hr | avatar
Illuminating Talent

Test Results and Interview Guide

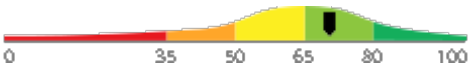
Candidate: **Elizabeth Wantsajob**
Assessment: Python Programming
Completed: June 27, 2026
Prepared for: Sara Maple
Example Company

What's Included

- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Python Programming assessment measures key factors related to high performance and tenure in this job. Attribute types measured vary by test, but can include cognitive ability, skills, knowledge, personality characteristics, emotional intelligence, and past behavioral history. This report includes a one page summary, followed by detailed results with an embedded interview guide. Note that these results should always be used as a part of a balanced candidate selection process that includes independent evaluation steps, such as interviews and reference checks.

Overall

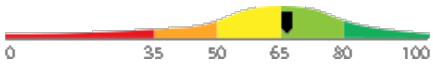
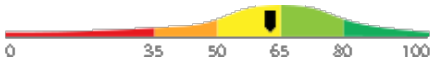
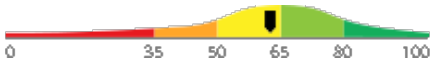
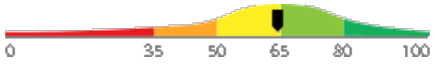
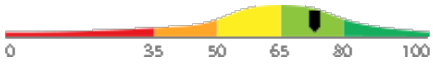
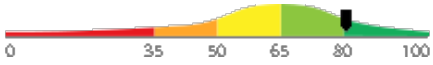
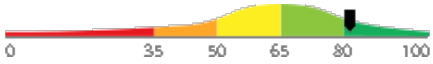
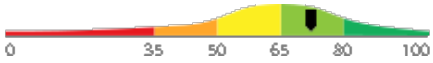
Candidate	Score	Interpretation
Elizabeth Wantsajob	70	

beth.wantsajob@gmail.com
Python Programming
June 27, 2026

The candidate exhibits a solid working knowledge of Python programming, including proficiency in data structures, object-oriented design, exception handling, file operations, and module organization. Minor gaps may exist in advanced areas such as performance analysis, testing utilities, or environment virtualization, but the candidate is well-suited for entry-level to mid-level programming responsibilities.

Key
- Candidate Score
- Higher Risk
- Lower Risk

Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Control Flow and Functions	67	
Control Flow and Functions (Coding Tasks)	62	
Core Data Structures (Coding Tasks)	62	
Core Data Structures	64	
Error Handling and Debugging	73	
File Handling and Data Processing	81	
Modules, Packages, and Environments	81	
Object-Oriented Programming (OOP)	72	

Comparison

Percentile scores indicate how the candidate compares to other test-takers within various groups. The candidate scored equal to or better than the fraction of test-takers indicated by the percentile.

Test-Taker Group	Percentile	0 10 20 30 40 50 60 70 80 90 100
Global	70th	
North America	58th	
United States	58th	
Example Company	65th	

Artificial Intelligence (AI) Generated Scores

This table includes one or more scores derived from a large language model AI query. AI-derived scores are non-deterministic. That is, they are not precisely repeatable. Therefore, these scores should always be treated as supplementary information and should never be used exclusively or compared to hard cutoff values.

