lab05

March 9, 2020

1 Lab 5: Randomization

Welcome to lab 5! This week, we will go over conditionals and iteration, and introduce the concept of randomness. All of this material is covered in Chapter 8 of the textbook.

First, set up the tests and imports by running the cell below.

[1]:
```
import numpy as np
from datascience import *

# These lines load the tests.
from client.api.notebook import Notebook
ok = Notebook('lab05.ok')
_ = ok.auth(inline=True)
```

```
=====================================================================
Assignment: Randomization
OK, version v1.14.20
=====================================================================

Successfully logged in as m.zareei@ieee.org
```

1.1 1. Nachos and Conditionals

In Python, Boolean values can either be `True` or `False`. We get Boolean values when using comparison operators, among which are `<` (less than), `>` (greater than), and `==` (equal to). For a complete list, refer to Booleans and Comparison at the start of Chapter 8.

Run the cell below to see an example of a comparison operator in action.

[2]:
```
3 > 1 + 1
```

[2]:
```
True
```

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

[3]:
```
result = 10 / 2 == 5
result
```

[3]:
```
True
```

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

[4]:
```
make_array(1, 5, 7, 8, 3, -1) > 3
```

[4]:
```
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 call `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.

[5]:
```
nachos = make_array('cheese', 'salsa', 'both', 'neither')
np.random.choice(nachos)
```

[5]:
```
'neither'
```

Question 1. Assume we took ten nachos at random, and stored the results in an array called `ten_nachos`. 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.

[6]:
```
ten_nachos = make_array('neither', 'cheese', 'both', 'both', 'cheese', 'salsa',␣
 ↪'both', 'neither', 'cheese', 'both')
number_cheese = np.count_nonzero(ten_nachos == 'cheese') #SOLUTION
number_cheese
```

[6]:
```
3
```

[7]:
```
_ = ok.grade('q1_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

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 8.1.

Question 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`.

[8]:
```
say_please = '?'

if number_cheese < 5: #SOLUTION
    say_please = 'More please'

say_please
```

[8]:
```
'More please'
```

[9]:
```
_ = ok.grade('q1_2')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 3. Write a function called `nacho_reaction` that returns a string based on the type of nacho passed in. From top to bottom, the conditions should correspond to: `'cheese'`, `'salsa'`, `'both'`, `'neither'`.

[10]:
```
def nacho_reaction(nacho):  #SOLUTION
    if nacho == 'cheese':
        return 'Cheesy!'
    # next condition should return 'Spicy!'
    elif nacho == 'salsa':
        return 'Spicy!'
    # next condition should return 'Wow!'
    elif nacho == 'both':
        return 'Wow!'
    # next condition should return 'Meh.'
    else:
        return 'Meh.'

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

[10]:
```
'Spicy!'
```

[11]:
```
_ = ok.grade('q1_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 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.

[12]:
```
ten_nachos_reactions = Table().with_column('Nachos', ten_nachos)
ten_nachos_reactions = ten_nachos_reactions.with_column('Reactions',␣
↪ten_nachos_reactions.apply(nacho_reaction, 'Nachos'))  #SOLUTION
ten_nachos_reactions
```

[12]:

Nachos	Reactions
neither	Meh.
cheese	Cheesy!
both	Wow!
both	Wow!
cheese	Cheesy!
salsa	Spicy!
both	Wow!

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

[13]:
```
_ = ok.grade('q1_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 5. Using code, find the number of `'Wow!'` reactions for the nachos in `ten_nachos_reactions`.

[14]:
```
number_wow_reactions = np.count_nonzero(ten_nachos_reactions.
 ↪column('Reactions') == 'Wow!') #SOLUTION
number_wow_reactions
```

[14]:
```
4
```

[15]:
```
_ = ok.grade('q1_5')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 6: Change just the comparison operators from `==` to some other operators so that `should_be_true` is `True`.

[16]:
```
should_be_true = number_cheese < number_wow_reactions > np.
 ↪count_nonzero(ten_nachos == 'neither') #SOLUTION
should_be_true
```

[16]:
```
True
```

[17]:
```
_ = ok.grade('q1_6')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 7. Complete the function `both_or_neither`, which takes in a table of nachos with reactions (just like the one from Question 4) and returns `'Wow!'` if there are more nachos with both cheese and salsa, or `'Meh.'` if there are more nachos with neither. If there are an equal number of each, return `'Okay!'`.

