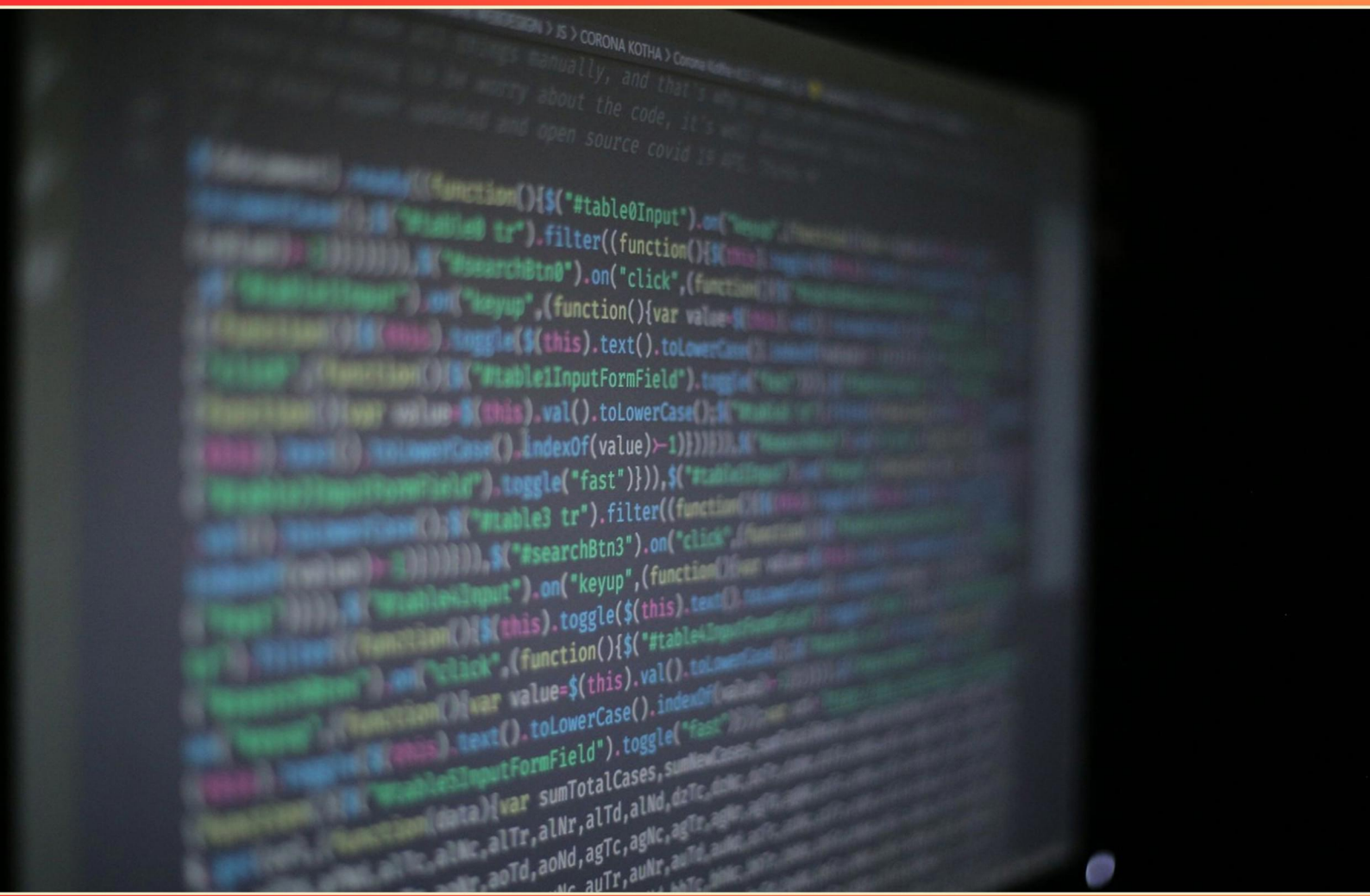
WESTERN GOVERNORS UNIVERSITY (WGU) ITSW 2113 D278 SCRIPTING AND PROGRAMMING FOUNDATIONS PRACTICE EXAM (SAMPLE)

STUDY GUIDE



SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://wgu-itsw2113-d278.examzify.com

Copyright © 2026 by Examzify - A Kaluba Technologies Inc. product.

