Exercise 3: Dask

Each assignment is graded out of 5 points. The topic for this assignment is Dask. This week, we will be returning to a more familiar environment and doing the assignment in a Jupyter notebook (hooray). The submission will be a single Jupyter notebook file with all of your work.
We do not intend to re-run the notebook. The assignment will be graded only by looking at the output.

Advice: Managing Conda Environments with Jupyter

Please make sure to run the notebook in the same conda environment you used to install dask. To make the environment visible to your Jupyter, there were two steps I needed to take on my system:

(1) Install the package nb_conda_kernels in your base environment, e.g.
$ conda activate base
(base) $ conda install install nb_conda_kernels

(2) Install the package ipykernel in every environment that you want available in Jupyter, e.g.
$ conda activate ac295
(ac295) $ conda install ipykernel

Advice: Please Don't Copy / Paste Commands; Type Them Yourself!

Much of this assignment consists of following along with tutorials in the Dask book.
It is possible to copy / paste the commands verbatim, and you will probably get full credit on the assignment wherever that works. However, we strongly suggest that you type in all commands at the keyboard. You will learn the material much better this way.

Question 1: Install dask & Download data set (0.5 points)

Install dask on your computer. We recommend using conda in an environment for this course
$ conda activate ac295
(ac295) $ conda install dask
Download the data set used in the Dask book:
(https://www.kaggle.com/new-york-city/nyc-parking-tickets)
On my computer, I downloaded the file into directory IACS/AC-295/Exercises/Dask
The zip folder inflates to ../Dask/nyc-parking-tickets
My jupyter notebook file is also in the /Dask directory.
Run the commands in Listing 2.1 (p. 65) of the Dask book from a Jupyter notebook.
The output should be a description of the metadata of the Dask DataFrame object

Submit:

1. Commands matching listing 2.1 that run successfully on your system

Example Submission:

```
import dask.dataframe as dd
from dask.diagnostics import ProgressBar
from matplotlib import pyplot as plt
```

```
df = dd.read_csv('nyc-parking-tickets/*2017.csv')
```

```
df
```

Dask DataFrame Structure:

	Summons Number	Plate ID	Registration State	Plate Type	Issue Date	Violation Code	Vehicle Body Type	Vehicle Make	Issuing Agency	Street Code1	Street Code2	Street Code3	Vehicle Expiration Date
npartitions=33													
	int64	object	object	object	object	int64	object	object	object	int64	int64	int64	int64
	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...	...	...

Dask Name: from-delayed, 99 tasks

Question 2: First Dask Computations (1.0 points)

This question follows the exercises in Chapter 2 of the Dask book starting with 2.1 on page 65.
Continue with the tutorial and count the missing values in the data set.
Estimate the percentage of missing data in each field (listings 2.2 - 2.4 in book)
Get the list of columns with a lot of missing data and drop them; persist the results. (listing 2.5)

Submit:

1. Commands matching listings 2.2-2.4 that run successfully on your system.

Example Submission:

```
: # perform the calculation of missing data by column
  with ProgressBar() as pb:
      missing_pct_out = missing_pct.compute()
```

```
[########################################] | 100% Completed |  1min 13.6s
```

```
: # display results of missing data
  missing_pct_out
```

```
: Summons Number                    0.000000
  Plate ID                          0.006739
  Registration State                0.000000
  Plate Type                        0.000000
  Issue Date                        0.000000
  Violation Code                    0.000000
  Vehicle Body Type                 0.395361
```

```
: # identify columns with more than 60% missing data
  columns_to_drop = missing_pct_out[missing_pct_out > 60].index
```

```
: # Review columns to be dropped
  columns_to_drop
```

```
: Index(['Time First Observed', 'Intersecting Street', 'Violation Legal Code',
         'Unregistered Vehicle?', 'Meter Number',
         'No Standing or Stopping Violation', 'Hydrant Violation',
         'Double Parking Violation'],
        dtype='object')
```

```
: # drop the bad columns and persist results in df_dropped
  with ProgressBar() as pb:
      df_dropped = df.drop(columns_to_drop, axis=1).persist()
```

```
[########################################] | 100% Completed |  1min  5.0s
```

Question 3: Visualize a DAG in Dask (0.5 points)

