Urban Data Science

USP 410/510 | Spring 2026

Dr. Liming Wang

Portland State University

Part 1: Course Overview

About This Course

An interdisciplinary approach to understanding, managing, and designing the city using **data-driven theories and methods**

- **Project-based** — learn by doing
- **AI-assisted** — learn to work with LLMs and coding agents
- **No prerequisites** — just curiosity and tolerance for experimentation
- Builds on information processing, statistics, and computer science for **urban applications**

Course Objectives

1. **Code** — Automate tasks with Python
2. **AI tools** — Use LLMs and AI agents for coding, analysis, and problem solving
3. **Workflow** — Project management best practices for data work
4. **Data skills** — Access, clean, visualize, and analyze urban data
5. **Domain knowledge** — Combine technical and substantive expertise to solve urban problems

Grading

COMPONENT	USP 410	USP 510
DataCamp exercises (4 x 5pts)	20%	20%
Data science show & tell (2 x 5pts)	10%	10%
Assignments (2)	30%	20%
Project presentation	10%	10%
Project report	30%	40%

Assignments

Both use **ODOT crash data**

A1 — Exploring Crash Data (due W4)

- Investigate questions of your choosing
- e.g., Does DST increase crashes?
- e.g., Nighttime pedestrian fatality patterns

A2 — Interactive Crash Map (due W8)

- Go beyond ODOT's existing viewer
- Build a map with a point of view
- e.g., Dangerous corridors, equity, safe routes near schools

AI Policy

ACTIVITY	AI USE
DataCamp	**Not permitted**
Show & tell	Research only
Assignments	**Encouraged**
Final project	**Encouraged**

> AI use on assignments is **encouraged and recommended** — this is how modern data science is done.

Class Project

The final product can be a **report**, **infographic**, or **dashboard**

- Generated using Python and/or Quarto
- Presentation: 20 min + 5 min Q&A

MILESTONE	DUE
Project idea	W3 (04/16)
Project proposal (1 page)	W6 (05/07)
Progress update	W8 (05/21)
Project presentation	W11 (06/11)
Final submission	06/12

Schedule at a Glance

WEEK	DATE	TOPIC
W1	**04/02**	**Overview, Setup, Intro to Python**
W2	04/09	LLMs and AI agents
W3	04/16	Data import/export, cleaning & processing
W4	04/23	Workflow & project management
W5	04/30	Exploring and visualizing data
W6	05/07	Reproducible research; Quarto & Jupyter
W7	05/14	Spatial data and maps
W8	05/21	Public data from the web and APIs

Textbooks & Resources

All freely available online:

- **Yu & Barter** — *Veridical Data Science* (vdsbook.com)
- **Downey** — *Think Python*, 3rd Ed. (allendowney.github.io/ThinkPython)
- **Turrell** — *Python for Data Science* (aeturrell.github.io/python4DS)
- **McKinney** — *Python for Data Analysis*, 3rd Ed. (wesmckinney.com/book)
- **Rey et al.** — *Geographic Data Science with Python* (geographicdata.science/book)

