Preguntas teóricas

- **¿Cuáles son los tipos de Datos en Python?**

  Python tiene una gran variedad de tipos básicos, entre los que podemos destacar los siguientes:

  - **Cadena o string** = ' El perro de san Roque'
  - **Cadena de caracteres** = 'gato', 'tortuga', 'hipopótamo'
  - **Número o number**: Entero (int) { -10, 6, 100}, punto flotante (float) {7.8, 3.6, 2.9} y número complejos {3j+8, 6j-90}
  - **Listas o list** = ['coche', 'casa', 35, '54']
  - **Conjuntos o set** = {'hola', 'adios', 9, 5}
  - **Tulpa o tuple** = (9, 0, 'piso', 'elefante')
  - **Booleano o boolean** = True o False

- **¿Qué tipo de convención de nomenclatura deberíamos utilizar para las variables en Python?**

  Para referirse a una variable se usa la nomenclatura llamada "snake_case", en la que escribimos el texto en minúsculas y separamos cada palabra con un guión bajo.

  También existen otras nomenclaturas. Las más importantes en Python son:

  - snake_case para variables, funciones y métodos.
  - PascalCase para clases.
  - SCREAMING_SNAKE_CASE para constantes.

- **¿Qué es un Heredoc en Python?**

El heredoc es una sintaxis especial que nos permite definir bloques de texto multilínea dentro de nuestro código de manera fácil y legible. Es decir, es una forma de representar un bloque de texto en código y nos permite definir cadenas multilínea sin necesidad de caracteres de escape.

Usamos la sintaxis heredoc, que consiste en tres comillas o tres comillas dobles antes y después de un texto de varios párrafos, para facilitar que el programa entienda que todo el texto que aparece entre esas comillas pertenece al mismo string.

Si usáramos comillas normales con el mismo texto, en el momento que exista un salto de línea, el programa espera un cierre de comillas y daría error.

- **¿Qué es una interpolación de cadenas?**

La interpolación de cadenas se refiere al proceso de insertar valores o variables en una cadena de texto, para crear cadenas dinámicas que cambian según los datos que se introduzcan.

```
animal = "caballo"
nombre = "Manchas"
Frase_interpolada = f "Mi {animal} se llama {nombre}"
Print(Frase_interpolada)    => Mi caballo se llama Manchas

animal = "pez"
nombre = "Glu-glu"
Frase_interpolada = f "Mi {animal} se llama {nombre}"
Print(Frase_interpolada)    => Mi pez se llama Glu-glu

animal = "dinosaurio"
nombre = "Blue"
Frase_interpolada =  "Mi {0} se llama {1}" .format(animal,nombre)
Print(Frase_interpolada)    => Mi dinosaurio se llama Blue
```

- **¿Cuándo deberíamos usar comentarios en Python?**

Los comentarios sirven para dar explicaciones sobre el código que hemos realizado. Como este código puede variar con el tiempo, hasta el punto que sea completamente distinto al original, se recomienda que si vamos a poner un cometario, este sea de carácter general o para dar indicaciones de cómo está organizado.

Se usan principalmente como documentación (explica la finalidad de las funciones, clases o bloques de código.), depuración (aislar e identificar errores) y colaboración (facilitar trabajo en equipo y revisión del código).

Ejemplo:

- Aquí aparecen los estilos relativos a la página principal de la web
- Aquí aparecen los estilos relativos al blog de la web
- Aquí aparecen los estilos relativos a la página de productos de la web
- También pueden servir como recordatorio de cosas que faltan de hacer (TODO) mientras se trabaja en un código.
- Comentamos un trozo de código en el que aún estamos trabajando, que da error o no nos sirve hasta que revisemos, depuremos o encontremos el error.

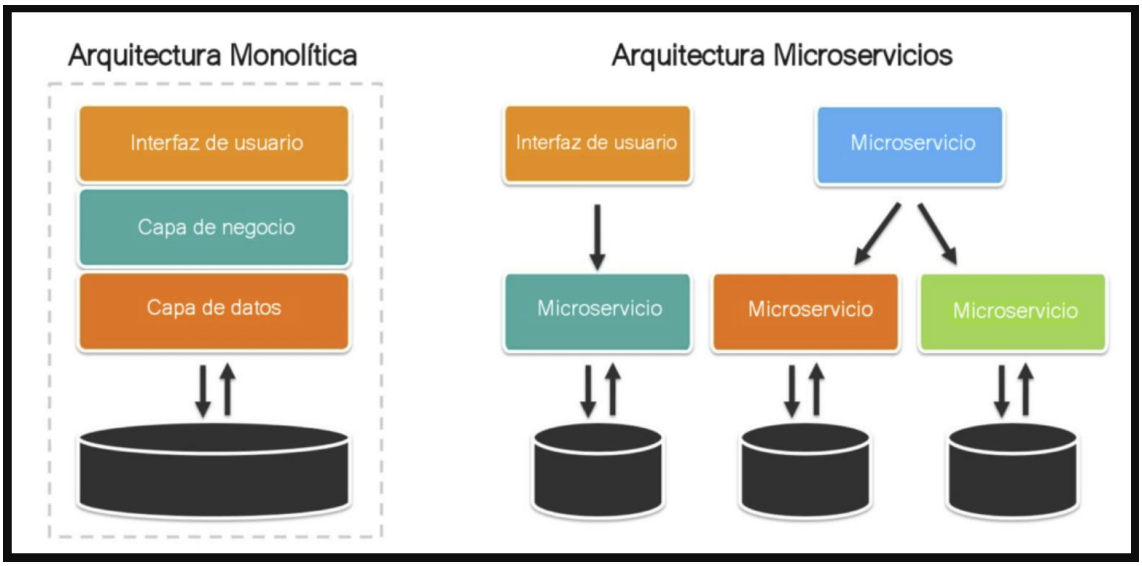
- **¿Cuáles son las diferencias entre aplicaciones monolíticas y de microservicios?**

Una aplicación monolítica se compila como una sola unidad unificada, mientras que una arquitectura de microservicios es una serie de servicios pequeños que se pueden implementar de forma independiente.

Una de las principales diferencias es que en la aplicación monolítica todo se encuentra dentro de la misma base de datos y en la de microservicios las aplicaciones se encuentran en diferentes bases de datos.

Otra diferencia es que hacer una modificación dentro de una arquitectura monolítica, puede implicar tener que realizar cambios en otras partes de la aplicación y que si se produce un error, existe la posibilidad de que caiga el sistema por completo. Por el contrario, una arquitectura de microservicios nos permite hacer modificaciones aisladas y el fallo en uno de los componentes no implica el fallo de toda la aplicación.

En general, construir una arquitectura monolítica lleva menos tiempo que una de microservicios.



Fuente imagen: web