Continue with the Dask book with listing 2.6, creating simple functions inc and add. Visualize the results of a simple calculation.
Hint: You will need to install pygraphviz first
(ac295) $ conda install graphviz python-graphviz

Submit:

1. Notebook that includes visualization of the DAG for z as in the exercise

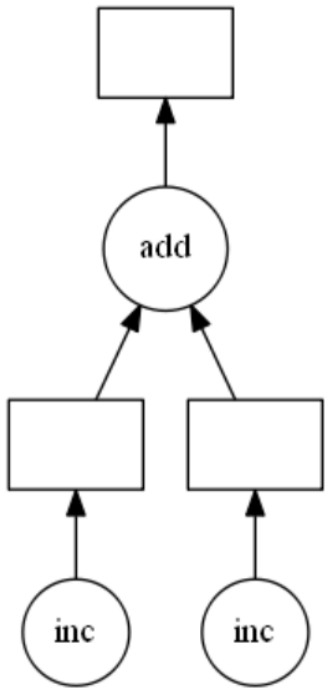
Example Submission:

```
# define delayed functions (listing 2.6)
def inc(i):
    return i+1

def add(x, y):
    return x + y
```

```
# Create a simple example computation
x = delayed(inc)(1)
y = delayed(inc)(1)
z = delayed(add)(x, y)

# Visualize the result
z.visualize()
```



Question 4: Compute Pi with a Slowly Converging Series (2.0 points)

Complete the remainder of the tutorial in chapter 2. You will learn to do simple arithmetic calculations and persist intermediate results.

One of the oldest known series for π was published by Leibniz in 1676. While this is easy to understand and derive, it converges very slowly.

https://en.wikipedia.org/wiki/Leibniz_formula_for_%CF%80

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \cdots$$

While this is a truly atrocious way to compute π if you were serious, it's a fun opportunity to use brute force on a problem instead of thinking.

Compute π using at least one billion terms in this sequence. On my PC, this took 27.5 seconds.

Submit:

1. Jupyter notebook calculation of π as described above

Example Submission:

```
: with ProgressBar() as pb:
      sum_flt = series_sum.compute()
```

```
[########################################] | 100% Completed | 27.5s
```

```
: # Error
error = np.abs(sum_flt - np.pi)

# Report results
print(f'Sum of 4*(1 - 1/3 + 1/5 - ...) for n over {terms} terms:')
print(f'series sum = {sum_flt:14.12f}')
print(f'pi         = {np.pi:14.12f}')
print(f'error      = {error:6.3e}')
```

```
Sum of 4(1 - 1/3 + 1/5 - ...) for n over 1000000000 terms:
series sum = 3.141592652590
pi         = 3.141592653590
error      = 9.999e-10
```

Question 5: Filter Parking Tickets Dataset (1.0 points)

Work through the examples in chapter 3 of the Dask book and learn how to filter a Dask DataFrame. According to the documentation for the parking tickets data set, the column called 'Plate Type' consists mainly of two different types: 'PAS' and 'COM', presumably for passenger and commercial vehicles respectively. (Maybe the rest are the famous parking tickets from the diplomats the UN, who take advantage of diplomatic immunity to not pay their fines.)
Create a filtered Dask DataFrame with only the commercial plates.
Persist it so it is available in memory for future computations.
Count the number of summonses in 2017 that were issued to commercial plate types.
Compute them as a percentage of the total data set.
Hint: This is easy; it's only about 5 lines of code

Submit:

1. Output in your Jupyter notebook with the count and percentage of commercial citations

Example Submission:

```
: # Calculate percentage of commercial vehicles
pct_commercial = num_commercial / num_total * 100.0

# Add random numbers to not spoil the answer
num_commercial_print = np.int(num_commercial + np.random.uniform(-500000, 500000))
pct_commercial_print = pct_commercial + np.random.uniform(-5.0, 5.0)

# Print the answers with random noise added
print(f'The number of NYC summonses with commercial plates in 2017 was {num_commercial_print}')
print(f'The percentage of NYC summonses with commercial plates in 2017 was {pct_commercial_print:5.2f}')
```

```
The number of NYC summonses with commercial plates in 2017 was 1750026
The percentage of NYC summonses with commercial plates in 2017 was 18.20
```