PassQueen
a.k.a dumpsqueen.com

WGU Scripting and Programming Foundations Exam

WGU Scripting-and-Programming-Foundations

Version Demo

Total Demo Questions: 15

Total Premium Questions: 155

Buy Premium PDF

https://passqueen.com

support@passqueen.com

QUESTION NO: 1

What are two examples of valid function calls?

Choose 2 answers.

A. function sample(float 2.0)

B. GetHeight(integer 3, 4)

C. round(4.723, 2)

D. PrintSample()

ANSWER: C D

Explanation:

`round(4.723, 2)` is a valid function call because it uses a function identifier followed by parentheses containing comma-separated argument values. It invokes `round` with the numeric value to round and a second numeric value indicating the desired number of decimal places. This follows the general function-call pattern: a function name followed by an argument list in parentheses. In Python specifically, `round(number, ndigits)` accepts these inputs and returns a rounded numeric result; in this case, the result is `4.72`.

`PrintSample()` is also a valid function call. The identifier `PrintSample` is followed by empty parentheses, which correctly represents invoking a function that has no required parameters. A function must have been defined or otherwise made available before it can successfully execute, but the call syntax itself is valid. Empty parentheses are the standard way to call a no-argument function in many programming languages, including Python, JavaScript, C#, and Java. These examples demonstrate the key distinction between calling a function and declaring it: a call supplies only argument expressions, if any, rather than parameter type declarations. See the Python documentation for round() and its overview of defining and calling functions.

QUESTION NO: 2

Which characteristic distinguishes a markup language from other languages

A. It supports decomposing programs into custom types that often combine with other variable types into more complicated concepts.

B. It does not perform complex algorithms, but instead describes the content and formatting of webpages and other documents.

C. It allows variables to change type during execution

D. It requires fewer variables and variable conversions than other languages because the types can change during execution

ANSWER: B

Explanation:

It does not perform complex algorithms, but instead describes the content and formatting of webpages and other documents. This is the distinguishing role of a markup language: it uses tags or other markup constructs to identify parts of a document and communicate their structure, meaning, or presentation. For example, HTML marks headings, paragraphs, links, images, and lists so that a browser can interpret and render the document. XML similarly represents structured data through elements and attributes. Markup itself is declarative: it specifies what document components are, rather than providing the procedural control flow, calculations, and algorithms used to execute general-purpose programs. While webpages can include executable behavior through languages such as JavaScript, that behavior is separate from HTML's markup function. The World Wide Web Consortium describes HTML as the language used to structure and present web content, and MDN

explains that HTML uses elements to enclose and mark up different parts of content. HTML Living Standard and MDN: Basic HTML syntax provide further details on this document-structuring purpose.

QUESTION NO: 3

The steps in an algorithm to calculate the positive difference in two given values, x and y, are given in no particular order:

What is the first step of the algorithm?

A. Set Diff = x - y

B. Put Diff to output

C. Declare variable Diff

D. If y > x, set Diff = y - x.

ANSWER: C

Explanation:

Declare variable Diff is the first step because the algorithm needs a named storage location for the result before it performs calculations or produces output. The values x and y are already identified as given inputs, while Diff is the working variable that will hold the calculated nonnegative difference. Declaring it establishes the result variable used by the later assignment steps: the algorithm can initially calculate one subtraction, test the relationship between the two inputs, and, when necessary, replace the stored value with the subtraction in the opposite order. After those operations, the same variable contains the positive difference that is sent to output. This follows the usual pseudocode organization of identifying inputs and variables, processing data through assignments and decisions, and then presenting the resulting value. In particular, a variable declaration communicates the intended purpose and scope of the data item before it is referenced by an assignment or output instruction. For background on algorithm structure and pseudocode, see OpenStax: Algorithms and IBM: Pseudocode.

QUESTION NO: 4

Review the following sequence diagram:

What does a sequence diagram do?