Estimated Value	Score	Confidence	Interpretation
Knowledge, Skills, and Abilities Summary	-	-	Summary Points (AI): • (Generic Text for Sample Report) Strong performer in Drag and Drop Files tasks, indicating comfort with file management and basic computer interactions. • Demonstrates solid numerical accuracy in Recognizing and Confirming Numbers, a valuable asset in detail-oriented roles. • Moderate overall performance in Analytical Thinking and Attention to Detail, with adequate grammar skills but room for improvement. • Struggles with Reading and Analyzing Problems, which may limit effectiveness in roles requiring critical reading and complex problem-solving. • Lowest performance in Navigating Between Screens, suggesting difficulty with multi-screen software workflows that could impact productivity in computer-intensive roles. Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies. Elizabeth shows a strong aptitude in Drag and Drop Files, performing well on this technical task and suggesting she is comfortable with this type of computer interaction. This is a notable strength that would translate well into roles requiring file management and basic computer navigation tasks. In the area of Analytical Thinking and Attention to Detail, Elizabeth performs at a moderate level. She demonstrates solid ability in Recognizing and Confirming Numbers, which suggests she is careful and accurate when working with numerical data — a valuable skill in detail-oriented work environments. Her Grammar performance is adequate but leaves room for improvement, indicating she may occasionally make written communication errors. Her weakest area within this competency is Reading and Analyzing Problems, where she struggled to consistently interpret and work through written problem scenarios. This may impact her effectiveness in roles that require critical reading, written comprehension, or complex problem-solving. Elizabeth's most significant area for development is Navigating Between Screens, where she scored considerably lower than the other competencies. This suggests she may have difficulty efficiently moving through software interfaces or multi-screen workflows, which could slow productivity in roles that rely heavily on navigating computer applications or data entry systems. Overall, Elizabeth brings some useful technical strengths, particularly in file management and numerical accuracy, but would benefit from targeted development in software navigation and analytical problem-solving to be fully effective in roles that demand these skills. Computed on: April 2, 2026, 11:09:49PM EDT

Detail

Candidate:	Elizabeth Wantsajob, beth.wantsajob@gmail.com
Assessment:	Python Programming
Authorized:	June 27, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 27, 2026, 1:26:42PM EDT
Completed:	June 27, 2026, 1:26:42PM EDT
Overall Score:	70

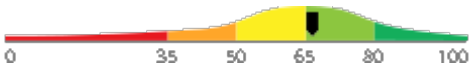
Knowledge and Skills Detail

This section contains a list of job-related knowledge areas and skills that have been evaluated. Low scores in these areas often indicate that additional learning may be required before top performance can be achieved.

Detail | Interview Guide

Control Flow and Functions

Score: 67



Description:

Covers the use of conditional statements, loops, and function definitions to control how a program executes. Includes writing reusable functions with parameters, return values, and scope, as well as using loops and conditionals to direct program logic.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate exhibits a solid understanding of Python control flow and functions, competently applying conditional logic, loops, and function definitions in most scenarios. They demonstrate reliable knowledge of parameters, return values, and scope, with only minor gaps in more advanced or complex applications.

How would you write a function that accepts a variable number of arguments, and how would you handle both positional and keyword arguments within it?

1	2	3	4	5
Unaware of *args and **kwargs syntax or confuses their purpose.		Correctly describes *args and **kwargs with a basic working example.		Demonstrates confident use of *args and **kwargs, explains unpacking, and provides a practical, well-structured example.

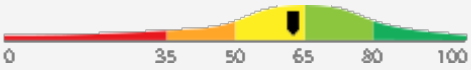
Can you explain the difference between a 'for' loop and a 'while' loop, and describe a situation where you would use each one?

1	2	3	4	5
Cannot clearly distinguish between the two or provides incorrect descriptions of how they work.		Correctly describes both loops and gives a basic example for each, though the use cases may be generic.		Clearly explains the difference, gives specific practical use cases, and mentions edge cases like infinite loops or iterables.

Detail

Control Flow and Functions (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

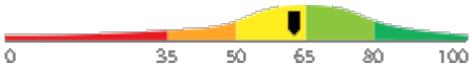
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Core Data Structures (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

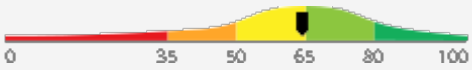
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Core Data Structures

Score: 64



Description:

Covers the use of Python's built-in data structures including lists, dictionaries, tuples, and sets. Includes creating, accessing, modifying, and iterating over these structures to store and manipulate data in everyday programming tasks.

Interpretation:

Candidate appears capable of average job performance in this area with little or no training.

