Computer Science — Python — HW #1

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

The current means of turning in assignments is to email them to [first.last@xxxx.](mailto:first.last@xxxx.)xxx.

[**Note:** The homework assignments in this document have been edited and re-ordered from their original versions used in Spring of 2017. What's presented here is what is recommended, based on student difficulties with earlier versions. For example, the problem involving Pascal's triangle was postponed and more introductory material was moved to lead up to it. Also, the problem was re-written, with more guidance given than was present in Spring of 2017.]

1. **[Turn in]** Send me an email at [first.last@xxxx.xxx](mailto:first.last@xxxx.xxx), with the phrase “HW #1” somewhere in the Subject. Briefly describe any programming experience you might have, and any concerns you might have about the course.

For example:

Yo! This is Al O’Vera. I’ve written a few small programs in Scratch, but nothing in Python. I once looked into writing apps for Android, but it looked like it would take forever, and didn’t have time for it.

**Note:** We might shift to your submitting homework by uploading it to xxxx.xxx. Or we might not.

You do not need to submit any evidence of having completed the remaining items from this assignment.

1. Install and test the Python distribution called “Anaconda”. This has Python itself, plus some additional packages.
   1. In a web browser, visit <https://www.continuum.io/downloads>, and install Anaconda from one of the links.
   2. Next, open a Windows command window by pressing the Windows key then typing “cmd” + <Enter>.
      * On Mac OS X, you would instead open a terminal windows under /Applications/Utilities.
   3. Next run the command

python --version # Note: There are two hyphens before the word “version”.

# The version of Python should start with 3.

* 1. Next, run the command “ipython”. You should get a prompt that reads “In [1]:”.
  2. Test the installation by entering the following command at the prompt.

In [1]: print(‘Hello, world!’)

Starting to learn a new programming language by running some sort of “Hello, world!” program dates back at least as far as 1972. That’s one long-running meme!

1. Download the book Think Python, Second Edition, from <http://greenteapress.com/wp/think-python-2e/>
2. Play lightbot at lightbot.com, up through level 1-8.
   1. First watch the brief video at lightbot.com.
   2. Next, click on the Demo Puzzles button labeled “Web”. This requires Adobe Flash to run, and might take a minute or so to start up.
   3. In the Flash program, click on the screen with the heading “1. Basics”.
   4. Complete all eight Basics boards. The goal is to light up \*all\* the blue squares. Note that you can turn off the music by clicking on the speaker icon on the left.

NWYH — Computer Science — Python — HW #2

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

The current means of turning in assignments is to email them to [first.last@nyhs.org](mailto:first.last@nyhs.org).

1. Run the following commands in ipython, and ensure that you understand their corresponding outputs.
   1. print('Hello, world!')
   2. print("Hello, world!") # Double quotes work also.
   3. print('Hello, ' + 'world!')
   4. print('Hello' \* 4)
   5. print(len('Hello'))
   6. ?len
   7. print(+2) # The single plus emphasizes that the value is positive, not negative.
   8. print(3\*\*2 + 4\*4) # An expression can be passed to print()
   9. x = 27; print(x) # Multiple statements on one line separated by semi-colon
   10. 3\*\*2 + 4\*4 # The resulting value is printed by default, even without the print() function.
   11. type(25)
   12. type(25.0)
   13. x = 42; print(x) # Commas can separate statements in the same line
   14. x = 42; y = 816357492; print('x={0:d}, y={1:d}'.format(x, y)) # Args to the formatted string are numbered
   15. x = 42; y = 816357492; print('x={0:d}, y={1:,d}'.format(x, y)) # 'd' for integers. Note the added comma.
   16. import math; print(math.pi)
   17. import math; print('pi={0:.2f}'.format(math.pi)) # 'f' for floating point numbers
   18. import IPython; IPython.\_\_version\_\_ # Two double underscores before and after “version”
   19. import this # <https://en.wikipedia.org/wiki/Easter_egg_(media)>
2. **[Turn in]** Run these commands in ipython and, in your own words and in complete sentences (one sentence for each part should suffice), briefly describe and explain the errors you get. Submit the results by email.
   1. True = True
   2. print 'Hello'
   3. print(2+)
   4. asdf
   5. (3\*5) + 7) + (4 + (2\*6)
3. Install PyCharm from <https://www.jetbrains.com/pycharm/> # This is a development tool for Python scripts.

