Python Essentials 1

Scope and Sequence

Version 1.0

Developed in collaboration with

Contents

Target Audience

The Python Essentials 1 course is designed for learners looking to gain fundamental skills in Python and computer programming, including:

- Learners with little or no prior programming knowledge;
- Students of secondary schools, universities, and vocational schools;
- Industry professionals exploring technologies connected with Python, or that use Python as a foundation for building complex projects;
- Team leaders, products managers, and project managers who want to understand terminology and processes in the software development cycle to more effectively manage and communicate with IT, testing, and development teams;
- Anyone interested in learning programming for fun or job-related purposes.

Prerequisites

There are no prerequisites for this course, although possessing basic knowledge in mathematics is helpful. All learners are welcome to sign up!

Certification Alignment

The Python Essentials 1 course prepares learners for the PCEP^{TM} – Certified Entry-Level Python Programmer. The PCEP^{TM} certification (Exam PCEP-30-0x) is a professional credential that measures the candidate's ability to accomplish coding tasks related to the essentials of programming in the Python language. A test candidate must demonstrate sufficient knowledge of the universal concepts of computer programming, the syntax and semantics of the Python language, as well as the skills in resolving typical implementation challenges with the help of the Python Standard Library.

Course Description

Python Essentials 1 was delveloped by the OpenEDG Python Institute, and is provided to all audiences to enhance, develop, and support professional careers in Python programming and related technologies. The course covers the basics of Python programming, as well as general computer programming concepts and techniques. The course familiarizes learners with the procedural approach to programming and covers the following topics:

- Fundamental terminology, concepts, and definitions
- Basic Python syntax, semantics, and the runtime environment
- Literals, variables, numeral systems, operators, and data types
- Basic I/O operations
- Control flow: conditional execution and loops

- Data collections: lists, tuples, dictionaries, and strings
- Functions
- Exceptions, troubleshooting, and debugging

The course is comprised of 4 modules. Learners have access to hands-on practice materials, quizzes, and tests as well as interact with some real-life programming tasks and situations –- all to develop the skills and knowledge gained within the course.

Course Objectives

1. Develop an awareness of programming languages

1.1 Identify programming language design approaches

1.2 Explain the components of programming languages

1.3 Examine connections between elements of mathematics and computer science, including binary numbers, logic (including the Boolean Algebra), sets, and functions

2. Demonstrate proficiency using specialized computer software

2.1 Use specialized computer programming software to solve problems (e.g., editor, console, IDLE, IDE, debugger)

2.2 Demonstrate proficiency using specialized computer software (e.g., Python, Edube Interactive Programming Environment)

3. Demonstrate knowledge, skills, and application of IT and communication systems to accomplish job objectives and enhance workplace performance

3.1 Develop keyboarding skills to enter and manipulate text and data (e.g., using keyboard shortcuts, code writing)

3.2 Describe and use current and emerging computer technology and software to perform personal and business-related tasks (e.g., prepare a to-do list, plan a project, write pseudo-code)

3.3 Perform a variety of operations such as data sorting and filtering, or organizing and displaying information in a variety of ways such as number formats (conversion to different numeral systems or units)

4. Demonstrate proficiency in using common software applications

4.1 Compare and contrast the appropriate use of various software applications and programs (e.g. compiler vs. interpreter)

4.2 Demonstrate proficiency in the use of various software applications and programs (e.g. code editors)

4.3 Explain why different file types exist (e.g., source files, semi-compiled files, .py files)

4.4 Identify the kinds of content associated with different file types

5. Demonstrate comprehension and communication

5.1 Use reading and writing skills and strategies to communicate effectively (reading documentation, writing self-commenting code)

5.2 Organize ideas and communicate written messages (e.g. using comments)

5.3 Recognize that more than one algorithm can solve a given problem

5.4 Create a program that implements an algorithm to achieve a given goal

6. Explore the characteristics, tasks, career paths, and tools available for starting a career in software development and related technologies

6.1 Explore a variety of careers which utilize computing and algorithmic skills, and investigate career opportunities in programming and related technologies

6.2 Discuss the impact of computing on business, commerce, and organizations (e.g., automated inventory processing, financial transactions, virtualization, cloud computing)

6.3 Discuss ethical responsibilities of the programmer and evaluate the impacts of plagiarism and copyright infringement on their projects, life, and career

6.4 Identify and analyze tasks performed by programmers and testers

6.5 Explain the need for continuing education of programmers

6.6 Understand, identify, and use the right tools and methods to support lifelong learning

6.7 Identify credentials and certifications that may improve employability for a computer programmer

7. Solve problems using critical thinking skills, creativity, and innovation

7.1 Employ critical thinking skills to solve problems and make decisions

7.2 Conduct technical research to gather information necessary for decision-making

7.3 Discuss and employ the digital tools or resources necessary for a real-world tasks based on their efficiency and effectiveness

8. Write and document code using the PEP8 guidelines, coding conventions, and best practices

8.1 Use appropriate naming conventions to define program variables, methods, functions, and classes

8.2 Use internal documentation (e.g., single-line and multi-line comments, module docstrings) to document a program according to accepted standards

8.3 List examples of good programming practices in Python (e.g. self-commenting code)

8.4 Use proper formatting and code layout (e.g. not exceeding the recommended line length)

8.5 Understand the importance of using coding conventions; use the most important PEP8 guidelines

9. Demonstrate proficiency in basic programming and Python

9.1 Describe the structure of a simple program, and explain why sequencing is important

9.2 Define the term algorithm and explain how it relates to problem solving

9.3 Describe, create, and use iterative programming structures (e.g. loops)

9.4 Describe, create, and use conditional structures and explain the logic used for them (if, else, else-if, finally)

9.5 Explain the types and use of variables in programming (e.g. naming, creating, initializing, and modifying variables in Python.)

