lab06

March 24, 2022

Name: Allan Gongora

Section: 0131

1 Lab 6: Randomization

Welcome to Lab 6!

We will go over iteration and simulations, as well as introduce the concept of randomness.

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests.

[1]:
```
pip install gofer-grader
```

```
Requirement already satisfied: gofer-grader in /opt/conda/lib/python3.7/site-
packages (1.1.0)
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (6.1)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (3.0.3)
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-
packages (from gofer-grader) (2.11.2)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-
packages (from jinja2->gofer-grader) (2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
import numpy as np
from datascience import *

from gofer.ok import check
```

Recommended Reading: * Randomness * Iteration * Simulation

1) For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. This can include:
   A) Sentence reponses to questions that ask for an explanation
   B) Numeric responses to multiple choice questions
   C) Programming code

2) Moreover, throughout this lab and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on. Otherwise, you will fail tests that you thought you were passing previously!

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing. Then click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

1.1 1. Nachos and Conditionals

In Python, Boolean values can either be `True` or `False`. We get Boolean values when using comparison operators such as `<` (less than), `>` (greater than), and `==` (equal to). A list of common comparison operators can be found below!

[3]:
```
3 > 1 + 1
```

[3]:
```
True
```

We can even assign the result of a comparison operation to a variable.

[4]:
```
result = 10 / 2 == 5
result
```

[4]:
```
True
```

Arrays are compatible with comparison operators. The output is an array of boolean values.

[5]:
```
make_array(1, 5, 7, 8, 3, -1) > 3
```

[5]:
```
array([False,  True,  True,  True, False, False])
```

Waiting on the dining table just for you is a hot bowl of nachos! Let's say that whenever you take a nacho, it will have cheese, salsa, both, or neither (just a plain tortilla chip).

Using the function `np.random.choice(array_name)`, let's simulate taking nachos from the bowl at random. Start by running the cell below several times, and observe how the results change.

[6]:
```
nachos = make_array('cheese', 'salsa', 'both', 'neither')
np.random.choice(nachos)
```

[6]:
```
'both'
```

Question 1.1 Assume we took ten nachos at random, and stored the results in an array called `ten_nachos` as done below. Find the number of nachos with only cheese using code (do not hardcode the answer).

Hint: Our solution involves a comparison operator and the `np.count_nonzero` method.

[7]:
```
ten_nachos = make_array('neither', 'cheese', 'both', 'both', 'cheese', 'salsa',␣
↪'both', 'neither', 'cheese', 'both')
number_cheese = len(ten_nachos[ten_nachos == "cheese"])
number_cheese
```

[7]: 3

[8]:
```
check('tests/q1_1.py')
```

[8]: <gofer.ok.OKTestsResult at 0x7f3fb2320450>

1.1.1 Conditional Statements

A conditional statement is made up of many lines that allow Python to choose from different alternatives based on whether some condition is true.

Here is a basic example.

```
def sign(x):
    if x > 0:
        return 'Positive'
```

How the function works is if the input `x` is greater than `0`, we get the string `'Positive'` back.

If we want to test multiple conditions at once, we use the following general format.

```
if <if expression>:
    <if body>
elif <elif expression 0>:
    <elif body 0>
elif <elif expression 1>:
    <elif body 1>
...
else:
    <else body>
```

Only one of the bodies will ever be executed. Each `if` and `elif` expression is evaluated and considered in order, starting at the top. As soon as a true value is found, the corresponding body is executed, and the rest of the expression is skipped. If none of the `if` or `elif` expressions are true, then the `else body` is executed. For more examples and explanation, refer to Section 9.1.

Question 1.2 Complete the following conditional statement so that the string `'More please'` is assigned to `say_please` if the number of nachos with cheese in `ten_nachos` is less than `5`. *Hint*: You should not have to reference the variable `ten_nachos`.

[9]:
```
say_please = '?'

if number_cheese < 5:
    say_please = 'More please'

say_please
```

[9]:
```
'More please'
```

[10]:
```
check('tests/q1_2.py')
```

[10]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c52ba90>
```

Question 1.3 Write a function called `nacho_reaction` that returns a string based on the type of nacho passed in as an argument. From top to bottom, the conditions should correspond to: `'cheese'`, `'salsa'`, `'both'`, `'neither'`.

[11]:
```
def nacho_reaction(nacho):
    x = {
        "cheese": "Cheesy!",
        "salsa": "Spicy!",
        "both": "Wow!",
        "neither": "Meh."
    }
    return x[nacho]

spicy_nacho = nacho_reaction('salsa')
spicy_nacho
```

[11]:
```
'Spicy!'
```

[12]:
```
check('tests/q1_3.py')
```

[12]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c3e0450>
```

Question 1.4 Add a column `'Reactions'` to the table `ten_nachos_reactions` that consists of reactions for each of the nachos in `ten_nachos`.

