Assignment 1 - Make a dataset

The dataset I created is image-based. I mainly used the `jmd_imagescraper` as the main function, which is an easy-to-use web-scraping method. With `jmd_imagescraper`, I collect 1,000 images of cats and dogs (500 each) from the DuckDuckGo search engine.

First things first, we need to create an environment for dataset creation. A virtual environment lets you have a stable, reproducible, and portable environment, making you are in control of which packages versions are installed and when they are upgraded, which is a huge benefit, especially when there could be tons of dependencies. The making of a virtual environment can be very flexible; for myself, I use `conda` to manage virtual environments. Here, I name the virtual environment to be `AI4Media`; I will finish all the actions in this environment.

```
conda create -n AI4Media python=3.9 anaconda
```

Then, we need to activate this environment to start our work.

```
conda activate AI4Media
```

After entering the `AI4Media` environment, we need to have dependencies installed. Here I use `pip` to install our required packages.

```
pip install jmd_imagescraper
```

After pressing the Enter button and waiting till the installation done, we should see this:

Successfully installed jedi-0.18.2 jmd_imagescraper-1.0.2.

Now, the preparation is finished.

For the main code, we have:

```python
from jmd_imagescraper.core import *
from pathlib import Path

root = Path().cwd()/"images"

duckduckgo_search(root, "Dogs", "a dog", max_results=500)
```

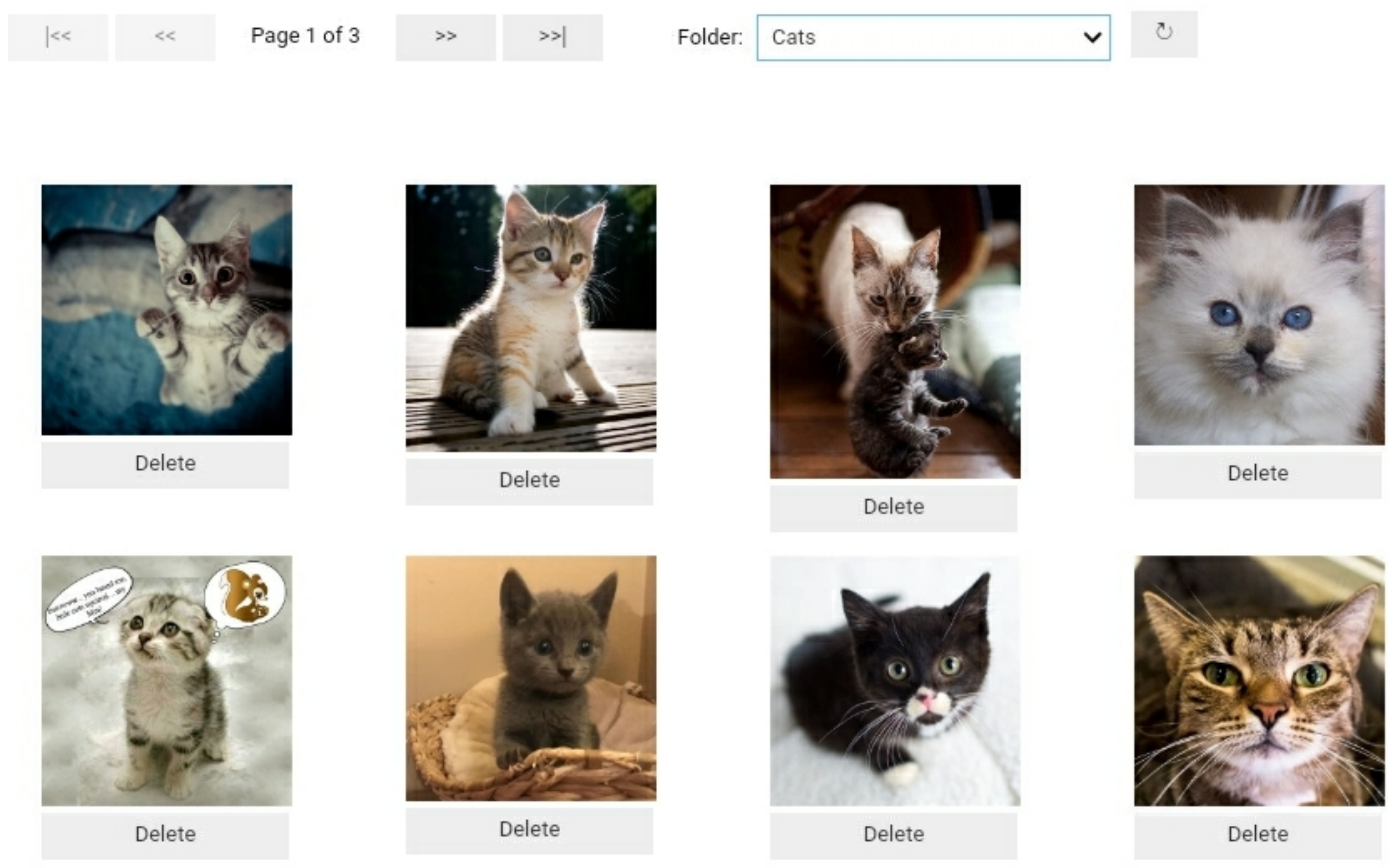
The code is simple and clear. We have `root` as the root path we expect our dataset to be under; and the calling of `duckduckgo_search()` automatically scrape and collect assigned images from the DuckDuckGo search engine. The parameters for `duckduckgo_search()` is simple, which expect the first to be the root path of the dataset, the second to be the name of the folder that holds the collected images, the third to be the searching prompt, the last to be the number of images to be scraped and collected, here we make it 500, but surely we can make it larger if needed.

We could examine our dataset with this:

```python
from jmd_imagescraper.imagecleaner import *

display_image_cleaner(root)
```

The above code visualizes the images like this:



So far, the raw, unlabeled dataset is created. The structure of the dataset is like this:

```
images
├── Cats
│   ├── 001_25df1ddf.jpg
│   ├── ...
│   └── 500_e3cc0b29.jpg
└── Dogs
    ├── 001_d98dcc47.jpg
    ├── ...
    └── 500_446c8f96.jpg
```

Reference

Dockrill, Joe, "jmd_imagescraper", Sep 20, 2020. https://joedockrill.github.io/jmd_imagescraper/