Introduction to Python (From N00B to N3RD) with Dartanion Williams

Episode Quiz Questions

Table of Contents:

[Chapter 1 - Introduction to Python 1](#_Toc140826517)

[Episode 1.01 - Introduction to Python and Programming 1](#_Toc140826518)

[Episode 1.02 - Setting Up Python 2](#_Toc140826519)

[Episode 1.03 - Getting Started with Programming 5](#_Toc140826520)

[Episode 1.04 - Conditional Logic in Python 7](#_Toc140826521)

[Chapter 2 - Introduction to Programming 10](#_Toc140826522)

[Episode 2.01 – Python Loops 10](#_Toc140826523)

[Episode 2.02 - Introduction to Python Data Structures 11](#_Toc140826524)

[Episode 2.03 - Python Dictionaries 13](#_Toc140826525)

[Episode 2.04 - Advanced Concepts with Data Structures 14](#_Toc140826526)

[Chapter 3 - Data Processing with Python 16](#_Toc140826527)

## Chapter 1 - Introduction to Python

### Episode 1.01 - Introduction to Python and Programming

1. What is Python used for?
2. Web development
3. Data analysis
4. Artificial intelligence
5. All of the above

Answer: d) All of the above

Explanation: Python is a versatile programming language used for various purposes, including web development, data analysis, machine learning, artificial intelligence, and more.

1. What is a script in Python?
2. A sequence of commands written in Python
3. A graphical user interface (GUI) application
4. A data storage format
5. A software testing tool

Answer: a) A sequence of commands written in Python

Explanation: A script in Python refers to a file containing a sequence of commands and instructions written in the Python programming language.

1. What is the purpose of algorithms in computer programming?
2. To provide step-by-step instructions for solving a problem
3. To define the visual appearance of a program
4. To store and retrieve data
5. To perform calculations and mathematical operations

Answer: a) To provide step-by-step instructions for solving a problem

Explanation: Algorithms are a set of well-defined instructions or steps used in computer programming to solve a specific problem or perform a particular task.

1. What is data analytics and how is it used in business and industry?
2. Data analytics refers to analyzing and interpreting data to gain insights and make informed decisions.
3. Data analytics is the process of collecting and storing data for future use.
4. Data analytics is used for creating data visualizations and charts.
5. Data analytics is primarily used in academic research.

Answer: a) Data analytics refers to analyzing and interpreting data to gain insights and make informed decisions.

Explanation: Data analytics involves the extraction, transformation, and analysis of data to discover patterns, trends, and insights that can be used for making data-driven decisions in business and industry.

1. What are the steps involved in the data analytics process?
2. Data collection, data preparation, data analysis, data visualization, data interpretation
3. Data visualization, data collection, data analysis, data preparation, data interpretation
4. Data interpretation, data visualization, data preparation, data collection, data analysis
5. Data analysis, data visualization, data collection, data preparation, data interpretation

Answer: a) Data collection, data preparation, data analysis, data visualization, data interpretation

Explanation: The data analytics process typically involves the following steps: data collection, data preparation (cleaning, transforming), data analysis (applying statistical or machine learning techniques), data visualization (creating visual representations), and data interpretation (drawing conclusions and making decisions based on the analysis).

### Episode 1.02 - Setting Up Python

1. How can you check the version of Python installed on your system?
2. python --version
3. python check-version
4. python -v
5. python version-check

Answer: a) python --version

Explanation: The command "python --version" allows you to check the version of Python installed on your system.

1. What is the difference between a Python interpreter and a Python environment?
2. There is no difference; the terms are used interchangeably.
3. A Python interpreter executes Python code, while a Python environment provides additional tools and libraries.
4. A Python environment executes Python code, while a Python interpreter provides additional tools and libraries.
5. A Python interpreter is used for web development, while a Python environment is used for data analysis.

Answer: b) A Python interpreter executes Python code, while a Python environment provides additional tools and libraries.

Explanation: A Python interpreter is the program that reads and executes Python code, while a Python environment typically includes the interpreter along with additional tools, libraries, and resources for a more comprehensive development or analysis environment.

1. How can you add Python to your system PATH?
2. It is automatically added during the installation process.
3. By modifying the system environment variables.
4. By running a specific Python command in the terminal.
5. Python cannot be added to the system PATH.

Answer: b) By modifying the system environment variables.

