

Reporte de Evaluación - Fork de GitHub

Información General

Estudiante: Emanuel Marin Alzate
Repositorio: Marinalzate/act_ntp_s4
Fecha de evaluación: 21/8/2025, 8:55:29
Evaluado por: Sistema de Evaluación Masiva

Resumen de Calificaciones

Calificación general: 2.1/5.0
Actividades completadas: 18/20
Porcentaje de completitud: 90.0%

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	5.0
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	2.0
8	TUPLAS - Ejercicio 8: Implementa una fun...	src/ejercicio_08.py	Sí	1.0
9	TUPLAS - Ejercicio 9: Crea una función q...	src/ejercicio_09.py	Sí	1.0
10	TUPLAS - Ejercicio 10: Desarrolla una fu...	src/ejercicio_10.py	Sí	1.0
11	CONJUNTOS - Ejercicio 11: Crea una funci...	src/ejercicio_11.py	Sí	2.0
12	CONJUNTOS - Ejercicio 12: Implementa una...	src/ejercicio_12.py	Sí	1.0
13	CONJUNTOS - Ejercicio 13: Desarrolla una...	src/ejercicio_13.py	Sí	1.0
14	CONJUNTOS - Ejercicio 14: Crea una funci...	src/ejercicio_14.py	Sí	0.0
15	CONJUNTOS - Ejercicio 15: Implementa una...	src/ejercicio_15.py	Sí	1.0
16	DICCIONARIOS - Ejercicio 16: Crea una fu...	src/ejercicio_16.py	Sí	1.0
17	DICCIONARIOS - Ejercicio 17: Desarrolla ...	src/ejercicio_17.py	Sí	1.0
18	DICCIONARIOS - Ejercicio 18: Implementa ...	src/ejercicio_18.py	No	0.0
19	DICCIONARIOS - Ejercicio 19: Crea una fu...	src/ejercicio_19.py	Sí	1.0
20	DICCIONARIOS - Ejercicio 20: Desarrolla ...	src/ejercicio_20.py	Sí	1.0

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:

La solución es correcta, clara y cumple con todos los requisitos. El código es legible y sigue las buenas prácticas de programación.

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: 5.0/5.0

Retroalimentación:

Excelente solución. El código es claro, funcional y maneja correctamente la entrada del usuario, incluyendo la validación y el caso de no ingresar calificaciones.

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 legible. El código incluye manejo de errores y buenas prácticas en general.

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 cumple con los requisitos y es funcional. La lógica para eliminar productos puede mejorarse para evitar errores si el producto no existe. Considera usar funciones para modularizar aún más 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:

Excelente solución. El código es funcional, legible y eficiente, evitando duplicados en la lista de resultados. Bien hecho.

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. Se podría mejorar la legibilidad separando la generación de coordenadas y el filtrado de puntos en funciones separadas para mejor organización 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: 2.0/5.0

Retroalimentación:

El código no cumple con la descripción del problema. No recibe una tupla de estudiantes como entrada y no devuelve una nueva tupla con los estudiantes destacados. El código simplemente imprime los estudiantes con promedio mayor o igual a 3.5.

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: 1.0/5.0

Retroalimentación:

El código no implementa la secuencia de Fibonacci ni cumple con los requisitos del ejercicio planteado. En cambio, realiza un conteo de colores favoritos. Debe rehacerse completamente.

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: 1.0/5.0

Retroalimentación:

El código proporcionado no resuelve el problema planteado (cálculo de la distancia total recorrida en un sistema de coordenadas). En cambio, elimina duplicados de una tupla de nombres. Es necesario implementar la lógica para calcular la distancia entre puntos y sumarlos.

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: 1.0/5.0

Retroalimentación:

El código entregado no resuelve el problema planteado sobre la suma de tuplas. En su lugar, filtra nombres que contienen la letra 'a'. Es necesario implementar la lógica de suma de tuplas usando un ciclo for.

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: 2.0/5.0

Retroalimentación:

El código solo maneja una lista fija y muestra un conjunto sin duplicados. No cumple con la descripción del ejercicio, que pide crear una función que reciba dos listas, las convierta en conjuntos usando ciclos for, y calcule la unión, intersección, diferencia y diferencia simétrica.

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: 1.0/5.0

Retroalimentación:

El código no resuelve el problema planteado. Implementa la intersección de dos conjuntos predefinidos en lugar de solicitar palabras al usuario e identificar 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: 1.0/5.0

Retroalimentación:

El código presentado no corresponde a la descripción de la actividad (Conjuntos con pares y múltiplos de 3). El código actual elimina un elemento de un conjunto de frutas, lo cual es incorrecto.

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: 0.0/5.0

Retroalimentación:

El código no resuelve el problema planteado de simular un sistema de votación. En cambio, imprime la edad de personas desde un diccionario. Es necesario revisar y rehacer la actividad.

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: 1.0/5.0

Retroalimentación:

La solución no resuelve el problema planteado (conjuntos y conteo de duplicados). El código presentado corresponde a un problema diferente relacionado con diccionarios e impresión de nombres. Es necesario implementar la lógica solicitada utilizando conjuntos y un ciclo for.

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: 1.0/5.0

Retroalimentación:

El código no implementa la funcionalidad requerida de un inventario con agregar/actualizar/eliminar. El código provisto calcula el total de una compra y tiene un error lógico en la iteración ``if producto in producto:``.

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: 1.0/5.0

Retroalimentación:

El código presentado no resuelve el problema planteado. En lugar de contar la frecuencia de las palabras en una frase, encuentra al estudiante con la nota más alta en un diccionario predefinido.

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 no encontrado

Calificación: 0.0/5.0

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: 1.0/5.0

Retroalimentación:

La solución no implementa la funcionalidad solicitada en la descripción del ejercicio (gestión de calificaciones con diccionarios). El código presentado imprime nombres de personas mayores de edad, que no corresponde con la actividad. Es necesario ajustar el código para cumplir con la gestión de estudiantes y sus calificaciones.

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: 1.0/5.0

Retroalimentación:

La solución presentada no corresponde a la descripción del ejercicio (registro de temperaturas por ciudad usando diccionarios anidados). En su lugar, calcula promedios de notas de estudiantes. Es necesario implementar la lógica solicitada en la descripción.

Resumen General

Necesita mejorar. Completó 18/20 actividades (90%) con una calificación promedio de 2.1/5. Se recomienda revisar los conceptos fundamentales.

Recomendaciones

- Completar los archivos faltantes: `src/ejercicio_18.py`
- Revisar y mejorar las actividades con calificación baja
- Enfocarse en mejorar la documentación y comentarios del código
- Aplicar mejores prácticas de programación