Course ID: CNIT R161
Curriculum Committee Approval Date: 10/24/2018
Catalog Start Date: Fall 2019

COURSE OUTLINE

OXNARD COLLEGE

I. Course Identification and Justification:
   A. Proposed course id: CNIT R161
      Banner title: Programming Essentials Python
      Full title: Programming Essentials in Python

   B. Reason(s) course is offered:
      This course is offered to expand the program into new technologies that are in demand in the field of computer networking/IT and as a response to a recommendation by the CNIT Industry Advisory Council. This course will also help prepare students for upper division coursework in the IT program at CSUCI.

   C. C-ID:
      1. C-ID Descriptor: ITIS 130
      2. C-ID Status: Approved

   D. Co-listed as:
      Current: None

II. Catalog Information:
   A. Units:
      Current: 3.00

   B. Course Hours:
      1. In-Class Contact Hours: Lecture: 43.75 Activity: 0 Lab: 26.25
      2. Total In-Class Contact Hours: 70
      3. Total Outside-of-Class Hours: 87.5
      4. Total Student Learning Hours: 157.5

   C. Prerequisites, Corequisites, Advisories, and Limitations on Enrollment:
      1. Prerequisites
         Current:

      2. Corequisites
         Current:

      3. Advisories:
         Current:

      4. Limitations on Enrollment:
         Current:

   D. Catalog description:
      Current:

How great would it be to write your own computer program or design a modern web or desktop application? Both are a possibility if you learn how to code in Python. Python is the very versatile, object-oriented programming language used by startups and tech giants, Google, Facebook, Dropbox and IBM. Python is also recommended for aspiring young developers who are interested in pursuing careers in security, networking and Internet-of-Things. This course utilizes the Cisco Networking Academy PCAP Python curriculum.

E. Fees:
Current: $ None

F. Field trips:
Current:
Will be required: [ ]
May be required: [X]
Will not be required: [ ]

G. Repeatability:
Current:
A - Not designed as repeatable

H. Credit basis:
Current:
Letter graded only [X]
Pass/no pass [ ]
Student option [ ]

I. Credit by exam:
Current:
Petitions may be granted: [X]
Petitions will not be granted: [ ]

III. Course Objectives:
Upon successful completion of this course, the student should be able to:
A. Summarize what distinguishes Python from other programming languages.
B. Identify basic programming concepts.
C. List what distinguishes the different versions of the Python programming language.
D. Create a computer program using the Python programming language.
E. Use primitive data types and data structures offered by the development environment.
F. Implement standard modules provided by Python.
G. Use Boolean values to compare difference values and control the execution paths.
H. Identify the basic methods of formatting and outputting data offered by Python.
I. Apply core program control structures.
J. Test Python created applications with sample data.
K. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.

IV. Student Learning Outcomes:
A. Summarize what distinguishes python from other programming languages.
B. Create a computer program using the python programming language.
C. Identify errors and problem-solve using an algorithmic approach.

V. Course Content:

Topics to be covered include, but are not limited to:

A. Introduction
   1. Python popularity
   2. Versions of the Python programming language
   3. What is object oriented programming
   4. Creating a basic Python program

B. Modular Design

C. Control and Evaluation
   1. Basic concepts: interpreting and the interpreter, compilation and the compiler, language elements, lexis, syntax and semantics, Python keywords, instructions, indenting
   2. Literals: Boolean, integer, floating-point numbers, scientific notation, strings
   3. Operators: unary and binary, priorities and binding
   4. Numeric operators: ** * / % // + –
   5. Bitwise operators: ~ & ^ | << >>
   6. String operators: * +
   7. Boolean operators: not and or
   8. Relational operators (== != > >= <
   9. Assignments and shortcut operators
   10. Accuracy of floating-point numbers
   11. Basic input and output: input(), print(), int(), float(), str() functions
   12. Formatting print() output with end= and sep= arguments
   13. Conditional statements: if, if-else, if-elif, if-elif-else
   14. The pass instruction
   15. Simple lists: constructing vectors, indexing and slicing, the len() function
   16. Simple strings: constructing, assigning, indexing, slicing comparing, immutability
   17. Building loops: while, for, range(), in, iterating through sequences
   18. Expanding loops: while-else, for-else, nesting loops and conditional statements
   19. Controlling loop execution: break, continue

