Welcome to your first set of Python coding problems. If this is your first time using Kaggle Notebooks, welcome!

Notebooks are composed of blocks (called "cells") of text and code. Each of these is editable, though you'll mainly be editing the code cells to answer some questions.

To get started, try running the code cell below (by pressing the ► button, or clicking on the cell and pressing ctrl+enter on your keyboard).

```
In [1]:
print("You've successfully run some Python code")
print("Congratulations!")
```

Try adding another line of code in the cell above and re-running it.

Now let's get a little fancier: Add a new code cell by clicking on an existing code cell, hitting the escape key, and then hitting the `a` or `b` key. The `a` key will add a cell above the current cell, and `b` adds a cell below.

Great! Now you know how to use Notebooks.

Each hands-on exercise starts by setting up our feedback and code checking mechanism. Run the code cell below to do that. Then you'll be ready to move on to question 0.

unfold_moreShow hidden cell

0.

This is a silly question intended as an introduction to the format we use for hands-on exercises throughout all Kaggle courses.

What is your favorite color?

To complete this question, create a variable called `color` in the cell below with an appropriate value. The function call `q0.check()` (which we've already provided in the cell below) will check your answer.

```
In [3]:
# create a variable called color with an appropriate value on the line below
# (Remember, strings in Python must be enclosed in 'single' or "double" quotes)
color = "blue"
q0.check()
```

Correct: What?! You got it right without needing a hint or anything? Drats. Well hey, you should still continue to the next step to get some practice asking for a hint and checking solutions. (Even though you obviously don't need any help here.)

Didn't get the right answer? How do you not even know your own favorite color?!

Delete the # in the line below to make one of the lines run. You can choose between getting a hint or the full answer by choosing which line to remove the # from.

Removing the # is called uncommenting, because it changes that line from a "comment" which Python doesn't run to code, which Python does run.

The upcoming questions work the same way. The only thing that will change are the question numbers. For the next question, you'll call `q1.check()`, `q1.hint()`, `q1.solution()`, for question 2, you'll call `q2.check()`, and so on.

1.

Complete the code below. In case it's helpful, here is the table of available arithmetic operations:

Operator	Name	Description
`a + b`	Addition	Sum of `a` and `b`
`a - b`	Subtraction	Difference of `a` and `b`
`a * b`	Multiplication	Product of `a` and `b`
`a / b`	True division	Quotient of `a` and `b`
`a // b`	Floor division	Quotient of `a` and `b`, removing fractional parts
`a % b`	Modulus	Integer remainder after division of `a` by `b`
`a ** b`	Exponentiation	`a` raised to the power of `b`
`-a`	Negation	The negative of `a`

In [5]:
```
pi = 3.14159 # approximate
diameter = 3
```

```
# Create a variable called 'radius' equal to half the diameter
radius=diameter/2
# Create a variable called 'area', using the formula for the area of a circle: pi tim
es the radius squared
area=pi * radius ** 2
# Check your answer
q1.check()
```

Correct

2.

Add code to the following cell to swap variables `a` and `b` (so that `a` refers to the object previously referred to by `b` and vice versa).

```
In [7]:
########### Setup code - don't touch this part ######################
# If you're curious, these are examples of lists. We'll talk about
# them in depth a few lessons from now. For now, just know that they're
# yet another type of Python object, like int or float.
a = [1, 2, 3]
b = [3, 2, 1]
q2.store_original_ids()
######################################################################

temp=a
a=b
b=temp

######################################################################

# Check your answer
q2.check()
```

Correct:

The most straightforward solution is to use a third variable to temporarily store one of the old values. e.g.:

If you've read lots of Python code, you might have seen the following trick to swap two variables in one line:

We'll demystify this bit of Python magic later when we talk about *tuples*.

```
In [8]:
```

3a.

Add parentheses to the following expression so that it evaluates to 1.

```
In [10]:
(5 - 3) // 2
```

3b. 🌶️□

Questions, like this one, marked a spicy pepper are a bit harder.

Add parentheses to the following expression so that it evaluates to 0.

```
In [13]:
8 - (3 * 2) - (1 + 1)
```

```
Out[13]:
```

4.

Alice, Bob and Carol have agreed to pool their Halloween candy and split it evenly among themselves. For the sake of their friendship, any candies left over will be smashed. For example, if they collectively bring home 91 candies, they'll take 30 each and smash 1.

Write an arithmetic expression below to calculate how many candies they must smash for a given haul.

```
In [16]:
# Variables representing the number of candies collected by alice, bob, and carol
alice_candies = 121
bob_candies = 77
carol_candies = 109

# Your code goes here! Replace the right-hand side of this assignment with an express
ion
# involving alice_candies, bob_candies, and carol_candies
total_candies=(alice_candies+bob_candies+carol_candies)
candies=total_candies//3
divided_candies=(candies*3)
to_smash = total_candies-divided_candies

# Check your answer
q4.check()
```

```
Correct
```

Keep Going

Next up, you'll **learn to write new functions and understand functions others write**. This will make you at least 10 times more productive as a Python programmer.