# CS 5293, Spring 2020 Project 0

*Due 2/25 end of day*

In this activity, we are going to stretch the super powers you have learned thus far. Use your knowledge of Python and the Linux command line tools to extract information from a scraped file and add it to a csv file.

The Norman, Oklahoma police department regularly reports of incidents arrests and other activity. This data is hosted on their website. This data is distributed to the public in the form of PDF files.

The website contains three types of summaries `arrests`, `incidents`, and `case summaries`. Your assignment in this project is to build a function that collects **only the incidents**. To do so, you need to write python-based functions to do the following:

- ☐ download the data from one incident pdf;
- ☐ extract the fields:
  - ☐ date/time
  - ☐ incident number
  - ☐ incident location
  - ☐ nature
  - ☐ incident ori (originating agency identifier)
- ☐ create a SQLite database to store the data;
- ☐ insert the data into the database;
- ☐ print each nature and the number of times it appears

Below we describe the assignment structure and each required function. Please read through this whole document before starting!

## README

The README file should be *all uppercase* with either no extension or a `.md` extension. You should write your name in it, and example of how to run it, any bugs that should be expected, and a list of any web or external libraries that are used. The README file should make it clear contain a list of any bugs or assumptions made while writing the program. Note that you should not be copying code from **any website not provided**

**by the instructor.** Using any code from previous semesters or students will results in an academic dishonesty violation. You should include directions on how to install and use the Python package. You should describe all functions and your approach to develop the database. You should describe any known bugs and cite any sources or people you used for help. **Be sure to include any assumptions you make for your solution.**

## COLLABORATORS file

This file should contain a pipe separated list describing who you worked with and a small text description describing the nature of the collaboration. This information should be listed in three fields as in the example is below:

```
Katherine Johnson | kj@nasa.gov, Helped me understand calculations
Dorothy Vaughan | doro@dod.gov, Helped me with multiplexed time ma
Stackoverflow | https://example | helped me with a compilation but
```

## Project Descriptions

Your code structure should be in a directory with the following format:

```
cs5293p19-project0/
├── COLLABORATORS
├── LICENSE
├── Pipfile
├── Pipfile.lock
├── README
├── project0
│   ├── __init__.py
│   └── main.py
│   └── ...
├── docs
├── setup.cfg
├── setup.py
└── tests
    ├── test_download.py
    └── test_date_times.py
    └── ...
```

Feel free to combine or optimize functions as long as you code preserves the behavior of main.py. Also, you may have several additioanl tests and modules in your code.

**main.py**

Here is an **example** main.py file. Calling the main function should download data insert it into a dataabse and print a summary of the incidents. You code will likely differ significantly. You may have more or less individual steps.

```
# -*- coding: utf-8 -*-
# Example main.py
import argparse

import project0

def main(url):
    # Download data
    project0.fetchincidents(url)

    # Extract Data
    incidents = project0.extractincidents()

    # Create Dataase
    db = project0.createdb()

    # Insert Data
    project0.populatedb(db, incidents)

    # Print Status
    project0.status(db)


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("--incidents", type=str, required=True,
                        help="The arrest summary url.")

    args = parser.parse_args()
    if args.arrests:
        main(args.arrests)
```

Your code should take a url from the command line and perform each operation. After the code is installed, you should be able to run the code using the command below.

```
pipenv run python project0/main.py --incidents <url>
```

Each run of the above command should create a new normandb database files. You can add other command line parameters to test each operation but the `--incidents <url>` flag is required.

Below is a discussion of each interface. Note, the **function names are a suggestions and should be changed to suite the programmer**.

#### Download Data

The function `fetchincidents()` takes no parameters, it uses the python urllib.request library to grab **all the incident** one of the pdfs for the norman police report webpage.

