Reporte de Evaluación - Fork de GitHub

Información General

Estudiante: Joel Stiven Mariños Rivera
Repositorio: JoelSMR/act_ntp_s3
Fecha de evaluación: 21/8/2025, 12:35:50
Evaluado por: Sistema de Evaluación de No Calificados

Resumen de Calificaciones

Calificación general: 4.5/5.0
Actividades completadas: 20/20
Porcentaje de completitud: 100.0%

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	Sí	5.0
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	Sí	5.0
6	Mediante un ciclo while, genera y muestr...	src/ejercicio_06.py	Sí	3.0
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	3.0
9	Con un ciclo for, imprime todos los núme...	src/ejercicio_09.py	Sí	4.0
10	Mediante un ciclo while, solicita al usu...	src/ejercicio_10.py	Sí	3.0
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	4.0
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	4.0
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	5.0
19	Con un ciclo for, cuenta cuántas vocales...	src/ejercicio_19.py	Sí	5.0
20	Utilizando un ciclo while, solicita al u...	src/ejercicio_20.py	Sí	4.0

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, clara y concisa. Cumple con todos los requisitos de la actividad, utilizando un ciclo for 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:

Excelente solución. El código es conciso, legible y cumple con el objetivo de la actividad de manera eficiente. No hay aspectos significativos a mejorar.

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 encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta, concisa y fácil de entender. El código es limpio y sigue las buenas prácticas para este tipo de problema.

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 funcional. El código es limpio y fácil de entender, cumple con los requisitos de la actividad.

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 encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es conciso, legible y cumple con el objetivo de la actividad utilizando un bucle for y formateo de strings adecuadamente.

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: 3.0/5.0

Retroalimentación:

El código genera los múltiplos de 3 correctamente, pero incluye una impresión adicional dentro del bucle que no es necesaria y afecta la claridad. Debes enfocarte en la concisión del código.

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:

La solución es correcta, clara y concisa. El código es legible y cumple con el objetivo propuesto sin problemas.

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: 3.0/5.0

Retroalimentación:

La solución calcula e imprime los cuadrados correctamente, pero imprime incorrectamente un mensaje adicional dentro del bucle. Deberías quitar la línea `print(f'El cuadrado de {i} es: {cuadrado}')`.

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: 4.0/5.0

Retroalimentación:

La solución es correcta e imprime los números pares del 2 al 50. Sin embargo, la impresión adicional dentro del ciclo `print(f"El número par en la posición {i // 2} es: {i}")` no estaba solicitada en la descripción del problema y reduce la claridad del output. Remueve esta línea para ajustarte a la consigna.

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: 3.0/5.0

Retroalimentación:

El código funciona correctamente, pero la impresión del total de palabras dentro del bucle es redundante y confusa. Debería moverse la impresión del total fuera del bucle para una mejor claridad.

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:

La solución es correcta, concisa y cumple con el objetivo de la actividad. 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 limpio, legible 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:

La solución es correcta, concisa y eficiente. Utiliza el ciclo `for` con el `range` de manera adecuada 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: 4.0/5.0

Retroalimentación:

La lógica del juego es correcta y funciona como se espera. Sin embargo, el mensaje de felicitación se imprime dentro del ciclo while, lo que hace que se muestre antes de tiempo. Mueve el `print("¡Felicidades! Adivinaste el número.")` fuera del bucle para que se ejecute solo cuando se adivine el número.

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:

La solución es correcta, concisa y cumple con el objetivo. El código es legible y eficiente.

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: 4.0/5.0

Retroalimentación:

La solución cumple con el objetivo principal. Considera añadir validaciones adicionales para cubrir todos los casos posibles y manejar errores, además de documentar el código.

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 es conciso, legible y cumple con todos los requisitos de la actividad. ¡Bien hecho!

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: 5.0/5.0

Retroalimentación:

La solución es correcta, eficiente y legible. El código genera la secuencia de Fibonacci hasta el primer número mayor que 1000 usando un ciclo `while` y asignación múltiple de manera concisa.

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: 5.0/5.0

Retroalimentación:

La solución es correcta y eficiente. El código es limpio, legible y cumple con el objetivo planteado.

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: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Podría mejorarse evitando la asignación inicial de `mayor = 0` y manejar el caso de la entrada -1 de forma más elegante para evitar que -1 sea considerada la edad mayor si es la primera entrada. También podrías considerar usar un `try-except` para manejar errores si el usuario ingresa algo que no es un número.

Resumen General

Excelente trabajo. Completó 20/20 actividades (100%) con una calificación promedio de 4.5/5. Demuestra buen dominio de los conceptos.

Recomendaciones

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