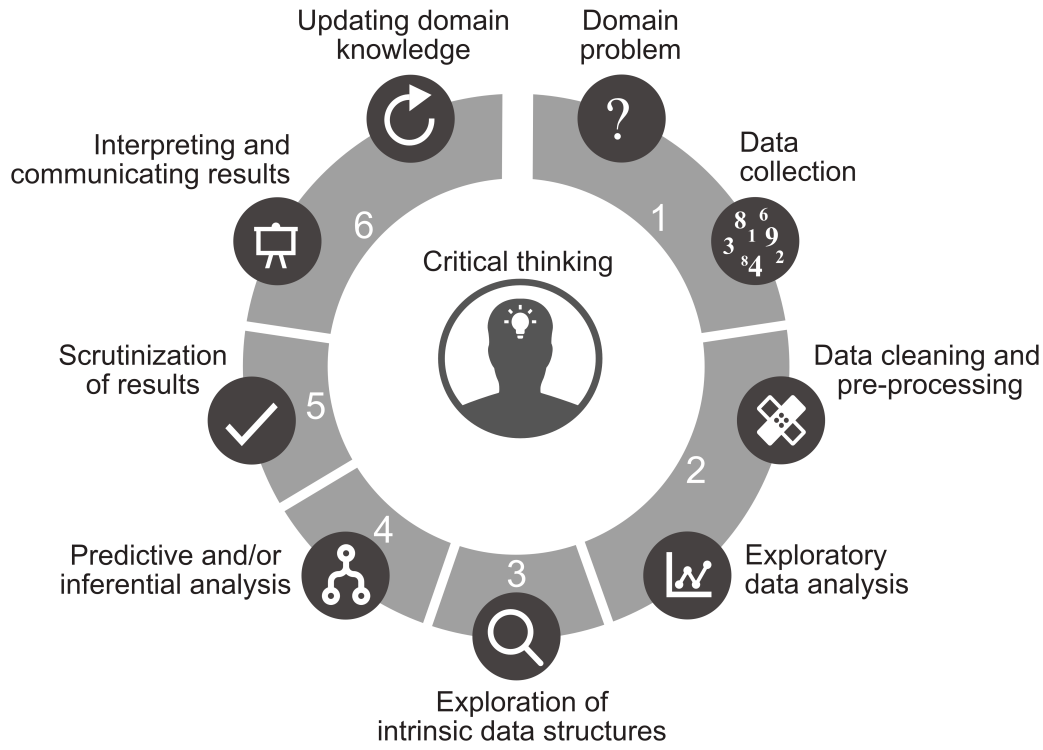
DataCamp: Free access via DataCamp Classroom program

Part 2: The Data Science Life Cycle

Based on Yu & Barter, *Veridical Data Science*, Chapter 2

What is the Data Science Life Cycle?

A **six-stage framework** for how data science projects progress



Source: Yu & Barter, *Veridical Data Science*, vdsbook.com

Stage 1: Problem Formulation & Data Collection

- **Collaborate with domain experts** to turn vague questions into answerable ones
- Identify what data is available — or what needs to be collected
- Understanding the project goals drives data requirements

Urban data science example

> "Is Portland getting less safe for pedestrians?"

becomes

> "Have pedestrian-involved crashes in Portland increased in frequency or severity from 2015 to 2024, controlling for changes in population and VMT?"

Stage 2: Data Cleaning & EDA

The stage where you spend **most of your time**

- **Data cleaning**: Make datasets tidy, properly formatted, unambiguous
- **Preprocessing**: Adapt cleaned data for specific analyses
- **Exploratory Data Analysis (EDA)**: Summaries and visualizations to understand patterns
- **Explanatory Data Analysis**: Refine exploratory work for external audiences

> "The process of data cleaning is necessarily subjective and involves making assumptions about the underlying real-world quantities being measured."
> — Yu & Barter

Stages 3–6

Stage 3: Intrinsic Structures (optional)

- Dimensionality reduction, cluster analysis
- Discover natural groupings in the data

Stage 4: Prediction & Inference (optional)

- Supervised learning, regression, classification
- Focus on accuracy with validation/test data

Stage 5: Evaluation

- Is the result *stable* and *predictable*? (PCS framework)
- Consult domain experts; use negative controls

Stage 6: Communication

- Reports, infographics, dashboards, applications
- Tailor the message to the audience

Key Concepts from DSLC

Data structure vocabulary

- **Variables/features** = columns; **observations** = rows
- **High-dimensional** = many variables (>100)

Data quality

- **Live data**: actively maintained, errors corrected
- **Dead data**: static snapshots, no monitoring

Data snooping ⚠️

- Presenting discovered patterns as proven conclusions
- Veridical Data Science counters this with *predictability* and *stability*

Non-linear progression

- You will cycle back through stages — that's normal and expected

Part 3: Computer Setup

Tools We'll Use

Python (via Miniconda or uv)

- The programming language for this course

Google Antigravity

- AI-powered IDE with built-in coding agents
- Free, based on VS Code, supports Gemini & Claude models

Quarto

- Render notebooks to HTML/PDF/Word

Git & GitHub

- Version control and collaboration

Setup checklist

- ☐ Install Python
- ☐ Install Google Antigravity
- ☐ Install Quarto
- ☐ Create a GitHub account
- ☐ Verify: `python --version`
- ☐ Verify: `quarto --version`

Why Python?

