Project Euler Challenge

Eckel, TJHSST AI1, Fall 2022

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, time, and heapq, if you want to use a heap). Code all algorithms yourself!

RED 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're doing this before I have lectured about list / set comprehensions, **look up Python list and set comprehensions online somewhere.** You will need them.

4) Solve **any fourteen** of the following Project Euler problems. There are sixteen options, that means **you may skip any two problems and still get full credit.** Your total runtime should be less than 30 seconds. (For what it's worth, my own solution code runs in about 5.)
   - Solve problem 1 in one line using a list comprehension (your one line should say print() with the whole solution inside the print parentheses)
   - Solve problem 2
   - Solve problem 3
   - Solve problem 4
   - 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:
     - 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. 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.
     - 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!
   - Solve problem 6 in one line
   - Solve problem 7 using your `is_prime(x)` function from #2
   - Solve problem 8 (the course website has the number copy/pasteable in Python-ready form)
   - Solve problem 9
   - Solve problem 11 (the course website has the grid copy/pasteable in Python-ready form)
   - Solve problem 12
   - Solve problem 14
   - Solve problem 18 – the brute force solution is fine, but if you can come up with the clever one you'll be really proud of yourself (the course website has the pyramid copy/pasteable in Python-ready form)
   - Solve problem 24
   - Solve problem 28
   - Solve problem 29 in one line

5) At the bottom of your code, print the **total amount of time the whole program took to run.** (I will independently verify this with my grader, so be sure to be honest!)

Specification for RED 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.

This assignment is **complete** if:

- You follow the instructions on the submission form to format your submission properly.
- You have completed at least 14 problems correctly. Reminder: **you may skip any two problems and still get full credit.**
- 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.) Please identify your answers; print something like "#1:" before the answer to number 1 so I can easily tell which is which.
- Your code is **clearly commented** so I can see which code goes with which problem.
- Your **total runtime** is less than 30 seconds. (That's on my computer; if you're on a school laptop, less than a minute is probably fine.)
- Your code **prints** your total runtime at the bottom of your run.

BLACK Credit: Extension - Your Choice of Four from #100 or Higher

Basically the same as the RED assignment, but different problems. For this option, 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!

This assignment is **complete** if:

- You follow the instructions on the submission form to format your submission properly.
- Your submission **does not include** the RED level problems, just the four new ones.
- 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).