Chapter 1

An introduction to Python programming

Objectives

Applied

1. Use IDLE to test Python expressions and statements in the interactive shell.
2. Use IDLE to open, compile, and run a Python source file.

Knowledge

1. List three reasons why Python is a good first language for new programmers.
2. Describe a console program.
3. Explain how Python compiles and runs a program in terms of source code, bytecode, and the virtual machine.
4. Explain how main memory and disk storage work together when a program is running.
5. Distinguish between systems software and application software.
6. Distinguish between testing and debugging.
7. Distinguish between syntax errors and runtime errors.
8. Describe an exception.

Chapter 2
How to write your first programs

Objectives

Applied

1. Use the IDLE shell to test numeric and string operations.
2. Code, test, and debug programs that require the skills that you've learned in this chapter. That includes the use of:

   comments for documenting your code and commenting out statements
   str, int, and float values and variables
   arithmetic expressions
   string concatenation
   special characters in strings
   the built-in print(), input(), str(), float(), int(), and round() functions
   function chaining

Knowledge

1. Describe the use of indentation when coding Python statements.
2. Describe the use of Python comments, including "commenting out" portions of Python code.
3. Describe these data types: str, int, float.
4. List two recommendations for creating a Python variable name.
5. Distinguish between underscore notation and camel case.
6. Describe the evaluation of an arithmetic expression, including order of precedence and the use of parentheses.
7. Distinguish among these arithmetic operators: /, //, and %.
8. Describe the use of += operator in a compound arithmetic expression.
9. Describe the use of escape sequences when working with strings.
10. Describe the syntax for calling one of Python's built-in functions.
11. Describe the use of the print(), input(), str(), float(), int(), and round() functions.
12. Describe what it means to chain functions.

Chapter 3
How to code control statements

Objectives

Applied

1. Code, test, and debug programs that require the skills that you've learned in this chapter. That includes the use of:

   if statements
   while statements
   for statements
   break and continue statements
   pass statements

2. Use pseudocode to plan your control structures and programs.

Knowledge

1. Distinguish between a Boolean variable and a Boolean expression.
2. Describe the evaluation of a Boolean expression, including order of precedence and the use of parentheses.
3. Describe the sort sequence of string values and the use of the lower() or upper() method of a string for comparing two string values.
4. Describe the flow of control of an if statement that has both elif and else clauses.
5. Distinguish between the flow of control in a while loop and the flow of control in a for loop.
6. Describe the use of break and continue statements.
7. Describe the use of pseudocode for planning a program and its control statements.

Chapter 4

How to define and use functions and modules

Objectives

Applied

1. Define and use functions in your programs including the use of default values, named arguments, local variables, and global variables.
2. Create, document, import, and use your own modules.
3. Import and use the random module.
4. Use a hierarchy chart or outline to plan the functions of a program.

Knowledge

1. In general terms, describe how to define a function, including the use of a return statement.
2. In general terms, describe how to call a function.
3. In general terms, describe how to define and call a main() function.
4. Describe the use of default values in a function definition and in the statements that call the function.
5. Describe the use of named arguments in calling statements.
6. Describe the recommended use of global variables, local variables, and global constants.
7. In general terms, explain how to create and document a module.
8. Distinguish among importing a module into the default namespace, a specified namespace, and the global namespace.
9. Describe how to use Python standard modules, such as the random module.
10. Describe the problem that can occur if you import a module into the global namespace.
11. Explain how to use a hierarchy chart or outline to plan the functions of a program.

Chapter 5
How to test and debug a program

Objectives

Applied

1. Plan the test runs for a program.
2. Trace the execution of a program with print() functions.
3. Use top-down coding and testing to simplify debugging.
4. Use the IDLE shell to test the functions of your programs and modules.
5. Use the IDLE debugger to set breakpoints, step through the statements of a program, and view the values of the data items at each step.
6. Use IDLE to view the stack for a program when an exception occurs.