Explanation: To add Python to the system PATH, you need to modify the system environment variables and include the path to the Python executable.

1. What is a python file extension and what is the standard naming convention for Python files?
2. Python files have the extension .txt, and the naming convention is all lowercase.
3. Python files have the extension .py, and the naming convention is all uppercase.
4. Python files have the extension .txt, and the naming convention is all uppercase.
5. Python files have the extension .py, and the naming convention is all lowercase.

Answer: d) Python files have the extension .py, and the naming convention is all lowercase.

Explanation: Python files have the extension .py, and it is recommended to follow the naming convention of using all lowercase letters for Python file names.

1. How can you run a Python script from the command line or terminal?
2. python run script.py
3. python execute script.py
4. python start script.py
5. python script.py

Answer: d) python script.py

Explanation: To run a Python script from the command line or terminal, you can use the command "python script.py" where "script.py" is the name of your Python script file.

1. What is a virtual environment and why is it used in Python development?
2. A virtual environment is a graphical user interface (GUI) for Python development.
3. A virtual environment is used to isolate Python packages and dependencies for different projects.
4. A virtual environment is a separate physical machine for Python development.
5. A virtual environment is used for debugging and testing Python code.

Answer: b) A virtual environment is used to isolate Python packages and dependencies for different projects.

Explanation: A virtual environment is a tool in Python that allows you to create isolated environments with their own set of Python packages and dependencies. It is used to manage and separate dependencies between different projects to avoid conflicts and ensure reproducibility.

1. How do you create a virtual environment in Python using virtualenv or venv?
2. virtualenv create myenv / venv create myenv
3. virtualenv myenv / venv myenv
4. create virtualenv myenv / create venv myenv
5. python -m virtualenv myenv / python -m venv myenv

Answer: d) python -m virtualenv myenv / python -m venv myenv

Explanation: To create a virtual environment using virtualenv or venv, you can use the command "python -m virtualenv myenv" or "python -m venv myenv" where "myenv" is the name you choose for your virtual environment.

1. How do you activate and deactivate a virtual environment in Python?
2. activate myenv / deactivate
3. source myenv/bin/activate / deactivate
4. python activate myenv / python deactivate
5. run myenv / stop myenv

Answer: b) source myenv/bin/activate / deactivate

Explanation: To activate a virtual environment, you can use the command "source myenv/bin/activate" where "myenv" is the name of your virtual environment. To deactivate the virtual environment, simply use the command "deactivate".

1. How can you verify that a virtual environment is currently active in Python?
2. Running the command "verify" in the terminal.
3. Checking the environment variable "VIRTUAL\_ENV".
4. Using the command "activeenv" in the virtual environment.
5. There is no way to verify if a virtual environment is active.

Answer: b) Checking the environment variable "VIRTUAL\_ENV".

Explanation: When a virtual environment is active, the environment variable "VIRTUAL\_ENV" is set. You can check the value of this variable in the terminal to verify if a virtual environment is currently active.

1. How can you install packages in a virtual environment using pip?
2. pip install package\_name
3. pip install -e package\_name
4. pip install --env package\_name
5. pip install --venv package\_name

Answer: a) pip install package\_name

Explanation: To install packages in a virtual environment using pip, you can use the command "pip install package\_name" where "package\_name" is the name of the package you want to install.

### Episode 1.03 - Getting Started with Programming

1. What is the correct syntax to print a message in Python?
2. display("Hello, World!")
3. log("Hello, World!")
4. print("Hello, World!")
5. echo("Hello, World!")

Answer: c) print("Hello, World!")

Explanation: In Python, the correct syntax to print a message or output to the console is by using the "print" function followed by the message or variable to be displayed.

1. How do you declare a variable in Python?
2. variable\_name = value
3. set variable\_name = value
4. declare variable\_name = value
5. create variable\_name = value

Answer: a) variable\_name = value

Explanation: To declare a variable in Python, you use the syntax "variable\_name = value", where "variable\_name" is the name you choose for your variable, and "value" is the initial value assigned to the variable.

1. How can you check the type of a variable in Python?
2. check\_type(variable\_name)
3. get\_type(variable\_name)
4. type(variable\_name)
5. variable\_name.type()

Answer: c) type(variable\_name)

Explanation: To check the type of a variable in Python, you can use the "type" function followed by the variable name in parentheses. It will return the type of the variable.