ALL RIGHTS RESERVED.

No part of this book may be reproduced or transferred in any form or by any means, graphic, electronic, or mechanical, including photocopying, recording, web distribution, taping, or by any information storage retrieval system, without the written permission of the author.

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

SAMPLE

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 15

SAMPLE

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.

SAMPLE

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!

SAMPLE

Questions

SAMPLE

1. What is a variable in programming?

A. A function that returns a value

B. A named storage location in memory that holds a value

C. A type of programming error

D. A structure that contains multiple data types

2. What is meant by the term "syntax error"?

A. A mistake that occurs during runtime

B. A violation of the programming language's rules that causes failure to compile or run

C. A logical error in the program's execution

D. An error caused by incorrect function use

3. What action does a program perform during the processing stage?

A. Inputting data from a user

B. Storing results into a file

C. Performing computations on the data

D. Displaying graphical outputs

4. What is meant by the term "syntax" in programming?

A. The meaning of code elements

B. The rules that define the structure of code

C. The arrangement of variables in memory

D. The performance of the code

5. Which of the following is an example of an iterative loop structure?

A. if-else statement

B. for loop

C. switch statement

D. function call

6. What differentiates a do-while loop from a while loop?

A. A do-while loop executes the body only if the condition is true

B. A do-while loop checks the condition before executing

C. A do-while loop executes at least once before checking the condition

D. A do-while loop cannot contain conditions

SAMPLE

7. What is meant by underscore separated notation?

A. Words are capitalized and joined together

B. Words are in uppercase letters

C. Words are lowercase and separated by underscores

D. All words are in camel case

8. What does "statically typed" mean in programming?

A. The type of each variable can change during execution

B. The type of each variable must be declared and does not change

C. No types are used in variable declarations

D. Type definitions are created at runtime

9. What does "debugging" mean in the context of programming?

A. The process of writing documentation

B. The process of optimizing code for performance

C. The process of identifying and fixing errors or bugs in code

D. The process of compiling code

10. How does a syntax error differ from a runtime error?

A. A syntax error occurs during execution, while a runtime error occurs during coding

B. A syntax error is a logical error, while a runtime error is a compilation error

C. A syntax error occurs during coding, while a runtime error occurs during execution

D. There is no difference; both refer to the same type of error

SAMPLE

Answers

SAMPLE

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

SAMPLE

Explanations

SAMPLE

1. What is a variable in programming?

A. A function that returns a value

B. A named storage location in memory that holds a value

C. A type of programming error

D. A structure that contains multiple data types

A variable in programming is essentially a named storage location in memory that holds a value. This is a fundamental concept in programming as it allows developers to store data, manipulate it, and retrieve it as needed. When a variable is created, it is given a specific name, which acts as a reference to that memory location. This enables programmers to access or update the stored value by using the variable name instead of dealing with memory addresses directly. Variables can hold different types of data, such as numbers, strings, or objects, depending on the programming language's capabilities. The ability to use variables makes it easier to write dynamic and flexible code since values can change during the execution of a program, allowing for more complex operations and functionality. In contrast, other options describe different concepts. A function that returns a value pertains to a specific programming construct designed for executing a block of code and providing a result, while a programming error refers to mistakes in the code that lead to incorrect behavior or crashes. Finally, a structure that contains multiple data types typically refers to more complex data structures, like arrays or objects, rather than the fundamental concept of a variable.

