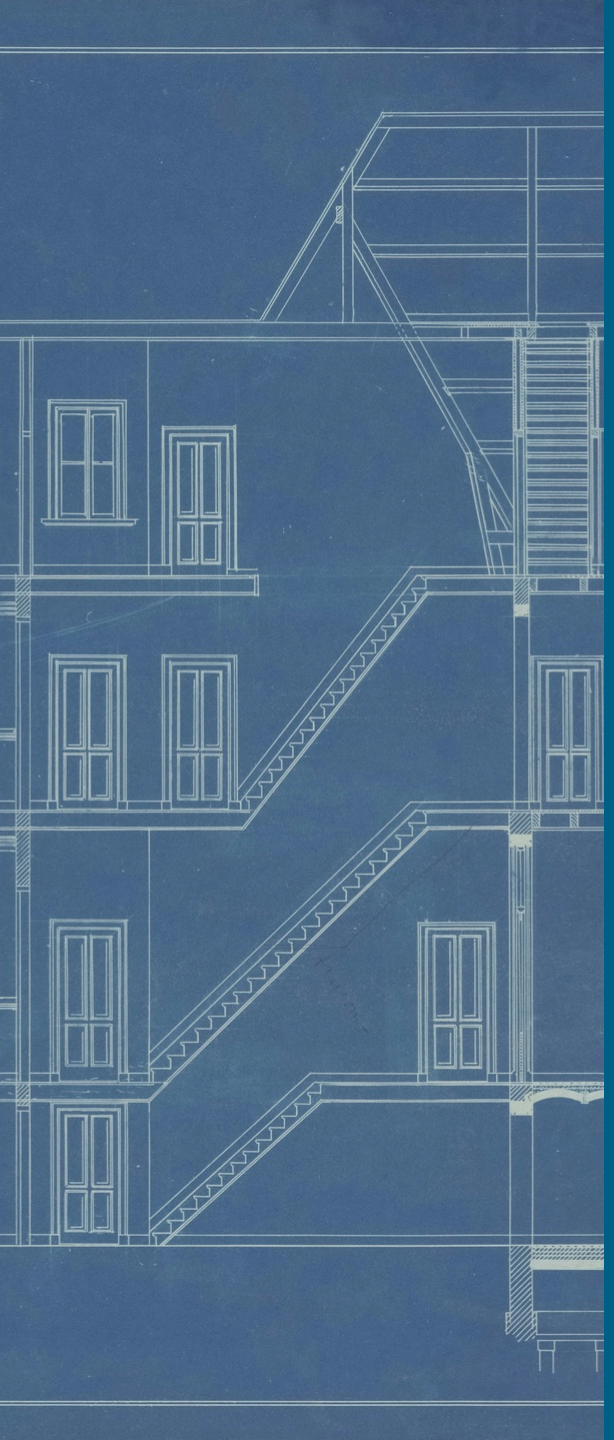
FEBRUARY 14TH, 2025

Modules, Packages, and Libraries

CE 311K - L14



Review: Classes

Classes are **blueprints** for objects

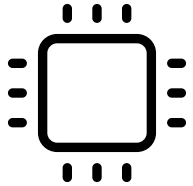
```
class Engineer:
    def __init__(self, name):
        self.name = name
    def describe(self):
        return "Hello, " + self.name
```

Classes define an object's **attributes** and **methods** which can be accessed via dot notation

- attributes: variables that have unique values per object instance
- **methods:** functions that perform operations for that object


Review Poll

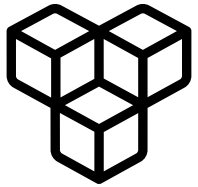
Imports: Modules, Packages, and Libraries



Module

A single file that contains code

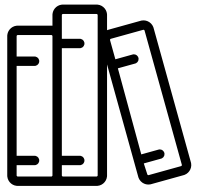
Functions, classes, variables, and runnable statements



Package

A collection of related modules grouped together in a directory

Represents a higher-level organization of functionality



Library

A collection of modules and packages providing ready-made functionality for a specific purpose

Libraries are usually shared for others' uses

Using Imports

Imports are useful for organization, reusability, simplification, and collaboration

import statement allows you to use external code in your program

```
import math  # Importing the math module
print(math.sqrt(16))  # Accessing a function from the math module
```

You can import specific functions or classes with *from*

```
from math import sqrt, pi
print(sqrt(16))  # Accessing directly, without `math.`
print(pi)        # Accessing the constant directly
```

import

Create a function, *pythag*, that takes two parameters corresponding to the lengths of the legs of a right triangle and returns the value of the hypotenuse.

$$c = \sqrt{a^2 + b^2}$$

Use the sqrt() function provided by the math library

Package Managers

A **package manager** is a tool that automates the process of installing, upgrading, configuring, and removing software

