¡Felicitaciones! ¡Aprobaste!

Calificación recibida 100 % **Para Aprobar** 80 % o más

Ir al siguiente elemento

Estás viendo una versión traducida automáticamente de esta evaluación
Puedes volver a ver el contenido en su idioma original si lo prefieres. No perderás el progreso que hayas conseguido si cambias el idioma.
Mostrar la versión en Inglés
Desestimar

1. ¿Para qué sirve la recursividad? 1 / 1 punto

- ○ La recursión se utiliza para crear bucles en lenguajes en los que no existen otros bucles.
- ○ Utilizamos la recursividad sólo para implementar fórmulas matemáticas en el código.
- ○ La recursión se utiliza para iterar por los archivos de un mismo directorio.
- ◉ La recursión se utiliza para llamar a una función desde dentro de la misma función.

Correcto
¡Lo ha clavado! Puede llamar a una función dentro de sí misma para iterar sobre una jerarquía de objetos, como directorios y subdirectorios.

2. ¿Cuáles de estas actividades son buenos casos de uso para los programas recursivos? Marque todas las que correspondan. 1 / 1 punto

- ☑ Recorrer un sistema de archivos recopilando información relacionada con directorios y archivos.

Correcto
Muy bien Dado que los directorios pueden contener subdirectorios que a su vez pueden contener más subdirectorios, recorrer estos contenidos es un buen caso de uso para un programa recursivo.

- ☐ Creación de una cuenta de usuario para un nuevo empleado.
- ☐ Instalación o actualización de software en un ordenador.
- ☑ Gestionar los permisos asignados a los grupos dentro de una empresa, cuando cada grupo puede contener tanto subgrupos como usuarios.

Correcto
¡Ya lo tiene! Como los grupos pueden contener tanto grupos como usuarios, éste es el tipo de problema que constituye un caso de uso ideal para una solución recursiva.

- ☐ Comprobación de si un ordenador está conectado a la red local.

3. Rellene los espacios en blanco para que la función `is_power_of` devuelva si el número es una potencia de la base dada. Nota: se supone que la base es un número positivo. Consejo: para las funciones que devuelven un valor booleano, puede devolver el resultado de una comparación. 1 / 1 punto

```
def is_power_of(number, base):
# Base case: when number is smaller than base.
    if number < base:
        # If number is equal to 1, it's a power (base**0).
        return number == 1

    # Recursive case: keep dividing number by base.
    return is_power_of(number//base, base)


print(is_power_of(8,2))     # Should be True
print(is_power_of(64,4))    # Should be True
print(is_power_of(70,10))   # Should be False
```

Ejecutar
Restablecer

```
True
True
False
```

Correcto
¡Buen trabajo! Ha hecho que el código compruebe potencias de números
reduciendo el problema a uno más pequeño.

4. La recursión es un proceso en el que una función se llama a sí misma una o más veces con valores modificados para realizar una tarea. Esta técnica puede ser especialmente eficaz a la hora de resolver problemas complejos que pueden descomponerse en problemas más pequeños y sencillos del mismo tipo. En el código proporcionado, la función `count_users` utiliza la recursión para contar el número de usuarios que pertenecen a un grupo dentro del sistema de una empresa. Lo hace iterando a través de cada miembro de un grupo, y si un miembro es de otro grupo, llama recursivamente a `count_users` para contar los usuarios dentro de ese subgrupo. Sin embargo, ¡hay un error en el código! ¿Puede detectar el problema y solucionarlo? 1 / 1 punto

```
def count_users(group):
  count = 0
  for member in get_members(group):
    #count += 1 <-- este es el error
    if is_group(member):
      count += count_users(member)
    else:
      count += 1
  return count


print(count_users("sales")) # Should be 3
print(count_users("engineering")) # Should be 8
print(count_users("everyone")) # Should be 18
```

Ejecutar
Restablecer

```
3
8
18
```

Correcto
Bien hecho Has detectado el problema que hacía que
¡que se contaran los grupos cuando sólo queríamos contar los usuarios!

5. En el cuestionario de práctica de bucles de `while` , se le pidió que escribiera una función para calcular la suma de todos los números positivos entre 1 y n. Vuelva a escribir la función utilizando recursividad en lugar de un bucle de `while`. Recuerde que cuando n es menor que 1, la función debe devolver 0 como respuesta. 1 / 1 punto

```
def sum_positive_numbers(n):
    if n < 1:
        return 0
    return n + sum_positive_numbers(n-1)


print(sum_positive_numbers(3)) # Should be 6
print(sum_positive_numbers(5)) # Should be 15
```

Ejecutar
Restablecer

```
6
15
```

Correcto
Aquí tiene su resultado:
6
15

¡Gran trabajo! Usted realmente ha clavado escribir recursiva
¡recursivas!