Reporte de Evaluación - Fork de GitHub

Información General

Estudiante: Cristian Andrés Sierra Duque
Repositorio: CristianSierra420/act_ntp_s3
Fecha de evaluación: 21/8/2025, 8:39:55
Evaluado por: Sistema de Evaluación Masiva

Resumen de Calificaciones

Calificación general: 4.3/5.0
Actividades completadas: 18/20
Porcentaje de completitud: 90.0%

Detalle de Actividades

#	Descripción	Archivo	Encontrado	Calificación
1	Usando un ciclo for, imprime los números...	src/ejercicio_01.py	Sí	5.0
2	Mediante un ciclo while, imprime los núm...	src/ejercicio_02.py	Sí	5.0
3	Con un ciclo for, calcula la suma de tod...	src/ejercicio_03.py	No	0.0
4	Utilizando un ciclo while, solicita al u...	src/ejercicio_04.py	Sí	5.0
5	Con un ciclo for, imprime la tabla de mu...	src/ejercicio_05.py	No	0.0
6	Mediante un ciclo while, genera y muestr...	src/ejercicio_06.py	Sí	5.0
7	Con un ciclo for, cuenta cuántas letras ...	src/ejercicio_07.py	Sí	5.0
8	Usando un ciclo while, calcula y muestra...	src/ejercicio_08.py	Sí	5.0
9	Con un ciclo for, imprime todos los núme...	src/ejercicio_09.py	Sí	5.0
10	Mediante un ciclo while, solicita al usu...	src/ejercicio_10.py	Sí	5.0
11	Con un ciclo for, imprime cada carácter ...	src/ejercicio_11.py	Sí	5.0
12	Utilizando un ciclo while, calcula el fa...	src/ejercicio_12.py	Sí	5.0
13	Con un ciclo for, imprime los números de...	src/ejercicio_13.py	Sí	5.0
14	Mediante un ciclo while, implementa un j...	src/ejercicio_14.py	Sí	5.0
15	Con un ciclo for, imprime un triángulo r...	src/ejercicio_15.py	Sí	5.0
16	Utilizando un ciclo while, simula un rel...	src/ejercicio_16.py	Sí	5.0
17	Con un ciclo for, solicita al usuario qu...	src/ejercicio_17.py	Sí	5.0
18	Mediante un ciclo while, genera y muestr...	src/ejercicio_18.py	Sí	4.0
19	Con un ciclo for, cuenta cuántas vocales...	src/ejercicio_19.py	Sí	4.0
20	Utilizando un ciclo while, solicita al u...	src/ejercicio_20.py	Sí	2.0

Retroalimentación Detallada

Actividad 1: Usando un ciclo for, imprime los números enteros del 0 al 9, cada uno en una línea.

Archivo esperado: src/ejercicio_01.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y concisa. El código es legible y cumple con el objetivo planteado de manera eficiente.

Actividad 2: Mediante un ciclo while, imprime los números enteros del 10 al 1 en orden descendente, cada número en una línea.

Archivo esperado: src/ejercicio_02.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta, clara y eficiente. Cumple con todos los requisitos del problema y utiliza buenas prácticas de programación.

Actividad 3: Con un ciclo for, calcula la suma de todos los enteros del 1 al 100 (inclusive) y muestra el resultado.

Archivo esperado: src/ejercicio_03.py

Estado: Archivo no encontrado

Calificación: 0.0/5.0

Retroalimentación:

Error al evaluar: got status: 503 . {"error":{"code":503,"message":"The model is overloaded. Please try again later.","status":"UNAVAILABLE"}}

Actividad 4: Utilizando un ciclo while, solicita al usuario que ingrese números. El proceso termina cuando el usuario escriba 0. Al final, muestra la suma total de todos los números ingresados.

Archivo esperado: src/ejercicio_04.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y eficiente. El código es legible y cumple con los requisitos del problema. Buen trabajo.

Actividad 5: Con un ciclo for, imprime la tabla de multiplicar del 7, es decir, 7 × 1, 7 × 2, …, 7 × 10, cada resultado en una línea.

Archivo esperado: src/ejercicio_05.py

Estado: Archivo no encontrado

Calificación: 0.0/5.0

Retroalimentación:

Error al evaluar: got status: 503 . {"error":{"code":503,"message":"The model is overloaded. Please try again later.","status":"UNAVAILABLE"}}

Actividad 6: Mediante un ciclo while, genera y muestra los primeros 15 múltiplos de 3, comenzando desde 3.

Archivo esperado: src/ejercicio_06.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y eficiente. El código es limpio, fácil de entender y cumple con los requisitos de la actividad. ¡Excelente trabajo!

Actividad 7: Con un ciclo for, cuenta cuántas letras 'a' (minúscula) hay en la cadena texto = "manzana" y muestra el total.

Archivo esperado: src/ejercicio_07.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

¡Excelente solución! El código es claro, conciso y resuelve el problema correctamente. Se adhiere a las buenas prácticas.

Actividad 8: Usando un ciclo while, calcula y muestra los cuadrados de los números del 1 al 20 (1^2, 2^2, …, 20^2), cada resultado en una línea.