Knowledge

1. Distinguish among syntax, runtime, and logic errors.
2. Distinguish between testing and debugging.
3. Describe the process of tracing the execution of a program with print() functions.
4. Describe top-down coding and testing.
5. Describe the use of the IDLE shell for testing the functions of a program or module.
6. Describe the use of breakpoints and the IDLE debugger for stepping through a program.
7. Describe the use of the stack when a program exception occurs.
8. Describe the debugging problems that can occur when you use floating-point numbers in arithmetic expressions.

Chapter 6
How to work with lists and tuples

Objectives

Applied

1. Use lists in your programs.
2. Use lists of lists in your programs.
3. Use tuples in your programs.

Knowledge

1. Describe how an item in a list is accessed.
2. Describe the use of these list methods: append(), insert(), remove(), index(), and pop().
3. Distinguish between the way mutable types like a list are passed to and returned by functions and the way immutable types like integers are passed to and returned by functions.
4. Describe the use of a list of lists.
5. Describe the use of these functions with lists: count(), reverse(), sort(), min(), max(), choice(), shuffle(), and deepcopy().
6. Differentiate between a shallow copy of a list and a deep copy.
7. Distinguish between a tuple and a list.
8. Describe the use of a multiple assignment statement when you unpack a tuple.

Chapter 7
How to work with file I/O

Objectives

Applied

1. Use text, CSV, or binary files to save and retrieve the data that's used by your programs.

Knowledge

1. Differentiate between text and binary files.
2. Describe the benefit of using a with statement for opening and closing a file.
3. Describe the use of the csv module, writer objects, and reader objects for writing a list of lists to a CSV file and reading a list of lists from a CSV file.
4. Describe the use of the pickle module and the load() and dump() methods for saving a list of lists to a binary file and reading a list of lists from a binary file.

Chapter 8
How to handle exceptions

Objectives

Applied

1. Add the proper level of exception handling to your programs.
2. Use raise statements to test the exception handling in your programs.

Knowledge

1. Describe the types of exceptions that need to be handled by a program.
2. Describe the operation of a try statement with one or more except clauses.
3. Describe the use of an exception object in your exception handling routines.
4. Describe the use of the exit() function in the sys module.
5. Describe the use of a finally clause in a try statement.
6. Explain why you may need to raise exceptions when testing your exception handling routines.

Chapter 9
How to work with numbers

Objectives

Applied

1. Code, test, and debug programs that work with numbers. That includes the use of:

   the math module
   the format() method of a string for formatting numbers
   the locale module for formatting currency values for specific countries
   the decimal module

Knowledge

1. Describe how the use of floating-point numbers can lead to inaccurate results.
2. Describe the purposes of the math, locale, and decimal modules.
3. Describe two ways to eliminate the types of errors that can occur when using floating-point numbers.

Chapter 10
How to work with strings

Objectives

Applied

1. Code, test, and debug programs that work with strings. That includes:

   slicing a string
   finding and replacing parts of a string
   splitting a string into a list of strings
   joining the items in a list into a string

Knowledge

1. In general terms, describe the coding that's used for Unicode characters.
2. Describe these built-in functions : the ord() function for working with characters and the len() function for working with strings.
3. Describe these string methods: islower(), isdigit(), startswith(), lower(), strip(), rjust(), find(), replace(), split(), and join().
4. Explain how delimiters work with the split() method and the join() method.

Chapter 11
How to work with dates and times

Objectives

Applied

1. Code, test, and debug programs that work with dates and times. That includes:

   creating date, time, and datetime objects
   formatting dates and times
   working with spans of time
   comparing datetime objects

Knowledge

1. Describe the three ways to create date, time, and datetime objects: with methods, with constructors, and by parsing.
2. Distinguish between aware and naïve date, time, and datetime objects.
3. Describe the way spans of times are used when working with dates and times.
4. Describe the way you compare date and time objects.

Chapter 12
How to work with dictionaries

Objectives

Applied

1. Use dictionaries in your programs.
2. Use dictionaries that contain complex objects like lists and other dictionaries.

