Tu calificación: 100 %

Tu calificación más reciente: **100 %** • Tu calificación más alta: **100 %** • Para aprobar necesitas al menos un 80 %. Guardamos tu puntaje más alto.

Próximo artículo

Estás viendo una versión traducida automáticamente de esta evaluación
Puedes volver a ver el contenido en su idioma original si lo prefieres. No perderás el progreso que hayas conseguido si cambias el idioma.
Mostrar la versión en Inglés
Desestimar

1. En Python, ¿qué hacen los bucles `while`? **1 / 1 punto**

- (•) `while` los bucles indican al ordenador que ejecute repetidamente un conjunto de instrucciones mientras una condición sea cierta.
- ( ) `while` los bucles ordenan al ordenador que ejecute un fragmento de código un número determinado de veces.
- ( ) `while` los bucles ramifican la ejecución en función de si una condición es cierta o no.
- ( ) `while` los bucles inicializan variables en Python.

Correcto
Correcto. `while` los bucles seguirán ejecutando el mismo grupo de instrucciones hasta que una condición de bucle especificada deje de ser cierta.

2. ¿Qué técnicas pueden evitar un bucle infinito `while`? Seleccione todas las que correspondan. **1 / 1 punto**

- [x] Cambiar el valor de una variable utilizada en la condición `while`

Correcto
Correcto. Si se utiliza una variable numérica para controlar las iteraciones de una condición `while`, esa variable debe cambiar finalmente a un valor que haga que la condición `while` sea Falsa, de lo contrario el bucle `while` seguirá ejecutándose indefinidamente.

- [ ] Utilice la palabra clave `stop`
- [x] Utilice la palabra clave `break`

Correcto
Correcto. La palabra clave `break` puede detener un bucle `while`. A menudo se utiliza en el cuerpo de una sentencia condicional anidada `if-else` dentro de un bucle `while`.

- [ ] Utilice la palabra clave continuar

3. El siguiente código contiene un bucle infinito, lo que significa que el intérprete de Python no sabe cuándo salir del bucle una vez completada la tarea. Para resolver el problema, tendrá que **1 / 1 punto**

1. Encontrar el error en el código
2. Arreglar el bucle while para que haya una condición de salida

Sugerencia: Pruebe a ejecutar su función con el número `0` como entrada y observe el resultado.

Tenga en cuenta que los bloques de código de Coursera agotarán el tiempo de espera después de 5 segundos de ejecutar un bucle infinito. Si obtiene este mensaje de error de tiempo de espera, significa que el bucle infinito no se ha solucionado.

```
def is_power_of_two(number):
  # This while loop checks if the "number" can be divided by two
  # without leaving a remainder. How can you change the while loop to
  # avoid a Python ZeroDivisionError?
  if number <=0:
    return False

  while number % 2 == 0 and number >= 1:
    number = number / 2
  # If after dividing by 2 "number" equals 1, then "number" is a power
  # of 2.
  if number == 1:
    return True
  return False

# Calls to the function
print(is_power_of_two(0)) # Should be False
print(is_power_of_two(1)) # Should be True
print(is_power_of_two(8)) # Should be True
print(is_power_of_two(9)) # Should be False
```

Ejecutar
Restablecer

Correcto
¡Impresionante! Ha solucionado un error difícil de encontrar y la función se comporta ahora correctamente.

4. Escriba una función que tome un argumento n y devuelva la suma de enteros de 1 a n. Por ejemplo, si n=5, su función debería sumar 1+2+3+4+5 y devolver 15. Si n es menor que 1, simplemente devuelve 0. Utilice un bucle `while` para calcular esta suma. **1 / 1 punto**

```
def sum_of_integers(n):
  if n < 1:
    return 0


  i = 1
  sum = 0
  while i <= n:
    sum = sum + i
    i = i + 1


  return sum


print(sum_of_integers(3))  # should print 6
print(sum_of_integers(4))  # should print 10
print(sum_of_integers(5))  # should print 15
```

Ejecutar
Restablecer

Correcto
¡Buen trabajo! Ha escrito un bucle while y ha conseguido que Python haga el trabajo por ti.

5. Rellene los espacios en blanco para completar la función, que debe dar como resultado una tabla de multiplicar. La función acepta una variable "número" a través de sus parámetros. Esta variable "número" debe multiplicarse por los números del 1 al 5, e imprimirse en un formato similar a "1x6=6" ("número" x "multiplicador" = "resultado"). El código también debe limitar el "resultado" para que no exceda de 25. Para satisfacer estas condiciones, deberá: **1 / 1 punto**

1. Inicializar la variable "multiplicador" con el valor inicial
2. Completar la condición del bucle `while`
3. Añadir un punto de salida para el bucle
4. Incrementar la variable "multiplicador" dentro del bucle `while`

```
def multiplication_table(number):
    # Initialize the appropriate variable
    multiplier = 1


    # Complete the while loop condition.
    while multiplier <= 5:
        result = number * multiplier
        if  result > 25:
            # Enter the action to take if the result > 25
            break
        print(str(number) + "x" + str(multiplier) + "=" + str(result))

        # Increment the appropriate variable
        multiplier+= 1



multiplication_table(3)
# Should print:
# 3x1=3
# 3x2=6
# 3x3=9
# 3x4=12
# 3x5=15


multiplication_table(5)
# Should print:
# 5x1=5
# 5x2=10
# 5x3=15
# 5x4=20
# 5x5=25


multiplication_table(8)
# Should print:
# 8x1=8
```

Ejecutar
Restablecer

Correcto
Correcto. Ha completado la tabla de multiplicar con todos los criterios requeridos, ¡y queda muy bien!