# **Hotel Overbooking – Flatiron Course**

## Objectives for this course

* Define a real world problem to solve with Python
* Understand the Bernoulli experiment
* Learn about a binomial distribution and apply it to the problem
* Use the above information to help determine the optimal number of rooms to overbook

The travel and hospitality industries have become reliant on technology to help them make and keep their operations profitable. For hotel owners and managers one thing they work towards, everyday, is to fill as many rooms as possible while getting the best price. In this course, we will explore one method to help maximize profitability by optimizing the booking of rooms in a hotel using Python.

# **THE PROBLEM**

You are the owner of a hotel with 100 rooms, and you would like to maximize your revenue for these rooms. In order to determine the maximum revenue, you first need to establish what information you know about your hotel. You take a look at the books and discovered the following information:

* You have 100 rooms in the hotel that can be sold.
* Each room costs $220 USD per night
* On an average night, you know that 8% of your hotel guests who book a room don’t show up.
* If overbooked, we have to place customers at another hotel, which costs us $400 USD.

Using all of this information, we need to determine how many rooms we should overbook in order to maximize expected revenue.

## Develop a game plan

In order to successfully solve this problem, there are a few things we’ll need to learn about. First of all, we’ll need to understand what it means numerically that “on average, 8% of our hotel guests don’t show up”.

It turns out that you can model this with a **binomial distribution**. To understand **the binomial distribution**, an important concept in probability theory which lies at the foundation of data science, you first need to learn a little bit more about the Bernoulli experiment\*.

# **THE BERNOULLI EXPERIMENT**

A Bernoulli experiment is an experiment for which the probability a certain event occurs is  or ; or in other words, the event has two possible outcomes: one event occurring with probability  and the other one with probability .

**EXAMPLE 1**: A classical example of the Bernoulli experiment is flipping a coin. When flipping a coin,  is equal to 0.5, and the probability of heads is 0.5, as well as the probability of tails, which is the other possible event, is .

**EXAMPLE 2**: The probability of scoring a point when being granted a penalty kick in soccer is 0.8. In this case the Bernoulli experiment is whether someone scores or not:  is equal to 0.8, and the probability of not scoring is .

We can use Python to design a Bernoulli experiment. Let’s look at the code we use to generate the Bernoulli experiment that equals tossing a coin:

**import** numpy **as** np

print(np.random.binomial(1, 0.5, 1))

The first line of code imports the Python NumPy library. A library is essentially an open-source reusable chunk of code that you may want to include in your programs / projects. All you need to know is that NumPy is a library widely used in Python, and will help you solve some of the problems you’ll see in this course.

The second line of code uses the random.binomial function in the NumPy library to run a Bernoulli experiment. The function random.binomial takes in three so-called “arguments” here:

* The first argument represents the number of trials - or how many coin flips we’re doing in each experiment
* The second argument represents  or the probability of “success”.
* The third argument represents how many experiments you’re running (if the difference between argument 1 and 3 is not entirely clear at this time, don’t worry - it will become clear later!)

Let’s get into running some code!

**IN JUPYTER NB!**

You’ll see that you either see “1” (= success, say heads) as an output, or “0” (no success, say tails). Let’s say that we consider heads as success, that means running this experiment (or running this code cell) is exactly the same as tossing a coin. Run the code a few times and see how you’ll sometimes get a 1 and sometimes a 0!