

# Reporte de Evaluación - Fork de GitHub

## Información General

Estudiante: Mateo Gonzalez Villegas  
Repositorio: Mateogv09/act\_ntp\_s4  
Fecha de evaluación: 20/8/2025, 9:29:19  
Evaluado por: Sistema de Evaluación Masiva

## Resumen de Calificaciones

Calificación general: 4.6/5.0  
Actividades completadas: 20/20  
Porcentaje de completitud: 100.0%

## Detalle de Actividades

#	Descripción	Archivo	Encontrado	Calificación
1	LISTAS - Ejercicio 1: Crea una función q...	src/ejercicio_01.py	Sí	5.0
2	LISTAS - Ejercicio 2: Implementa una fun...	src/ejercicio_02.py	Sí	4.0
3	LISTAS - Ejercicio 3: Crea una función q...	src/ejercicio_03.py	Sí	5.0
4	LISTAS - Ejercicio 4: Desarrolla una fun...	src/ejercicio_04.py	Sí	4.0
5	LISTAS - Ejercicio 5: Implementa una fun...	src/ejercicio_05.py	Sí	5.0
6	TUPLAS - Ejercicio 6: Crea una función q...	src/ejercicio_06.py	Sí	4.0
7	TUPLAS - Ejercicio 7: Desarrolla una fun...	src/ejercicio_07.py	Sí	5.0
8	TUPLAS - Ejercicio 8: Implementa una fun...	src/ejercicio_08.py	Sí	4.0
9	TUPLAS - Ejercicio 9: Crea una función q...	src/ejercicio_09.py	Sí	5.0
10	TUPLAS - Ejercicio 10: Desarrolla una fu...	src/ejercicio_10.py	Sí	5.0
11	CONJUNTOS - Ejercicio 11: Crea una funci...	src/ejercicio_11.py	Sí	5.0
12	CONJUNTOS - Ejercicio 12: Implementa una...	src/ejercicio_12.py	Sí	5.0
13	CONJUNTOS - Ejercicio 13: Desarrolla una...	src/ejercicio_13.py	Sí	5.0
14	CONJUNTOS - Ejercicio 14: Crea una funci...	src/ejercicio_14.py	Sí	5.0
15	CONJUNTOS - Ejercicio 15: Implementa una...	src/ejercicio_15.py	Sí	5.0
16	DICCIONARIOS - Ejercicio 16: Crea una fu...	src/ejercicio_16.py	Sí	4.0
17	DICCIONARIOS - Ejercicio 17: Desarrolla ...	src/ejercicio_17.py	Sí	5.0
18	DICCIONARIOS - Ejercicio 18: Implementa ...	src/ejercicio_18.py	Sí	4.0
19	DICCIONARIOS - Ejercicio 19: Crea una fu...	src/ejercicio_19.py	Sí	4.0
20	DICCIONARIOS - Ejercicio 20: Desarrolla ...	src/ejercicio_20.py	Sí	4.0

## Retroalimentación Detallada

**Actividad 1: LISTAS - Ejercicio 1: Crea una función que reciba una lista de números y use un ciclo for para devolver una nueva lista con solo los números pares. Prueba la función con la lista [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].**

Archivo esperado: src/ejercicio\_01.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es claro, conciso y cumple con los requisitos del ejercicio. Buena práctica al incluir una prueba con la lista proporcionada.

**Actividad 2: LISTAS - Ejercicio 2: Implementa una función que solicite al usuario ingresar calificaciones usando un ciclo while hasta que escriba 'fin'. Almacena las calificaciones en una lista y calcula el promedio, la nota más alta y más baja.**

Archivo esperado: src/ejercicio\_02.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Podrías mejorar la estructura separando la lógica de entrada/validación de la de cálculo y salida. Además, la función `sistema\_calificaciones` no debería retornar nada, lo cual hace innecesario el `print` al final.

**Actividad 3: LISTAS - Ejercicio 3: Crea una función que reciba dos listas de igual tamaño y use un ciclo for para combinarlas elemento por elemento en una nueva lista. Ejemplo: [1,2,3] + ['a','b','c'] = [1,'a',2,'b',3,'c'].**

Archivo esperado: src/ejercicio\_03.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta, eficiente y fácil de entender. El código es limpio y sigue buenas prácticas. ¡Excelente trabajo!