1. What is the purpose of comments in Python code?
2. To execute specific lines of code.
3. To define variables.
4. To improve code readability and provide explanations.
5. To skip code execution.

Answer: c) To improve code readability and provide explanations.

Explanation: Comments in Python are used to add explanations or notes within the code. They are ignored by the interpreter and are meant to provide information for developers to understand the code more easily.

1. How do you add a comment in a Python script?
2. // This is a comment
3. /\* This is a comment \*/
4. # This is a comment
5. <!-- This is a comment -->

Answer: c) # This is a comment

Explanation: In Python, comments are indicated using the "#" symbol. Anything after the "#" symbol on a line is considered a comment and is ignored by the interpreter.

1. What is the difference between a string and an integer in Python?
2. Strings represent text data, while integers represent numeric data.
3. Strings can only contain letters, while integers can only contain numbers.
4. Strings are mutable, while integers are immutable.
5. Strings are used for mathematical calculations, while integers are used for text manipulation.

Answer: a) Strings represent text data, while integers represent numeric data.

Explanation: In Python, strings are used to represent text data, while integers are used to represent numeric data. Strings are enclosed in quotation marks, while integers are whole numbers without decimals.

1. How can you convert a string to an integer in Python?
2. int(string\_variable
3. string\_variable.toInt()
4. convert\_to\_int(string\_variable)
5. str\_to\_int(string\_variable)

Answer: a) int(string\_variable)

Explanation: To convert a string to an integer in Python, you can use the "int" function followed by the string variable in parentheses. It will attempt to convert the string into an integer representation.

1. What is the difference between = and == in Python?
2. Both are used for assignment.
3. Both are used for comparison.
4. = is used for assignment, while == is used for comparison.
5. = is used for comparison, while == is used for assignment.

Answer: c) = is used for assignment, while == is used for comparison.

Explanation: In Python, the "=" operator is used for variable assignment, where a value is assigned to a variable. The "==" operator is used for comparison, checking if two values are equal.

1. What is the correct way to compare two values in Python for equality?
2. value1 == value2
3. value1 = value2
4. value1 === value2
5. value1 is value2

Answer: a) value1 == value2

Explanation: To compare two values for equality in Python, you use the "==" operator. It checks if the values on both sides are equal and returns a Boolean result.

### Episode 1.04 - Conditional Logic in Python

1. What is the purpose of the else statement in Python?
2. It is used to handle exceptions.
3. It is used to define a fallback option when no condition is met.
4. It is used to break out of a loop.
5. It is used to define the starting point of a function.

Answer: b) It is used to define a fallback option when no condition is met.

Explanation: In Python, the "else" statement is used in conjunction with an "if" statement to provide an alternative code block to execute when the condition of the "if" statement is not met.

1. What is the syntax of an if statement in Python?
2. if condition:
3. if (condition)
4. if condition then
5. if {condition}

Answer: a) if condition:

Explanation: The syntax of an if statement in Python follows the pattern "if condition:", where "condition" is the expression that evaluates to either True or False.

1. How can you check multiple conditions using elif statements in Python?
2. By nesting if statements within each other.
3. By using the "else" keyword.
4. By using the "elif" keyword.
5. By combining conditions with the "and" operator.

Answer: c) By using the "elif" keyword.

Explanation: In Python, you can check multiple conditions using the "elif" (short for "else if") keyword. It allows you to specify additional conditions to be evaluated if the previous conditions are not met.

1. How can you use nested if-elif-else statements in Python?
2. By indenting the code blocks within each statement.
3. By separating each statement with semicolons.
4. By using curly braces to enclose each statement.
5. By using parentheses to enclose each statement.

Answer: a) By indenting the code blocks within each statement.

Explanation: In Python, nested if-elif-else statements are created by indenting the code blocks within each statement. Indentation is crucial in Python to define the scope and hierarchy of statements.

1. What happens if multiple conditions are true in an if-elif-else chain in Python?
2. The code block of the first true condition is executed.
3. The code blocks of all true conditions are executed.
4. The code block of the last true condition is executed.
5. An error is thrown, and the program terminates.

Answer: a) The code block of the first true condition is executed.

Explanation: In an if-elif-else chain in Python, once a condition is evaluated as true, the corresponding code block is executed, and the rest of the chain is skipped. Only the code block of the first true condition is executed.