2. What is meant by the term "syntax error"?

A. A mistake that occurs during runtime

B. A violation of the programming language's rules that causes failure to compile or run

C. A logical error in the program's execution

D. An error caused by incorrect function use

The term "syntax error" refers to a specific type of mistake in programming that arises when the code does not conform to the grammatical rules of the programming language being used. This kind of error is identified by the compiler or interpreter during the compilation or interpretation process, meaning that it prevents the program from running at all. When a syntax error occurs, the program does not successfully compile, which means that none of the code can execute until the error is corrected. Common examples of syntax errors include missing punctuation, misspelled keywords, and incorrect use of operators or functions. Understanding syntax errors is crucial for programmers as they are often the first barrier encountered when writing code. Recognizing and fixing these errors enables smoother development and ensures that the code adheres to the standards of the programming language, allowing it to function correctly when executed.

SAMPLE

3. What action does a program perform during the processing stage?

A. Inputting data from a user

B. Storing results into a file

C. Performing computations on the data

D. Displaying graphical outputs

During the processing stage, a program primarily focuses on performing computations on the data that it has received. This stage is where the logic of the program takes place, applying algorithms and operations to manipulate the input data. For instance, if a program is designed to calculate the total costs based on user inputs, the processing stage would involve adding, subtracting, or performing other calculations with that data. This phase is crucial because it transforms raw input into meaningful output, which might then be stored or displayed in later steps of the program. The other actions listed, such as inputting data, storing results, and displaying outputs, occur in different phases of program execution: they are pertinent to the input, output, and data storage stages, respectively. Understanding the separation of these stages helps clarify the flow of data through a program and the specific roles each stage plays in overall program functionality.

4. What is meant by the term "syntax" in programming?

A. The meaning of code elements

B. The rules that define the structure of code

C. The arrangement of variables in memory

D. The performance of the code

The term "syntax" in programming specifically refers to the rules that define the structure and arrangement of code. It encompasses the set of rules that govern how to write code in a specific programming language, including how to correctly form statements, declare variables, and utilize various operators and constructs. Adhering to these rules is essential because improper syntax will result in errors, preventing the code from executing properly. Programming languages have their own syntax, much like human languages have grammar. For example, in Python, the use of indentation is part of the syntax that determines the structure of the code blocks. Thus, understanding and following the syntax rules for a programming language is crucial for creating valid programs that the computer can understand and execute.

5. Which of the following is an example of an iterative loop structure?

A. if-else statement

B. for loop

C. switch statement

D. function call

An iterative loop structure is designed to repeat a block of code multiple times based on a specified condition. A for loop is a prime example of this, allowing programmers to execute a set of instructions repeatedly. It typically includes three main components: an initialization step, a condition that determines how long to loop, and an increment or decrement statement that updates the loop control variable. For instance, a for loop can be structured like this: ```python for i in range(5): print(i) ``` In this example, the loop will print numbers from 0 to 4, iterating five times in total. This repetition is the hallmark of an iterative loop structure. In contrast, the other choices do not embody iterative looping: - An if-else statement is a conditional structure used to execute different blocks of code based on certain conditions, but it does not inherently repeat. - A switch statement, similar to an if-else, directs execution to specific code blocks based on the value of an expression but doesn't iterate. - A function call executes a particular block of code once and does not involve repeating that execution unless explicitly called multiple times, which is not the same as iterative looping. This distinction highlights why the for loop is the appropriate answer in

SAMPLE

6. What differentiates a do-while loop from a while loop?

A. A do-while loop executes the body only if the condition is true

B. A do-while loop checks the condition before executing

C. A do-while loop executes at least once before checking the condition

D. A do-while loop cannot contain conditions

The key distinction that sets a do-while loop apart from a while loop is that a do-while loop always executes its body at least once, regardless of whether the condition evaluates to true or false. This behavior is defined by the structure of the do-while loop, which first executes the block of code, and then checks the specified condition. If the condition is true, it will continue executing the loop; if false, it will exit after that single execution. This characteristic is particularly useful in scenarios where an initial action must occur before any condition can be evaluated, such as prompting a user for input at least once before checking if the input meets a specific criterion. In contrast, a while loop checks the condition prior to executing its body, meaning it may not run at all if the condition is initially false. Thus, understanding that a do-while loop guarantees at least one execution of the loop body is crucial for effectively utilizing this structure in programming.