D. Data Aggregates
   1. Strings in detail: ASCII, UNICODE, UTF-8, immutability, escaping using the \ character, quotes and apostrophes inside strings, multiline strings, copying vs. cloning, advanced slicing, string vs. string, string vs. non-string, basic string methods (upper(), lower(), isxxx(), capitalize(), split(), join(), etc.) and functions (len(), chr(), ord()), escape characters
   2. Lists in detail: indexing, slicing, basic methods (append(), insert(), index()) and functions (len(), sorted(), etc.), del instruction, iterating lists with the for loop, initializing, in and not in operators,list comprehension, copying and cloning
   3. Lists in lists: matrices and cubes
   4. Tuples: indexing, slicing, building, immutability
   5. Tuples vs. lists: similarities and differences, lists inside tuples and tuples inside lists
   6. Dictionaries: building, indexing, adding and removing keys, iterating through dictionaries as well as their keys and values, checking key existence, keys(), items() and values() methods

E. Functions and Modules
   1. Defining and invoking your own functions and generators
   2. Return and yield keywords, returning results, the None keyword, recursion
   3. Parameters vs. arguments, positional keyword and mixed argument passing, default parameter values
   4. Converting generator objects into lists using the list() function
   5. Name scopes, name hiding (shadowing), the global keyword
   6. Lambda functions, defining and using

7. Map(), filter(), reduce(), reversed(), sorted() functions and the sort() method
8. The if operator
9. Import directives, qualifying entities with module names, initializing modules
10. Writing and using modules, the __name__ variable
11. Pyc file creation and usage
12. Constructing and distributing packages, packages vs. directories, the role of the __init__.py file
13. Hiding module entities
14. Python hashbangs, using multiline strings as module documentation

F. Classes, Objects, and Exceptions
1. Defining your own classes, superclasses, subclasses, inheritance, searching for missing class components, creating objects
2. Class attributes: class variables and instance variables, defining, adding and removing attributes, explicit constructor invocation
3. Class methods: defining and using, the self-parameter meaning and usage
4. Inheritance and overriding, finding class/object components
5. Single inheritance vs. multiple inheritance
6. Name mangling
7. Invoking methods, passing and using the self-argument/parameter
8. The __init__ method
9. The role of the __str__ method
10. Introspection: __dict__, __name__, __module__, __bases__ properties, examining class/object structure
11. Writing and using constructors
12. Hasattr(), type(), issubclass(), isinstance(), super() functions
13. Using predefined exceptions and defining your own ones
14. The try-except-else-finally block, the raise statement, the except-as variant
15. Exceptions hierarchy, assigning more than one exception to one except branch
16. Adding your own exceptions to an existing hierarchy
17. Assertions
18. The anatomy of an exception object
19. Input/output basics: opening files with the open() function, stream objects, binary vs. text files, newline character translation, reading and writing files, bytearray objects
20. Read(), readinto(), readline(), write(), close() methods

G. Program Development Lifecycle

H. Requirements Determinants and Analysis

VI. Lab Content:

A. The fundamentals of computer programming, i.e. how the computer works, how the program is executed, how the programming language is defined and constructed, what the difference is between compilation and interpretation, what Python is, how it is positioned among other programming languages, and what distinguishes the different versions of Python.

B. The basic methods of formatting and outputting data offered by Python, together with the primary kinds of data and numerical operators, their mutual relations and bindings; the concept of variables and variable naming conventions; the assignment operator, the rules governing the building of expressions; the inputting and converting of data.

C. Boolean values to compare difference values and control the execution paths using the *if* and *if-else* instructions; the utilization of loops (*while* and *for*) and how to control their behavior using the *break* and *continue* instructions; the difference between logical and bitwise operations; the concept of lists and list processing, including the iteration provided by the *for* loop, and slicing; the idea of multi-dimensional arrays.

D. The defining and using of functions – their rationale, purpose, conventions, and traps; the concept of passing arguments in different ways and setting their default values, along with the mechanisms of returning the function's results; name scope issues; new data aggregates: tuples and dictionaries, and their role in data processing

E. Python modules: their rationale, function, how to import them in different ways, and present the content of some standard modules provided by Python; the way in which modules are coupled together to make packages; the concept of an exception and Python's implementation of exceptions, including the *try-except* instruction, with its applications, and the *raise* instruction; strings and their specific methods, together with their similarities and differences compared to lists.

