Western Governors University (WGU) C173 Scripting and Programming Practice Test (Sample)

Study Guide

BY EXAMZIFY

Everything you need from our exam experts!

Notice: Examzify makes every reasonable effort to obtain accurate, complete, and timely information about this product from reliable sources.

Table of Contents

Copyright 1

Table of Contents 2

Introduction 3

How to Use This Guide 4

Questions 5

Answers 8

Explanations 10

Next Steps 16

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- **Practice answering questions under realistic conditions,**
- **Improve accuracy and speed,**
- **Review explanations to strengthen weak areas, and**
- **Approach the exam with greater confidence.**

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here’s how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don’t just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There’s no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

Questions

1. How do you save a value to a variable?

A. By assigning the value using the equals sign

B. By declaring its type explicitly

C. By enclosing it in quotation marks

D. By creating a function

2. What keyword is used to define a method in Python?

A. method

B. define

C. def

D. function

3. Which of the following best describes a logical operator?

A. An operator that combines boolean expressions

B. An operator that performs arithmetic operations

C. An operator that compares two values

D. An operator that modifies strings

4. In which phase of an agile approach are program goals modified based on client feedback?

A. Testing

B. Implementation

C. Analysis

D. Design

5. What is the primary distinction between global and local variables?

A. Global variables can only be accessed within their defining function

B. Local variables can be accessed from any function

C. Global variables can be accessed from any function, while local variables are restricted to their defining function

D. Both global and local variables are accessible from anywhere within the application

6. What does the term 'variable' refer to in programming?

- **A. A reserved memory location to store data**
- **B. A specific function within a program**
- **C. A part of the user interface**
- **D. A constant value that cannot change**

7. How is asynchronous programming primarily managed in JavaScript?

- **A. Using synchronous function calls**
- **B. Using callback functions only**
- **C. Using Promises or async/await syntax**
- **D. Using global variables**

8. Which phase of the waterfall approach specifies the required order of events between program components?

- **A. Implementation**
- **B. Testing**
- **C. Design**
- **D. Requirements gathering**

9. What is the primary distinction between a compiled language and an interpreted language?

- **A. Compiled languages are faster than interpreted languages**
- **B. Compiled languages require a compiler, interpreted languages do not**
- **C. Interpreted languages produce machine code, compiled ones do not**
- **D. Compiled languages can only run on specific platforms**

10. What is the main feature of a class in object-oriented programming?

- **A. Provides a structure for data management**
- **B. Encapsulation of related functions and data**
- **C. Allows for global access of variables**
- **D. Defines loop structures**

Answers

1. **A**
2. **C**
3. **A**
4. **B**
5. **C**
6. **A**
7. **C**
8. **B**
9. **B**
10. **B**

Explanations

1. How do you save a value to a variable?

A. By assigning the value using the equals sign

B. By declaring its type explicitly

C. By enclosing it in quotation marks

D. By creating a function

Saving a value to a variable involves the process of assignment, which is achieved through the use of the equals sign. In programming, when you assign a value to a variable, you are essentially creating a label for that value that can be referenced later in your code. The syntax typically looks like this: ``` variable_name = value; ``` In this syntax, the variable name is assigned the value on the right side of the equals sign. This means that the variable now holds that value, and it can be used elsewhere in the program for various operations, such as calculations or data manipulation. Other options, while related to programming concepts, do not directly pertain to the act of saving a value to a variable. Declaring a type explicitly refers to defining what kind of data a variable can hold, but it does not itself involve the assignment of a value. Enclosing a value in quotation marks is relevant for string literals, but again, it does not encompass the broader notion of saving a value to a variable if not assigned. Creating a function is a separate mechanism used to group and reuse code, but it does not relate to variable assignment. Therefore, using the equals sign is the fundamental principle behind assigning values to variables.

2. What keyword is used to define a method in Python?

A. method

B. define

C. def

D. function

In Python, the keyword used to define a method is "def." This keyword indicates the start of a function definition and is a crucial part of the syntax in Python programming. When you use "def," you specify the name of the method, followed by parentheses that may include parameters, and then a colon. This allows Python to recognize that a new block of code is being defined as a function or method. For example, a simple method definition would look like this: ```python def my_function(parameter): # method body pass ``` The use of "def" is essential because it informs the Python interpreter about the following indented block of code that constitutes the function body, marking the beginning of the function's scope. The other choices do not represent valid Python syntax for defining methods; therefore, they do not function as commands within the language.