A. Shows interactions and indicates an order of events

B. Shows interactions but does not specify an order of events

C. Shows sialic elements of software

D. Shows an order of events but does not specify all interactions

ANSWER: A

Explanation:

“Shows interactions and indicates an order of events” is correct because a UML sequence diagram models how participating objects, components, actors, or other lifelines communicate during a scenario. It represents the messages exchanged between those participants and, crucially, the sequence in which those messages occur. Lifelines are conventionally arranged across the diagram, while time progresses from the top toward the bottom. A message placed lower on the diagram therefore occurs after messages above it, making the flow of a use case, operation, or business process easy to follow.

Sequence diagrams are especially useful for describing the dynamic behavior of a system: for example, a user submits a request, an application validates it, a service retrieves data, and the application returns a response. The diagram captures both the participants involved and the ordered communication among them. The UML specification defines interactions as

behavior focused on the exchange of messages among connected lifelines, which is the core purpose represented by a sequence diagram. See the OMG UML 2.5.1 specification and IBM's overview of sequence diagrams.

QUESTION NO: 5

Consider the given function:

function K(string s1, string s2)

Put s1 to output

Put " and " to output

Put s2 to output

What is the total output when K("sign", "horse") is called 2 times?

A. sign and horse and sign and horse

B. sign and horsesign and horse

C. sign and horse

D. sign and horse

E. sign and horse sign and horse

ANSWER: B

Explanation:

`sign and horsesign and horse` is correct because each execution of the function performs three consecutive output operations. With `s1` set to `"sign"` and `s2` set to `"horse"`, one execution writes `sign`, then the literal string `and ` (which includes one leading and one trailing space), and then `horse`. The resulting character sequence from one call is therefore `sign and horse`.

Calling the function twice repeats precisely those same output operations. The pseudocode does not contain a command to output a newline, an additional space, or any delimiter after the final value of the first call. Consequently, the second call begins immediately after the first call's final `horse`, producing `sign and horsesign and horse`. This follows the general output-stream principle that separately written values are appended in the order written; a separator or line break appears only when the program explicitly writes one. For a comparable example of direct output to a stream without automatic spacing, see Python's sys.stdout documentation and the TextIOBase.write documentation.

QUESTION NO: 6

A program stores information about deliveries. Which two declarations use an appropriate data type for the value being stored?

Choose 2 answers.

A. integer packagesDelivered

B. float packageWeight

C. integer customerName

D. string invoicePaid

E. boolean invoiceTotal

ANSWER: A B

Explanation:

`integer packagesDelivered` is appropriate because the number of delivered packages is a count of whole items. `float packageWeight` is appropriate because a package weight can include a fractional amount, such as 2.5 kilograms.

A customer name is text and should use a string type. Whether an invoice has been paid is a true-or-false condition and should use a Boolean type. A monetary amount may contain a fractional component, so an integer is not the best choice among the options.

QUESTION NO: 7 - (SIMULATION)

Oder the tasks needed to safely replace a lamp's light bulb from first (1) to last (4).

Select your answer from the pull down list.

ANSWER: Check the explanation for the answer

Explanation:

Set the task order as follows:

1. **Turn off the lamp.**
2. **Remove/unscrew the old (burned-out) light bulb.**
3. **Install/screw in the replacement light bulb.**
4. **Turn on the lamp to test the new bulb.**

This sequence prevents replacing a bulb while the lamp is energized and verifies operation only after the new bulb is installed.

QUESTION NO: 8

A particular sorting algorithm takes integer list [10, 6, 8] and incorrectly sorts the list to [6, 10, 8] . What is true about the algorithm’s correctness for sorting an arbitrary list of three integers?