EDIT: Install the Community edition, not the

1. Play lightbot, levels 2-1 through 2-4. Or further, if you’d like.
2. **[Optional]** Last Monday (2017-01-30) we mentioned leap seconds (<https://en.wikipedia.org/wiki/Leap_second>), and how there can be bugs relating to them.
3. Using the Gregorian calendar in the United States, how many different possible month lengths are there? There are months with 28, 29, 30, and 31 days, so the number of different lengths would be at least four.
4. Do you expect that the answer could be different outside of the United States? If so, how?

NWYH — Computer Science — Python — HW #3

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

The current means of turning in assignments is to email them to [first.last@nyhs.org](mailto:first.last@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 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, x)** that uses a loop to return True if the list called **items** contains the element called **x**, and otherwise returns False.

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

NWYH — Computer Science — Python — HW #4

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

The current means of turning in assignments is to email them to [first.last@nyhs.org](mailto:first.last@nyhs.org).

1. Read chapter 2 of Think Python, 2nd ed.
2. See if you can determine the values of the following Python expressions before evaluating them.
3. [2\*\*p for p in range(0, 5)]
4. [(p, 2\*\*p) for p in range(0, 5)]
5. ['Hello'[k] for k in range(0, 5)]
6. ['KMGTE'[k] + '=10\*\*' + str(3\*k+3) for k in range(0, 5)]
7. **[Turn in]** Write a function called **rps\_winner** that causes all the assert statements in following script to be evaluated without crashing. Try to write the function using as few lines as you can.

**Script:**

NONE = 0

ROCK = 1

PAPER = 2

SCISSORS = 3

def **rps\_winner**(a, b):

pass # TODO: Replace this line with the body of the function.

assert(rps\_winner(PAPER, PAPER) == NONE)

assert(rps\_winner(PAPER, ROCK) == PAPER)

assert(rps\_winner(PAPER, SCISSORS) == SCISSORS)

assert(rps\_winner(ROCK, PAPER) == PAPER)

assert(rps\_winner(ROCK, ROCK) == NONE)

assert(rps\_winner(ROCK, SCISSORS) == ROCK)

assert(rps\_winner(SCISSORS, PAPER) == SCISSORS)

assert(rps\_winner(SCISSORS, ROCK) == ROCK)

assert(rps\_winner(SCISSORS, SCISSORS) == NONE)

1. **[Turn in]** In books, tables of contents usually consist of list of section names aligned against the left-hand side of the page, with the corresponding page numbers aligned against the right-hand side of the page, often with periods in-between, like so:

Alligators.......................................1

Bears...........................................12

Yaks...........................................437

Zebras.........................................451

Define a function called print\_toc\_line that takes three arguments:

**width:** the number of columns the table of contents is supposed to fill

**name:** the name of the given section of the book

**page\_num:** the first page number of the given section of the book

Define it so that the following lines give rise to the table of contents shown above.

print\_toc\_line(50, 'Alligators', 1)

print\_toc\_line(50, 'Bears', 12)

print\_toc\_line(50, 'Yaks', 437)

print\_toc\_line(50, 'Zebras', 451)

**Note:**

* The function **str** converts integers to strings. For example, str(437) returns the string '437'.
* I'll leave it to your imagination whether some sections were left out of the table of contents above, or whether the author just had a lot to say about bears.

NWYH — Computer Science — Python — HW #5

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

Nothing to turn in.

1. Read chapter 3 of Think Python, 2nd ed. Continue reading two chapters per week.
2. Consider the following function:

def maxmin(a, b, x):

return max(a, min(b, x))

What will be the output of the following statements? (Try to figure this out before running it in Python.)