Hint: Use the `apply` method.

[13]:
```
ten_nachos_reactions = Table().with_column('Nachos', ten_nachos)
ten_nachos_reactions["Reactions"] = ten_nachos_reactions.apply(nacho_reaction,␣
↪"Nachos")
ten_nachos_reactions
```

[13]:
```
Nachos  | Reactions
neither | Meh.
cheese  | Cheesy!
```

```
both    | Wow!
both    | Wow!
cheese  | Cheesy!
salsa   | Spicy!
both    | Wow!
neither | Meh.
cheese  | Cheesy!
both    | Wow!
```

[14]:
```
check('tests/q1_4.py')
```

[14]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c3e8d90>
```

Question 1.5 Using code, find the number of `'Wow!'` reactions for the nachos in `ten_nachos_reactions`.

[15]:
```
number_wow_reactions = ten_nachos_reactions.where("Reactions", "Wow!").num_rows
number_wow_reactions
```

[15]:
```
4
```

[16]:
```
check('tests/q1_5.py')
```

[16]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c52b6d0>
```

1.2 2. Simulations and For Loops

Using a `for` statement, we can perform a task multiple times. This is known as iteration. Here, we'll simulate drawing different suits from a deck of cards.

[17]:
```
suits = make_array(" ", " ", " ", " ")

draws = make_array()

repetitions = 6

for i in np.arange(repetitions):
    draws = np.append(draws, np.random.choice(suits))

draws
```

[17]:
```
array([' ', ' ', ' ', ' ', ' ', ' '], dtype='<U32')
```

The unrolled version of this `for` loop can be found below.

[18]:
```
draws = make_array()

draws = np.append(draws, np.random.choice(suits))
```

```
draws = np.append(draws, np.random.choice(suits))
draws = np.append(draws, np.random.choice(suits))
draws = np.append(draws, np.random.choice(suits))
draws = np.append(draws, np.random.choice(suits))
draws = np.append(draws, np.random.choice(suits))

draws
```

[18]:
```
array([' ', ' ', ' ', ' ', ' ', ' '], dtype='<U32')
```

In the example above, the `for` loop appends a random draw to the `draws` array for every number in `np.arange(repetitions)`.

Here's a nice way to think of what we did above. We had a deck of 4 cards of different suits, we randomly drew one card, saw the suit, kept track of it in `draws`, and put the card back into the deck. We repeated this for a total of 6 times without having to repeat code, thanks to the `for` loop. We simulated this experiment using a `for` loop.

Another use of iteration is to loop through a set of values. For instance, we can print out all of the colors of the rainbow.

[19]:
```
rainbow = make_array("red", "orange", "yellow", "green", "blue", "indigo",␣
 ↪"violet")

for color in rainbow:
    print(color)
```

```
red
orange
yellow
green
blue
indigo
violet
```

We can see that the indented part of the `for` loop, known as the body, is executed once for each item in `rainbow`. Note that the name `color` is arbitrary; we could easily have named it something else. The important thing is that we stay consistent throughout the for loop.

[20]:
```
for another_name in rainbow:
    print(another_name)
```

```
red
orange
yellow
green
blue
indigo
violet
```

In general, however, we would like the variable name to be somewhat informative.

Question 2.1 Clay is playing darts. His dartboard contains ten equal-sized zones with point values from 1 to 10. Write code that simulates his total score after 1000 dart tosses. Make sure to use a `for` loop.

Hint: There are three steps to this problem (and most simulations): 1. Deciding the possible values you can take in the experiment (point values in this case) 2. Running through the experiment a certain amount of times (running through 1000 dart tosses, and randomly getting a value per toss in this case) 3. Keeping track of the total information of each time you run through the experiment (the total score in this case)

[21]:
```
from random import choice
```

[22]:
```
possible_point_values = range(1, 11)
tosses = 1000
total_score = sum(choice(possible_point_values) for _ in range(tosses))

total_score
```

[22]:
```
5456
```

[23]:
```
check('tests/q2_1.py')
```

[23]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c50cbd0>
```

Question 2.2 In the following cell, we've loaded the text of *Pride and Prejudice* by Jane Austen, split it into individual words, and stored these words in an array. Using a `for` loop, assign `longer_than_five` to the number of words in the novel that are more than 5 letters long.

Hint: You can find the number of letters in a word with the `len` function.

[24]:
```
austen_string = open('Austen_PrideAndPrejudice.txt', encoding='utf-8').read()
p_and_p_words = np.array(austen_string.split())

longer_than_five = len([i for i in p_and_p_words if len(i) > 5])

# a for loop would be useful here


longer_than_five
```

[24]:
```
35453
```

[25]:
```
check('tests/q2_2.py')
```

[25]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c404f90>
```

Question 2.3 Using simulation with 10,000 trials, assign `chance_of_all_different` to an estimate of the chance that if you pick three words from Pride and Prejudice uniformly at random

(with replacement), they all have different lengths.