[20]:
```
def both_or_neither(nacho_table): #SOLUTION
    reactions = nacho_table.column('Reactions')
    number_wow_reactions = np.count_nonzero(reactions == 'Wow!')
    number_meh_reactions = np.count_nonzero(reactions == 'Meh.')
    if number_wow_reactions > number_meh_reactions:
        return 'Wow!'
    # next condition should return 'Meh.'
    elif number_wow_reactions < number_meh_reactions:
        return 'Meh.'
    # next condition should return 'Okay!'
    else:
        return 'Okay!'

many_nachos = Table().with_column('Nachos', np.random.choice(nachos, 250))
many_nachos = many_nachos.with_column('Reactions', many_nachos.
 ↪apply(nacho_reaction, 'Nachos'))
result = both_or_neither(many_nachos)
result
```

[20]:
```
'Meh.'
```

[21]:
```
_ = ok.grade('q1_7')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 3
    Failed: 0
[oooooooooook] 100.0% passed
```

1.2 2. Iteration

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.

[22]:
```
suits = make_array(" ", " ", " ", " ")

draws = make_array()

repetitions = 6

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

draws
```

[22]:
```
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)`. 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.

[23]:
```
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.

Question 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.

[26]:
```
possible_point_values = np.arange(1, 11) #SOLUTION
tosses = 1000
total_score = 0

for i in np.arange(tosses):
```

```
    total_score = total_score + np.random.choice(possible_point_values)

total_score
```

[26]: 5555

[27]:
```
_ = ok.grade('q2_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 2. What is the average point value of a dart thrown by Clay?

[28]:
```
average_score = total_score / tosses #SOLUTION
average_score
```

[28]: 5.555

[29]:
```
_ = ok.grade('q2_2')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 3. 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.

[30]:
```
austen_string = open('Austen_PrideAndPrejudice.txt', encoding='utf-8').read()␣
↪#SOLUTION
p_and_p_words = np.array(austen_string.split())

longer_than_five = 0
```

```
for word in p_and_p_words:
    if len(word) > 5:
        longer_than_five = longer_than_five + 1

longer_than_five
```

[30]: 35453

[31]:
```
_ = ok.grade('q2_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 4. 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.

[32]:
```
trials = 10000  #SOLUTION
different = 0

for i in np.arange(trials):
    words = np.random.choice(p_and_p_words, 3)
    if len(words.item(0)) != len(words.item(1)) != len(words.item(2)) !=␣
 ↪len(words.item(0)):
        different = different + 1

chance_of_all_different = different / trials

chance_of_all_different
```

[32]: 0.6262

[33]:
```
_ = ok.grade('q2_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

1.3 3. Finding Probabilities

After a long day of class, Clay decides to go to Crossroads 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 Crossroads.

Question 1. What is the probability that Clay will be able to eat pizza at Crossroads?

[34]:
```
pizza_prob = 0.7 #SOLUTION
```

[35]:
```
_ = ok.grade('q3_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2. What is the probability that Clay will be able to eat all four of these foods at Crossroads?

[36]:
```
all_prob = 0.7 ** 4 #SOLUTION
```

[37]:
```
_ = ok.grade('q3_2')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 3. What is the probability that Crossroads will have run out of something before Clay can get there?

[38]:
```
something_is_out = 1 - all_prob #SOLUTION
```

[39]:
```
_ = ok.grade('q3_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

To make up for their unpredictable food supply, Crossroads decides to hold a contest for some free Cal Dining swag. 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 4. What is the probability of Clay winning the contest?

[40]:
```
winning_prob = 1 * 4/5 * 2/4 #SOLUTION
```

[41]:
```
_ = ok.grade('q3_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

[42]:
```
# For your convenience, you can run this cell to run all the tests at once!
import os
_ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')]
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 3
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
```

```
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
```

```
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

[ ]:
```
_ = ok.submit()
```