M2C3 Python Assignment – Preguntas teóricas

Además necesito que me crees una cadena que contenga la palabra "Hola". Usando la palabra clave en el método de búsqueda o el índice, busque y seleccione "Hola" en su cadena. Y usando la función de reemplazo, reemplace "Hola" en su cadena con "adiós".

```
#Exercise 10
sentence = 'Hola esta es la tarea adicional'
saludo = sentence[0:4]
sentence = sentence.replace('Hola', 'adiós')
print(saludo)
print(sentence)
```

Ejercicio añadido como '*Exercise 10*' en el documento "Checkpoint_3.py" subido a Github.

Este ejercicio de Python debes subirlo a tu Git-Hub o Replit para poder revisarlo

---

Y por último, las preguntas teóricas son:

¿Cuáles son los tipos de Datos en Python?

Los tipos de datos de Python incluyen:

Booleans

Numbers

Strings

Bytes and byte arrays

None

Lists

Tuples

Sets

Dictionaries

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

De acuerdo con lo estudiado debemos usar un guión bajo para formar los nombres de las variables y hacer que sean lo más descriptivas posible ('*Snake case*' naming convention).

‘*camel case*’ es usado por muchos lenguajes de programación (JSON) pero los desarrolladores que trabajan con Python asumen por convención que se trata de clases y no de variables. PEP 8 – Style Guide for Python Code.

¿Qué es un Heredoc en Python?

Un ‘Heredoc’ es una cadena de valores (‘*string*’) con varias líneas de texto (‘*multi-line strings*’). Es un literal de entrada utilizado por muchos lenguajes de programación y *scripting*. Un "*heredoc*" se puede traducir como ‘*here*’ (aquí) ‘*doc*’ (documento) y es una forma de especificar un bloque de texto, preservando los saltos de línea, sangrías y otros espacios en blanco dentro del texto (conservando su formato de entrada). Dicho código utiliza comillas triples para definir una cadena de varias líneas, también conocida como cadena *heredoc*. El resultado será la cadena exactamente como aparece en el código. Puede ser usado para asignar una cadena con varios saltos de línea o para realizar un comentario que requiere un texto extenso. *Heredoc* no solo sirve para hacer comentarios sino que en algunas funciones como la de ‘*string interpolation*’ con un texto (mensaje de una aplicación o correo electrónico automático) con variables indexadas que se pueden modificar dinámicamente teniendo un texto compuesto y funcional de varias líneas. Lo más habitual es usar los *heredocs* para citar contenido dentro de una aplicación o web. *Heredoc* admite delimitadores directamente en el texto como EOD (End of Data), EOT (End of Text), END así como el uso de sangrías. Heterodoc además ayuda a la legibilidad, sin el uso comillas desordenadas ni unión de varias líneas, menos posibilidades de cometer errores por la forma en que se presenta el texto y admite los caracteres especiales dentro del bloque (por ejemplo acentos en Castellano).

¿Qué es una interpolación de cadenas?

Una interpolación de cadenas (‘string interpolation’) permite operar el código de Python dentro de una cadena (‘string’), es decir, llamar variables que tengo en mi código dentro de una cadena por medio de su índice o de la variable encerrada en llaves {}. Esto hace que el código deje de ser estático en dichas variables y se transforme en dinámico al poder cambiar los valores de estas variables sin alterar el texto que se incluye en la cadena.

¿Cuándo deberíamos usar comentarios en Python?

Como buena práctica tanto las variables como el código debe ser preparado de una forma autoexplicativa y no requerir el uso de comentarios porque el comportamiento del código suele cambiar mientras se refina. Una de las veces donde sí deberíamos usar los comentarios es para organizar el código y así tener los diferentes elementos accesibles de manera visual.

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

Las aplicaciones monolíticas generalmente se construyen como un todo, el funcionamiento del sistema se basa en una única aplicación mientras que las aplicaciones de microservicios dividen el trabajo en varios procesos independientes (servicio) pero conectados para dar sentido a la aplicación como un todo, cada servicio puede ser probado y desarrollado de manera independiente sin alterar el funcionamiento de la aplicación completa porque cada uno funciona separadamente.