Miami Dade College

Course Description

COP2041C | Python Programming 2 | 4.00 Credits

This is an advanced-level programming course using Python. Students will learn how to code, compile, and execute programs. Topics include problem-solving, object-oriented programming, database programming, data visualization, and secure software development. Prerequisites COP1047C.

Course Competencies

Competency 1: The student will demonstrate an understanding of problem-solving by:

1. Describing what problem-solving is and what common problem-solving strategies are.
2. Identifying inputs, outputs, and actions necessary to solve a problem.
3. Formulating an algorithm to solve a problem using pseudocode, UML, flowcharts, etc.
4. Implementing an algorithm.
5. Describing what use cases are and their role in the development process.
6. Identifying the different elements of a use case (e.g., actors, stakeholders, preconditions, triggers, success scenarios, alternative paths, etc.)

Competency 2: The student will demonstrate knowledge of the software development process by:

1. Describing software development process life cycle models (such as waterfall, spiral, agile, incremental, etc.).
2. Describing process activities within a specified life cycle process model.
3. Designing or tailoring a software process to the needs of a project team or development activity.
4. Designing and implementing a software test plan.
5. Designing, implementing, and executing test cases.
6. Identifying test objectives.
7. Identifying, collecting, and storing appropriate data resulting from testing/demonstration.
8. Identifying, assigning, and performing necessary corrective actions such as debugging, refactoring, etc.
9. Analyzing test data for test coverage, effectiveness, and process improvement.

Competency 3: The student will demonstrate an understanding of the object-oriented programming concepts of class and object by:

1. Designing classes, objects, interactions, and attributes using UML.
2. Creating programs that use classes, objects, and attributes.
3. Creating a class with the appropriate behaviors and attributes.
4. Identifying and using instance variables and instance methods.
5. Creating and using the appropriate in-it method.
6. Explaining the process of object instantiation.
7. Recognizing the difference between class and object.
8. Defining and implementing encapsulation.

Competency 4: The student will demonstrate an understanding of inheritance by:

1. Explaining the benefits of inheritance and polymorphism.
2. Creating a sub-class that inherits from a parent class.
3. Explaining the restrictions imposed when using inheritance.
4. Explain the pros and cons of multiple inheritance.
5. Overriding and overloading parent class methods within a sub-class.

Competency 5: The student will demonstrate an understanding of exception programming techniques by:

1. Describing exceptions.
2. Using try-except blocks to handle built-in exceptions.

3. Using else and finally clauses.
4. Demonstrating the use of raising exceptions.
5. Creating new exceptions from existing ones.

Competency 6: The student will demonstrate an understanding of modules and packages by:

1. Demonstrating how to create functions grouped into modules.
2. Demonstrating the ability to import custom modules.
3. Describing “from” and “as” keywords.
4. Documenting a module and all its published functionality.
5. Using built-in modules (such as OS, Math, Random, etc.)
6. Using popular Python modules and packages (such as NumPy, Pandas, Matplotlib, etc.)

Competency 7: The student will demonstrate knowledge of Python database programming by:

1. Describing what a database is and the different types of databases.
2. Explaining database concepts such as schema, constraints, table, query, record, field, etc.
3. Creating programs connecting and accessing databases using common Python DB-APIS such as SQLite3, MySQL, and cx-oracle.
4. Using SQL statements to create and drop tables.
5. Using SQL statements to retrieve and modify records.

Competency 8: The student will demonstrate knowledge of data visualization by:

1. Describing what data visualization is and its uses in data analysis.
2. Describing different pictorial representations (scatter plot, line chart, bar chart, histogram, etc.) And their uses.
3. Creating programs using standard Python data visualization libraries such as Matplotlib, seaborn, and Plotly.

Competency 9: The student will demonstrate knowledge of secure software development by:

1. Creating programs that use mitigation techniques, such as input validation, sanitation, and input size checking, to deal with common input manipulation errors.
2. Purging sensitive information from exceptions to avoid information exposure through error messages.
3. Writing programs that follow proper coding style guidelines, such as using proper variable names, commenting, code formatting conventions, etc.
4. Limiting the life of sensitive data in the program by restricting the scope of variables and objects.
5. Avoiding code duplication by properly using methods and classes with specific functionality.
6. Explaining coupling and how to achieve loose coupling.
7. Explaining cohesion and how to achieve high cohesion.
8. Describe the concepts of encryption and hashing and their role in secure software development.

Competency 10: The student will demonstrate an understanding of how Python is used in different fields/industries by:

1. Explaining why Python is so helpful and popular across many different fields and industries.
2. Describing the use of Python in different fields and industries such as AI and machine learning, cybersecurity, data analytics, finance, quantum programming, etc. course

Competency 11: The student will demonstrate an understanding of graphical user interface (GUI) programming in Python by:

1. Describing the fundamentals of GUI programming, including event-driven programming, widgets, and user interface elements.
2. Creating basic GUI applications using Tkinter or a similar library, including windows, buttons, labels, text fields, and layout management.

3. Implementing event handling in GUI applications to respond to user actions such as button clicks and text input. D) designing a user-friendly interface that adheres to good design and usability principles.
4. Incorporating advanced widgets into GUI applications, such as menus and dialogs, to enhance functionality and user experience.
5. Applying best practices for secure GUI development, including validating user input and handling sensitive data appropriately.
6. Evaluating and debugging GUI applications to ensure reliability, efficiency, and user satisfaction.

Learning outcomes:

1. Communicate effectively using listening, speaking, reading, and writing skills.
2. Solve problems using critical and creative thinking and scientific reasoning.
3. Use quantitative analytical skills to evaluate and process numerical data.
4. Use computers and emerging technologies effectively.
5. Formulate strategies to locate, evaluate, and apply information.