**Actividad 4: LISTAS - Ejercicio 4: Desarrolla una función que simule un carrito de compras. Usa una lista para almacenar productos y un ciclo while para mostrar un menú que permita agregar, eliminar, mostrar productos y calcular el total.**

Archivo esperado: src/ejercicio\_04.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es funcional y resuelve el problema planteado. Se podría mejorar la gestión de errores (ej: validación de la entrada del precio) y usar funciones para modularizar el código.

**Actividad 5: LISTAS - Ejercicio 5: Implementa una función que reciba una lista de palabras y use ciclos anidados para encontrar y devolver todas las palabras que contienen una letra específica ingresada por el usuario.**

Archivo esperado: src/ejercicio\_05.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta, clara y concisa. El código es funcional y sigue buenas prácticas al convertir a minúsculas para la búsqueda insensible a mayúsculas.

**Actividad 6: TUPLAS - Ejercicio 6: Crea una función que genere una tupla con las coordenadas (x, y) de 10 puntos aleatorios. Usa un ciclo for para calcular cuáles puntos están dentro de un círculo de radio 5 centrado en el origen.**

Archivo esperado: src/ejercicio\_06.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Podría retornar la tupla de puntos dentro del círculo en lugar de imprimirla para mejorar la reutilización del código.

**Actividad 7: TUPLAS - Ejercicio 7: Desarrolla una función que reciba una tupla de estudiantes (nombre, edad, promedio) y use un ciclo for para encontrar y devolver una nueva tupla solo con los estudiantes que tienen promedio mayor a 8.0.**

Archivo esperado: src/ejercicio\_07.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y bien estructurada. El código es legible y cumple con la descripción del problema. Se puede mejorar la eficiencia usando una comprensión de lista para crear la tupla resultante.

**Actividad 8: TUPLAS - Ejercicio 8: Implementa una función que cree una tupla con los primeros 20 números de la secuencia de Fibonacci. Usa un ciclo while para generar la secuencia y luego un ciclo for para mostrar solo los números impares.**

Archivo esperado: src/ejercicio\_08.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Podrías mejorar la legibilidad separando la lógica de generación de Fibonacci de la lógica de impresión, y evitar usar listas por comprensión cuando un simple bucle for es suficiente para imprimir los impares.

**Actividad 9: TUPLAS - Ejercicio 9: Crea una función que simule un sistema de coordenadas. Recibe una tupla de puntos (x, y) y usa ciclos para calcular la distancia total recorrida si se visitan todos los puntos en orden.**

Archivo esperado: src/ejercicio\_09.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta, legible y bien estructurada. Implementa la lógica solicitada de manera eficiente y sigue buenas prácticas.

**Actividad 10: TUPLAS - Ejercicio 10: Desarrolla una función que reciba dos tuplas de igual longitud y use un ciclo for para crear una nueva tupla con la suma de elementos correspondientes. Ejemplo: (1,2,3) + (4,5,6) = (5,7,9).**

Archivo esperado: src/ejercicio\_10.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Solución correcta y eficiente. El uso de comprensión de tuplas es conciso y legible. Buen manejo de la validación de longitudes.

**Actividad 11: CONJUNTOS - Ejercicio 11: Crea una función que reciba dos listas y use ciclos for para convertirlas en conjuntos. Luego calcula y muestra la unión, intersección, diferencia y diferencia simétrica entre ambos conjuntos.**

Archivo esperado: src/ejercicio\_11.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Solución correcta y completa. El código es legible y cumple con la descripción del problema. Excelente uso de conjuntos y operaciones.

**Actividad 12: CONJUNTOS - Ejercicio 12: Implementa una función que solicite al usuario ingresar palabras usando un ciclo while hasta que escriba 'salir'. Almacena las palabras en un conjunto y muestra cuántas palabras únicas se ingresaron y cuáles se repitieron.**

Archivo esperado: src/ejercicio\_12.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es claro, funcional y cumple con todos los requisitos de la actividad. Buena utilización de conjuntos para identificar palabras únicas y repetidas.