1. print(maxmin(0, 10, -5))
2. print(maxmin(0, 10, 0))
3. print(maxmin(0, 10, 5))
4. print(maxmin(0, 10, 10))
5. print(maxmin(0, 10, 15))
6. What would be the output of the following script? (Try to figure this out before running it in Python.)

def print\_half\_word(word):

for i in range(0, len(word)):

if i % 2 == 0:

print(word[i], end='')

print()

print\_half\_word('ions')

print\_half\_word('pulsations')

print\_half\_word('slingshots')

1. What would be the output of the following script? (Try to figure this out before running it in Python.)

s = 'antes'

for i in range(0, len(s)):

for j in range(0, i):

print(s[j], end='')

print()

1. What would be the output of the following script? (Try to figure this out before running it in Python.)

s = 'banana'

for i in range(0, len(s)):

for j in range(i, len(s)):

print(s[j], end='')

print()

1. Write a function called **c2f** that takes one float argument that is a measure of degrees Celsius (a.k.a. Centigrade), and returns the same temperature in degrees Fahrenheit. Note that both Celsius and Fahrenheit are linear scales, and have the following equivalences.

|  |  |  |
| --- | --- | --- |
| Celsius | Fahrenheit | Notes |
| -40 | -40 | School would be *so* closed |
| 0 | 32 | Freezing point of water |
| 100 | 212 | Boiling point of water |

1. Download the PowerPoint deck for Unit #1 from <<URL>>.

Study the examples, up through paren\_balanced.py, but not those after it. (They're roughly in order of increasing complexity.) For each example, become familiar with the script by imagining it running (with various inputs, if that applies). For example:

* Mentally trace what happens when the function **capitalize\_a** is called with the argument 'banana'.
* Mentally trace the function **capitalize\_ends** being called with the argument 'banana'.
* Mentally trace the function **is\_palindrome\_v2** being called with the argument 'revolver'.
* Mentally trace the function **dedupe\_v1** being called with the argument ['A', 'B', 'A', 'B']

NWYH — Computer Science — Python — HW #6

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

No code to turn in.

1. Read chapter 4 of Think Python, 2nd ed. Continue reading two chapters per week.
2. [By Bob O'Hara] Consider the following program.

k = 0

while k < 20

if k % 3 == 1:

print(k)

k = k + 2

When it is executed, what is printed?

1. B. C. D. E.

4 4 0 1 0

16 10 6 4 2

16 12 7 4

18 10 6

13 8

16 10

19 12

14

16

18

1. Here are two slightly different ways to print out a list. Do you have a preference? If so, for which one, and why?

def print\_list\_v1(items):

print('[', end='') **# Reminder: Setting end='' prevents a new line from being added.**

for i in range(0, len(items)):

if i > 0: # Skip the separator before the first item

print(', ', end='')

print(items[i], end='')

print(']')

def print\_list\_v2(items):

sep = '' # Start with the separator "disabled"

print('[', end='')

for i in range(0, len(items)):

print('{0:s}{1:s}'.format(sep, items[i]), end='')

sep = ', ' # Turn on the separator

print(']')

1. What does the following script do?

from collections import Counter

word2counter = dict() # Create an empty dictionary

for word in open('wordlist.txt').readlines():

word = word.rstrip() # Strip off extra spaces+newline at end of string

word2counter[word] = Counter(word)

for w1 in word2counter.keys():

for w2 in word2counter.keys():

if w2 == w1:

continue

if word2counter[w1] == word2counter[w2]:

print('Found a pair: {0:s}, {1:s}'.format(w1, w2))

1. Download the most recent version of the course slides for Unit #1 from <<URL>>.

* Study the loop examples not covered by the previous homework assignment. Specifically, mentally walk through the execution of the loop examples in the slides word\_count.py, convert\_int2str.py, convert\_str2int.py, and shuffle.py.

NWYH — Computer Science — Python — HW #7

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

The current means of turning in assignments is to email them to [first.last@nyhs.org](mailto:first.last@nyhs.org).