3. Which of the following best describes a logical operator?

- A. An operator that combines boolean expressions
- B. An operator that performs arithmetic operations
- C. An operator that compares two values
- D. An operator that modifies strings

A logical operator is defined as an operator that combines boolean expressions to produce a true or false outcome based on particular logical relationships. In programming, common logical operators include AND, OR, and NOT. These operators assess the truth values of the expressions they operate on, allowing for complex conditional statements that can control the flow of a program or determine the result of a boolean expression. For example, in an expression like `A AND B`, the logical operator checks whether both A and B are true. If they are, the outcome is true; if either is false, the outcome is false. This functionality is crucial in decision-making processes within code, such as in `if` statements or loops, where the result dictates the course of execution. The other options describe different types of operators: arithmetic operators perform mathematical calculations (like addition and subtraction), comparison operators evaluate how two values relate to each other (like greater than or equal to), and string operators manipulate or modify text. Each serves a specific role in programming but does not fulfill the function of combining boolean expressions like logical operators do.

4. In which phase of an agile approach are program goals modified based on client feedback?

- A. Testing
- B. Implementation
- C. Analysis
- D. Design

In the agile approach, the phase where program goals are modified based on client feedback is typically the implementation phase. During this stage, the development team actively engages with clients to gather feedback on the product as it's being developed. This feedback can lead to adjustments in the program goals, ensuring that the final product closely aligns with client expectations and needs. Implementation is characterized by iterative development cycles where features are built, demonstrated, and refined based on user input. This continuous incorporation of feedback allows for flexibility in adapting to changes and improving the product incrementally. By incorporating feedback during this stage, the development process remains responsive to real-world uses and client requirements, promoting a more effective final outcome.   In contrast, other phases such as testing, analysis, and design focus on different aspects of the project. Testing primarily assesses the product's functionality and ensures that it meets quality standards, while analysis focuses on understanding requirements and defining what the project should achieve before development begins. The design phase involves creating the architecture and user interfaces based on gathered requirements, but does not specifically address the adaptation of goals in response to client feedback as effectively as the implementation phase does.

5. What is the primary distinction between global and local variables?

A. Global variables can only be accessed within their defining function

B. Local variables can be accessed from any function

C. Global variables can be accessed from any function, while local variables are restricted to their defining function

D. Both global and local variables are accessible from anywhere within the application

The primary distinction between global and local variables lies in their scope of accessibility within a program. Global variables are declared outside of any function and can be accessed from any part of the program, meaning any function can use them once they have been defined. This allows for shared data across multiple functions, which can be useful in managing state or configuration throughout a program's execution. In contrast, local variables are defined within a specific function and can only be accessed within that function. They exist in memory only while the function is executing, and once the function completes, those variables are no longer available. This encapsulation is vital for creating modular code, as it helps prevent unintended interactions between different parts of the program. Thus, the correct answer emphasizes that global variables have a broader reach across the program, while local variables are confined to their specific function context, highlighting the fundamental differences in their scope and lifetime.

6. What does the term 'variable' refer to in programming?

A. A reserved memory location to store data

B. A specific function within a program

C. A part of the user interface

D. A constant value that cannot change

The term 'variable' in programming refers to a reserved memory location used to store data. This is fundamental to programming because variables allow developers to create programs that can handle and manipulate dynamic information. When a variable is defined, it is associated with a specific data type, which dictates what kind of data it can hold (such as integers, strings, or floating-point numbers). Once a variable is declared, it can be used to store values that may change during the execution of a program. For example, a variable might store user input, calculations, or the results of operations, enabling a program to be flexible and responsive to changing data. This concept is essential because it allows a programmer to write code that can process different values without having to hard-code each possible value. The other options represent concepts that do not accurately define a variable in programming. A function refers to a block of code designed to perform a specific task, not a storage location for data. The user interface is concerned with the elements a user interacts with to use a program, while a constant value cannot change once defined, contrasting with the nature of a variable, which is inherently designed to hold values that can change over time.

