Definir rutas (/, /login, /api) y asignarles funciones
- Manejar **peticiones HTTP** (GET, POST, PUT, DELETE)
- Generar **respuestas dinámicas** (texto, JSON, HTML)
- **Renderizar plantillas HTML** con variables dinámicas
- Conectar a bases de datos mediante extensiones
- Construir APIs REST de manera sencilla



HTTP= protocolo de transferencia de hipertexto

Métodos HTTP Principales

conjunto de reglas para la comunicación entre navegadores web (clientes) y servidores web



Método	Uso Principal	Ejemplo
GET	Solicitar información del servidor (no modifica datos)	Obtener la página principal /home o una lista de usuarios /users
POST	Enviar datos al servidor para crear o procesar algo nuevo	Registrar un usuario con /register
PUT	Reemplazar un recurso existente con nuevos datos	Actualizar todos los datos de un usuario en /users/1
PATCH	Modificar parcialmente un recurso	Cambiar solo el email del usuario en /users/1
DELETE	Eliminar un recurso del servidor	Borrar un usuario en /users/1

Ejercicios en Casa Flask



Ejercicio 1

Crea una aplicación con las siguientes rutas:

- / → muestra una “Página principal”
- /about → muestra “Acerca de esta app”
- /contact → muestra “Página de contacto”

Ejercicio 2

Respuesta en HTML, en la ruta /html, devuelve una respuesta con HTML muy sencillo:

- <h1>Bienvenidos</h1>
- <p>Esta es mi primera página con HTML en Flask</p>

Ejercicio 3

Parámetros en la URL, crea una ruta /saludo/<nombre> que muestre un saludo personalizado en HTML

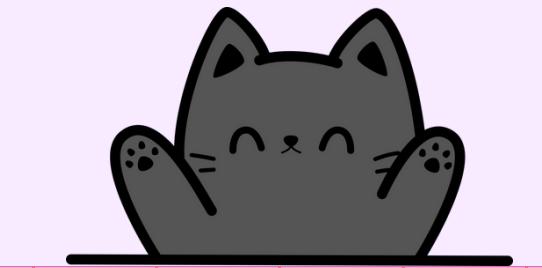
- /saludo/Brisa → “Hola Brisa”
- /saludo/Ana → “Hola Ana”

Ejercicio 4

Página con enlaces, en la ruta /links, devuelve HTML con enlaces a otras rutas de tu app:

- <h2>Menú</h2>
- Inicio

- Contacto.





Requisitos Próxima Clase

1

Revisar capsula Flask

2

Buscar información sobre HTML y CSS

3

Revisar capsula Google Colab





Mis entregables: https://colab.research.google.com/drive/1CX1hCdEXih0J8VDDZWbXizgl1Yj_sCGA
<https://colab.research.google.com/drive/1uSDd49pfXeE7zFzCjqwO1ITZO4DgHpva>

Primer Entregable

En equipos de 3 a 5 mujeres, tendrán que resolver una serie de ejercicios para profundizar en python. La entrega puede ser a través de GitHub o Google Colab (recomendamos este porque empezaremos a ocuparlo las siguientes clases). En ella deben aparecer los nombres de las integrantes y la resolución de los ejercicios. La fecha máxima de entrega es hasta el martes 23/09 Deben enviar los trabajos al correo de latinas in cloud con los respectivos enlaces.

