Reporte de Evaluación - Fork de GitHub

Información General

Estudiante: Emanuel Marin Alzate
Repositorio: Marinalzate/act_ntp_s4
Fecha de evaluación: 21/8/2025, 7:52:47
Evaluado por: Sistema de Evaluación de No Calificados

Resumen de Calificaciones

Calificación general: 2.0/5.0
Actividades completadas: 17/20
Porcentaje de completitud: 85.0%

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	4.0
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	0.0
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	0.0
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	Sí	1.0
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 y eficiente. El código es legible y sigue las buenas prácticas al separar la lógica en una función y luego aplicarla a la lista de prueba.

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 limpio, funcional y cumple con todos los requisitos de la actividad. Bien manejado el control de errores.

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 y eficiente. El código es legible, bien estructurado y maneja el caso de listas de diferente tamaño. ¡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 cumple con los requisitos principales y funciona correctamente. Se podría mejorar el manejo de la eliminación de productos para evitar el `if not encontrado` dentro del bucle. Considera usar un diccionario en lugar de una lista de tuplas para una mejor estructura.

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: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Bien por el uso del `break` para evitar duplicados. Considera agregar manejo de errores para entradas inválidas (e.g., entrada vacía).

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 encapsulando la lógica de verificación dentro del círculo en una función separada para mejor modularidad y reutilización.

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:

La solución no cumple con la descripción del problema (tupla de estudiantes con nombre, edad y promedio). Además, debería devolver una nueva tupla en lugar de imprimir directamente en la función.

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: 0.0/5.0

Retroalimentación:

El código entregado no corresponde con la descripción del ejercicio (generar la secuencia de Fibonacci y mostrar los impares). El código dado cuenta la frecuencia de colores favoritos, lo que indica un error en la entrega o un malentendido del problema.

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:

La solución provista elimina duplicados de una tupla de nombres, pero no cumple con el objetivo principal del ejercicio que era calcular la distancia total recorrida en un sistema de coordenadas. El código necesita ser reescrito para implementar la lógica solicitada en la descripción del problema.

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 no resuelve el problema planteado (suma de tuplas). Implementa una función diferente que filtra nombres. Debes enfocarte en entender y cumplir los requisitos de la actividad.

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:

La solución solo muestra un conjunto a partir de una lista y su longitud, pero no recibe dos listas como entrada, ni calcula la unión, intersección, diferencia y diferencia simétrica. Además, el código debería seguir la descripción de la actividad (usar ciclos for para convertir listas en conjuntos) y no usar directamente `set()`.

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 cumple con la descripción de la actividad. En lugar de solicitar palabras al usuario y almacenarlas en un conjunto hasta que escriba 'salir', simplemente calcula la intersección de dos conjuntos predefinidos. Debe implementar la lógica de entrada de usuario y el uso del conjunto como se describe en la consigna.

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 no resuelve el problema planteado (generar conjuntos de números pares y múltiplos de 3 y realizar operaciones entre ellos). En su lugar, realiza operaciones con un conjunto de frutas. El código debe ser reescrito para cumplir con la descripción de la actividad.

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 entregado no resuelve el problema planteado. El código provisto imprime la edad de personas desde un diccionario, en lugar de simular un sistema de votación utilizando conjuntos y un ciclo while.

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: 0.0/5.0

Retroalimentación:

El código proporcionado no resuelve el problema planteado en la descripción del ejercicio. Implementa una función para imprimir nombres de personas mayores de 25 años, en lugar de crear un conjunto a partir de una lista con duplicados y contar los duplicados.

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 inventario con agregar, actualizar, eliminar productos y mostrar el inventario. Hay un error lógico `if producto in producto:` que siempre será falso y el diccionario `productos` está hardcodeado y no permite la manipulación interactiva.

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 entregado no resuelve la actividad descrita (conteo de frecuencia de palabras en una frase). En cambio, encuentra el estudiante con la nota más alta en un diccionario predefinido. Necesitas implementar la lógica solicitada en la descripción del ejercicio.

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: 1.0/5.0

Retroalimentación:

El código entregado no corresponde a la descripción del ejercicio (agenda telefónica). Implementa un cálculo del total de una compra, lo cual no cumple con lo solicitado. Debes enfocarte en implementar la funcionalidad requerida usando diccionarios y un ciclo while.

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:

El código no cumple con la descripción de la actividad. Se esperaba la gestión de calificaciones de estudiantes usando diccionarios y funciones para agregar, calificar y calcular promedios, en cambio, el código dado solo imprime nombres de personas mayores de edad a partir de un diccionario predefinido.

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:

El código presentado calcula el promedio de notas de estudiantes, no simula un sistema de registro de temperaturas por ciudad como se solicitaba. La implementación no cumple con los requisitos del ejercicio planteado.

Resumen General

Necesita mejorar. Completó 17/20 actividades (85%) con una calificación promedio de 2.0/5. Se recomienda revisar los conceptos fundamentales.

Recomendaciones

- Revisar y mejorar las actividades con calificación baja
- Enfocarse en mejorar la documentación y comentarios del código
- Aplicar mejores prácticas de programación