← Back **Practice Quiz: Understanding the Problem**
Practice Quiz • 10 min • 5 total points

Congratulations! You passed!

Grade received 100% To pass 80% or higher

Go to next item

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? **1 / 1 point**

- 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.

2. What is a heisenbug? **1 / 1 point**

- [✓ 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.

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**

```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"[.?!,;:\-']"
  string1 = re.sub(punctuation, "", string1)
  string2 = re.sub(punctuation, "", string2)

  #DEBUG CODE GOES HERE
  print(f"string1: {string1}")
  print(f"string2: {string2}")

  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
```

Run

Reset

```
string1: have a great day
string2: have a great day
True
string1: its raining again
string2: its raining again
True
string1: learn to count 1 2 3
string2: learn to count one two three
False
string1: they found some body
string2: they found somebody
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

Correct

Woohoo! If we can recreate the circumstances of the issue, we can verify whether the problem continues to occur.

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
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

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)

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

  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
```

Run

Reset

```
2024-01-30
2022-01-01
2024-02-29
```

Correct

Excellent! Debugging multiple functions is more challenging
than working with just one function, and you've done it!