A. The algorithm is incorrect.

B. The algorithm only works for [10, 6, 8].

C. The algorithm’s correctness is unknown.

D. The algorithm is correct.

ANSWER: A

Explanation:

The algorithm is incorrect. Correctness for a sorting algorithm means that it must meet its specification for every input within its stated domain—in this case, every list of three integers. For an ascending sort, the output must have each element no greater than the one that follows it. On the supplied input, the algorithm produces [6, 10, 8]. Although 6 precedes 10 appropriately, 10 is greater than 8, so the resulting list is not in ascending order. The required sorted result for those same values is [6, 8, 10].

Because the input [10, 6, 8] is a valid list of three integers and produces an output that violates the sorting requirement, it is a counterexample to the claim that the algorithm correctly sorts arbitrary lists of three integers. A single valid counterexample is sufficient to establish that an algorithm is not correct for all inputs in its domain. This follows the standard definition of algorithm correctness: an algorithm must produce the specified result whenever it is given an allowed input. See the Cornell CS 2110 discussion of specifications and correctness and the NIST Dictionary of Algorithms and Data Structures definition of sorting.

QUESTION NO: 9

Which two operators can be used for checking divisibility of a number?

Choose 2 answers.

A. ^

B. *

C. +

D. $

E. /

F. %

ANSWER: E F

Explanation:

The correct operators are `/` and `%`. Division, written as `/`, obtains a quotient and can be used in a divisibility test when the result is an integer under the language's numeric rules. For example, dividing 12 by 3 produces the whole-number quotient 4, which indicates an exact division. In programming, however, the modulus operator, written as `%`, is the most direct and dependable way to check divisibility. It returns the remainder after division: an expression such as `number % divisor == 0` is true precisely when the divisor divides the number evenly. For instance, `12 % 3` evaluates to 0, so 12 is divisible by 3; likewise, `14 % 3` evaluates to 2, showing that it is not evenly divisible. These are standard arithmetic operators commonly introduced in foundational programming courses. The JavaScript reference describes remainder behavior for `%`, while Python's language reference defines the modulo operation and its relationship to division. See MDN's remainder operator reference and Python's arithmetic-operations reference.

QUESTION NO: 10

What are two examples of equality operators?

Choose 2 answers.

A. -

B. ==

C. /

D. not

E. <=

F. !=

ANSWER: B F

Explanation:

The equality operators are `==` and `!=`. Equality operators evaluate the relationship between two values and produce a Boolean result: true or false. The `==` operator tests whether its operands have equal values. For example, an expression such as `total == 10` evaluates to true when `total` currently contains the value 10. The `!=` operator tests whether its operands have different values. For example, `status != "complete"` evaluates to true when the value stored in `status` is anything other than `"complete"`. These tests are fundamental to decision-making in programs because they are commonly used in conditional statements and loops to select behavior based on whether values match or differ. Operator terminology can vary slightly by language, but introductory scripting and programming curricula consistently identify equal-to and not-equal-to as equality tests. Python's language reference lists `==` and `!=` among its comparison operations, and its

tutorial demonstrates their use in conditional expressions. See the Python comparison-operations reference and the Python tutorial on conditional statements.

QUESTION NO: 11

Which two statements accurately describe dynamically typed programming languages?

Choose 2 answers.

A. A variable can later refer to a value of a different type.

B. Many type-related checks occur while the program runs.

C. Every variable must have a fixed type declaration before execution.

D. The language can contain only text and formatting instructions.

E. Functions cannot receive values of different types.

ANSWER: A B

Explanation:

In a dynamically typed language, a variable name can be bound to a numeric value and later be bound to a string value. The type is associated with the value at runtime rather than being permanently declared for the variable name in advance.

Dynamic languages generally determine whether an operation is valid when the program executes. They do not require every variable to have a fixed declared type before the program runs, and dynamic typing does not mean that the language is markup or that a program cannot contain functions.

QUESTION NO: 12

An algorithm should output `"Eligible"` when an applicant's score is from 70 through 100, inclusive. Otherwise, it should output `"Not Eligible"`.

Which two test cases specifically verify the boundary values of the eligible range? Choose 2 answers.