1. How does the if statement evaluate non-Boolean values in Python?
2. It considers all non-zero values as True and zero as False.
3. It considers all non-empty values as True and empty values as False.
4. It raises an error when non-Boolean values are used in an if statement.
5. It automatically converts non-Boolean values to Boolean before evaluation.

Answer: a) It considers all non-zero values as True and zero as False.

Explanation: In Python, the if statement treats non-Boolean values in a truthy or falsy manner. Non-zero values are considered true, while zero is considered false. This behavior is known as truthiness and falsiness.

1. What is the difference between "if x" and "if x is not None" in Python?
2. There is no difference; both statements are equivalent.
3. "if x" checks if x is True, while "if x is not None" checks if x is not None.
4. "if x" checks if x is not None, while "if x is not None" checks if x is True.
5. "if x" raises an error if x is None, while "if x is not None" does not.

Answer: b) "if x" checks if x is True, while "if x is not None" checks if x is not None.

Explanation: In Python, "if x" checks the truthiness of x, evaluating if it is considered True. "if x is not None" specifically checks if x is not equal to None, which is a distinct value in Python.

1. Can you use a single line if-else statement in Python?
2. Yes, by separating the if and else statements with a comma.
3. Yes, by using the semicolon as a separator between the if and else statements.
4. No, a single line if-else statement is not valid in Python.
5. Yes, by using a type of operator or expression that takes three operands or arguments
6. Answer: d) Yes, by using a type of operator or expression that takes three operands or arguments

Explanation: In Python, you can use a single line if-else statement by using the ternary operator syntax, which follows the pattern "x if condition else y". It allows you to evaluate a condition and choose between two expressions based on the result.

1. How does the ternary operator (a type of operator or expression that takes three operands or arguments (x if c else y)) work in Python?
2. It evaluates both x and y and returns the result of the condition.
3. It raises an error if the condition is False.
4. It returns x if the condition is True; otherwise, it returns y.
5. It performs a logical OR operation between x and y based on the condition.

Answer: c) It returns x if the condition is True; otherwise, it returns y.

Explanation: The ternary operator (x if c else y) in Python evaluates the condition c. If the condition is True, it returns the value of x; otherwise, it returns the value of y. It provides a concise way to choose between two expressions based on a condition.

1. What is the difference between the and and or operators in an if statement in Python?
2. The and operator checks if both conditions are True, while the or operator checks if either condition is True.
3. The and operator checks if either condition is True, while the or operator checks if both conditions are True.
4. The and operator checks if at least one condition is True, while the or operator checks if both conditions are False.
5. The and operator checks if both conditions are False, while the or operator checks if either condition is False.

Answer: a) The and operator checks if both conditions are True, while the or operator checks if either condition is True.

Explanation: In Python, the "and" operator is used in an if statement to check if both conditions are true. It returns True only if both conditions evaluate to True. The "or" operator, on the other hand, checks if at least one of the conditions is true. It returns True if either of the conditions evaluates to True.

## Chapter 2 - Introduction to Programming

### Episode 2.01 – Python Loops

1. Create a for loop that outputs each character of "Faustus Blackwood".
2. for char in "Faustus Blackwood":
3. for char in "Faustus Blackwood".split():
4. for char in range("Faustus Blackwood"):
5. for char in ["Faustus Blackwood"]:

Answer: a) for char in "Faustus Blackwood":

Explanation: The correct syntax for iterating over each character of a string in a for loop is "for char in <string>". The loop variable "char" will take on each character of the string in each iteration.

1. Create a for loop that outputs only the vowels in "Zelda Spellman".
2. for char in "Zelda Spellman":
3. for char in "Zelda Spellman".split():
4. for char in "Zelda Spellman": if char in "aeiouAEIOU":
5. for char in ["Zelda Spellman"]:

Answer: c) for char in "Zelda Spellman": if char in "aeiouAEIOU":

Explanation: To output only the vowels in a string using a for loop, you can iterate over each character of the string and use an if statement to check if the character is a vowel.

1. Create a while loop that outputs "Sabrina" ten times.
2. while i < 10: print("Sabrina")
3. while i <= 10: print("Sabrina")
4. while i < 11: print("Sabrina")
5. while i <= 11: print("Sabrina")

Answer: c) while i < 11: print("Sabrina")

Explanation: To output "Sabrina" ten times using a while loop, you can set an iterative variable, such as "i", to 0 and use a condition "while i < 10" to control the loop.

