If `person.name` is not a string, then Python will complain of a type error when it tries to concatenate it with other strings. This is one difference between Python and Java -- Python insists that you use a conversion operation like `str()`, whereas Java will automatically convert any type into a `String` when you try to concatenate it with another `String`.

Submit

Answers are displayed within the problem

documenting assumptions, part 2

1/1 point (graded)

If you were writing Java instead of Python, and your Java code needed to make *all* the assumptions below, then which of them could be documented by type declarations and statically checked by the Java compiler?

- [x] `person` must be an object with `age` and `name` instance variables
- [ ] `person` is not `null`
- [ ] `person.age` must be a nonnegative number
- [x] `person.age` must be an integer
- [x] `person.name` must be a string

✓

Explanation

The `person` variable would be declared with some class type, perhaps called `Person`, and the definition of that class would have instance variables `name` and `age` declared with types `String` and `int` respectively.

But we can't use a type declaration to forbid `person` from being `null`. Any object reference might be `null` in Java, just like any variable might be `None` in Python. Similarly, we can't forbid `age` from being negative using a type declaration. These assumptions would have to be documented in comments instead.

Submit