9.6 Write simple programs in pseudo code

9.7 Find, analyze, describe, troubleshoot, and debug errors in code

9.8 Explain the fundamental programming concepts, such as compiler, interpreter, source code, machines code, Ide, etc., and use skills learned to install and configure basic development tools

9.9 Discuss the difference between syntax and semantics of a programming language; describe the basic data types, identifiers, and keywords in Python

9.10 Demonstrate knowledge of Python's development history and its main traits and features

9.11 Design, create, edit, and run Python source files using IDLE

9.12 Use Python numeral literals, their syntax, types, and formats

9.13 Understand and use Python assignment operators, arithmetic operators, and expressions

9.14 Perform basic I/O operations in Python programs

9.15 Work with Boolean data types and their features

9.16 Work with relational operators in Python

9.17 Perform bitwise operations in Python

9.18 Operate with lists to perform indexing, slicing, and content manipulation

9.19 Employ multidimensional lists in Python

9.20 Apply the concept of functions and be able to create and invoke user-created functions

9.21 Employ the main features of structural programming

9.22 Understand the concept of name scopes, name shadowing, and distinguish between global and local variables

9.23 Utilize the principles of tuples, including the immutability notion

9.24 Use dictionaries effectively and in appropriate circumstances

9.25 Manipulate strings using basic processing techniques

9.26 Employ various techniques to import and work with modules in code/namespace

9.27 Use selected useful standard Python modules

9.28 Use various techniques to identify and handle runtime errors in Python

9.29 Utilize the control statements for exception handling (try, except)

9.30 Use Python in everyday life applications including DIY activities

9.31 Use Python in several work-related scenarios

10 Perform program design, coding, and testing activities

10.1 Demonstrate knowledge of computer concepts and terminology

10.2 Create and document a program design

10.3 Communicate design specifications

10.4 Develop prototype

10.5 Perform revisions and enhancement of software systems (e.g., code refactoring, updates, add-ons)

10.6 Demonstrate proficient use of programming development tools

10.7 Test code by validating inputs, expected outcomes, determining boundary test cases

10.8 Revise program code and make code fixes

11. Use oral and written communication skills in creating, expressing, and interpreting information and ideas

11.1 Select and employ appropriate communication concepts and strategies to enhance oral and written communication in the workplace

11.2 Locate, organize, analyze, adopt, adapt, and reference written information from various sources (e.g. documentation)

11.3 Construct communication using the right development terminology

11.4 Analyze the positive and negative impacts of technology and IT terminology on popular culture and personal life (e.g. the use of sensitive terminology such as master-slave, blacklist-whitelist; interconnection of things through technology, etc.)

11.5 Discuss the influence of technology on the way people build, develop, and manage organizations and projects

12. Create and document a computer program that uses a variety of internal and control structures for manipulating various data types

12.1 Use a program editor to write the source code for a program

12.2 Write programs that use nested structures (e.g. recursions, nested loops, nested if statements)

12.3 Document a program and use consistent style to increase readability, improve maintenance, and explain the purpose and operation of given program elements

12.4 Write programs that use a variety of common data types and operators

12.5 Write programs that perform data conversion between numeric and string data types, and other data types (e.g. typecasting)

12.6 Write programs and functions that accept and process arguments

12.7 Write programs and functions that return values

12.8 Participate in peer code review to verify program functionality, programming styles, program usability, and adherence to common programming standards

Equipment Requirements

The course can be accessed online using an Internet browser and alaptop/computer/tablet/mobile device. For the best learning experience, use the latest version of Mozilla Firefox, Internet Explorer/Microsoft Edge, or Google Chrome.

Course Outline

Learners completing Python Essentials 1 will gain:

- an ability to design, develop, and improve simple computer programs written in Python
- knowledge suitable for learning another programming language
- sufficient competence for taking the PCEP^{TM} exam to obtain the corresponding certification
- skills needed to participate in the Python Essentials 2 course
- skills and knowledge to perform at a junior-level job
- the possibility to continue personal development through self-education and self-improvement

The table below details the course modules and their associated competencies. Each module is an integrated unit of learning that consists of content, activities, and assessments that target a specific set of competencies. The size of the module depends on the depth of knowledge and skill needed to master the competency.

Table 1: Module Title and Objective

Module Title / Topic Title	Objective
Module 1: Develop an awareness of programming languages	After completing Module 1, the learner will: • have a basic knowledge of computer programming and software development; • understand the fundamental programming concepts, such as: compiler, interpreter, source code, machine code, IDE; • have an orientation in Python's development history, its main traits and features; • gain skills to install and configure basic development tools as well as code, and run the very first Python program.
Module 2: Python Data Types, Variables, Operators, and Basic I/O Operations	After completing Module 2, the learner will: • gain skills to create, edit and run Python source files using IDLE; • develop knowledge of Python's numeral literals, their syntax, types and formats; • have an orientation in issues related to Python arithmetic operators and expressions; • gain the ability to name, create, initialize and modify variables;

Module Title / Topic Title	Objective
	• build skills to perform basic input/output operations in a Python program.
Module 3: Boolean Values, Conditional Execution, Loops, Lists, and List Processing. Logical and Bitwise Operations	After completing Module 3, learners will: • know basic features of the Boolean data type; • gain skills to work with relational operators in Python; • have the ability to effectively use control statements: if, if-else, if-elif-else; • understand the role of a loop and be able to use the control statements: while, for; • have an orientation in bitwise operations in Python; • know the role of lists and be able to operate with them to perform actions including indexing, slicing and content-manipulation; • understand multidimensional lists in Python.