F. The fundamentals of OOP (Object Oriented Programming) and the way they are adopted in Python, showing the difference between OOP and the classical, procedural approach; the standard objective features: inheritance, abstraction, encapsulation, and polymorphism, along with Python-specific issues like instance vs. class variables, and Python's implementation of inheritance; objective nature of exceptions; Python's generators (the *yield* instruction) and closures (the *lambda* keyword); the means Python developers can use to process (create, read, and write) files.

VII. Methods of Instruction:

Methods may include, but are not limited to:

A. Lecture on course topics including but not limited to control and evaluations, data aggregates, functions and modules, and classes, objects, and exceptions.

B. Demonstrations on Python programming including but not limited to the following: A. Simple lists: constructing vectors, indexing and slicing, the len() function B. Simple strings: constructing, assigning, indexing, slicing comparing, immutability C. Building loops: while, for, range(), in, iterating through sequences D. Expanding loops: while-else, for-else, nesting loops and conditional statements E. Controlling loop execution: break, continue F. Dictionaries: building, indexing, adding and removing keys, iterating through dictionaries as well as their keys and values, checking key existence, keys(), items() and values() methods G. Converting generator objects into lists using the list() function H. Import directives, qualifying entities with module names, initializing modules I. Writing and using modules, the __name__ variable J. Constructing and distributing packages, packages vs. directories, the role of the __init__.py file K. Class attributes: class variables and instance variables, defining, adding and removing attributes, explicit constructor invocation L. Invoking methods, passing and using the self-argument/parameter

C. Videos on Python programming topics from the Cisco Networking Academy and the Python Institute.

VIII. Methods of Evaluation and Assignments:

A. Methods of evaluation for degree-applicable courses:
Essays [ ]
Problem-solving assignments (Examples: Math-like problems, diagnosis & repair) [X]
Physical skills demonstrations (Examples: Performing arts, equipment operation) [ ]

For any course, if "Essays" above is not checked, explain why.
This is a computer programming course and writing essays is not a requirement for students to achieve the stated student learning outcomes.

B. Typical graded assignments (methods of evaluation):

1. Quizzes and exams on topics in the course content section.
2. Lab assignment answers will be evaluated and graded to determine the correctness of the answers.
3. Python programming code will be submitted to determine proper techniques and execution.

C. Typical outside of classroom assignments:
   1. Reading
      a. Students are required to read and study the information in the assigned chapter of the curriculum in between classes in order to be prepared for the lecture and classroom lab activities. A typical reading activity would be for the students to read the material on Python functions and then create and run Python functions.
   2. Writing
      a. Students are required to answer the questions in the lab assignments to demonstrate that they grasp the material.
      b. Students are required to respond to questions posed at the current Oxnard College LMS portal. An example would be for a student to respond to a question explaining why Python is such a popular programming language with the most popular Internet companies.
   3. Other
      a. In order to prepare for the Python Institute PCAP certification, students will be required to answer certification preparation questions included in the Cisco Networking Academy Python course.

IX. Textbooks and Instructional Materials:
   A. Textbooks/Resources:
      1. Cisco Networking Academy (2017). *Programming Essentials in Python* Cisco Networking Academy.
      2. "Edube ." Python Institute, 1st ed.
         Description: A browser-based Python sandbox that allows users to create and test Python code.
   B. Other instructional materials:

X. Minimum Qualifications and Additional Certifications:
   A. Minimum qualifications:
   B. Additional certifications:
      1. Description of certification requirement:
         PCAP – Certified Associate in Python Programming
      2. Name of statute, regulation, or licensing/certification organization requiring this certification:
         Cisco Networking Academy

XI. Approval Dates
Curriculum Committee Approval Date: 10/24/2018
Board of Trustees Approval Date: 11/13/2018
State Approval Date:
Catalog Start Date: Fall 2019

XII. Distance Learning Appendix

A. Methods of Instruction

Methods may include, but are not limited to:

1. A VCCCD approved learning management system (LMS) will be utilized to facilitate synchronous and asynchronous communication which includes but is not limited to the following: Live chat, asynchronous discussion forums, direct messages, grading assignments and providing feedback to students regarding performance on the assignments. The instructor may also utilize the LMS to provide course progress updates. Distance education instructors may also use videoconferencing for the course (CCCConfer/Skype) as they deem appropriate.

B. Information Transfer

Methods may include, but are not limited to:

1. Chat/IM
2. Course announcements
3. Discussion boards
4. E-Mail
5. Instructor-provided online materials
6. Messaging via the LMS
7. Modules on the LMS
8. Personalized feedback
9. Textbooks
10. Videoconferencing/CCCConfer/Skype