1. Read chapter 5 of Think Python, 2nd ed. Continue reading two chapters per week.
2. **[Turn in]** Write three small scripts. **Please submit them as email attachments.**
   1. **Script name:** print\_char\_count.py.
      * Let **print\_char\_count** be a function that takes a string argument, and prints out the distribution of characters in that string. Have this function convert its argument to lower case, so that the arguments 'abcdABCD' and 'aabbCCDD' yield the same output.
      * Write a script that calls this function with the word “masslessness”.
      * Submit both the script and its output.
   2. **Script name:** reachone.py
      * Let f be the (recursive) function that takes a positive integer n, and then:

If n is even, then f(n) = n / 2

If n is odd, then f(n) = 3 \* n + 1

* Note that if you start with any positive integer, and apply f repeatedly, that you will eventually arrive at the value 1. For example, if you start with n = 5, you next get the values 16, then 8, then 4, 2, 1. So, if you start with n=5, then you arrive at 1 after 12 applications of f.
* Let g be the function that takes a positive integer, n, and returns the number of applications of f it takes to arrive at 1. So, for example, g(17) = 12. Write a Python script that defines the function g, and prints out the value of g(5777).
* Submit both the script and its output—the value of g(5777).
  1. **Script name:** pascal.py

|  |  |
| --- | --- |
| Pascal's Triangle is a triangle formed as shown in the diagram to the right, with centered rows of increasing length. The top row is called Row #0, the next is called Row #1, etc. Row #<n> has n+1 items.  The first and last items in each row equal one. Items in-between are the sum of the item directly to the "northwest" and the item directly to the "northeast". |  |

* Write a (recursive) Python function called **pascal(row, col)** that takes two arguments, **row** and **col**, and returns the value of the specified row and column of Pascal's triangle.
* Define another function called **print\_pascal\_row(row)** that takes one integer argument representing a row number, and uses a loop to print out the specified row of Pascal's Triangle. Add commas to separate the values on each row, but not at the end of a row.
* **Note:** Adding end='' to a print statement suppresses the newline that is output by default. Make sure that end is a parameter to the **print** function, and not to the **format** function.
* Have a loop repeatedly call **print\_pascal\_row** in order to print out the first seven rows of Pascal's Triangle. The output should be left-aligned, and look like this:

1

1,1

1,2,1

1,3,3,1

1,4,6,4,1

1,5,10,10,5,1

1,6,15,20,15,6,1

NWYH — Computer Science — Python — HW #8

Assigned on Day, YYYY-MM-DD. Due on Day, YYYY-MM-DD.

1. Read chapters 6 and 7 of Think Python, 2nd ed. (Continuing with two chapters per week.)
2. **[Turn in]**
   1. Write a function **count\_palindromes(a, b)** that takes two integer arguments, **a** and **b**, and returns a count of how many integers **p** there are for which

* **p** is a palindrome, meaning a number whose digits are the same whether the number is read forwards or backwards.
* **p** is greater than or equal to **a.**
* **p** is less than **b**.

Assume that the argument **b** is greater than **a.**

* 1. Call the function to find how many palindromes there are between 10 and 1,000,000.
  2. Submit both the function **count\_palindromes(a, b)**, and the output from (b).

1. What is the output of the following script?

months = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']

m2days = { 'Jan':31, 'Feb':28, 'Mar':31, 'Apr':30, 'May':31, 'Jun':30,

'Jul':31, 'Aug':31, 'Sep':30, 'Oct':31, 'Nov':30, 'Dec':31 }

for m in range(0, len(months)):

print('{0:s} -> {1:d}'.format(months[m], m2days[months[m]]))

1. Download the most recent version of the course slides for Unit #1 from <<URL>>.

Review all the slides. Remind yourself of detail we've seen and discussed that you might have forgotten.

1. Work through the **Unit #1 Practice Exam**. Compare your answers with others in the class, but only \*after\* you've come up with your own answers.