Preguntas teóricas

1. ¿Cuáles son los 4 tipos de datos en Python?

   Los 4 principales tipos de datos en Python son:

   - String
   - Number
   - List
   - Boolean

2. ¿Qué tipo de convención de nomenclatura debemos usar para las variables en Python?

   La convención de nomenclatura señala que en el caso de variables con múltiples nombres es recomendable colocar "guiones_bajo" (underscores) entre ellos. Dicha convención busca evitar problemas de interpretación entre programadores o desarrolladores. De no ser así, podría darse el caso, por ejemplo, que un determinado nombre fuera interpretado como *variable* por un programador, pero como *clase* por otro, generando como consecuencia eventuales errores.

3. ¿Qué es un heredoc de Python?

   Un "heredoc" es, básicamente, un "string" (si no me equivoco, "cadena" en castellano) de varias líneas. Ahora bien, requiere de una sintaxis particular distinta a aquella empleada en el caso de una palabra o línea; de no ser así, se produce un error.

   En el caso de una sola palabra o línea, la sintaxis de Pyhton requiere de una sola comilla ("...") tanto al inicio como al final del texto para transformarlo en string. Distinto es lo que sucede cuando tenemos mùltiples o varias líneas. En ese escenario, se requiere colocar tres comillas (""" ... """) al inicio y al final.

4. ¿Qué es la interpolación de cadenas?

   La interpolación de cadenas es un tipo método o proceso que entrega la posibilidad de ejecutar un código Python dentro del centexto de una cadena.

   A continuación veámos uno de los ejemplos presentado por el profesor en el video. En primer lugar define una variable (name) a la cual le asigna un valor específico; en este caso, una cadena (string) con el nombre propio 'Kristine'.

   name = 'Kristine'

   Una vez definido lo anterior, crea una segunda variable (greeting) la cual, podríamos decir, envuelve o hace referencia, de alguna manera, al proceso de interpolación de cadenas.

greeting = f'Hi name'

Como se puede ver de la variable anterior, la sintaxis utilizada para ejecutar dicho proceso se caracteriza por dos cosas. Primero, por colocar una letra f al inicio de la cadena. Segundo, por poner las variables entre "llaves" ({ }, curly brackets).

Por último, imprimimos la variable greeting (print(greeting)) para ver el resultado de dicho proceso o método de interpolación de cadenas. Si todo está correcto, debiera imprimirse: Hi Kristine.

5. ¿Cuándo debemos usar comentarios en Python?

   A menudo se piensa que mientras más comentarios al código, mejor. Como si los comentarios fueran buenos por sí mismos como consecuencia de un intento de querer explicar o fundamentar cada expresión que se escribe. Me parece que sucede algo parecido con las notas al pie de página de textos más especializados. A menudo hay un exceso de notas al pie de página que, o bien, podrían ir en el texto principal, o bien, no debieran estar ya que tienen poco o nada que ver con el texto en general.

   Los comentarios en los programas debieran estar limitados a los componentes más relevantes, y a aquellos que pueden generar problemas de interpretación.

6. ¿Cuáles son las diferencias entre las aplicaciones monolíticas y de microservicios

   Las aplicaciones monolíticas se caracterizan por poner todas las funciones dentro de un solo proceso. Aquella estructura tiene varias ventajas. En primer lugar, es más rápido de desarollar, ya que es posible crear una aplicación con caracterìsticas básicas, para luego con el tiempo ir mejorándola o "escalándola". Y en segundo lugar, suelen ser más rápidas ya que no deben comunicarse con múltiples servicios. Uno de los inconvenientes de este tipo de aplicaciones, eso sí, es que su mantención y ajuste puede resultar complejo si no están diseñados correctamente. No es poco probable que un simple cambio puede generar modificaciones relevantes y no deseadas en otras partes del código.

   Las aplicaciones de microservicios, a diferencia de las anteriores, desarrollan varios servicios dentro de una aplicación; cada uno de los cuales se puede evaluar y desarrollar de manera independiente. Uno de los beneficios más importantes de este tipo de aplicación es la posibilidad de "escalar" cada servicio de manera individual, sin la necesidad de tener que hacerlo en su conjunto. Asimismo, es posible evaluar cada componente de manera individual, pudiendo aislar el problema de manera rápida y sencilla.