The candidate possesses a moderate understanding of Python programming concepts, demonstrating partial familiarity with data structures, control flow, and standard libraries. While capable of handling some entry-level programming tasks, gaps in areas such as exception handling, testing, environment configuration, or performance analysis may limit independent effectiveness without additional support.

Interview Guide

Walk me through how you would use a dictionary to count the frequency of words in a list of strings, and explain any built-in tools you might use to simplify the task.

1	2	3	4	5
Describes a vague or incorrect approach; unaware of tools like collections.Counter or dict.get().		Provides a working loop-based solution using a dictionary but does not mention built-in helpers.		Provides a clean solution using collections.Counter or setdefault/get, and explains trade-offs clearly.

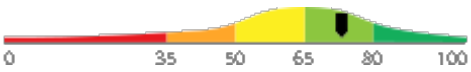
Can you describe what a Python dictionary is and give an example of when you would use one instead of a list?

1	2	3	4	5
Cannot distinguish between a dictionary and a list or provides a vague, incorrect explanation.		Correctly describes a dictionary as key-value pairs and gives a basic but valid use case.		Clearly explains key-value structure, mutability, and gives a specific, practical use case with correct syntax.

Detail

Error Handling and Debugging

Score: 73



Description:

Covers the use of try, except, and finally blocks to catch and manage errors without crashing a program. Includes identifying common error types, raising custom exceptions, and using debugging and logging tools to track down and fix problems in code.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid and competent understanding of Python error handling and debugging practices. They are proficient in using try, except, and finally blocks, recognizing common error types, raising custom exceptions, and applying debugging and logging tools to resolve issues in code.

Interview Guide

How would you use Python's logging module to track events in an application, and how is it more useful than using print statements for debugging?

1	2	3	4	5
Unaware of the logging module or cannot explain its advantage over print statements.		Knows the logging module exists, can set a basic log level, and explains that logs can be filtered or redirected.		Configures logging with levels, handlers, and formatters; clearly articulates benefits like severity filtering and persistent log output.

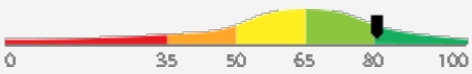
What is a try-except block, and why would you use one in your code?

1	2	3	4	5
Cannot explain what a try-except block does or confuses it with a conditional statement.		Correctly explains that try-except catches errors and prevents crashes, with a basic example.		Explains try, except, and finally clearly, mentions catching specific exception types, and gives a realistic use case.

Detail

File Handling and Data Processing

Score: 81



Description:

Covers reading from and writing to files using Python's built-in file tools. Includes working with plain text and common data formats, processing numeric and string data, and managing dates and times using standard Python libraries.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate demonstrates a strong and comprehensive mastery of Python file handling and data processing. They are highly proficient in reading from and writing to files, handling a variety of data formats, and effectively leveraging standard Python libraries to process numeric, string, and date and time data with accuracy and confidence.

Interview Guide

Describe how you would read data from a CSV file, process its contents, and write a modified version of the data back to a new file.

1	2	3	4	5
Unsure how to read a CSV or manually parses it without using appropriate standard library tools.		Uses the csv module to read and write files correctly with a basic processing step.		Uses csv.DictReader/DictWriter fluently, handles edge cases like headers and encoding, and describes a clear data transformation step.

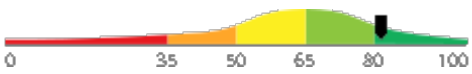
How do you open and read the contents of a text file in Python, and why is it recommended to use a 'with' statement when doing so?

1	2	3	4	5
Cannot describe how to open a file or is unaware of the 'with' statement and its purpose.		Correctly shows how to open and read a file using 'with open()' and explains that it closes the file automatically.		Demonstrates correct syntax, explains context managers, mentions file modes, and discusses handling large files efficiently.

Detail

Modules, Packages, and Environments

Score: 81



Description:

Covers how to organize Python code into reusable modules and packages, and how to manage project dependencies. