✓ **Congratulations! You passed!**
TO PASS 80% or higher

Keep Learning

GRADE
100%

Practice Quiz: Understanding the Problem

TOTAL POINTS 5

1. When a user reports that an "application doesn't work," what is an appropriate follow-up question to gather more information about the problem?

    ○ Is the server plugged in?

    ○ Why do you need the application?

    ○ Do you have a support ticket number?

    ⦿ What should happen when you open the app?

    ✓ **Correct**
    Awesome! Asking the user what an expected result should be will help you gather more information to understand and isolate the problem.

    1 / 1 point

2. What is a heisenbug?

    ⦿ The observer effect.

    ○ A test environment.

    ○ The root cause.

    ○ An event viewer.

    ✓ **Correct**
    Right on! The observer effect is when just observing a phenomenon alters the phenomenon.

    1 / 1 point

3. The compare_strings function is supposed to compare just the alphanumeric content of two strings, ignoring upper vs lower case and punctuation. But something is not working. Fill in the code to try to find the problems, then fix the problems.

    1 / 1 point

    ```
    1   import re
    2 ▾ def compare_strings(string1, string2):
    3       #Convert both strings to lowercase
    4       #and remove leading and trailing blanks
    5       string1 = string1.lower().strip()
    6       string2 = string2.lower().strip()
    7
    8       #Ignore punctuation
    9       punctuation = r"[.?!,;:\-']"
    10      string1 = re.sub(punctuation, r"", string1)
    11      string2 = re.sub(punctuation, r"", string2)
    12
    13      #DEBUG CODE GOES HERE
    14      print(punctuation)
    15
    16      return string1 == string2
    17
    18  print(compare_strings("Have a Great Day!", "Have a great day?")) # True
    19  print(compare_strings("It's raining again.", "its raining, again")) # True
    20  print(compare_strings("Learn to count: 1, 2, 3.", "Learn to count: one, two,
            three.")) # False
    21  print(compare_strings("They found some body.", "They found somebody.")) # False
    ```

    Run

    Reset

    ```
    [.?!,;:\-']
    True
    [.?!,;:\-']
    True
    [.?!,;:\-']
    False
    [.?!,;:\-']
    False
    ```

    ✓ **Correct**
    Great job! These bugs don't stand a chance with you around!

4. How do we verify if a problem is still persisting or not?

    1 / 1 point

    ○ Restart the device or server hardware

    ⦿ Attempt to trigger the problem again by following the steps of our reproduction case

    ○ Repeatedly ask the user

    ○ Check again later

5. The datetime module supplies classes for manipulating dates and times, and contains many types, objects, and methods. You've seen some of them used in the dow function, which returns the day of the week for a specific date. We'll use them again in the next_date function, which takes the date_string parameter in the format of "year-month-day", and uses the add_year function to calculate the next year that this date will occur (it's 4 years later for the 29th of February during Leap Year, and 1 year later for all other dates). Then it returns the value in the same format as it receives the date: "year-month-day".

`1 / 1 point`

Can you find the error in the code? Is it in the next_date function or the add_year function? How can you determine if the add_year function returns what it's supposed to? Add debug lines as necessary to find the problems, then fix the code to work as indicated above.

```python
1   import datetime
2   from datetime import date
3
4   def add_year(date_obj):
5       try:
6           new_date_obj = date_obj.replace(year = date_obj.year + 1)
7       except ValueError:
8           # This gets executed when the above method fails,
9           # which means that we're making a Leap Year calculation
10          new_date_obj = date_obj.replace(year = date_obj.year + 4)
11      return new_date_obj
12
13  def next_date(date_string):
14      # Convert the argument from string to date object
15      date_obj = datetime.datetime.strptime(date_string, r"%Y-%m-%d")
16      next_date_obj = add_year(date_obj)
17
18      # Convert the datetime object to string,
19      # in the format of "yyyy-mm-dd"
20      next_date_string = next_date_obj.strftime("%Y-%m-%d")
21      return next_date_string
22
23  today = date.today()  # Get today's date
24  print(next_date(str(today)))
25  # Should return a year from today, unless today is Leap Day
26
27  print(next_date("2021-01-01")) # Should return 2022-01-01
28  print(next_date("2020-02-29")) # Should return 2024-02-29
```

```
2021-04-23
2022-01-01
2024-02-29
```

[Run]

[Reset]