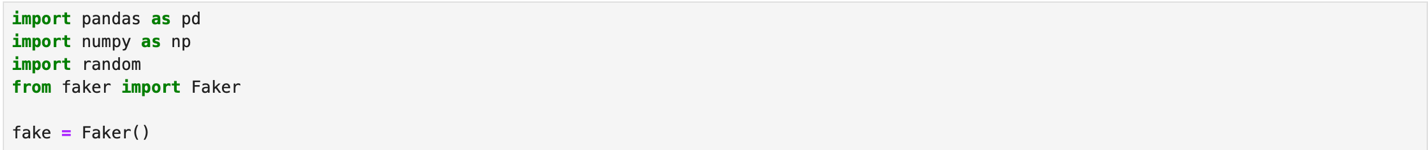
**Python Generation Script and a few of my Thoughts:**

A screenshot of a computer

AI-generated content may be incorrect.

* This excerpt installs the Faker, Pandas, and NumPy libraries



* This excerpt of script imports pandas and numpy libraries for data manipulation and analysis and imports random and faker libraries for data creation. I also initialize the Faker instance for generating fake names.
  + Random library for generating randomized values
  + Faker for generating fake but realistic personal data (names, etc.)



* The script above defines categorical variables such as Jobs, Industries, Locations, and AI tools, to use when generating structured fake data with Faker.
  + Job titles used are jobs commonly named as at risk of replacement by AI
  + Industries used are major industries which use AI to better their practice but this could also potentially end up affecting the jobs titles listed.
  + Locations listed are major cities home to hubs of the major industries listed
  + The AI tools are relevant AI tools used in/made for the different industries listed, some have applications across many industries

A screenshot of a computer

AI-generated content may be incorrect.

* In this excerpt of the script, I created a Python dictionary called job\_industry\_map that associates each job role with a list of relevant industries. Each key in the dictionary is a job title from our job\_titles category we just defined, and the corresponding value is a list of industries (also from the previously defined categories) where that role typically exists. For example, 'Doctor' is mapped to 'Healthcare'.
* This block of code is designed to support realistic fake data generation, where jobs are paired with a logically consistent industry and the fake data generated feels like you’re looking at a database of real data from different people across industries.

A screenshot of a computer program

AI-generated content may be incorrect.

* Here I built a function with conditional logic to assign AI tools commonly associated with specific job titles (Copilot for Software Engineers, DoNotPay for Paralegals, etc.)
  + The jobs are grouped by similarity of industry and the function returns a randomly selected tool from the relevant options for each person based on their role
  + This simulated the realistic threat of AI or automation to one’s job
    - Unfortunately, this project is using fake data partially because I want to become better at using python and I want to learn by doing but also because I don’t want to wait forever for google form responses and I don’t have the audience to get enough responses. BUT in theory, if this data pipeline were to use real data gathered from responses from a google form with the categories, this would lead to great insights on the workforce’s perception of AI like which jobs/industries feel most threatened.

A screenshot of a computer

AI-generated content may be incorrect.

* This script essentially generates 1000 fake employee records. The Faker library generates a name which is accompanied with a randomly selected job which is associated with an industry and a random salary, location, and years of experience.
  + The (assign\_ai\_tool) function is used to assign an AI threat based on role.
  + The (job\_industry\_map) function is a dictionary that maps each job role to one or more relevant industries. It is used in the data generation loop to assign an industry that aligns with the selected job, ensuring that each fake person's profile remains realistic and consistent.
  + The core of the script is a loop that runs 1,000 times, generating a new fake job record on each iteration.
  + Stored each record as a dictionary and appended it to a master list (data).
  + This simulates a structured dataset with realistic variation.
  + It’s just like if I received 1000 responses on googles forms with answers to general questions gauging perception of AI while also collecting data on the person filling out the form.

A screenshot of a computer

AI-generated content may be incorrect.

* Once all 1,000 records were generated, I converted the list of dictionaries into a pandas DataFrame, which is a tabular data structure ideal for analysis and visualization. Then I ran a print function to display the first 25 rows which offered a quick way to inspect the quality and completeness of the generated fake data.



* Here I simply exported the finalized fake dataset to a CSV file and added an (index = false) to ensure that the dataset would be clean and shareable.
  + I could put this directly into a database or analytics platform such as Power BI