- **Free and open source** — no license barriers
- **General purpose** — not just for statistics (unlike R)
- **Massive ecosystem** — pandas, matplotlib, geopandas, folium, streamlit, …
- **Industry standard** — used at Google, Meta, NASA, city governments
- **AI tools work best with Python** — first-class support in Antigravity, Claude Code, and other AI coding agents
- **Readable syntax** — designed to be close to natural language

Google Antigravity

Our IDE for the course — an **agent-first** development environment

- **Free** with generous Gemini rate limits; also supports Claude models
- Built on VS Code — familiar editor interface with all the extensions you know
- **Two views**:
  - **Editor view** — write code with an AI agent sidebar (like Copilot, but deeper)
  - **Manager view** — orchestrate multiple agents working in parallel
- Agents can **plan, execute, and verify** tasks across your editor, terminal, and browser
- Download at antigravity.google

Part 4: Introduction to Python

Based on Downey, *Think Python*, Chapter 1

Programming as a Way of Thinking

Programming combines features from:

- **Mathematics** — formal notation, abstract reasoning
- **Engineering** — building things that work, testing and debugging
- **Natural science** — observing behavior, forming hypotheses, experimenting

> The most important skill is **learning to experiment** — try things, break things, read error messages, and try again.

Arithmetic in Python

Python as a calculator:

```
30 + 12         # Addition → 42
43 - 1          # Subtraction → 42
6 * 7           # Multiplication → 42
84 / 2          # Division → 42.0 (always returns a float!)
85 // 2         # Integer division → 42
7 ** 2          # Exponentiation → 49
```

Order of operations: same as math (PEMDAS)

```
(12 + 5) * 6    # → 102  (parentheses first)
12 + 5 * 6      # → 42   (multiplication before addition)
```

Data Types

Every value in Python has a **type**

```
type(42)       # → int     (integer — whole numbers)
type(42.0)     # → float   (floating point — decimals)
type('hello')  # → str     (string — text)
```

Type conversion:

```
int(42.9)      # → 42     (truncates, does not round!)
float(42)      # → 42.0
str(42)        # → '42'
```

Watch out: division always returns a float

```
84 / 2         # → 42.0   (not 42)
```

Strings

Text is enclosed in quotes (single `'...'` or double `"..."`)

```
'Hello' + ' ' + 'World'   # Concatenation → 'Hello World'
'Spam, ' * 3              # Repetition → 'Spam, Spam, Spam, '
len('Portland')           # Length → 8
```

Note: `+` and `*` behave differently for strings vs. numbers

```
3 + 4          # → 7        (arithmetic)
'3' + '4'      # → '34'     (concatenation)
```

Common error:

```
'3' + 4        # → TypeError! Can't mix strings and numbers
```

Built-in Functions

Functions take inputs and produce outputs:

```
round(3.14159)       # → 3
round(3.14159, 2)    # → 3.14
abs(-7)              # → 7
len('data science')  # → 12
type(42)             # → <class 'int'>
print('Hello!')      # displays: Hello!
```

Calling a function: `function_name(argument1, argument2, ...)`

You've already been using functions — `type()`, `len()`, `int()`, `float()`, `str()` are all functions

Formal vs. Natural Languages

	NATURAL LANGUAGE	PROGRAMMING LANGUAGE
Ambiguity	Common and tolerated	Not allowed
Redundancy	Verbose for clarity	Concise and precise
Literalness	Full of idioms and metaphor	Means exactly what it says

Implication for learning:

- You can't skim code the way you skim English
- Small differences in spelling and punctuation matter
- **Error messages are your friend** — read them carefully

Let's Try It

Open a Python interpreter or Jupyter notebook and try:

```
# What is your age in days (approximately)?
age_years = 25
age_days = age_years * 365
print(age_days)

# How many seconds in a year?
seconds_per_year = 365 * 24 * 60 * 60
print(seconds_per_year)

# What type is each of these?
print(type(age_years))
print(type(age_years / 2))
print(type('Portland'))
```

Experiment: What happens if you type `age_years / 0`?

For Next Week

Read:

- Karpathy, How I Use LLMs (video)
- Evkaya & de Carvalho, Using ChatGPT for Data Science Analyses (HDSR, 2026)

Do:

- Complete computer setup (Python, VS Code, Quarto, GitHub)
- Start **DataCamp DC1**: Introduction to Python (due W3)
- Sign up for show & tell slots in the shared Google Doc