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

1. Cuando un usuario informa de que una "aplicación no funciona", ¿cuál es una pregunta de seguimiento adecuada para recabar más información sobre el problema? 1 / 1 punto

- ¿Está enchufado el servidor?
- ¿Por qué necesita la aplicación?
- ¿Tiene un número de ticket de asistencia?
- (•) ¿Qué debería ocurrir al abrir la aplicación?

Correcto
¡Impresionante! Preguntar al usuario cuál debería ser el resultado esperado le ayudará a reunir más información para comprender y aislar el problema.

2. ¿Qué es un heisenbug? 1 / 1 punto

- (•) El efecto observador.
- Un entorno de pruebas.
- La causa principal.
- Un visor de eventos.

Correcto
Muy cierto El efecto observador se produce cuando la mera observación de un fenómeno altera el fenómeno.

3. Se supone que la función `compare_strings` compara sólo el contenido alfanumérico de dos cadenas, ignorando mayúsculas frente a minúsculas y la puntuación. Pero hay algo que no funciona. Rellene el código para tratar de encontrar los problemas y, a continuación, arréglelos. Su objetivo es buscar el carácter "-", pero - suele reservarse para los rangos. Encuentre una solución que le permita comparar las dos cadenas. 1 / 1 punto

```python
import re
def compare_strings(string1, string2):
  #Convert both strings to lowercase
  #and remove leading and trailing blanks
  string1 = string1.lower().strip()
  string2 = string2.lower().strip()

  #Ignore punctuation
  ## punctuation = r"[.?!,;:-']"
  punctuation = r"[.?!,;:'-]"
  string1 = re.sub(punctuation, r"", string1)
  string2 = re.sub(punctuation, r"", string2)

  #DEBUG CODE GOES HERE
  """
  change r"[.?!,;:-']" with r"[.?!,;:'-]" in punctuation variable
  because of pattern error (Character range is out of order ('-' pattern))
  """

  return string1 == string2

print(compare_strings("Have a Great Day!", "Have a great day?")) ## True
print(compare_strings("It's raining again.", "its raining, again")) ## True
print(compare_strings("Learn to count: 1, 2, 3.", "Learn to count: one, two, three.")) ## False
print(compare_strings("They found some body.", "They found somebody.")) ## False
```

Ejecutar
Restablecer

Correcto
¡Buen trabajo! ¡Estos bichos no tienen ninguna oportunidad contigo cerca!

4. ¿Cómo podemos verificar si un problema persiste o no? 1 / 1 punto

- Reinicie el dispositivo o el hardware del servidor
- (•) Intente desencadenar de nuevo el problema siguiendo los pasos de nuestro caso de reproducción
- Preguntar repetidamente al usuario
- Vuelva a comprobarlo más tarde

Correcto
¡Woohoo! Si podemos recrear las circunstancias de la incidencia, podremos verificar si el problema sigue produciéndose.

5. El módulo datetime proporciona clases para manipular fechas y horas, y contiene muchos tipos, objetos y métodos. Ya ha visto utilizar algunos de ellos en la función dow, que devuelve el día de la semana para una fecha concreta. Los utilizaremos de nuevo en la función `next_date`, que toma el parámetro `date_string` en el formato "año-mes-día", y utiliza la función `add_year` para calcular el próximo año en que se producirá esta fecha (es 4 años más tarde para el 29 de febrero durante el año bisiesto, y 1 año más tarde para todas las demás fechas). Después devuelve el valor en el mismo formato en el que recibe la fecha: "año-mes-día". 1 / 1 punto

¿Puede encontrar el error en el código? ¿Está en la función `next_date` o en la función `add_year`? ¿Cómo puede determinar si la función `add_year` devuelve lo que se supone que debe devolver? Añada las líneas de depuración que sean necesarias para encontrar los problemas y, a continuación, corrija el código para que funcione como se indica más arriba.

```python
import datetime
from datetime import date

def add_year(date_obj):
  try:
    new_date_obj = date_obj.replace(year = date_obj.year + 1)
  except ValueError:
    ## This gets executed when the above method fails,
    ## which means that we're making a Leap Year calculation
    new_date_obj = date_obj.replace(year = date_obj.year + 4)
  return new_date_obj ## OK

def next_date(date_string):
  ## Convert the argument from string to date object
  date_obj = datetime.datetime.strptime(date_string, r"%Y-%m-%d")
  next_date_obj = add_year(date_obj)
  ## print(f'{date_obj} | {next_date_obj}') ## OK

  ## Convert the datetime object to string,
  ## in the format of "yyyy-mm-dd"
  ## next_date_string = next_date_obj.strftime("yyyy-mm-dd")
  next_date_string = next_date_obj.strftime("%Y-%m-%d")
  return next_date_string

today = date.today()  ## Get today's date
print(next_date(str(today)))
## Should return a year from today, unless today is Leap Day

print(next_date("2021-01-01")) ## Should return 2022-01-01
print(next_date("2020-02-29")) ## Should return 2024-02-29
```

Ejecutar
Restablecer

Correcto
¡Excelente! Depurar múltiples funciones es más difícil
que trabajar con una sola función, ¡y usted lo ha conseguido!