Project Euler Challenge

Eckel, TJHSST AI1, Fall 2021

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!

BLUE Credit: 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) If you need a refresher, look up examples of list and set comprehensions (easily googleable); I'm requiring them for a couple problems.

4) Solve the following Project Euler problems. To make sure you're ready to be successful later in the course, I have a few extra criteria for some problems; be sure to refer back to this page as you work. Your code for each problem should execute in less than a minute.
    a. Solve problem 1 (optional extra challenge: solve in one line using a list comprehension)
    b. Solve problem 2
    c. Solve problem 3
    d. Solve problem 4
    e. Problem 5 is too easy; 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:
        i. 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.
        ii. 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*).
        iii. 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. You don't have to be as fast as my code, but it shouldn't be massively worse; more than a factor of 10 is bad.
        iv. Set your code to print the answer to the actual Project Euler #5, but be aware I will examine your code to check that your solution really is universal!
    f. Solve problem 6 (optional extra challenge: solve in one line using list comprehensions)
    g. Solve problem 7 using your `is_prime(x)` function from #2
    h. Solve problem 8
    i. Solve problem 9
    j. Solve problem 29 (optional extra challenge: solve in one line; maybe you don't want **list** comprehensions though…)

Specification for BLUE Credit

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.

RED Credit Option #1: Additional Selected Problems from 11 to 30

If you'd like RED credit for this assignment, after finishing the BLUE credit portion you need to either do this option or the next one.

Here are 9 more particularly nice problems out of the next 20; if you solve at least 6, that is worth RED 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. 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 "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 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.

RED Credit Option #2: Your Choice of Four from #100 or Higher

The same as the above option, but there's a different submission link on the website. For this option, I want you to pick four questions from question 100 or higher to solve yourself. Basically, you're choosing if you want to do 6 more straightforward problems or go hunting for 4 more interesting problems that you're independently interested in.

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!

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 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 four** problems from #100 or higher are solved.
- **Each problem individually** runs in less than one minute.
- You **also submit the requested screenshot** (see above).

BLACK Credit: Be One of the First People to Solve a Project Euler Problem

Pick a Project Euler problem that less than 200 people have solved. Solve it! Project Euler says that all problems have solutions that will run in less than a minute; this is a good guideline, but not necessary for this problem. If you solve it in less than, oh, say, a couple hours, I'm fine with it.

Once you've solved the problem and the website has verified, create either a video or a document that explains your code in detail, so that someone who was totally stumped would be able to understand your solution. I might accept an **extraordinarily** well-commented Python file by itself, but you'll have to show me in class before you submit so I can agree that's ok.

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 this problem.

Submit your Python file, your screenshot, and a document or VIDEO.txt file with a link to your video (just so it's easier for me to keep track of).

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 your chosen problem runs and 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.)
- You **also submit the requested screenshot** (see above).
- Your **explanation is clear enough that I understand it**.