Programming with Python

Lesson planning and learning goals:

# Lecture design

1. look at the course learning goals, and see which are relevant to this lecture

2. look at content from last year, see which goals we might want to add  
Aim: 1 major and 2 minor learning goals

3. given the learning goals, plan the activities that can help you reach those

# Unit 3.1 (Object-Oriented Programming)

Learning goals:

After this lecture, students should be able to:

* Explain the fundamentals of object-oriented programming (OOP) in Python

Topics:

* classes

Teaching activities and assessment methods:

* Review data structures
  + Make them store a list of students in a list
  + Make them store them in a dictionary so they can look up by name
* Motivate object-oriented programming: list of students that need to be enrolled in multiple courses
* LibreOffice presentation
* Go through Jupyter notebook

# Unit 3.2 (Files and Pandas)

Learning goals:

After this lecture, students should be able to:

* open/close and read/write files
* summarize what the Pandas library is used for

Topics:

* File I/O with open
* Working with CSV files using pandas

Teaching activities:

* question on slide to get them started
* review of last lecture
* general motivation for rest of course
* Why: original question. Take ideas and write them down
* Show dataset, pose question again, give them 3 minutes
* Jupyter notebook

Assessment methods:

* gather ideas on how to tackle the problem, multiple times in this lecture
* challenges embedded in the Jupyter notebook
* exercises after lecture

# Unit 3.3 (Advanced Pandas)

Learning goals:

After this lecture, students should be able to:

* effectively perform advanced DataFrame operations in Pandas, including grouping and handling missing data

Topics:

* Pandas join
* Pandas handling missing data

Teaching activities:

* Jupyter notebook with embedded challenges

Assessment methods:

* Exercises on the Jupyter notebook

# Unit 4.1 (Data visualization with matplotlib)

Learning goals:

After this lecture, students should be able to:

* (must) represent a distribution as a histogram
* (nice) correlate two variables visually with a scatter plot
* (nice) make use of the matplotlib documentation to find how to run a method

Topics:

* Why do we need to visualize?
* Histograms
* Scatter plots
* Modify invocations to change plot properties

Teaching activities:

* One-minute paper about Pandas
* Think-Pair-Share about why we need visualization
  + Say we had the McDonald’s menu; we want to visualize calories
* Jupyter notebook
* Think-Pair-Share about modifying the hist invocation
* Final discussion:
  + One-minute paper
  + Think-Pair-Share

Assessment methods:

* Exercises on the Jupyter notebook

# Unit 4.2 (Working with date and time)

Learning goals:

After this lecture, students should be able to:

* (must) show the current date and its individual attributes
* (nice) compare two datetime objects
* (nice) make use of the datetime documentation to find how to run a method

Topics:

* Datetime.date basics:
  + get the current date
  + retrieve the attributes of a date
* Datetime versus datetime.datetime
* timedelta
* datetime.date advanced:
  + date from ISO
  + date.replace
  + date.weekday()

Teaching activities:

* Jupyter notebook with embedded activities
* Final discussion:
  + One-minute paper

Assessment methods:

* Exercises on the Jupyter notebook

# Unit 4.3 (Unit 4.3: Matrix computation)

Learning goals:

After this lecture, students should be able to:

* (must) perform basic operations (+-\*/) on elements of ndarrays
* (nice) carry out addition or subtraction of two ndarrays
* (nice) perform matrix multiplication

Topics:

* + Ndarray instantiation
    - Array, zeros, ones, itentity
  + Incrementing all elements by 1
  + Adding two matrices
  + Multiplying two matrices

Teaching activities:

* Jupyter notebook with embedded activities
* Final discussion:
  + Think-Pair-Share (of the whole day)

Assessment methods:

* Exercises on the Jupyter notebook

# Unit 5.1:(Lecture on error handling)

Learning goals:

After this lecture, students should be able to:

* (must) implement a function that catches and handles an exception
* *(nice to have)* implement a function that catches and handles a specific class of exception
* *(nice to have)* write a custom Error class

Topics:

* Error handling using try and except

Teaching activities:

* Recap: Pandas dataframes
* Now we have the information in a dataframe. Next? 3 minutes. What about errors?
* Show how to handle errors. What do we do about those errors?

Assessment methods:

* gather ideas on how to tackle the problem, multiple times in this lecture
* challenges embedded in the Jupyter notebook
* exercises after lecture