The first two tasks will use a CSV file of Pokemon data. Download it from GitHub, and move it into the location of your Python code, before continuing.

Task 1:

Write a function that will return the name, and stats total, of the Pokemon that has the highest total stats points. **Note**: this will require you to look up *two* columns from the CSV file.

The signature of this function is `(None) ---> tuple<string, int>`.

Note the somewhat odd syntax here – `tuple<string, int>` indicates that this is a two-element tuple, containing a string as its first element, and an int as its second element.

Task 2:

Write a function that will return the count of how many Pokemon have the specified type as either their primary, or secondary, type. For instance, if we specify the type "Grass", then your function should include both Bulbasaur (which is a Grass/Poison type), and Parasect (which is a Bug/Grass) type when calculating the total. Make sure that your program still works even for Pokemon with a single type (for instance, Tangrowth, which is only Grass type).

The signature of this function is `(string) ---> int`

Task 3:

A hospital calculates salaries for its doctors in the following way. Each gets a base pay of $200,000 a year. In addition to the base pay each doctor gets an additional $10,000 for every 5 patients treated in a year (rounded down). Download `patients.txt` from GitHub, which contains the data for this exercise.

It lists the number of patients treated by each doctor. Write a function `salary_dist(filename)` to calculate each doctor's pay and display it as a histogram. It should be formatted like the following (**Your result may look different**)

```
<= 200,000      |*
200,001-300,000 |**********
300,001-400,000 |**************
```

```
400,001-500,000 |*********
>=500,001       |**************************************************
*************
```

After this is working properly, modify `salary_dist()` to write it to a file `histogram.txt`

The signature of this function is `(string) ---> None`.

Task 4:

In this activity you are going to estimate the likelihood of occurrence of alphabet characters in normal English text. In order to find the likelihood of occurrence of a particular character analyse a large piece of English text (provided to you as `sample.txt`) and count the number of times a certain character occurs in the text. The probability of each character is the count of that character divided by the total number of characters in the text. Write a function `get_likelihood(filename)` to calculate the likelihood of occurrence of alphabetic characters in normal English text; your function should return a dictionary mapping characters to their probability. You'll probably find the `isalpha` method useful for checking whether a character is alphabetic or not.

The signature of this function is `(string) ---> dict`.