A. Input 69. Ensure output is `"Not Eligible"`.

B. Input 70. Ensure output is `"Eligible"`.

C. Input 71. Ensure output is `"Eligible"`.

D. Input 100. Ensure output is `"Eligible"`.

E. Input 101. Ensure output is `"Not Eligible"`.

ANSWER: B D

Explanation:

The boundary values of the inclusive eligible range are 70 and 100. A score of 70 must produce `"Eligible"` because it is the lowest allowed score, and a score of 100 must also produce `"Eligible"` because it is the highest allowed score.

Scores such as 71 and 99 are valid interior test cases, but they do not test the endpoints where comparison errors are most likely. A score of 101 is useful for testing an out-of-range value, not the eligible-range boundary itself.

QUESTION NO: 13

A programmer has been hired to create an inventory system for a library. What is the Waterfall phase in which outlining all the functions that need to be written to support the inventory system occurs?

A. Testing

B. Analysis

C. Design

D. Implementation

ANSWER: C

Explanation:

Design is the Waterfall phase in which the system's requirements are translated into a technical blueprint that developers can implement. For a library inventory system, this work includes identifying and specifying the functions needed to fulfill the requirements, such as functions for adding inventory records, searching the catalog, recording checkouts and returns, updating availability, and generating reports. The design artifacts may define each function's purpose, inputs, outputs, processing logic, data structures, interfaces, and relationships to other parts of the system. This gives the programmer an organized plan before code is written.

In a traditional Waterfall lifecycle, work progresses from understanding what the system must accomplish to designing how it will accomplish it. Function outlines are therefore part of the solution design: they turn broad needs such as tracking books and loans into implementable program components. The resulting design specification helps ensure that implementation is systematic and that the finished inventory system supports the agreed business needs. IBM's overview of the Waterfall model describes design as the stage where requirements are used to create the system design, while the Software Engineering Body of Knowledge identifies detailed design as defining software units and their interfaces. See IBM: What is the Waterfall Model? and IEEE Computer Society SWEBOK.

QUESTION NO: 14

A program stores a customer's name, account balance, and account number together. It also provides operations to deposit funds and withdraw funds from that account.

Which programming approach is best represented by this design?

A. Object-oriented programming

B. Procedural programming

C. Compiled programming

D. Markup programming

ANSWER: A

Explanation:

The design represents object-oriented programming because related data and operations are grouped into an account object. The account's name, balance, and number represent its state, while deposit and withdrawal operations represent behavior that acts on that state.

A procedural design can contain functions, but the defining feature in this scenario is combining the data with the operations that manage it as one program entity. Compilation and interpretation describe execution approaches, not this way of organizing code.

QUESTION NO: 15

What are two example of valid function calls?

Choose 2 answers.

A. round_number(4.723, 2)

B. convort_value(12) returns cVa1

C. Printsample()

D. CountFactors(96 integer)

E. function Sample (float 2.0)

F. GetHeight(integer 3, integer 4)

ANSWER: A C

Explanation:

`round_number(4.723, 2)` is a valid function-call expression: it supplies a function identifier followed by parentheses containing a comma-separated argument list. The arguments are values, and the call does not include parameter data types or a function-definition keyword. Whether a function with that name has been declared elsewhere affects whether a complete program can execute successfully, but it does not change the fact that this line uses valid call syntax.

`Printsample()` is also a valid function call. Empty parentheses correctly represent a call with no arguments. Function names are generally case-sensitive in many programming languages, so the spelling and capitalization must match a corresponding definition when the program is compiled or run; nevertheless, the call's structure is correct.

A function call invokes an already-defined operation by writing its name and providing zero or more argument expressions inside parentheses. In contrast, declarations and definitions use syntax that identifies parameter types, and descriptive phrases such as a return-value statement are not part of the call itself. See the Python documentation's discussion of call expressions and Oracle's overview of calling methods.