M3C5 Python Assignment 3

Preguntas teóricas.

1. ¿Qué es un condicional?

Es uno de los conceptos fundacionales, que generan código, para conseguir que un programa de Python sea dinámico. Y por dinámico se entiende que el programa puede empezar a tomar decisiones.

Los condicionales son una forma simple de decir, “si ocurre una situación, quiero que, como programa, realices una tarea. Pero si ocurre una situación diferente, quiero que realices esta otra tarea”.

Los condicionales utilizados son “if”, “else” y “elif”.

Un ejemplo muy común de este tipo de estructuras de control son los sistemas de autenticación de las páginas web, en las que, se nos pide identificarnos para saber si tenemos permiso o acceso a la totalidad de la información como usuarios autorizados o se nos limita, o niega, el acceso a esa información si no estamos autorizados por no disponer de la identificación correspondiente.

La sintaxis para codificar el ejemplo anterior podría ser como sigue:

```
role = ‘admin’
if role == ‘admin’
        auth = ‘can access’
else:
        auth = ‘cannot access’
print (auth)
```

2. ¿Cuáles son los diferentes tipos de bucles en Python?

Tenemos dos tipos de bucles en Python: “for” y “while”.

La sintaxis “for” nos permite repetir, o iterar, el código las veces que le indiquemos, de partida. La sintaxis “while” nos permite ejecutar un código repetidas veces pero no tiene definido, de antemano, el número de veces que lo ha de hacer. Es por ello que hemos de añadir un código para que la condición tenga un final, si no, la condición será repetida sin fin.

La sintaxis de “for”, tanto en listas como en tuplas, será:

```
for elemento in lista o tupla:
    print(item)
```

Esto nos devolverá el listado de los elementos que están en esa lista o tupla, en su caso.

Cuando se trata de diccionarios, la sintaxis será:

```
for clave in elemento in diccionario.items():
    print(‘Clave’, clave)
    print(‘Elemento’, elemento)
```

En el primer caso optendremos el contenido de todas la claves.

En el segundo, el contenido de todos los elementos.

La sintaxis “while” será:

```
num = (0, 1, 2, 3, 4)
counter = 0
while counter < len(num):
    print(counter)
    counter +=1
```

En este caso obtendremos el conteo de los números hasta llegar el último, y parará el bucle porque se lo hemos indicado en el código. Si no lo hubiésemos codificado con “counter +=1”, seguiría indefinidamente.

3. ¿Por qué son útiles?

Nos permite la ejecución de parte de un programa las veces que se lo pidamos. Facilita y simplifica codificar.

Pueden ser utilizados en en cadenas, listas, en tuplas y en diccionarios.

4. ¿Qué es una lista por comprensión en Python?

Es esencialmente un conjunto de bucles “for” y condicionales que se pueden colocar todos dentro de una línea de código. Si tenemos el siguiente código inicial:

```
num_list = range(1, 11)
cubed_num = [ ]
```

Y queremos realizar la siguiente codificación:

```
for num in num_list:
cubed_nums.append(num**3)
    print(cubed_nums)
```

Si utilizamos lista por comprensión, para obtener lo mismo, se verá de la siguiente forma:

```
cubed_nums = [num**3 for num in num_list]
```

5. ¿Qué es un argumento en Python?

Un argumento es el valor que se otorga a una función para que realice una tarea específica.

Hasta ahora hemos estado utilizando en la programación argumentos posicionales. Esto significa que el mapeo entre el valor y la forma en que se usa ese valor en la función, está determinado por la posición y el orden en que pasamos los valores.

Ejemplo:

```
full_name('Juan', 'Pérez')
```

Para desarrollos mayores y más complejos, lo más indicado son los argumentos con nombre, que son los que se pasan a una función especificando su nombre en la llamada a la función. Esto permite que el código sea más claro, legible y más flexible a la hora de trabajar, precindiendo del orden en el que se encuentran esos valores.

Por ejemplo, haciendo referencia al código anterior, este se reflejará de la siguiente manera:

```
full_name(first='Juan', last='Pérez')
```

6. ¿Qué es una función Lambda en Python?

Es una función que nos permite empaquetar un comportamiento, en general, un comportamiento pequeño, y luego introducirlo en otras funciones.

Un lambda es muy similar a una variable que se puede introducir en lugar de un valor básico, como una cadena, un diccionario o algo parecido. Nos permite empaquetar un proceso y que sea escalable. Son muy móviles y fáciles de usar.

Ejemplo:

```
full_name = lambda first, last: f '{first}, {last}'
print (full_name('Juan', 'Pérez')
```

7. ¿Qué es un paquete pip?

Es un sistema de gestión de paquetes o gestor de paquetes que nos permite acceder desde nuestro editor de lenguaje Python a una gran cantidad de códigos que están disponibles online en Python Package Index (PyPI), repositorio de software oficial para aplicaciones de terceros en el lenguaje de programación Python. Se usa “pip” como soporte de Python en la nube.

También podemos descargar los que más nos interesen como archivos y utilizarlo sin necesidad de conexión.

Ejercicio práctico.

Adicional a esta asignación de crear una documentación, necesito que realices los siguientes ejercicios prácticos, recuerda subirlos a Git-Hub o Replit para revisarlos:

- Cree un bucle For de Python.
- Cree una función de Python llamada suma que tome 3 argumentos y devuelva la suma de los 3.
- Cree una función lambda con la misma funcionalidad que la función de suma que acaba de crear.
- Utilizando la siguiente lista y variable, determine si el valor de la variable coincide o no con un valor de la lista. *Sugerencia, si es necesario, utilice un bucle for in y el operador in. nombre = 'Enrique' lista_nombre = 'Jessica', 'Paul', 'George', 'Henry', 'Adán'