CS 5010 – Final Exam – Study Guide

Topics that may be covered on the Final Exam *(note, this is **not** an exhaustive list): Exam 1 and Exam 2 topics may show up again in the context of the Final Exam. Focus will be on Modules 7-10 (not Module 11). Reference Exam 1 and Exam 2 Study Guides for topics covered.*

1. Python Coding / CS Programming Concepts (**General**) – *Should any Python coding/reading of code or related CS programming topics be present on the exam, follow these general guidelines*
    a. Be able to read / write Python code
    b. Be able to show the output of a piece of code
    c. Be able to compare two pieces of code
    d. Describe the (many!) benefits of applying object-oriented principles
    e. Describe what the major data structures are and their properties (and scenarios in which they would be best suited – e.g. lists, dictionaries, and tuples)
    f. Decide on an appropriate data structure(s) to use given the scenario
    g. Correctly implement the appropriate data structure(s) for the given scenario and manipulate data within this structure correctly and appropriately
    h. Describe the major functions used and their properties (and scenarios in which they would be best suited)
    i. Be familiar with searching (*linear* and *binary*) and sorting algorithms. You *will* be expected to demonstrate knowledge of the algorithms by answering questions about them, or deciding which one is most appropriate. Built-in sorting algorithms will not be covered on this exam.
    j. A high-level question might be asked on NumPy & Pandas (*no coding* will be involved, however) (Related: querying data: high-level questions; no code.)
    k. Explain some of the elements that results in "good quality" code
2. Software Development Methods/Models
    a. Waterfall model
        i. Pros/Cons
        ii. Understand it is "historical" (not in use today as-is)
    b. Spiral model
        i. Pros/Cons
        ii. Understand it is "historical" (not in use today as-is)
    c. Why learn about these methods?
        i. Inject quality into software
        ii. Avoiding problems early that cause huge problems later
        iii. Software engineering is not just about writing code
    d. "Plan-driven" methodologies
        i. "Think it through to the end"
        ii. Document-driven
        iii. Heavy-weight
        iv. How "traditional" engineering has been done

e. "Agile" methodologies
   i. *"Focus on mostly just the next thing"*
   ii. Less focus on documentation
   iii. Frequent interaction with customer
   iv. Iterative development
   v. Flexible, ability to change
   vi. Light-weight
   vii. Relatively new (since 1990s)

f. Agile approaches
   i. Extreme Programming (XP)
   ii. XP made popular techniques that have been widely adopted:
      - Pair-programming
      - Unit-testing
      - Test First (test driven development)
   iii. SCRUM (see next section)

3. SCRUM

a. Explain what Scrum is– (Agile process)
b. Focus on delivering the highest business value in the shortest time
c. User Stories (and Acceptance criteria) – *"As a ___, I want ___."*
d. Product Backlog
e. Sprint and Sprint Backlog
f. Tasks and task assignment

4. Python: Aggregates

An understanding of the most common methods with respect to aggregates using NumPy and Pandas:

a. Aggregates using NumPy
   i. Use of 'reduce' method with another function
      - If you would like to reduce an array with a particular operation you can use the reduce method with another function. The outcome of using reduce will repeatedly apply a given operation to the elements of an array until only a single result remains
   ii. Use of 'accumulate' method with another function for keeping intermediate results
      - If you want to store all the intermediate results of the computation, use, instead, the 'accumulate' function
   iii. Computing summary statistics
      - Common summary statistics include mean and standard deviation, but other aggregates are useful as well, such as: sum, product, median, minimum, maximum, etc..
      - Summing the values in an array
   iv. Understanding that NumPy versions of methods (*vs. built-in functions*) execute quicker
   v. Using multidimensional aggregates (along a row or column)
      - E.g. with sum, min, and max
   vi. Familiarity with other aggregate functions that exist, such as:
      - np.mean, np.std, np.var, np.argmin, np.argmax, np.median, np.percentile, ...

b. **Aggregates using Pandas**
   i. Simple Aggregation in Pandas
      - Previously we explored some of the data aggregations available for NumPy arrays. As with a one-dimensional NumPy array, for a Pandas Series the aggregates return a single value.
      - Examples using sum(), mean(), and min()
   ii. Remember that for a DataFrame, by default the aggregates return results within *each* column
   iii. Familiarity with some built-in Pandas aggregations, such as:
      - count(), first(), last(), mean(), median(), min(), max(), std(), var(), prod(), and sum()
   iv. GroupBy
      - Split, Apply, Combine
      - What do each of these steps involve?
      - Groupby a desired key column
      - Groupby and then apply an aggregate (e.g. sum())

c. **GroupBy Mechanics**
   i. General syntax: Selecting the data and then selecting the grouping (followed by applying an aggregate function)
      - E.g. can use sum(), mean(), etc as the aggregate function
   ii. Familiarity with the fact you can group by more than one key
      - E.g. group first by key1 then by key2)
   iii. Remember, a single value is no longer produced, rather the result of the aggregate function for EACH grouping. Output looks mostly like a *table*. Often times a hierarchical row or column axis