Package managers are common and unique to a language

Python: pip, conda

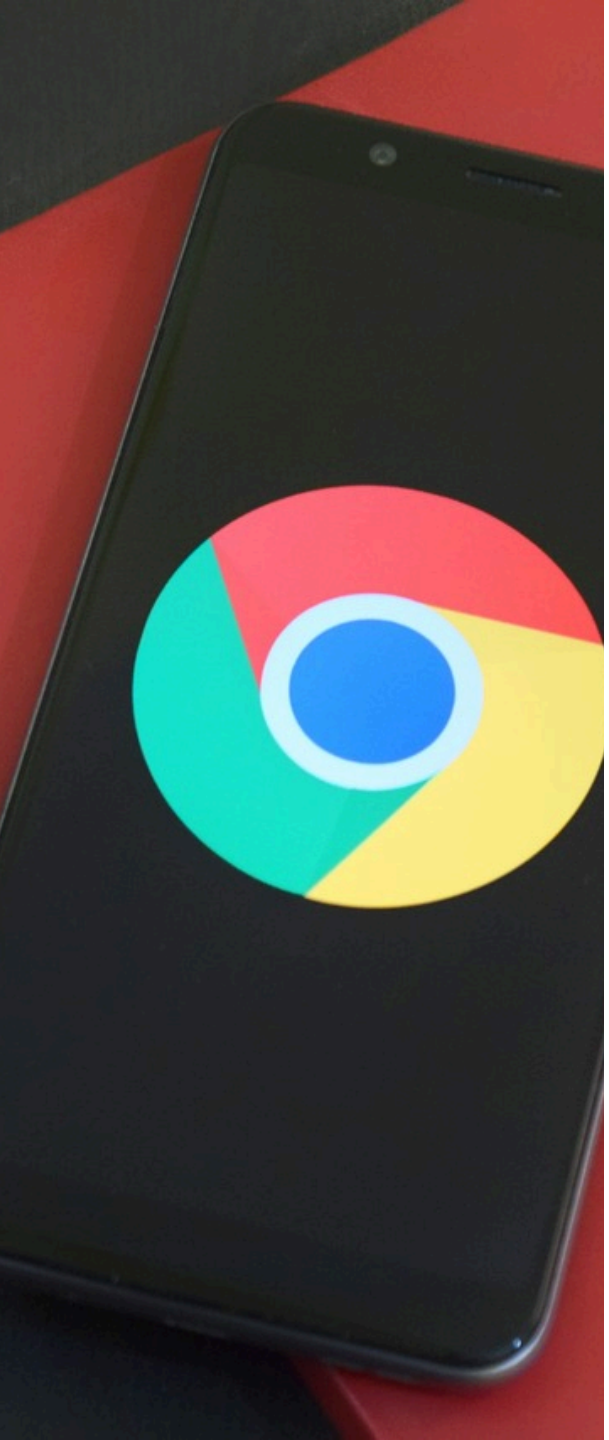
JavaScript: npm, yarn

Ruby: gem

Package managers are additional tools that are either bundled together with the language or downloaded separately

Use command line interface (CLI) to install libraries

```
pip install requests
```



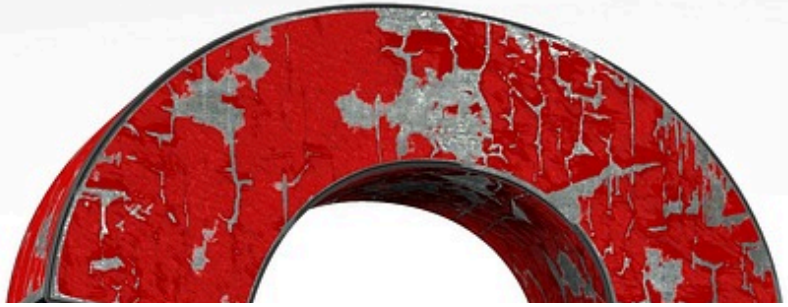
Installing Packages in Colab

Colab has many popular libraries pre-installed - you can install additional packages using *%pip*

```
%pip install pandas
```

Use *%pip list* to see all installed packages

```
%pip list
```



Install Packages/Libs

In Colab, use *pip* to install the *ProPyCore* library

Import ProPyCore in a new cell after installing to check if it was successful

General Purpose Libraries

os

Interacting with the operating system i.e. file paths, environment variables, etc.

sys

Accessing system-specific parameters and functions i.e. command-line arguments, Python paths, etc.

re

Working with regular expressions for pattern matching in strings

math

Performing advanced arithmetic, trigonometry, logarithmic, and statistical calculations

datetime

Working with time and dates

Data and File Handing

csv

For reading and writing CSV files

json

For working with JSON data

openpyxl

For working with Excel files

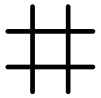
sqlite3, sqlalchemy, mongodb

Built-in library for SQL-based databases

Data Analysis and Visualization

pandas

Powerful tools for data manipulation and analysis, using DataFrames for structured data



numpy

Fast, efficient operations on arrays and matrices, plus mathematical functions

matplotlib

Low-level plotting library for creating static, animated, and interactive plots

seaborn

High-level statistical data visualization built on Matplotlib

scipy

Provides advanced scientific and engineering tools i.e. optimization, integration, etc.

statsmodels

Statistical models and hypothesis testing

Machine Learning

scikit-learn

Tools for machine learning, including classification, regression, clustering, and preprocessing

tensorflow

A library for deep learning and neural networks

pytorch

An alternative to TensorFlow, popular in the research community

Web Scraping

requests

For sending HTTP requests to interact with web services

beautifulsoup4

For parsing and extracting data from HTML and XML

selenium

For automating web browsers to scrape or test web applications.

datetime

Create, manipulate, and format date and time objects for tasks like scheduling, time calculations, and timestamping events

Python Datetime Documentation

```python
from datetime import datetime

now = datetime.now()                                  # 2024-12-23 10:30:45.123456
specific_dt = datetime(2025, 12, 25, 0, 5)            # 2025-12-25 00:05:00
formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")  # 2024-12-23 10:30:45
dt_str = "2025-01-01 14:30:00"
parsed = datetime.strptime(dt_str,"%Y-%m-%d %H:%M:%S") # 2024-12-23 14:30:00
```

Summary

modules< package < library

- Module: single file that contains code
- Package: collection of related modules
- Library: collection of modules and packages providing ready-made functionality

Use *import* to bring in external code

Include *from* to access specific classes/functions

You can **alias** an import name with *as*

Package Managers help you install third-party libraries

For Python, use *pip* or *conda*

Don't reinvent the wheel

Modern languages have thousands of libraries to handle common and niche applications