Learning Quiz 21: Random

Due Oct 30 at 1pm **Points** 5 **Questions** 5 **Available** until Dec 4 at 11:59pm **Time Limit** None
Allowed Attempts Unlimited

Instructions

Prior to completing this quiz, be sure to read:

- Section 6.4: Module random (p. 186-190)

Please also go over Practice Problems 6.9 and 6.10 in the textbook (solutions at the end of the chapter) before attempting this quiz.

This quiz was created for learning purposes. You may attempt this quiz as many times as you would like. The highest score *prior to the deadline* will count towards the final course grade. No late submissions will be accepted.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	1 minute	4 out of 5
LATEST	Attempt 2	1 minute	4 out of 5
	Attempt 1	8 minutes	2.5 out of 5

Score for this attempt: **4** out of 5
Submitted Nov 1 at 1:24pm
This attempt took 1 minute.

Question 1

0 / 1 pts

In Chapter 2, we saw examples of the 'math' and 'fractions' modules. In this section, we'll look at the 'random' module, which serves as a *pseudorandom* number generator. Pseudorandom number generators are programs that produce a sequence of numbers that look random, but are not truly random. Truly random numbers are actually very difficult to obtain. In general, pseudorandom number generators are good enough for most applications.

Suppose I wanted to generate a random number between 1 and 10. The following code will help me achieve this task:

```
import random
random.randrange(1,11)
```

Notice that the code above reads uses (1,11) as arguments. That is because random.randrange(a, b) will generate numbers between a and b, including a and excluding b.

Which of the following will generate numbers between -3 and 3?

You Answered

- [✓ random.randrange(-3,3)]
- random.randrange(-4,4)

Correct Answer

- random.randrange(-3,4)
- random.randrange(-4,3)
- -random.randrange(0,4)

Question 2

1 / 1 pts

Again, the 'random' module is only a pseudorandom number generator. The numbers look random, but they are reproducible.

Consider the following code:

```
import random
random.seed(85677665)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
```

At first glance, the numbers look random. However, if we run the exact same code multiple times, we get the exact same numbers.

```
import random
random.seed(85677665)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)

random.seed(85677665)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)

random.seed(85677665)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
random.randrange(1,101)
```

The key here is the "seed()" where we've picked a starting point for our random numbers. Each time we type random.seed(___), we are restarting at that same starting point.

When we randomly generate numbers without setting seed(), we are still starting at some starting point. We just don't know exactly where. This can be both good and bad because it would look random from our point of view. However, if we want to reproduce the same results, we can't.

seed() is typically used when you want to be able to reproduce the exact same results. Reasons for doing so can vary.

Suppose I wanted to generate an random integer between 1 and 500 inclusive of both:

```
import random
random.seed(85677665)
random.randrange(1,501)
```

What number did I get?

Correct!

148

Be cause we used random.seed(), everyone should get the same number.

Correct Answers 148 (with margin: 0)

Question 3

1 / 1 pts

To generate a decimal number between a and b use the *uniform(a, b)* method in random. All numbers between a and b are equally likely.

The following line of code generates a decimal number between 10 and 20.

```
random.uniform(10,20)
```

Among which of the following would it be possible to get 3.1415 as a randomly generated number? Select all.

- ☐ random.randrange(2,4)
- Correct! [✓ random.uniform(3,4)]
- ☐ random.uniform(2,3)
- ☐ random.randrange(0,5)
- ☐ random.uniform(0,3)
- Correct! [✓ random.uniform(1,5)]

Question 4

1 / 1 pts

The random module can also shuffle items in a list:

```
import random

values = [1, 2, 3, 4, 5, 6]
random.shuffle(values)
values
```

As a reminder, modifying a COPY of the list will still modify the original.

```
import random

values = [1, 2, 3, 4, 5, 6]
copy = values
random.shuffle(copy)
values
```

Try the following line of code, which sets seed to 85677665 and then shuffles the list:

```
import random

values = [1, 2, 3, 4, 5, 6, 7, 8, 9]
random.seed(85677665)
random.shuffle(values)
```

What is the order of values now? [6, 4, 1, 9, 2, 7, 8, 3, 5]?

☐ It is not [6, 4, 1, 9, 2, 7, 8, 3, 5]

Correct!

[✓ It is also [6, 4, 1, 9, 2, 7, 8, 3, 5]!]

Question 5

1 / 1 pts

Working off of your list, suppose you don't want to shuffle the order, but you just want to randomly pick numbers from your list. You have two options.

You can use the choice() method, which will pick one item from a list with all items having an equal chance of being picked.

```
random.choice(values)
```

You can also use the sample() method, which will pick n items from a list with all items having an equal chance of being picked with no replacement (e.g. you cannot pick numbers that you've previously picked). The following will pick three items from the list values.

```
random.sample(values, 3)
```

For sampling with replacement, you can always use choice() multiple times.

```
random.choice(values)
random.choice(values)
random.choice(values)
```

Suppose we roll a dice three times. Which of the following lines of code will help us simulate three dice rolls?

Correct!

[✓]
```
die = [1,2,3,4,5,6]
random.choice(die)
random.choice(die)
random.choice(die)
```

[ ]
```
die = [1,2,3,4,5,6]
random.sample(die, 3)
```

[ ]
```
die = [1,2,3,4,5,6]
random.choice(die, 3)
```

Correct!

[✓]
```
die = [1,2,3,4,5,6]
random.sample(die, 1)
random.sample(die, 1)
random.sample(die, 1)
```

Quiz Score: **4** out of 5