# Creating a Computer Model for US Presidential Elections

Consider the base program provided for this exam. It contains the basic structure of how a presidential election could turn out based on random factors. For example, the total number of people voting per state is random and which party gets a vote from that population is also random. There are 4 types of party categories built into this program: Democrat, Republican, Libertarian, Independent, and Invalid.

Invalid votes are discarded and not considered. Votes for the libertarian and Independent parties are counted separately, not together. The same probability of outcome is randomly generated to select any of the 4 party categories.

Base project can be downloaded from here: <https://github.com/fedcalderon/python1-Midterm>.

When complete, commit/push your code to your repo. Send Mr Fed an email with a link to it for evaluation.

1. (10 pts) Create a GitHub repository on your account called “Python1-Midterm” and copy the code over to your repo.
   1. Clone the repo to your PyCharm IDE, upgrade your project
   2. Make sure the project runs as demonstrated in class
2. (10 pts) Write a 1-page paper explaining how the program works.
   1. Place breakpoints and step through the code carefully
   2. Study the program as is. Place a break point and step through the code to gain understanding of how the logic works.
3. (20 pts) Modify the function that selects the party vote, get\_random\_number(), to match historical data. The function should return a list of selected parties that match historical data. For example, Republicans get between 42% to 52% of the vote, Democrats get between 43% to 54% of the vote, other parties get the rest. For any given state, as well as nationally, the distribution must follow the same pattern. Consider the following websites as a reference for historical data analysis on party victories. Your changes must match this historical trend.
   1. <https://www.statista.com/statistics/1035521/popular-votes-republican-democratic-parties-since-1828/>
   2. <https://en.wikipedia.org/wiki/List_of_United_States_presidential_elections_by_popular_vote_margin>.
4. (30 pts) In the results\_processor.py module, write logic that does the following:
   1. Tabulate each state by electorate values
   2. Determine which party won
   3. Print a well formatted report stating which party won and by how many electorate points ahead of the other parties
5. (30 pts) Suggest 1 way to manipulate the logic of this program to rig the election results for any party and implement the logic to show how it can be manipulated.
   1. For example, a vote for the Libertarian party turns into a vote for the Independent party.
   2. In main.py, write a function call to this logic where I can pass an argument to rig the election for such party. For example,
      1. elect.run\_rigged\_model(“Libertarian”)
         1. this makes the Libertarian party win
      2. elect.run\_rigged\_model(“Republican”)
         1. this makes the Republican party win

Extra Credit:

1. 20 pts
   1. (10 pts) Make a class Die with one attribute called sides, which has a default value of 6. Write a method called roll\_die() that prints a random number between 1 and the number of sides the die has. Make a 6-sided die and roll it 10 times.
   2. (10 pts) Write a function that calculates the probability of getting 4 dots on the dice.