Questions | Reading 1: Static Checking | 6.005.1x Courseware

1. Course , current location
2. Progress

Questions

reading the hailstone code

1/1 point (graded)

```
n = 4
while n != 1:
    if n % 2 == 0: # if n is even...
        n = n / 2
    else:
        n = 3 * n + 1
    print n
```

What is the first number that this code will print?

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

programming terminology

1/1 point (graded)

In the program, what kind of a thing is `n % 2 == 0`?

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Questions | Reading 1: Static Checking | 6.005.1x Courseware

openlearninglibrary.mit.edu/courses/course-v1:MITx+6.005.1x+3T2016/courseware/Readings_Videos/01-Static-Checking

1. Course , current location
2. Progress

Questions

how types affect execution

1/1 point (graded)

```
# assume Python 2
data = [ 2, 4, 6 ]
total = 0
average = 0
n = 0
for value in data:
    n += 1
    total += value
    average = total / n
    print "average:", average
```

What averages are printed?

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

how types affect execution, part 2

1/1 point (graded)

Here is the same program, with different starting values for the data list.

```
# assume Python 2
data = [ 1, 2, 3 ]
total = 0
average = 0
n = 0
for value in data:
    n += 1
    total += value
    average = total / n
    print "average:", average
```

Now what averages are printed?

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

how types affect execution, part 3

1/1 point (graded)

Here is the same program one more time, again with different starting values for the data list.

```
# assume Python 2
data = [ "1", "2", "3" ]
total = 0
average = 0
n = 0
for value in data:
    n += 1
    total += value
    average = total / n
    print "average:", average
```

Now what averages are printed?

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Questions | Reading 1: Static Checking | 6.005.1x Courseware

Questions

kinds of error checking

1/1 point (graded)

Let's try some examples of buggy code and see how they behave in Java.

Are these bugs caught statically, dynamically, or not at all?

```
int n = 5;
if (n) {
  n = n + 1;
}
```

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

kinds of error checking, part 2

1/1 point (graded)

```
int big = 200000; // 200,000
big = big * big;  // big should be 4 billion now
```

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

kinds of error checking, part 3

1/1 point (graded)

```
double probability = 1/5;
```

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

kinds of error checking, part 4

1/1 point (graded)

```
int sum = 0;
int n = 0;
int average = sum/n;
```

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

kinds of error checking, part 5

1/1 point (graded)

```
double sum = 7;
double n = 0;
double average = sum/n;
```

correct

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Questions | Reading 1: Static Checking | 6.005.1x Courseware

openlearninglibrary.mit.edu/courses/course-v1:MITx+6.005.1x+3T2016/courseware/Readings_Videos/01-Static-Checking

Questions

list vs. array

1/1 point (graded)

Rewrite these variable declarations using Lists instead of arrays.

We're only declaring the variables, not initializing them with any value.

correct
List<String> names **or** List<String> names;

Explanation

(edX might not display the answer correctly, it's List<String> names)

The translation from String[] to List<String> is pretty straightforward in Java.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

list vs. array, part 2

1/1 point (graded)

correct
List<Integer> numbers **or** List<Integer> numbers;

Explanation

(edX might not display the answer correctly, it's List<Integer> numbers)

We can create arrays of primitive types, but not Lists. Use the Integer wrapper.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

list vs. array, part 3

1/1 point (graded)

correct
List<List<Character>> grid **or** List<List<Character>> grid;

Explanation

(edX might not display the answer correctly, it's List<List<Character>> grid)

There's nothing wrong with a List<List<Character>> -- but if this is a fixed-size grid, it might be simpler to use a 2-dimensional array instead of a list-of-lists.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

x marks the spot

3/3 points (graded)

Java Maps work like Python dictionaries.

