Computer Science — Python — HW #3

Assigned on Mon, 2017-01-30. Due on Tue, 2017-02-~~07~~ 09.

The current means of turning in assignments is to email them to [jay.coskey@xxxx.xxx](mailto:jay.coskey@nyhs.org).

1. Read chapter 1 of Think Python, 2nd ed.
2. Find PEP 8, the Style Guide for Python Code, at <https://www.python.org/dev/peps/pep-0008>. Read up through, but not including the second titled “Prescriptive: Naming Conventions”.
   1. FYI — PEP stands for "Python Enhancement Proposal".
   2. FYI — the later sections of PEP 8 cover parts of Python we haven't seen yet.
3. Run a program in PyCharm
   1. Open PyCharm. It might take a few moments.

[**Edit:** The very first time you run it after installation, it starts with a window asking if you want to import settings from a previous installation.  Obviously, you can select the second option "I do not have a previous version....".  It then starts up PyCharm, and opens a "PyCharm Community Edition Initial Configuration Window".  The default settings here work just fine, so you can click on "OK" without changing anything.]

* 1. Click on "Create New Project".
  2. In the "Location" field, set the right-most element (the directory name) to "append\_test".
  3. Click on the "Create" button in the lower-right corner.
  4. Read as many "Tip of the Day" topics as you'd like, then click on "Close".
  5. Create a new file by ~~either clicking "Ctrl+Shift+N" or~~ selecting File 🡪 New 🡪 File.
  6. Enter the filename as append\_test.py
  7. Enter the following script:

def append7(items):

items.append(7)

if \_\_name\_\_ == '\_\_main\_\_': # Remember that these are double underscores.

my\_list = [ ]

print(my\_list)

append7(my\_list)

append7(my\_list)

append7(my\_list)

print(my\_list)

* 1. Run the script by selecting Run 🡪 Run… 🡪 append\_test
     + If a "Run/Debug Configurations" window appears, then enter the script name **append\_test**
  2. Note whether or not the call to append7 modifies the value of my\_list in the main body of the script.

1. **[Turn in]** Write three short Python functions.
   1. Python has a Boolean operator called “and”. If both b1 and b2 are Boolean (i.e., True/False) values, then here are the values of the expression “b1 and b2”:

|  |  |  |
| --- | --- | --- |
| b1 | b2 | b1 and b2 |
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

Without using the logical operators "and" or "or", write a function called **bool\_and(b1, b2)** that takes two arguments, and returns the same value as the expression “b1 and b2”. In other words, write a function without using logical operators that has the same output values as shown in the above table.

* 1. Write a function **product(items)**, which takes a list of numbers and returns their product.
  2. Write a function **does\_contain(items, item)** that uses a loop to return True if the list called **items** contains the element called **item**, and otherwise returns False.

**Note:** Please submit all three parts of exercise #4 in the same email, rather than in separate emails.