Some Suggested Resources/References:

[Module 1] Data conceptualization and Python
- ➢ 01-Data Conceptualization.pdf
- ➢ 01-The Challenger-An Information Disaster.pdf
- ➢ Python 3-X Basics.pdf
- ➢ Module exercises
- ➢ Python Scripts 1-13
- ➢ Homework 1: Data Quiz

[Module 2] Python Part II
- ➢ 02-Python – Map Filter Reduce.pdf
- ➢ Python Scripts 14-21
- ➢ Python Scripts sphere, spheretest, newton, newtontest, newtontest2
- ➢ Homework 2: Python

[Module 3] Python Part III

- 1-Private, Protected, and Public in Python.pdf
- 2-Delete in Python.pdf
- 3-Functions and Parameter List in Python.pdf
- Python Scripts 22-26
- Python exception hierarchy: https://docs.python.org/3/library/exceptions.html#exception-hierarchy

[Module 4] Intro. To Searching, Sorting, and Data Wrangling

- 03-Additional Resource - Binary Search.pdf
- pyScript27_SortingIntro.py
- Review the resources for NumPy and Panda (4.6)
- 04-Brief Introduction to NumPy.pdf
- 05-Brief Introduction to Pandas.pdf
- Querying data Discussion exercise:
  - Key_Measures_December_2013_Mod.csv ; read_dataMod.py
- Homework 3: Python and Web Scraper (Module 4)

[Module 5] Software Testing and Debugging

- AStudent_Class_courses_test.py
- AStudent_Class_numCourses_test.py
- AStudent_Class.py
- primes.py ; primes_fail1.py ; primes_fail2.py ; primes_test.py
- Behind Pythons unittest-main.pdf
- Homework 4: Testing and Debugging (Module 5)
- Module 5 Knowledge Check: Python Unit Testing

[Module 6] Software Engineering

- 08-SWEngineering-(1).pdf
- 08-Functional and non-functional reqs.pdf
- 08-Functional and non-functional reqs - Solutions.pdf
- Module 6 Knowledge Check: Software Engineering Quiz
- Module 6 Knowledge Check: Is This a Requirements Statement?

[Module 7] Software Development Methods and Models

- Module 7 Knowledge Check: Software Development Methods and Models

[Module 8] SCRUM: Agile Methodology

- SIS User Story Example.pdf
- Word User Story Example.pdf
- Homework 5: SCRUM (Module 8)

[Module 9] SCRUM Activity

- Module 9 Knowledge Check: The Scrum Process

[Module 10] Aggregates Using NumPy and Pandas

- Module 10 - Aggregates _Part I - NumPy_.ipynb
- Module 10 - Aggregates _Part II - Pandas_.ipynb
- Module 10 - Aggregates _Part III - More Examples_.ipynb

Notes:

- It might be clear but worth stating, knowledge on *how to use or install* Anaconda or Spyder ***will not*** be tested on this or any future exam
- Advanced printing techniques (including "format string") ***will not*** be explicitly tested on the exam, however you're welcome to use it as part of your solution if you wish
- Redefining/redirecting standard output (**stdout**) to print from the screen (standard output) to a file (see `pyScript06.py` and other related scripts) ***will not*** be explicitly tested on the exam, however you're welcome to use it as part of your solution if you wish
- Web Scraping / Web Crawling ***will not*** be explicitly tested on the exam.
- Specifics of NumPy / Pandas ***will not*** be explicitly tested on the exam – only high-level questions may be asked.
- Python visualizations ***will not*** be explicitly tested on the exam, however you're welcome to use it as part of your solution if you wish.

--*--

Like the first two exams, Exam 3 – the Final Exam, will take place online on the Ford testing site. Remember, clicking on the exam link will open up the exam immediately and the timer will begin (so only click the provided link once you're ready to begin!) The Final Exam is also open-book/open-notes but absolutely no collaboration is permitted – it must be individual work. Always state assumptions next to the question you are answering when in doubt.

Good Luck! ☺

--*--