# **¿Qué aprendimos en la sesión anterior?**

Hablemos sobre lo que aprendimos en la sesión de la semana 1.

# **Respuestas de los ejercicios (semana 1)**

1. ¿Cuál de las siguientes es una variable y cuál es una cadena?  
   **R:** 'spam'
2. ¿Por qué esta expresión causa un error? ¿Cómo puedes arreglarlo?  
   'He comido' + 99 + 'burritos'.  
   **R:** 'He comido' + str(99) + 'burritos'
3. ¿Cuáles son los dos valores de tipo de datos booleanos? ¿Cómo los escribes?  
   **R: True** y **False**. Los escribes con T y F en mayúsculas y el resto en minúsculas.
4. Escriba las tablas de verdad de cada operador booleano (es decir, cada posible combinación de valores booleanos para el operador y lo que evalúan)  
   **R:   
   AND**  
   True and True is True.  
   True and False is False.  
   False and True is False.  
   False and False is False.  
   **OR**  
   True or True is True.  
   True or False is True.  
   False or True is True.  
   False or False is False.  
     
   **NOT**  
   not True is False.  
   not False is True.  
   Ver la [documentación de Python 3](https://docs.python.org/3/reference/expressions.html#boolean-operations)
5. ¿Qué evalúan las siguientes expresiones?  
   **R:**  
   (5 > 4) and (3 == 5) **False**  
   not (5 > 4) **False**  
   (5 > 4) or (3 == 5) **True**  
   not ((5 > 4) or (3 == 5)) **False**  
   (True and True) and (True == False) **False**  
   (not False) or (not True) **True**
6. Escriba el código que imprime ‘Hello’ si 1 está almacenado en la variable **spam**, imprime ‘Howdy’ si 2 está almacenado en **spam** e imprime ‘Greetings!’ si algo más está almacenado en **spam**.  
   **R:**  
   if spam == 1:  
    print('Hello')  
   elif spam == 2:  
    print('Howdy')  
   else:  
    print('Greetings!')
7. ¿Cómo se puede evitar que un programa falle cuando se produce un error?  
   **R:** Coloque la línea de código que podría causar un error en una cláusula try.  
   Ver [documentación de Python manejo de errores](https://docs.python.org/3/tutorial/errors.html)
8. Escriba una función llamada **collatz()** que tenga un parámetro llamado number. Si number es par, entonces **collatz()** debería imprimir number // 2 y devolver este valor. Si number es impar, entonces collatz() debería imprimir y devolver 3 \* number + 1.  
   **R:**  
   def collatz(number):  
    if number % 2 == 0:  
    result = number // 2  
    elif number % 2 == 1:  
    result = 3 \* number + 1  
    print(str(result))  
     
   *Para las siguientes tres preguntas, digamos que bacon contiene la lista [3.14, 'cat', 11, 'cat', True].*
9. ¿A qué evalúa **bacon.index('cat')**?  
   **R:** 1
10. ¿Qué hace **bacon.append(99)** en los valores de la lista bacon?  
    **R:** Agrega el 99 a la lista. Quedando de esta forma:  
    [3.14, 'cat', 11, 'cat', True, 99]
11. ¿Qué hace **bacon.remove('cat')** en los valores de la lista bacon ?  
    R: [3.14, 11, 'cat', True]
12. **El código de las comas**  
    Digamos que tiene una lista como esta:  
      
    **spam** = ['manzanas', 'bananas', 'tofu', 'gatos']  
      
    Escriba una función que tome una lista como argumento y devuelva una cadena con todos los elementos separados por una coma y un espacio, con una ‘y’ insertado antes del último elemento. Por ejemplo, pasando la lista previa de **spam** a la función, debería devolver **'manzanas, bananas, tofu, y gatos'**. Su función debería poder funcionar con cualquier lista que se le pase.  
      
    def comma\_code(shopping\_list):  
     result = ''  
     for i in range(len(shopping\_list)):  
     element = shopping\_list[i]  
     if i == len(shopping\_list) -1:  
     result += 'and ' + str(element)  
     else:  
     result += str(element) + ', '   
     print(result)  
      
    # Probar en consola con estas dos listas  
    compras = ['apples', 'bananas', 'tofu', 'cats']  
    lista\_escuela = ['libros', 'cuadernos', 'lapices', 'colores', 'limpion', 'goma']  
    >>> comma\_code(lista\_escuela)  
    >>> comma\_code(compras)
13. ¿Cómo luce un diccionario con una clave 'foo' y un valor 42?  
    **R:** {'foo': 42}
14. ¿Qué sucede si intenta acceder a **spam['foo']** si **spam** es {'bar': 100}?  
    **R:** Obtiene un error KeyError.  
      
    Wiki sobre el [error KeyError](https://wiki.python.org/moin/KeyError)
15. **Inventario de Videojuego de Fantasía**  
    Estás creando un videojuego de fantasía. La estructura de datos para modelar el inventario del jugador será un diccionario donde las llave son strings que describen el artículo en el inventario y el valor es un valor entero que detalla cuántos de ese elemento tiene el jugador. Por ejemplo, el valor del diccionario **{'cuerda': 1, 'antorcha': 6, 'moneda de oro': 42, 'daga': 1, 'flecha': 12}** significa que el jugador tiene 1 cuerda, 6 antorchas, 42 monedas de oro, etc.  
    Escriba una función llamada **mostrarInventario()** que tomaría cualquier posible "inventory" y lo mostraría de la siguiente manera:  
      
    Inventario:  
    12 flecha  
    42 moneda de oro  
    1 cuerda  
    6 antorcha  
    1 daga  
    Número total de objetos: 62  
      
    Pista: puede usar un bucle for para recorrer todas las teclas de un diccionario.  
      
    **# inventario.py**  
    cosas = {'cuerda': 1, 'antorcha': 6, 'moneda de oro': 42, 'daga': 1, 'flecha': 12}  
      
    def mostrarInventario(inventario):  
     print("Inventario:")  
     total\_objetos = 0  
     for k, v in inventario.items():  
     print(str(v) + ' ' + str(k))  
     total\_objetos += v  
     print("Numero total de objetos: " + str(total\_objetos))  
    mostrarInventario(cosas)