Hint: Remember that `!=` only checks for non-equality between two items, not three. However, you can use `!=` more than once in the same line.

For example, `2 != 3 != 4` first checks for non-equality between 2 and 3, then 3 and 4, but NOT 2 and 4.

[26]:
```
trials = 10000
different = 0

for _ in range(trials):
    words = []
    words.append(choice(p_and_p_words))
    words.append(choice(p_and_p_words))
    words.append(choice(p_and_p_words))
    if (len(words[0]) != len(words[1])) and (len(words[1]) != len(words[2]))␣
 ↪and (len(words[0]) != len(words[2])):
        different += 1

chance_of_all_different = different / trials

chance_of_all_different
```

[26]:
```
0.6318
```

[27]:
```
check('tests/q2_3.py')
```

[27]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c4003d0>
```

1.3 3. Finding Probabilities

After a long day of class, Clay decides to go to a food court for dinner. Today's menu has Clay's four favorite foods: enchiladas, hamburgers, pizza, and spaghetti. However, each dish has a 30% chance of running out before Clay can get to the food court.

Question 3.1 What is the probability that Clay will be able to eat pizza at the food court?

[28]:
```
pizza_prob = 1 - .3
```

[29]:
```
check('tests/q3_1.py')
```

[29]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c40f990>
```

Question 3.2 What is the probability that Clay will be able to eat all four of these foods at the food court?

[30]:
```
all_prob = 0.7**4 # because there is a 30% chance a given food is done, there␣
 ↪is a 70% chance a food is still availible
# so .7 * .7 * .7 *.7 or .7**4
```

[31]:
```
check('tests/q3_2.py')
```

[31]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c41dcd0>
```

Question 3.3 What is the probability that the food court will have run out of something before Clay can get there?

[32]:
```
something_is_out = 1 - 0.7**4
```

[33]:
```
check('tests/q3_3.py')
```

[33]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c41d2d0>
```

To make up for their unpredictable food supply, the food court decides to hold a contest for some free food. There is a bag with two red marbles, two green marbles, and two blue marbles. Clay has to draw three marbles separately. In order to win, all three of these marbles must be of different colors.

Question 3.4 What is the probability of Clay winning the contest?

[34]:
```
# 1st marble doesn't matter
# 2nd marble has a 4/5 chance of not being 1st marble color
# 3rd marble has a 2/4 chance of not being 1st or 2nd marble color
```

[35]:
```
winning_prob = 4/5 * 1/2
```

[36]:
```
check('tests/q3_4.py')
```

[36]:
```
<gofer.ok.OKTestsResult at 0x7f3f9c50f750>
```

1.4 4. Submission

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing.

Double check that you have completed all of the free response questions as the auto-grader does NOT check that and YOU are responsible for knowing those questions are there and completing them as part of the grade for this lab. When ready, click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

[37]:
```
# For your convenience, you can run this cell to run all the tests at once!
import glob
from gofer.ok import grade_notebook
if not globals().get('__GOFER_GRADER__', False):
    display(grade_notebook('lab06.ipynb', sorted(glob.glob('tests/q*.py'))))
```

```
red
orange
yellow
green
blue
indigo
violet
red
orange
yellow
green
blue
indigo
violet
['tests/q1_1.py', 'tests/q1_2.py', 'tests/q1_3.py', 'tests/q1_4.py',
'tests/q1_5.py', 'tests/q2_1.py', 'tests/q2_2.py', 'tests/q2_3.py',
'tests/q3_1.py', 'tests/q3_2.py', 'tests/q3_3.py', 'tests/q3_4.py']
Question 1:

<gofer.ok.OKTestsResult at 0x7f3f9ae54b10>

Question 2:

<gofer.ok.OKTestsResult at 0x7f3f9bc32d50>

Question 3:

<gofer.ok.OKTestsResult at 0x7f3f9c404490>

Question 4:

<gofer.ok.OKTestsResult at 0x7f3f9ae54b50>

Question 5:

<gofer.ok.OKTestsResult at 0x7f3f9ae5c4d0>

Question 6:

<gofer.ok.OKTestsResult at 0x7f3f9ae5c0d0>

Question 7:

<gofer.ok.OKTestsResult at 0x7f3f9ae5c190>

Question 8:

<gofer.ok.OKTestsResult at 0x7f3f9ae5c950>
```

```
Question 9:
<gofer.ok.OKTestsResult at 0x7f3f9ae5c910>
Question 10:
<gofer.ok.OKTestsResult at 0x7f3f9ae5c610>
Question 11:
<gofer.ok.OKTestsResult at 0x7f3f9ae5ca10>
Question 12:
<gofer.ok.OKTestsResult at 0x7f3f9ae5c6d0>
1.0
```

Name: Allan Gongora

Section: 0131