7. How is asynchronous programming primarily managed in JavaScript?

A. Using synchronous function calls

B. Using callback functions only

C. Using Promises or async/await syntax

D. Using global variables

Asynchronous programming in JavaScript is primarily managed through Promises and the async/await syntax because these constructs provide a more manageable and readable approach to handling asynchronous operations. Promises represent a value that may be available now, or in the future, or never. They allow developers to write cleaner and more organized code compared to traditional callback functions. With Promises, you can chain operations and handle both success and error conditions more elegantly. The async/await syntax, introduced in ES2017, builds on Promises and further simplifies asynchronous code. By allowing developers to write asynchronous code that looks synchronous, it enhances readability and makes it easier to work with complex asynchronous flows. The async keyword is used to declare a function as asynchronous, and within that function, the await keyword can pause execution until a Promise is resolved. In contrast, synchronous function calls block code execution until the function completes, which is not suitable for asynchronous operations. Relying solely on callback functions can lead to callback hell, making code difficult to read and maintain. Global variables do not inherently manage asynchronous behavior and can introduce risks related to data integrity and state management. This emphasis on Promises and async/await thus solidifies their role as the primary means of handling asynchronous tasks in JavaScript

8. Which phase of the waterfall approach specifies the required order of events between program components?

A. Implementation

B. Testing

C. Design

D. Requirements gathering

The correct choice is related to the design phase of the waterfall approach. In this phase, the architecture and design of the system are addressed, which includes establishing the necessary order of events and interactions between various program components. It involves creating detailed specifications and working on a blueprint for the system, which is crucial for successful implementation later on. The design phase delves into how the components will work together, including defining data flow, control flow, and any dependencies that exist. This structured planning is essential to ensure that once the implementation begins, developers understand the sequence of operations necessary for the system to function correctly. In contrast, while the implementation phase focuses on coding the system based on the earlier designs, it does not primarily address the sequencing of events. Similarly, the testing phase occurs after implementation, where the system is evaluated for functionality and bugs without modifying the order of events defined in design. The requirements gathering phase mainly concentrates on defining what the system should do rather than how the components interact and execute their functions. Thus, establishing the required order of events between program components is a critical aspect of the design phase, making it the correct choice in this context.

9. What is the primary distinction between a compiled language and an interpreted language?

A. Compiled languages are faster than interpreted languages

B. Compiled languages require a compiler, interpreted languages do not

C. Interpreted languages produce machine code, compiled ones do not

D. Compiled languages can only run on specific platforms

The primary distinction between a compiled language and an interpreted language lies in the requirement of a compiler for compiled languages. Compiled languages need to be translated into machine code before execution. This translation is handled by a compiler, which processes the entire code and produces an executable file. Once compiled, the program can run directly on the target machine without needing the original source code or compiler. In contrast, interpreted languages do not undergo this pre-compilation step. Instead, an interpreter reads and executes the code line-by-line at runtime. This means interpreted languages can be more flexible and easier to work with during development, allowing for dynamic changes and immediate execution. Understanding this distinction is vital because it directly impacts aspects such as performance, development speed, and flexibility. For instance, since compiled languages are converted to machine code, they tend to execute faster than interpreted languages, which must be processed during execution. The other choices, while they touch on relevant aspects of compiled and interpreted languages, do not capture the core distinction effectively. For instance, while it's true that compiled programs can be faster, it's not an inherent rule applicable in all cases, and some interpreted languages are optimized for speed. The assertion that interpreted languages produce machine code is also inaccurate since they typically execute the

10. What is the main feature of a class in object-oriented programming?

A. Provides a structure for data management

B. Encapsulation of related functions and data

C. Allows for global access of variables

D. Defines loop structures

The main feature of a class in object-oriented programming is the encapsulation of related functions and data. A class serves as a blueprint for creating objects, which can represent real-world entities. This encapsulation means that a class can bundle data (attributes) and methods (functions) that operate on that data together, promoting organized and modular code. This encapsulation allows for better data security and integrity, as the internal state of an object can be protected from unintended interference and misuse. By grouping related functions and data together, classes enable developers to create more complex applications while maintaining readability and reducing complexity. Classes also facilitate code reuse through inheritance, where one class can inherit properties and behaviors from another class, further strengthening the concept of encapsulation. The other options, while related to programming concepts, do not specifically capture the essence of what a class does in object-oriented programming. For instance, data management structures can exist outside of the context of classes. Global access of variables is contrary to good object-oriented practices, as it can lead to issues with variable state management and maintainability. Loop structures are control flow constructs and not features directly associated with classes. Thus, the encapsulation of related functions and data is the defining characteristic of a class in this paradigm.

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://wgu-c173.examzify.com

We wish you the very best on your exam journey. You've got this!