Knowledge

1. Differentiate between a list and a dictionary.
2. Describe the use of these dictionary methods when creating view objects: key(), items(), and values().
3. Describe the use of the dict() method for converting a list or tuple to a dictionary.
4. Describe the way you access items when working with a dictionary of dictionaries, a dictionary of lists, or a list of dictionaries.

Chapter 13

How to work with recursion and algorithms

Objectives

Applied

1. Use recursion in a program.

Knowledge

1. Describe an algorithm.
2. Distinguish between the way the stack is used in a recursive function and the way it is used in an iterative function.
3. Distinguish between the base case for a recursive function and a deferred action.

Chapter 14

How to define and use your own classes

Objectives

Applied

1. Code the constructor for a class that has attributes and methods.
2. Import a class, create objects from it, access the attributes of the objects, and call the methods of the objects.
3. Use object composition to combine simple objects into more complex data structures.
4. Use encapsulation to hide the data attributes of an object.

Knowledge

1. Describe a UML class diagram.
2. Describe the relationship between a class and an object.
3. In general terms, describe the identity, state, and behavior of an object.
4. In general terms, describe the way Python code is used to define a constructor, its attributes, and its methods.
5. In general terms, describe the way Python code is used to create an object from a class.
6. Describe the concept of object composition.
7. Describe the concept of encapsulation.
8. Distinguish between public and private attributes.
9. Describe the use of getter and setter methods.

Chapter 15
How to work with inheritance

Objectives

Applied

1. Define and use a subclass that inherits a superclass and overrides one or more of the methods of the superclass.
2. Define and use a class that overrides one or more methods of the object class.

Knowledge

1. Describe the way inheritance works.
2. In general terms, explain how to override a method in the superclass when you're defining a subclass.
3. Describe the concept of polymorphism.
4. Describe the use of the isinstance() method when working with objects.
5. Describe the use of the object class and these methods of the object class: __str__(), __iter__(), and __next__().
6. Describe three factors that help determine when it is appropriate to use inheritance

Chapter 16

How to design an object-oriented program

Objectives

Applied

1. Design an object-oriented program and create a UML diagram for it.
2. Given the UML diagram for a program, develop the program with a three-tier architecture.

Knowledge

1. Describe each of these steps for designing the model for an object-oriented program:

   Identify the data attributes
   Subdivide each attribute into its smallest components
   Identify the classes
   Identify the methods
   Refine the classes, attributes, and methods

2. Describe the relationship between a class in an object-oriented program and an entity in the real world.
3. Distinguish between the presentation tier, the database tier, and the business tier in a three-tier architecture.

Chapter 17
How to work with a database

Objectives

Applied

1. Use SQLite Manager to test SQL statements against a SQLite database.
2. Develop Python programs that use SQLite databases to store the data of the programs.

Knowledge

1. Describe the organization of a relational database in terms of tables, rows, columns, primary keys, and foreign keys.
2. Describe a one-to-many relationship between two tables.
3. Describe the way the columns in a table are defined in terms of data types, null values, default values, primary keys, and foreign keys.
4. Describe the use of these SQL statements: SELECT, INSERT, UPDATE, and DELETE.
5. Describe the use of these clauses in SQL statements: FROM, WHERE, ORDER BY, and JOIN.
6. Describe a result set.
7. Describe the use of these methods of a cursor object: execute(), fetchone(), and fetchall().
8. Describe the use of the commit() method of a connection object.
9. In general terms, explain how to handle database exceptions.

Chapter 18
How to build a GUI program

Objectives

Applied

1. Develop a GUI program that has a user interface that consists of frames, buttons, labels, and text entry fields in a grid format.

Knowledge

1. Describe the need for the mainloop() method of a tkinter root window in terms of the event processing loop.
2. Describe the way an event handler works with a GUI component like a button.
3. Describe how the grid() method is used to lay out the components in a frame.
4. Describe the reason for creating a subclass of the ttk.Frame class when you're building a GUI.