7. What is meant by underscore separated notation?

A. Words are capitalized and joined together

B. Words are in uppercase letters

C. Words are lowercase and separated by underscores

D. All words are in camel case

Underscore separated notation refers to a naming convention where multiple words are combined into a single expression, with each word written in lowercase letters and separated by underscores. This format is commonly used in programming and data representation to enhance readability. For example, a variable name like "user_name" clearly indicates the two distinct components, "user" and "name," while maintaining a seamless connection. The choice highlights that this particular style focuses on using lowercase and underscores, making it distinct from other naming conventions that utilize capitalization or different formats, such as camel case or all uppercase letters. These other styles may serve different purposes or fit different programming languages but do not adhere to the underscore separated notation being discussed.

8. What does "statically typed" mean in programming?

A. The type of each variable can change during execution

B. The type of each variable must be declared and does not change

C. No types are used in variable declarations

D. Type definitions are created at runtime

The term "statically typed" refers to programming languages in which the type of each variable must be explicitly declared at the time of variable creation, and these types do not change throughout the program's execution. This means that once a variable is defined as a specific type, it can only hold values of that type. For example, if a variable is declared as an integer, trying to assign a string or another incompatible type to that variable would result in a compilation error. This approach allows for type checking at compile time, which can help catch errors early in the development process and enhances code reliability by ensuring type consistency. In contrast, options that suggest types can change during execution or that no types are used in variable declarations describe characteristics associated with dynamically typed languages. Additionally, the concept of type definitions being created at runtime aligns more with dynamic typing, where types are determined when the code is executed rather than at compile time. This distinction is crucial for understanding the difference between static and dynamic typing in programming.

SAMPLE

9. What does "debugging" mean in the context of programming?

A. The process of writing documentation

B. The process of optimizing code for performance

C. The process of identifying and fixing errors or bugs in code

D. The process of compiling code

In programming, debugging refers to the process of identifying and fixing errors or bugs in code. This is a crucial step in software development, as errors can cause programs to behave unexpectedly or crash altogether. Debugging involves a range of techniques, such as using debugging tools or inserting print statements to track the flow of execution and monitor variable values. A programmer may encounter various types of bugs, including syntax errors, logic errors, and runtime errors, and successfully debugging ensures that the code operates as intended and meets the requirements of the project. The iterative nature of debugging is vital because it helps improve not only the quality of the software but also enhances the programmer's understanding of the code's behavior. The other options refer to different aspects of software development. Writing documentation involves explaining how the code works and is used but does not address errors. Optimizing code for performance focuses on improving the efficiency or speed of the code rather than identifying issues. Compiling code is the process of translating code written in a high-level programming language into machine code that can be executed, which does not include bug identification or resolution. Thus, the most appropriate definition of debugging is the process of identifying and fixing errors or bugs in code.

10. How does a syntax error differ from a runtime error?

A. A syntax error occurs during execution, while a runtime error occurs during coding

B. A syntax error is a logical error, while a runtime error is a compilation error

C. A syntax error occurs during coding, while a runtime error occurs during execution

D. There is no difference; both refer to the same type of error

A syntax error occurs during coding because it happens when the code is not written according to the rules of the programming language. This could include issues like missing punctuation, incorrect use of keywords, or failing to adhere to the language's grammar rules. These errors are typically caught by the compiler or interpreter before the program is executed, highlighting that they prevent the code from running successfully from the very beginning. On the other hand, a runtime error takes place during execution after the code has been successfully compiled or interpreted. Runtime errors can arise due to various unforeseen conditions that could not be predicted at the code-writing stage, such as trying to divide by zero, accessing an invalid index in an array, or running out of memory. Understanding the distinction between the two types of errors is crucial for debugging. Syntax errors must be resolved in the coding phase, while runtime errors necessitate careful testing and handling during the program's execution. This clarity helps programmers write more robust and error-tolerant code.

SAMPLE

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-itsw2113-d278.examzify.com

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

SAMPLE