1. Create an infinite while loop that outputs "Sabrina" with a condition to break the loop if the user enters "break" (case insensitive).
2. while True: if user\_input == "break": break print("Sabrina")
3. while user\_input != "break": print("Sabrina")
4. while user\_input != "break": if user\_input == "break": break print("Sabrina")
5. while True: print("Sabrina")

Answer: a) while True: if user\_input == "break": break print("Sabrina")

Explanation: To create an infinite while loop that outputs "Sabrina" and breaks the loop when the user enters "break", you can use a condition "while True" to create an infinite loop, and within the loop, check if the user input equals "break" and use the "break" statement to exit the loop.

### Episode 2.02 - Introduction to Python Data Structures

1. Create a list that contains the names Hilda, Zelda, Rosalind, and Dorian, then perform the following:
2. Append Dorcas to the list.
3. Insert Theo to the beginning of the list.
4. Delete Zelda from the list.
5. Sort the list in ascending order.
6. Insert Sabrina at the beginning of the list after it's sorted.
7. Create a tuple containing the names Harvey, Theo, and Rosalind.
8. Loop through the tuple and output each name in uppercase.

Answer a)

names\_list = ["Hilda", "Zelda", "Rosalind", "Dorian"]

names\_list.append("Dorcas")

names\_list.insert(0, "Theo")

names\_list.remove("Zelda")

names\_list.sort()

names\_list.insert(0, "Sabrina")

names\_tuple = ("Harvey", "Theo", "Rosalind")

for name in names\_tuple: print(name.upper())

Answer b)

names\_list = ["Hilda", "Zelda", "Rosalind", "Dorian"]

names\_list.append("Dorcas")

names\_list.insert(0, "Theo")

names\_list.delete("Zelda")

names\_list.sort()

names\_list.insert(0, "Sabrina")

names\_tuple = ("Harvey", "Theo", "Rosalind")

for name in names\_tuple: print(name.upper())

Answer c)

names\_list = ["Hilda", "Zelda", "Rosalind", "Dorian"]

names\_list.append("Dorcas")

names\_list.add("Theo")

names\_list.remove("Zelda")

names\_list.sort()

names\_list.add("Sabrina")

names\_tuple = ("Harvey", "Theo", "Rosalind")

for name in names\_tuple: print(name.upper())

Answer d)

names\_list = ["Hilda", "Zelda", "Rosalind", "Dorian"]

names\_list.append("Dorcas")

names\_list.insert(1, "Theo")

names\_list.delete("Zelda")

names\_list.sort(reverse=True)

names\_list.insert(0, "Sabrina")

names\_tuple = ("Harvey", "Theo", "Rosalind")

for name in names\_tuple: print(name.upper())

Correct Answer: a)

names\_list = ["Hilda", "Zelda", "Rosalind", "Dorian"]

names\_list.append("Dorcas")

names\_list.insert(0, "Theo")

names\_list.remove("Zelda")

names\_list.sort()

names\_list.insert(0, "Sabrina")

names\_tuple = ("Harvey", "Theo", "Rosalind")

for name in names\_tuple: print(name.upper())

Explanation: The correct approach to perform the given operations is as follows:

Append "Dorcas" to the list using the append() function.

Insert "Theo" to the beginning of the list using the insert() function.

Delete "Zelda" from the list using the remove() function.

Sort the list in ascending order using the sort() function.

Insert "Sabrina" at the beginning of the list after it's sorted using the insert() function.

Create a tuple containing the names "Harvey", "Theo", and "Rosalind".

Loop through the tuple and output each name in uppercase using a for loop and the upper() function.