**Actividad 13: CONJUNTOS - Ejercicio 13: Desarrolla una función que genere dos conjuntos: uno con números pares del 2 al 20 y otro con múltiplos de 3 del 3 al 30. Usa ciclos for para crear los conjuntos y muestra todas las operaciones entre ellos.**

Archivo esperado: src/ejercicio\_13.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y eficiente. El código es limpio y cumple con los requisitos de la actividad, utilizando ciclos for y operaciones de conjuntos de manera efectiva.

**Actividad 14: CONJUNTOS - Ejercicio 14: Crea una función que simule un sistema de votación. Usa un conjunto para almacenar los votos únicos y un ciclo while para permitir que múltiples usuarios voten. Al final, muestra los candidatos que recibieron votos.**

Archivo esperado: src/ejercicio\_14.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es claro, conciso y cumple con todos los requisitos de la actividad. El uso de un conjunto es apropiado para evitar votos duplicados.

**Actividad 15: CONJUNTOS - Ejercicio 15: Implementa una función que reciba una lista de números con duplicados y use un ciclo for para crear un conjunto con números únicos. Luego compara el tamaño original vs el conjunto para mostrar cuántos duplicados había.**

Archivo esperado: src/ejercicio\_15.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y eficiente. El código es legible y cumple con el objetivo de eliminar duplicados y mostrar la cantidad eliminada.

**Actividad 16: DICCIONARIOS - Ejercicio 16: Crea una función que simule un inventario de productos. Usa un diccionario para almacenar producto:cantidad y un ciclo while para mostrar un menú que permita agregar, actualizar, eliminar productos y mostrar el inventario completo.**

Archivo esperado: src/ejercicio\_16.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es funcional y cumple con los requisitos básicos. Se podría mejorar la validación de la entrada del usuario (e.g., evitar errores si la cantidad no es un número) y la presentación del inventario para que sea más legible.

**Actividad 17: DICCIONARIOS - Ejercicio 17: Desarrolla una función que reciba una frase y use un ciclo for para crear un diccionario que cuente la frecuencia de cada palabra. Ignora mayúsculas/minúsculas y muestra las palabras ordenadas por frecuencia.**

Archivo esperado: src/ejercicio\_17.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es claro, eficiente y cumple con todos los requisitos del ejercicio. La función ``contador_palabras`` implementa correctamente el conteo y ordenamiento de las palabras.

**Actividad 18: DICCIONARIOS - Ejercicio 18: Implementa una función que simule una agenda telefónica usando un diccionario. Usa un ciclo while para mostrar un menú que permita agregar contactos, buscar por nombre, mostrar todos los contactos y eliminar contactos.**

Archivo esperado: src/ejercicio\_18.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución cumple con los requisitos de la actividad. Se podría mejorar la validación de la entrada del usuario (por ejemplo, verificar que el teléfono sea un número) y modularizar el código en funciones más pequeñas para mayor legibilidad.

**Actividad 19: DICCIONARIOS - Ejercicio 19: Crea una función que gestione las calificaciones de estudiantes. Usa un diccionario donde la clave sea el nombre del estudiante y el valor una lista de calificaciones. Implementa funciones para agregar estudiantes, agregar calificaciones y calcular promedios.**

Archivo esperado: src/ejercicio\_19.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución funciona correctamente y cumple con los requisitos. Se podría mejorar la organización separando la lógica en funciones más pequeñas para mejor legibilidad y reutilización.

**Actividad 20: DICCIONARIOS - Ejercicio 20: Desarrolla una función que simule un sistema de registro de temperaturas por ciudad. Usa un diccionario anidado donde cada ciudad tenga un diccionario con días de la semana y temperaturas. Calcula estadísticas por ciudad y día.**

Archivo esperado: src/ejercicio\_20.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Podría mejorarse encapsulando el cálculo de promedios en funciones separadas para mayor claridad y reutilización. Además, generalizar los días de la semana evitaría hardcoding.

## Resumen General

Excelente trabajo. Completó 20/20 actividades (100%) con una calificación promedio de 4.6/5. Demuestra buen dominio de los conceptos.

## Recomendaciones

- Continuar con el excelente trabajo y mantener la calidad del código