M2C3 Python Assignment+6Theoricals

During the first section of Module 2, you have learned all about Python. Python is a powerful programming language that serves a lot of purposes. To practice what you have learned in this section, you will complete some Python exercises. You may use Visual Studio Code, Repl.it, or another text editor/environment of your choice. Please complete the following assignment and reach out on the Support App to have a mentor review your work. If you have any questions or need any help, please reach out so we can help you! This assignment must be completed to pass this section of the coursework.

*Concepts included

Exercise 1: Create a string, number, list, and boolean, each stored in their own variable.

Exercise 2: Use an index to grab the first 3 letters in your string, store that in a variable.

Exercise 3: Use an index to grab the first element from your list.

Exercise 4: Create a new number variable that adds 10 to your original number.

Exercise 5: Use an index to get the last element in your list.

Exercise 6: Use split to transform the following string into a list.

```
names = 'harry,alex,susie,jared,gail,conner'
```

Exercise 7: Get the first word from your string using indexes. Use the upper function to transform the letters into uppercase. Create a new string that takes the uppercase word and the rest of the original string.

Exercise 8: Use string interpolation to print out a sentence that contains your number variable.

Exercise 9: Print “hello world”.

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".
Este ejercicio de Python debes subirlo a tu Git-Hub o Replit para poder revisarlo. Primeramente debes realizar unos ejercicios prácticos sobre lo aprendido, y luego responder algunas preguntas teóricas, necesito que las respondas con tus propias palabras, agregando ejemplos, buenas prácticas, y todo lo que has aprendido.

M2C3. 6 preguntas teóricas

1.¿Cuáles son los tipos de Datos en Python?

Booleans, strings, numbers, bytes & byte arrays, ~~heredocs~~, none or null, lists, sets, tuples y, dictionaries

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

variable_underscore = “mejor snake_case con underscore para variables que con CamelCase ya que se suelen usar mayúsculas para classes”

No mezclar snake_case y CamelCase.

3. ¿Qué es un Heredoc en Python?

Heredoc es multiple-line string. Funciona teniendo en cuenta todas las nuevas líneas de carácteres, también “retornos de carro”

Contenido = **“““**

Párrafo Lorem ipsum…

Párrafo Lorem ipsum…

Párrafo Lorem ipsum…

”””.strip()

print(contenido)

el output (return) se pueden **limpiar caracteres del principio y fin,** includos **“retornos de carro”**, con la función **.strip()**

Puede valer, por ejemplo, para cuando metes contenido de la descripción de un blog post y quieres limpiarlo de los carácteres del principio y final.

4. ¿Qué es una interpolación de cadenas?

Sirve para que se pueda interpretar python dentro de cadenas.

Puedes usar cualquier función dentro de {} como operaciones matemáticas e incluso con **heredocs para personalizar emails** por ejemplo con productos y más datos

input:

name = Mikel

personal_greeting = f ‘Hello {name}’

prin(personal_greeting)

Output:

Hello Mikel

5. ¿Cuándo deberíamos usar comentarios en Python?

Solo para explicar instrucciones o describir un comportamiento.

Lo ideal es nombrar bien cada componente (variables, classes, etc) para que no haga falta comentarios.

Convenciones Standard:

#TODO = not done yet

Multiline comentarios (3 dobles comillas) “““ ……. ”””

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

Monolíticas:

tienen mayor peligro de caer globalmente ya cualquier parte afecta al todo. Más compleja de mantener si no está bien hecha.

Más rápida de empezar a funcionar algo básico y después ir escalando con el tiempo.

Más rápida porque no se tiene que conectar a otros módulos y servicios.

Microservicios:

menor peligro de caer entera por estar formado por bloques y detectar los fallos y aislarlos.

Mayor escalabilidad por modulos y a la demanda de la necesidad concreta.

Más lenta y costosa de empezar, para proyectos más grandes.

Incompatibilidades entre servicios cuando se actualizan los componentes individualmente