# CHECKPOINT 3

**Maite Ekhiñe Mora**

*PT Full Stack Development with JavaScript, Python, React*

# PREGUNTAS TEÓRICAS

## ¿Cuáles son los tipos de datos en Python?

Los nueve principales tipos de datos de Python son los boleanos, números, cadenas, bytes y matrices de bytes, none, listas, tuplas, conjuntos, y diccionarios.

Los booleanos devuelven únicamente dos valores, 'verdadero o falso', es decir, puedes preguntarle al programa si un acontecimiento ha sucedido o no (por ejemplo, si es de noche), y te responderá, bien True, o bien False.

Python permite utilizar los números, así como las librerías numéricas, para desarrollar innumerables funciones dentro de la programación, de la ciencia de datos, del aprendizaje automático, etc. Un número puede ser entero (5), flotante (5.9), decimal…

Las cadenas deben ser contenidas entre unas 'comillas únicas' o "comillas dobles", y pueden estar formadas por caracteres, ya sea en forma de palabras o frases simples, o de valores más complejos como interpolaciones o archivos de HTML.

Los bytes y las matrices de bytes se utilizan al trabajar con imágenes y en un nivel avanzado de programación.

*None* o *null* se utilizan cuando defines una variable que todavía carece de valor alguno, pero se lo quieres otorgar más adelante.

Las listas, los conjuntos, y las tuplas almacenan colecciones de valores, y los diccionarios contienen colecciones almacenadas mediante parejas clave-valor ( {marca: Apple} ).

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

Es recomendable seguir las convenciones de desarrollo de Python porque, si otra persona va a acceder y utilizar tu código, será más sencillo y rápido para ella, y se generarán menos errores, si puede comprender fácilmente los distintos elementos por los que está compuesto tu código, sus funciones, etc.

Teniendo esto en cuenta, algunas de las prácticas recomendables son:

- Los nombres de las variables sean lo más descriptivos posibles.

    *edad_mínima = 25*

- Si se componen de múltiples palabras, estas se separen entre sí con un '_', para que quede claro que se trata de una variable, y no de una clase.

    *primer_apellido = Mora*

- La L minúscula, la i mayúscula o la o mayúscula no deben usarse como elementos únicos para nombrar variables, ya que pueden confundirse entre ellas o con el cero.

## ¿Qué es un heredoc en Python?

Según el funcionamiento de Python, si estoy tratando de trabajar con un texto compuesto por diversos párrafos, por ejemplo, el siguiente:

*palabras = 'Esta es una primera frase, con muchas, muchas, muchas palabras, que ocupan mucho espacio.*

*Seguida por más frases en un párrafo diferente.*

*Y, en un tercer párrafo, va a tratar más temas todavía.'*

Python va a leer la variable *palabras,* el =, la primera comill*a,* y va a buscar, después de la primera comilla, cuál es el contenido de *palabras*, sin embargo, al llegar al final de la línea, después de *espacio*, y encontrarse con que no hay una comilla de cierre, va a dar error. A esa cadena compuesta por múltiples líneas se le llama heredoc, de modo que debemos escribir el código de la siguiente manera:

*palabras = '''Esta es una primera frase, con muchas, muchas, muchas palabras, que ocupan mucho espacio.*

*Seguida por más frases en un párrafo diferente.*

*Y, en un tercer párrafo, va a tratar más temas todavía.'''*

O de la siguiente manera, más ordenada:

*palabras = '''*
*Esta es una primera frase, con muchas, muchas, muchas palabras,*
*que ocupan mucho espacio.*

*Seguida por más frases en un párrafo diferente.*

*Y, en un tercer párrafo, va a tratar más temas todavía.*
*'''.strip()*

Así Python sabe que debe esperar un salto de línea a través del cual la cadena va a continuar.

## ¿Qué es una interpolación de cadenas?

Una interpolación de cadenas nos permite crear un cuerpo dinámico por medio de la unión de diferentes cadenas. Por ejemplo, si estamos creando una invitación para una fiesta, podríamos crear una variable en la que contendremos la lista de personas invitadas (persona_invitada = 'persona'), un texto fijo ('ojalá puedas venir a mi fiesta improvisada de mañana, si no, te echaré de menos'), y unirlas con la siguiente función: f'{persona_invitada}, ojalá puedas venir a mi fiesta improvisada de mañana, si no, te echaré de menos'.

## ¿Cuándo deberíamos usar comentarios en Python?

El código que construyamos debe ser lo más claro y conciso posible, es decir, según las buenas prácticas, la nomenclatura que le demos a las variables, funciones, etc, debe ser lo suficientemente descriptiva como para que no sea necesario añadir comentarios explicativos, que, además, podrían quedar obsoletos según fuésemos escalando el código, y crearían confusión ante una relectura o una nueva persona trabajando en su construcción. De esta manera, los comentarios deben guardarse únicamente para situaciones en las que aporten un valor adicional sobre un código ya construido sobre las mejores prácticas, como, por ejemplo, para organizarlo (barra de navegación, columna central, columnas laterales, pie de página…).

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

Las aplicaciones monolíticas tienen unificadas sus diferentes funciones en un único código, mientras que las aplicaciones de microservicios están integradas por distintos grupos de códigos, independientes entre sí, que se centran, cada uno, en una función especifica de la aplicación. Es decir, Instagram tiene un código especifico para el intercambio de mensajes, otro separado para el feed de contenido, otro separado para las sugerencias de perfiles, etc. Las diferencias entre ambos sistemas radican en que, si bien una aplicación monolítica es más rápida de construir, ya que puedes crear un código con funciones básicas e irlo optimizando con el tiempo, una aplicación de microservicios es más fiable y fácilmente escalable, dado que puedes testar sus distintas partes obteniendo una idea más precisa de en qué función / parte del código hay un error, y puedes ir optimizando sus funciones individualmente, acorde a las necesidades o posibilidades específicas de esa función, sin afectar al resto del código, además, si por alguna razón la capacidad de Instagram para compartir contenido dejase de funcionar momentáneamente, las personas usuarias seguirían pudiendo consumir contenido ajeno y comunicarse mediante DMs, ya que el error recaería sobre una sola función, la de crear nuevo contenido, y no sobre la aplicación entera. Por otro lado, las aplicaciones de microservicios pueden tener cierto letargo respecto a las monolíticas, dado que deben conectarse a más servidores.