DevCamp 2025 – Module 2 – Checkpoint 3: Questions
Date: 2025, Feb. 11th
Student: Alexandr Gomez.

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

Datos:

Cadenas (Caracteres alfanuméricos)

Booleanos (Valores Verdadero o Falso para una consulta)

Numeros (Integrales, Decimales, ...)

Estructuras de Datos:

Bytes / Arrays de Bytes

Listas

Tuplas

Sets

Diccionarios

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

En general, hay varias formas, por ejemplo:

- Tipo "*CamelCase*"

- Tipo "*kebab-case*"

- Tipo "*pascalCase*".

- Tipo "*snake_case*". (Es la que usamos en Python)

3. ¿Qué es un Heredoc en Python?

Además de servir para documentar/comentar algo extensamente, en varias lineas, realmente usamos heredocs para introducir código a ejecutar con gran cantidad de saltos de linea sin tener que recurrir a las indicaciones manuales de salto de linea (\n)

Un heredoc mantendrá la estructura de idéntica manera.

Ejemplo:

```
html_parser= """<html>
…."""
```

4. ¿Qué es una interpolación de cadenas?

Ocurre en casi cualquier lenguaje de programación de alto nivel, orientado a objetos:

→ Permite acceder a variables previamente definidas dentro de otras variables o partes del código.

Tenemos varias formas de interpolación:

* Simple, usando variables.
* A través de índices.

Ejemplo con variables simples:

```
name_1 = 'Pepito'

name_2 = 'Jose'

place = 'casa'


string_interpolated = f’¡Hola Don {name_1}!\n¡Hola Don {name_2}!\n¿Pasó Vd. por mi {place}?\nPor su {place} yo pasé.'
```

Ejemplo con índice:

```
client = '#453 - Elisabeth/Sue'

age = 50

product = 'The Substance (Starter Kit)'

from_account = 'José from Don Pepito\'s Store'


mail = 'Product Purchased: {0}\n Greetings, {1}!\nYou are listed as {2} years old. \nRemember you both are one.\nCheers!\n{3}'.format(product, client, age, from_account)
```

5. ¿Cuándo deberíamos usar comentarios en Python?

Una buena práctica es documentar el código siempre. Esto es, las funciones, qué realizan, qué argumentos se podrán utilizar; información para comprender cada parte del código.

En cambio, una mejor práctica reside en la creación de código lo suficientemente auto-descriptivo como para prescindir de la mayoría de comentarios típicos. Es decir, puedo nombrar mis funcionas, variables, de tal manera que alguien que revise el código pueda entender fácilmente el paso a paso de su ejecución.

Ejemplo:

```
# This function will load both heredocs merging them into a HTML with CSS/JS embebbed to be loaded on a local
server by the previous function

# This is the callback to Flask in order to set a local server up with access on tcp 3000

def flask_config_1:

          ….


def parser(content_1, content_2):

          …

# This is the html content

content_1 = “””

<html>

…

“””

# This is the JS scriot

content_2 “””

parser(heredoc_1, heredoc_2)
```

Ejemplo mejor:

```
def serverConfig:

          ….


def html_css_js_JSON_join(html-part, css-part, js-part):

          …


html_part = “””<html>

…“””

# This is the JS scriot

js_part = “””function…

...”””

htmk_css_js_JSON_join(html_part, css_part, js_part)
```

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

Es un concepto muy similar al que identifica las arquitectura de sistemas, pero aplicado al entorno de desarrollo de aplicaciones.

Aquí, en el back-end, las aplicaciones monolíticas basan toda la estructura de su implementación en la misma capa, es decir, en un “gran bloque de código” unificado que integra todas y cada una de las funciones/servicios.

Esto no es suficientemente eficiente respecto a mantenimiento, escalabilidad ni disponibilidad pues, si ocurre una falla en ejecución será toda la aplicación la que no pueda funcionar.

El enfoque de las aplicaciones microservicios, en cambio, basa su estructura en un concepto “modular” donde todas las implementaciones, servicios, están separados entre sí, como diferentes capas, a las que la aplicación accede.

Esto mejora las deficiencias señaladas sobre las aplicaciones monolíticas.

Por ejemplo, en una implementación de “Tienda Online”, mi porción de código encargada de conectar con la API de un tercero para realizar pagos puede fallar temporalmente que, esto, no detendrá el resto de servicios integrados como realizar búsquedas en el catálogo, configurar la cesta, etc.