C S 21B: INTERMEDIATE PYTHON PROGRAMMING

Foothill College Course Outline of Record

Heading	Value
Effective Term:	Summer 2021
Units:	4.5
Hours:	4 lecture, 2 laboratory per week (72 total per quarter)
Advisory:	C S 3A or 21A or relevant experience.
Degree & Credit Status:	Degree-Applicable Credit Course
Foothill GE:	Non-GE
Transferable:	CSU/UC
Grade Type:	Letter Grade (Request for Pass/No Pass)
Repeatability:	Not Repeatable

Student Learning Outcomes

- A successful student will be able to develop a Python program that runs other programs, accesses a database, and transfers files over a network.
- A successful student will be able to develop an event driven Python program that interacts with the user through a graphical user interface that employs windows, dialog boxes, buttons, menus and text fields.

Description

This course builds on the student's prior knowledge of the Python programming language by offering a more in-depth and advanced approach to building effective Python applications. Specific topics include user interfaces, networked applications, databases, multithreading and regular expressions. The course reinforces object oriented design, thorough documentation, testing and conventional programming style.

Course Objectives

The student will be able to:
A. Understand Python's memory model and issues with mutability.
B. Recognize various aspects of Python code that exhibit better performance.
C. Discuss implementation differences between the standard data types.
D. Distinguish between Python 2 and 3, use migration tactics, discuss porting issues, and write code compatible with both versions.
E. Write code that executes other (Python and non-Python) programs as well as programmatically-generated Python code.
F. Use the standard Python developer and testing tools.
G. Write Python code with fewer bugs and other issues.

Course Content

A. Intensive review and expansion of C S 21A
1. Python standard data types: numbers, sequences, hashed types
2. Function (and method) decorators
3. Files and input/output
4. Anonymous functions and lambda
5. Exception handling
6. Functions: defining, execution, parameters (default and keyword)
7. Modules and packages
8. Object-oriented programming
B. Python 2 vs. Python 3
1. Differences/changes
2. Migration tactics and tools
3. Porting issues
4. Writing compatible code
5. Compatibility libraries
6. Changes to integer literals
7. Changes to built-in functions and methods
C. Exceptions
1. Updated 3.x syntax
2. Errors vs. warnings
3. Context management
D. Functions and functional programming
1. Anonymous functions
2. Variable arguments
3. Parameter expansion
4. Inner functions
5. Closures
E. Language constructs and data structures
1. List comprehensions vs. generator expressions
2. Iterators vs. generators
F. Best practices
1. Programmer tools
2. Testing and test tools
3. Documentation and documentation tools
4. Review of operating systems theory
5. Performance
G. Various areas of application development (topics time-permitting)
1. Regular expressions
2. Network programming
3. Multithreaded programming
4. GUI programming
5. Microsoft Office (COM client) programming
6. Databases (SQL and NoSQL)
7. Python extensions
8. Web programming
9. Internet client programming
10. Accessing Google APIs

Lab Content

A. Writing functional programs
1. Write functions that contain keyword and variable arguments
2. Write an inner/nested function
3. Use function and method decorators
4. Define an anonymous function using lambda
5. Use currying and partial function application
6. Distinguish between variable scope and namespaces
7. Employ closures
8. Write an advanced generator
B. Object oriented programming
1. Define a new class using composition
2. Define a new class using inheritance
3. Write programs that take advantage of polymorphism, introspection, delegation and wrapping
C. Execution environment
1. Employ callable objects and code objects
2. Write a program that executes another program

3. Make calls to the operating system
D. Write a program that is executable by both Python 2 and Python 3 interpreters
E. Write a program that contains list comprehensions, generator expressions
F. Write a program that employs iterators and generators
G. Write a program that opens, closes, deletes, reads and writes files
H. Write a program that communicates with a relational or a nonSQL database
I. Write a program that performs unit and other tests
J. Write programs that contain one or more of the following:
1. Regular expressions
2. Network programming
3. Multithreaded programming
4. GUI programming
5. Microsoft Office (COM client) programming
6. Databases (SQL and NoSQL)
7. Python extensions
8. Web programming
9. Internet client programming
10. Accessing Google APIs

Special Facilities and/or Equipment

A. When offered on/off campus: Access to a PSME-provided Unix/Linux shell account. The shell account should reside on a Unix/Linux host with the latest stable version of Python. Lecture room equipped with white/black board, and instructor workstation with internet connectivity attached to an LCD projector.
B. A website or course management system with an assignment posting component (through which all lab assignments are to be submitted) and a forum component (where students can discuss course material and receive help from the instructor). This applies to all sections, including on-campus (i.e., face-to-face) offerings.
C. When taught via Foothill Global Access on the Internet, the college will provide a fully functional and maintained course management system through which the instructor and students can interact.
D. When taught via Foothill Global Access on the Internet, students must have currently existing email accounts and ongoing access to computers with internet capabilities.

Method(s) of Evaluation

Programming assignments and projects
Midterm exam
Final exam

Method(s) of Instruction

Lectures which include motivation for syntax and use of the Python language and programming concepts, example programs, and analysis of these programs
Online labs (for all sections, including those meeting face-to-face/on-campus), consisting of:
1. A programming assignment webpage located on a college-hosted course management system or other department-approved internet environment. Here, the students will review the specification of each programming assignment and submit their completed lab work
2. A discussion webpage located on a college-hosted course management system or other department-approved internet environment. Here, students can request assistance from the instructor and interact publicly with other class members
Detailed review of programming assignments which includes model solutions and specific comments on the student submissions
Face to face or online discussion which engages students and instructor in an ongoing dialog pertaining to all aspects of designing, implementing and analyzing programs
When course is taught fully online:
1. Instructor-authored lecture materials, handouts, syllabus, assignments, tests, and other relevant course material will be delivered through a college-hosted course management system or other department-approved internet environment
2. Additional instructional guidelines for this course are listed in the attached addendum of CS department online practices

Representative Text(s) and Other Materials

Chun, Wesley. Core Python Language Fundamentals, 3rd ed.. 2012.

This text is still current and considered to be well-suited for instruction in this course.

Types and/or Examples of Required Reading, Writing, and Outside of Class Assignments

A. Textbook assigned reading averaging 30 pages per week.
B. Reading the supplied handouts and modules averaging 10 pages per week.
C. Reading online resources as directed by instructor though links pertinent to programming.
D. Reading library and reference material directed by instructor through course handouts.
E. Writing technical prose documentation that supports and describes the programs that are submitted for grades.

Discipline(s)

Computer Science