You can use the snippet below to grab the daily activity web page.

```
url = ("http://normanpd.normanok.gov/filebrowser_download/"
       "657/2020-02-05%20Daily%20Incident%20Summary.pdf)

data = urllib.request.urlopen(url).read()
```

The locations of files can be stored to the file system perhaps in the `/tmp` directory, as a local variable, a config file, any other method of your choosing. As long as the next method can read the data from the arrest page. For official ways of handeling config and tempoprary files, see the an example here. Please discuss your choice in your README file.

#### Extract Data

The function `extractincidents()` takes no parameters and it reads data from the pdf files and extracts the incidents. The each incident includes a `date_time`, `incident number`, `incident location`, `nature`, and `incident ori`. To extract the data from the pdf files, use the PyPdf2.PdfFileReader class. It will allow you to

extract pages and pdf file and search for the rows. Extract each row and add it to a list.

Here is an example Python snippet that takes data from a bytes object that contained pdf data, writes it to a temporary file, and reads it using the PyPdf2 module.

```
import tempfile
fp = tempfile.TemporaryFile()

# Write the pdf data to a temp file
fp.write(data.read())

# Set the curser of the file back to the begining
fp.seek(0)

# Read the PDF
pdfReader = PdfFileReader(fp)
pdfReader.getNumPages()

# Get the first page
page1 = pdfReader.getPage(0).extractText()
# ...
```

This function can return a list of rows so another function can easily insert the data into a database. **For this assignment you wil need to consider each page of any pdf.**

#### Create Database

The `createdb()` function creates an SQLite database file named `normanpd.db` and inserts a table with the schema below.

```
CREATE TABLE incidents (
    incident_time TEXT,
    incident_number TEXT,
    incident_location TEXT,
    nature TEXT,
    incident_ori TEXT
);
```

Note, some “cells” have information on multiple lines, your code should take care of these edge cases.

#### Insert Data

The function `populatedb(db, incidents)` function takes the rows created in the `extractincidents()` function and adds it to the `normanpd.db` database. Again, the signature of this function can be changed as needed.

#### Status Print

The `status()` function prints to standard out, a list of the nature of incidentsandthe number of times they have occured. The list should be sorted alphabetically by the nature. Each field of the row should be separated by the pipe character (`|`).

```
Abdominal Pains/Problems|2
Alarm|14
Animal at Large|2
Animal Complaint|2
Animal Inured|1
Animal Vicious|1
Assult EMS Needed|2
...
```

#### Test

We expect you to create your own test files to test each function. Some tests involve downloading and processing data. To create your own test you can download and save a file locally. This is recommended, particularly because Norman PD will irregularly remove the arrest files. Tests should be runnable by using `pipenv run pytest` or `pipenv run python -m pyest`. You should discuss your tests in you README.

## Create a repository for your project on your instance and GitHub

Create a **private** repository GitHub called `cs5293sp20-project0`

Add collaborators `cegme` and `mghirsch42` and `kbanweer` by going to `Settings > Collaborators`.

### Submitting your code

When ready to submit, create a tag on your repository using git tag on the latest commit:

```
git tag v1.0
git push origin v1.0
```

The version v1.0 lets us know when and what version of code you would like us to grade. If you would like to submit a a second version before the 24 hour deadline, use the tag `v2.0`.

If you need to update a tag, view the commands in the following StackOverflow post.

We will update the submission instructions so keep an eye out!

## Deadline

Your submission/release is due on Thursday, Feb 25st at 11:59pm. Submissions arriving between 12:00am and 11:59pm on the following day will receive a 10% penalty; meaning these scores will only receive 90% of its value. Any later submissions will receive not receive credits. You must have your instance running and code available otherwise you will not receive credit.

## Grading

Grades will be assessed according to the following distribution:

- 60%: Correctness.
  - This will be assessed by giving your code a range of inputs and checking the output.
  - Use the creation of tests to prove correctness.

- 40%: Documentation.
  - Your README file should fully explain your process for developing your code.
  - All other commands should be well-documented.

## Addendum

- 2020-02-14 Corrected the table name from `arrests` to `incidents`.

Back to Project List