College of San Mateo
Official Course Outline

1. **COURSE ID:** CIS 502 **TITLE:** Applied Python Programming
   Units: 4.0 units **Hours/Semester:** 64.0-72.0 Lecture hours; and 128.0-144.0 Homework hours
   Method of Grading: Grade Option (Letter Grade or Pass/No Pass)
   Prerequisite: CIS 117

2. **COURSE DESIGNATION:**
   Degree Credit
   Transfer credit: CSU; UC

3. **COURSE DESCRIPTIONS:**
   Catalog Description:
   The course introduces advance topics in Python programing and Python software development. The course also introduces some important Python libraries.

4. **STUDENT LEARNING OUTCOME(S) (SLO'S):**
   Upon successful completion of this course, a student will meet the following outcomes:
   1. Use Comprehensions, Metaclasses, Closure and Decorators in developing Python programs.
   2. Build programs using Generators and Descriptors.
   3. Describe some of Python design patterns.
   4. Use Lambdas , Multithread and Multiprocessing in Python programming.

5. **SPECIFIC INSTRUCTIONAL OBJECTIVES:**
   Upon successful completion of this course, a student will be able to:
   1. Use Comprehensions, Metaclasses, Closure and Decorators in developing Python programs.
   2. Build programs using Generators and Descriptors.
   3. Describe some of Python design patterns.
   4. Use Lambdas , Multithread and Multiprocessing in Python programming.

6. **COURSE CONTENT:**
   Lecture Content:

   Introduction
   1. Jupyter Notebook System

   Everything in Python is Object
   1. Objects, methods, and attributes

   Comprehensions
   1. List Comprehensions
   2. Dictionary Comprehensions
   3. Set Comprehensions
   4. Generator Comprehensions

   Metaclasses
   1. The Metaclass model
   2. Class statement protocol
   3. Declaring Metaclasses
   4. Coding Metaclasses
   5. Metaclass versus superclass
   6. Metaclass methods

   Extended Keyword Arguments (*args, **kwargs)
   1. *args
   2. **kwargs

   Closures and Decorators
   1. Function Decorators

2. Class Decorators
3. Decorator nesting
4. Decorator arguments
5. Decorator state retention options
6. Replacing or tweaking the original object
7. Decorator implemented as classes and as functions
8. Closure

Generators, Generator Expressions, and Iterators

1. Iterators
2. Generator expressions
3. Generators
4. Bidirectional communication
5. Chaining generators

Context Managers

1. Catching exceptions
2. Using generators to define context manager

Descriptor

1. Descriptor Protocol
2. Invoking Descriptor

Patterns

1. Singleton
2. Adapter
3. Proxy
4. Facade
5. Observer
6. Visitor
7. Template

Operator Overloading

1. Index Iteration
2. Iterable objects
3. User-defined Iterables
4. Attribute assignment and deletion
5. String representation
6. Right-Side and In-Place
7.Boolean test
8. Object desctruction

Anonymous functions, Lambda expressions, and Lambda functions

1. Anonymous functions
2. Lambda functions
3. Lambda expressions
4. Lambda abstractions
5. Lambda form
6. Functions literals
7. Map, Filter, and Reduce

Conventions

1. Wrapping instead of inheritance
2. Dependency injections
3. Factories
4. Duck typing
5. Monkey patching
6. Callbacks

Concurrency, Parallelism, and Mutiprocessing

1. Concurrency versus Parallelism
2. Thread versus Process versus Task
3. Multithreading versus Multiprocessing
4. Levels of Concurrency
5. Fetch-Decode-Excecute-Cycle
6. Multiprocessing
7. Asynchronous programming
8. Working with Threads
9. Producer-Consumer Threading
10. The Process class
11. Exchanging objects between Processes
12. Synchronization between Processes
13. Sharing state between Processes
14. Using Pool of Workers
15. Explicit shared memory parallelism using pthreads, or fork (), pipe()
16. Implicit shared memory parallelism

Modern approaches to Python development

1. Application-level isolation of Python environments
2. Virtual environment
3. System-level environment isolation
4. Containerization versus virtualization
5. Popular productivity tools

Deploying Code

1. The filesystem hierarchy
2. Isolation
3. Using process supervision tools
4. Code instrumentation and monitoring

Managing Code

1. Version control systems
2. Continuous development process
3. Building the documentation

Test-Driven development

1. Test-driven developmental principles
2. Preventing software regression
3. Improving code quality
4. Acceptance tests
5. Unit tests
6. Integration tests
7. Load and performance testing
8. Code quality testing
9. Python standard test tools

Optimization and Profiling

1. Optimization strategy
2. Finding bottlenecks
3. Reducing the complexity
4. Simplifying
5. Catching

Introduction to some Python Libraries

1. Numpy
2. Pandas
3. SciKit-Learn
4. Keras
5. NLTK
6. PyTorch

7. **REPRESENTATIVE METHODS OF INSTRUCTION:**
   Typical methods of instruction may include:
   A. Lecture
   B. Activity
   C. Discussion
   D. Observation and Demonstration
   E. Other (Specify): Student participation in short in-class projects. Students working in small groups to solve problems.

8. **REPRESENTATIVE ASSIGNMENTS**
   Representative assignments in this course may include, but are not limited to the following:
   Writing Assignments:
   Students will be assigned weekly homework problems from the required textbook.
   Reading Assignments:
   Students will read all chapters of the required textbook, reading parallel current assignments, and lecture content.
   Other Outside Assignments:
   Weekly homework problems
   Internet research

9. **REPRESENTATIVE METHODS OF EVALUATION**
   Representative methods of evaluation may include:
   A. Class Participation
   B. Class Performance
   C. Class Work
   D. Exams/Tests
   E. Homework
   F. Quizzes
   G. Written examination

10. **REPRESENTATIVE TEXT(S):**
    Possible textbooks include:
    A. Ziade, Tarek. *Expert Python Programming*, 3 ed. Packt Publishing, 2019
    B. Zaccone, Giancarlo. *Python Parallel Programming Cookbook*, 2 ed. Packt Publishing, 2019
    C. Lanaro, Gabriel. Kasampalis, Sakis. *Advanced Python Programming*, ed. Packt Publishing, 2019

Origination Date: August 2020
Curriculum Committee Approval Date: November 2020
Effective Term: Fall 2021
Course Originator: Kamran Eftekhari