Python Practice for MIT’s 6.00
Functions, exceptions
Frédo Durand

For vs. while

Try both a while and a for version of the following. Recall that you can exit a for loop early with break. Careful: in some cases, only while is possible. Decide for yourself which version is simpler.

1. Compute the sum of integers from 0 to N.

2. Sum of the odd integers between 1 and N.

3. Brute-force search of a perfect square. (Given $S$, for example 36, Start at 1 and increment by 1 until you find the square root).

4. Heron algorithm for square root. Ask the user for any positive number $s$. Start with a guess $g$, and refine it as $g' = g + s/g$ at each iteration until the difference between $g*g$ and $s$ is less than a given epsilon.

5. (Harder) Keep asking a user to enter an integer as long as they enter strings that are not digits. Hint: use the function .isdigit() on strings. For example, when $s$='abcd'; $s$.isdigit() returns False. There are a couple different ways to do this.

6. Syracuse numbers and Collatz conjecture: Ask the user to enter a number $N$. Then as long as $N$ is not 1, do the following. If $N$ is even, divide it by two. If $N$ is odd, multiply by 3 and add one. Print $N$ for each iteration. While nobody has yet been able to prove that this series always converges, there is no known example where it doesn’t.

More Pi and loops

7. Modify the pi computation code from lecture 1 to use for loops. There are a couple different ways to do it. You can loop over integers and divide inside the loop to get coordinates. (More advanced: to loop over floats, you will need numpy’s arrange, which we will cover later)

8. (Harder) You can compute pi by considering one eighth of the original points. First only look at the upper right quadrant between 0 and 1 for both $x$ and $y$. Then, you can ignore half of these points that are always inside the disk: the ones in the lower-left triangle.
9. (Harder.) Let’s try to optimize our computation further. For each row of points, the points at the beginning are always in, and at the end always out. All we need to do is figure out when points leave the circle.

10. (Harder and more open-ended.) For each row, efficiently find the inside/outside boundary starting from the previous row.

Functions

1. Write a function `double` that takes one number as input and returns twice the number. Test it by calling it on a few numbers. Write a different function `print_double` that prints twice the input rather than returning it.

2. Write a function that converts from Fahrenheit and to Celsius.

3. Write a function `even` that returns whether a number is odd or even.

4. Write a function `maxi` that takes two numbers and returns the bigger one.

5. Modify the pi computation code to make it a function `approx_pi(N)`

6. Write a function `print_alert` that takes a string as input, and prints it with exclamation marks and surrounded by a fixed number of stars. For example, `print_alert('pset due soon')` should show:

   **********************************
   pset due soon !!!!!
   **********************************

7. Modify the above function to make the number of stars the same as the size of the string plus the !!! (hint, use `len`, and recall that `*` works on strings)

8. (Harder.) Implement a function that computes the square root of a float S at a precision epsilon using Heron’s algorithm. Start with a guess g=1.0, and at each step, refine the guess as \( g' = 0.5(g + S/g) \), until the difference between S and g\*g is less than epsilon.

9. (Harder.) Modify the above code to return, instead, the number of iterations it takes to reach a good enough square root. Write a loop that calls this function on numbers from 1.0 to N in given steps (for example, 0.1) and prints the biggest number of iterations.

10. (Harder.) Write a function that takes N as input and returns the number smaller than N with the max number of divisors. For this, define a helper
function inside your function that returns the number of divisors of an integer N. Make sure you understand the difference between the N parameter of each function and their scope. Use the online Python tutor if needed.

Other resources

I used inspiration from the following resources, which have many more:

http://www.upriss.org.uk/python/PythonCourse.html
http://www.ling.gu.se/~lager/python_exercises.html
http://basicprogrammingexercises.blogspot.com/
http://projecteuler.net/
http://interactivepython.org/runestone/static/thinkcspy/index.html
http://www.codecademy.com/tracks/python