Archivo esperado: src/ejercicio_08.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y eficiente. El código es legible y cumple con todos los requisitos del ejercicio.

Actividad 9: Con un ciclo for, imprime todos los números pares del 2 al 50 (ambos inclusive), cada número en una línea.

Archivo esperado: src/ejercicio_09.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es conciso, eficiente y cumple con todos los requisitos del ejercicio. El uso de `range(2, 51, 2)` es la forma más elegante de resolverlo.

Actividad 10: Mediante un ciclo while, solicita al usuario que escriba palabras. El proceso termina cuando el usuario escriba la palabra "fin". Al final, muestra cuántas palabras se leyeron (sin contar "fin").

Archivo esperado: src/ejercicio_10.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y cumple con los requisitos. El código es limpio y fácil de entender, aplicando buenas prácticas como usar `.lower()` para hacer la comparación insensible a mayúsculas.

Actividad 11: Con un ciclo for, imprime cada carácter de la palabra "python" en una línea separada.

Archivo esperado: src/ejercicio_11.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

¡Excelente! La solución es correcta, concisa y cumple con todos los requisitos del ejercicio. El código es limpio y fácil de entender.

Actividad 12: Utilizando un ciclo while, calcula el factorial de un número entero n introducido por el usuario y muestra el resultado.

Archivo esperado: src/ejercicio_12.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y eficiente. El código es claro, bien estructurado y cumple con el objetivo de calcular el factorial utilizando un ciclo while.

Actividad 13: Con un ciclo for, imprime los números del 1 al 30 saltando de 3 en 3 (1, 4, 7, …, 28), cada número en una línea.

Archivo esperado: src/ejercicio_13.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es conciso, claro y cumple con los requisitos del problema utilizando la función `range` de forma correcta para generar la secuencia deseada.

Actividad 14: Mediante un ciclo while, implementa un juego de adivinanza: el programa genera un número aleatorio del 1 al 10 y solicita al usuario que lo adivine. El proceso se repite hasta que el usuario acierte. Muestra un mensaje de felicitación al final.

Archivo esperado: src/ejercicio_14.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

¡Excelente solución! El código resuelve correctamente el problema planteado, es legible y bien estructurado. Se puede mejorar la experiencia de usuario añadiendo pistas (mayor/menor) al jugador.

Actividad 15: Con un ciclo for, imprime un triángulo rectángulo de 5 filas usando el carácter '*'.

Archivo esperado: src/ejercicio_15.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

¡Excelente solución! El código es conciso, correcto y cumple con la descripción de la actividad. Buena utilización del ciclo for y la multiplicación de cadenas.

Actividad 16: Utilizando un ciclo while, simula un reloj digital que muestre cada segundo desde 00:00 hasta 00:59 en formato MM:SS, cada valor en una línea.

Archivo esperado: src/ejercicio_16.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 simular el reloj digital, incluyendo el uso de time.sleep(1) para simular el paso del tiempo.

Actividad 17: Con un ciclo for, solicita al usuario que ingrese un número entero positivo y calcula la suma de sus dígitos, mostrando el resultado final.

Archivo esperado: src/ejercicio_17.py

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código resuelve correctamente el problema, es legible y aplica buenas prácticas como la validación de dígitos. ¡Muy bien!

Actividad 18: Mediante un ciclo while, genera y muestra la secuencia de Fibonacci empezando por 1, 1, 2, 3, 5, … y termina cuando se alcance el primer valor mayor que 1000.

Archivo esperado: src/ejercicio_18.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Se puede mejorar la legibilidad inicializando numero2 con 1 para seguir estrictamente la secuencia de Fibonacci descrita (1, 1, 2, 3...). La condición `if numero1 > 1000` dentro del bucle es redundante.

Actividad 19: Con un ciclo for, cuenta cuántas vocales (sin distinción de mayúsculas/minúsculas) hay en la frase frase = "programacion es divertida" y muestra el total.

Archivo esperado: src/ejercicio_19.py

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

El código funciona correctamente y cuenta las vocales. Sin embargo, la frase utilizada en el código no coincide con la descripción de la actividad. Se debe utilizar la frase 'programacion es divertida' para cumplir con los requisitos de la tarea.

Actividad 20: Utilizando un ciclo while, solicita al usuario que ingrese edades una a una. El proceso termina cuando se introduzca -1. Al final, muestra la edad mayor que se haya ingresado.

Archivo esperado: src/ejercicio_20.py

Estado: Archivo encontrado

Calificación: 2.0/5.0

Retroalimentación:

El código tiene errores lógicos. No itera correctamente para obtener múltiples edades y siempre muestra solo la primera edad ingresada, además de que la condición `edad_mayor = edad` dentro del bucle solo actualiza si la edad es válida, pero no compara si es la mayor hasta el momento, y el bucle se rompe inmediatamente después de la primera edad válida. El mensaje de 'No se ingresó una edad válida' es incorrecto.

Resumen General

Excelente trabajo. Completó 18/20 actividades (90%) con una calificación promedio de 4.3/5. Demuestra buen dominio de los conceptos.

Recomendaciones

- Completar los archivos faltantes: src/ejercicio_03.py, src/ejercicio_05.py
- Revisar y mejorar las actividades con calificación baja