Project Euler Challenge

Eckel, TJHSST AI1, Fall 2020

Background & Explanation

A huge part of AI this year is building your skills in modeling and solving problems from scratch. You also need to practice your Python skills. A great way to build both of these skills is by solving problems on the website Project Euler. Project Euler is a long-running set of programming problems that are language agnostic, meaning they're problems well suited to programming solutions but that don't require solution code from any particular programming language. We will of course use Python. If you want to check your answers, you can optionally make an account and submit them; the website will verify each correct solution. You can make an anonymous account that has no association with your name.

Important notes:

- This is our first assignment subject to the collaboration & plagiarism rules. A refresher:
    - If students A and B both have *incorrect or incomplete code* they **may** collaborate together on algorithms, pseudocode, and finding Python commands that might be helpful. They **may not** write code together.
    - If student A has *incorrect code* and student B has *correct code*, student B **may not** show, send, copy, or dictate their *correct code* to student A. Student A **may not** look at student B's *correct code* for any reason. Student B **may** look at student A's *incorrect code* to help debug or offer advice.
    - If students A and B both have *correct code*, they **may** look at each other's code to compare solutions. Of course, after doing so, each student **may not** replace their own solution with the other student's.
- Solutions to PE problems are *easily* available online, so it is important that we agree on the purpose of this assignment. When programming, understanding someone else's code ***is not a valid substitute*** for writing it yourself. Generating an idea yourself is an entirely different learning experience. So, it is ***not acceptable*** to search solutions on the internet, understand them, and then code them yourself. This defeats the purpose. If you get stuck, ask me or another student or try another problem for a while. I know many of you are excited to see examples of elegant Python code, so I promise that we will all discuss canonical solutions as a class later on. For now, I want ***your*** work and thinking (subject to the collaboration rules listed above).
- You **may not** use any import statements (aside from import sys). Code all algorithms yourself!

Required Tasks

1) Go to ProjectEuler.net. If you'd like to check your answers, you'll need to make an account. This is **optional.** An important note: this website is ***unhelpful*** when it comes to recovering your password; email it to yourself or something so that you don't forget it. If you forget it, you might be locked out of your account forever.

2) Write a function `is_prime(x)` that will determine if `x` is a prime number or not.
    (To make this a little more efficient, there is a nice trick for this that's worth mentioning. To check if a number is prime, you only have to check possible factors less than or equal to the square root of that number. Can you see why this is true? Also, aside from "2", you only have to check odd numbers!)

3) Solve the following Project Euler problems. Your code for each problem should execute in less than a minute.
    a. Use your `is_prime(x)` function from #2 to solve Project Euler problem 7
    b. Solve problem 1
    c. Solve problem 2
    d. Solve problem 3
    e. Solve problem 4
    f. Solve problem 8
    g. Solve problem 9

Optional Challenges

This part is entirely optional: you will be able to start the course work if you found solutions to the questions above. However, if I were teaching a class where every student had coded in Python before, I'd probably require these additional problems, just to add a few more Python strategies to your toolkit. Also, lots of last year's students recommended taking time to be confident in Python at the beginning of the course; this is a chance to do that.

So, to be clear, these optional challenges won't be graded, but I will give answers to these in class after the Project Euler assignment is due. Try and play around with them if you have time, and then when you see the answers, you'll have a few more Python strategies to draw on.

I also highly recommend doing these if you plan on later trying for Project Euler Outstanding Works below.

- Solve problem 1 in one line using a list comprehension.
- You'll notice I didn't assign problem 5; it's too easy to hardcode the answer just by understanding how prime factorization works and not using any algorithm at all. Most of you would probably be able to figure this out with your brains and a pocket calculator pretty quickly. Let's fix this by making 5 a little harder:
    - Write a function gcd(x, y) that returns the gcd of x and y. I recommend one of the versions listed under "Implementations" on the Wikipedia page for Euclidean Algorithm.
    - Use this to solve Problem 5 in a universal way (ie, write an algorithm that could quickly solve the problem as stated for 1 to *n* for **any integer** *n*).
    - Test your speed. For comparison, mine is able to find the smallest integer divisible by everything from 1 to 10,000 almost instantly and takes less than 10 seconds to do 1 to 100,000.
- Solve problem 6 (not previously assigned) in one line using a list comprehension.
- See if your is_prime function is efficient enough to solve problem 10 in less than one minute. If it isn't, revise.
    - **Note:** If you do this one, **please comment it out when you submit your code** – I don't want to be stuck waiting a long time to grade each student's submission!
- Problem 29 has a lovely one-line solution that is relevant to our future labs; this is too easy to code otherwise, but trying to get that one-line solution is a lovely challenge.

Specification

Submit a single Python script to the link given on the course website. When executed, it should run your code for all of the required tasks and output the correct answer for each one. Feel free to include the optional challenges if you did them, except #10 (see above). Do **not** include the Outstanding Work below; if you do that, submit it separately, to the separate Outstanding Work submission links.

This assignment is **complete** if:

- The "Name" field on the Dropbox submission form contains your **class period**, then your **last name**, then your **first name**, in that order.
- The code for each problem runs and every problem is **solved correctly**. (No credit for a line of code that simply outputs the hardcoded answer; the code you submit must be the code that solved the problem.)
- Your code is **clearly commented** so I can see which code goes with which problem.

For **resubmission**:

- Complete the specification correctly.

Outstanding Work Specification: Additional Selected Problems from 11 to 30

If you're enjoying this, have extra time, or wish to push yourself further in straightforward Python use, here are some more challenges. You probably only want to do this if you were able to complete the optional challenges above (aside from maybe #29); this is a pretty hard OW to get if you're not super comfortable with Python ins and outs. On the other hand, if Python is new to you and you have some extra time in the first couple weeks of school, this is a great way to boost your skills.

In any case, here are 9 more particularly nice problems out of the next 20; if you solve at least 6, that is worth an Outstanding Work credit.

- Problem 11
- Problem 12
- Problem 14
- Problem 17
- Problem 18 using the brute force solution; the clever solution is probably too hard for week 1
- Problem 21
- Problem 24
- Problem 28
- Problem 30

Submit a single Python script to the link given on the course website (note: each OW *has its own separate submission link*. Do **not** submit this to the link for the normal assignment!)

When executed, your script should run your code for at least six of the problems listed above and output the correct answer for each one.

This assignment is **complete** if:

- The "First Name" field on the Dropbox submission form contains your **class period**, not your name.
- The "Last Name" field on the Dropbox submission form contains your **last name then a comma then your first name** (like, for example, "Eckel, Malcolm").
- The code for each problem runs and every problem is **solved correctly**. (No credit for a line of code that simply outputs the answer; the code you submit must be the code that solved the problem.)
- Your code is **clearly commented** so I can see which code goes with which problem.
- **At least six** of the problems are solved.
- **Each problem individually** runs in less than one minute.

For **resubmission**:

- Complete the specification correctly.

Outstanding Work Specification: Your Choice of Four from #100 or Higher

The same as the above OW, but there's a different submission link on the website, and I want you to pick four questions from question 100 or higher to solve yourself. Note this must still be your original work!

There is **one additional submission criterion** – in addition to submitting your .py file, I need you to submit **a screenshot of your Project Euler profile** demonstrating that the website has given you credit for these four problems. I don't have the answers to all of them available, so I need proof yours are correct!

You can do both OW assignments for two separate OW credits if you wish.