### Episode 2.03 - Python Dictionaries

1. Create one dictionary containing the first and last name for Dorian Gray. Then add a new entry with the key "hometown" with the value "Greendale". Add another entry with the key "species" with the value "witch". Delete the key "species". Create a list of dictionaries that contains first and last name for these characters: Agatha Night, Dorcas Night, Prudence Blackwood.

a) dorian\_dict = {"first\_name": "Dorian", "last\_name": "Gray"} dorian\_dict["hometown"] = "Greendale" dorian\_dict["species"] = "witch" del dorian\_dict["species"] characters\_list = [ {"first\_name": "Agatha", "last\_name": "Night"}, {"first\_name": "Dorcas", "last\_name": "Night"}, {"first\_name": "Prudence", "last\_name": "Blackwood"} ]

b) dorian\_dict = {"first\_name": "Dorian", "last\_name": "Gray"} dorian\_dict.append("hometown", "Greendale") dorian\_dict.append("species", "witch") dorian\_dict.remove("species") characters\_list = [ {"first\_name": "Agatha", "last\_name": "Night"}, {"first\_name": "Dorcas", "last\_name": "Night"}, {"first\_name": "Prudence", "last\_name": "Blackwood"} ]

c) dorian\_dict = {"first\_name": "Dorian", "last\_name": "Gray"} dorian\_dict["hometown"] = "Greendale" dorian\_dict.update("species", "witch") dorian\_dict.remove("species") characters\_list = [ {"first\_name": "Agatha", "last\_name": "Night"}, {"first\_name": "Dorcas", "last\_name": "Night"}, {"first\_name": "Prudence", "last\_name": "Blackwood"} ]

d) dorian\_dict = {"first\_name": "Dorian", "last\_name": "Gray"} dorian\_dict["hometown"] = "Greendale" dorian\_dict["species"] = "witch" dorian\_dict.delete("species") characters\_list = [ {"first\_name": "Agatha", "last\_name": "Night"}, {"first\_name": "Dorcas", "last\_name": "Night"}, {"first\_name": "Prudence", "last\_name": "Blackwood"} ]

Answer: a) dorian\_dict = {"first\_name": "Dorian", "last\_name": "Gray"} dorian\_dict["hometown"] = "Greendale" dorian\_dict["species"] = "witch" del dorian\_dict["species"] characters\_list = [ {"first\_name": "Agatha", "last\_name": "Night"}, {"first\_name": "Dorcas", "last\_name": "Night"}, {"first\_name": "Prudence", "last\_name": "Blackwood"} ]

Explanation: The correct approach to perform the given operations is as follows:

1. Create a dictionary named "dorian\_dict" with the key "first\_name" and value "Dorian" and the key "last\_name" and value "Gray".

Add a new entry to "dorian\_dict" with the key "hometown" and the value "Greendale" using the square bracket notation.

Add another entry to "dorian\_dict" with the key "species" and the value "witch" using the square bracket notation.

Delete the entry with the key "species" from "dorian\_dict" using the del keyword.

Create a list named "characters\_list" containing dictionaries for the characters with their first names and last names. The correct code snippet that accomplishes these tasks is option a.

### Episode 2.04 - Advanced Concepts with Data Structures

characters = ['Sabrina', 'Ambrose', 'Hilda', 'Zelda', 'Harvey'] ages = [16, 75, 270, 290, 17]

1. Given the character and ages data sets above, conduct the following: Create a dictionary through either a dictionary comprehension or the zip() function. Loop through the dictionary items. Use the keys and values to generate the following for each entry: {character} is {age} years old.

a) character\_ages = {character: age for character, age in zip(characters, ages)} for character, age in character\_ages.items(): print(f"{character} is {age} years old.")

b) character\_ages = {age: character for character, age in zip(characters, ages)} for character, age in character\_ages.items(): print(f"{character} is {age} years old.")

c) character\_ages = {character: age for age, character in zip(ages, characters)} for character, age in character\_ages.items(): print(f"{character} is {age} years old.")

d) character\_ages = {age: character for age, character in zip(ages, characters)} for character, age in character\_ages.items(): print(f"{character} is {age} years old.")

Answer: a) character\_ages = {character: age for character, age in zip(characters, ages)} for character, age in character\_ages.items(): print(f"{character} is {age} years old.")

Explanation: The correct approach to create a dictionary from the given character and ages data sets is by using a dictionary comprehension or the zip() function. Option a uses a dictionary comprehension to create a dictionary named "character\_ages" where the characters are the keys and the ages are the values. The loop then iterates over the items of "character\_ages" using the items() method, and for each entry, it prints the character and the corresponding age using f-string formatting. This will generate the desired output where each character is followed by their respective age.

## Chapter 3 - Data Processing with Python

Note: Currently there are no quiz questions for Chapter 3, as it is a programming-heavy chapter that doesn’t lend itself well to short, multiple-choice questions.