After we run this code:

```
Map<String, Double> treasures = new HashMap<>();
String x = "palm";
treasures.put("beach", 25.);
treasures.put("palm", 50.);
treasures.put("cove", 75.);
treasures.put("x", 100.);
treasures.put("palm", treasures.get("palm") + treasures.size());
treasures.remove("beach");
double found = 0;
for (double treasure : treasures.values()) {
    found += treasure;
}
```

What is the value of...

treasures.get(x)

correct
54

54

treasures.get("x")

correct
100

100.

found

correct
229

229

Explanation

After the first four `put()` calls, the map has stored the pairs ("beach", 25), ("palm", 50), ("cove", 75), ("x", 100). The fifth `put()` call adds the size of the map (4) to the entry for `"palm"`, so that entry is now ("palm", 54). Finally the entry for "beach" is removed from the map, so the final state of the map is ("palm", 54), ("cove", 75), ("x", 100).

Now that we know what the map looks like, we can answer the questions. `treasures.get(x)` returns the value stored for the key `"palm"`, which is 54. `treasures.get("x")` returns the value stored for `"x"`, which is 100. Finally, `found` sums up all the values currently stored in the map, which is 54+75+100 = 229.

You can see this code in action in Online Java Tutor.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

Questions | Reading 1: Static Checking | 6.005.1x Courseware

1. Course , current location
2. Progress

Questions

reading javadocs

1/1 point (graded)

After we run this code:

```
Map<String, Integer> treasures = new HashMap<>();
treasures.put("beach", 25);
Integer result = treasures.putIfAbsent("beach", 75);
```

What is result?

correct

To answer this question, you will have to search the web for Map's putIfAbsent method, and read its specification carefully.

Explanation

Good search keywords for finding the Java documentation for this method would be "java Map putIfAbsent". Then, according to the `putIfAbsent` documentation: "If the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, **else returns the current value**."

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

Questions | Reading 1: Static Checking | 6.005.1x Courseware

1. Course , current location
2. Progress

Questions

snapshot diagrams and final

2/2 points (graded)

Suppose we want to add `final` to both variable declarations in the `hailstoneSequence` method, as shown:

```
public static List<Integer> hailstoneSequence(final int n) {
      final List<Integer> list = new ArrayList<Integer>();
      while (n != 1) {
          list.add(n);
          if (n % 2 == 0) {
              n = n / 2;
          } else {
              n = 3 * n + 1;
          }
      }
      list.add(n);
      return list;
  }
```

Which of the following are true statements about putting `final` on `n`?

correct

Which of the following are true statements about putting `final` on `list`?

correct

Explanation

`final` can't be used on `n` because `n` needs to be reassigned in the body of the method. But `final` can indeed be used on `list`.

`final` can be used on both parameters and local variables. When used on a parameter, `final` means that the parameter is assigned when the method is called, and then can't be reassigned during the body of the method. When used on a local variable, `final` means that the variable can't be reassigned after its first assignment, until the variable's scope ends.

`final` can be used on variables of any type -- not just immutable types like int, but also mutable types like `List`. If a `final` variable points to a mutable object, then the variable cannot be reassigned, but the object it points to can still be mutated, say by calling `add()` on a `List`.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

Questions | Reading 1: Static Checking | 6.005.1x Courseware

Questions

documenting assumptions

1/1 point (graded)

Consider the following simple Python function:

```
from math import sqrt
def funFactAbout(person):
  if sqrt(person.age) == person.age:
    print("The age of " + person.name + " is a perfect square: " +
str(person.age))
```

Which of the following are assumptions made by this code, which must be true in order for it to run without errors?

correct

Explanation

If person is not an object (or if it's `None`), then the code will fail as soon as it tries to refer to `person.age`.

If `person.age` is not a number, or if it's a negative number, then `sqrt()` will fail. But it doesn't necessarily need to be an integer, because `sqrt()` can handle that.

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`.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

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?

correct

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.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

Open Learning Library

Give

Course › Readings/Videos › Reading 1: Static Ch... › Questions

Questions

reading the hailstone code

1/1 point (graded)

```
n = 4
while n != 1:
    if n % 2 == 0: # if n is even...
        n = n / 2
    else:
        n = 3 * n + 1
    print n
```

What is the first number that this code will print?

- 1
- [✓ 2]
- 6
- 7

✓

Submit

✓ Correct (1/1 point)

programming terminology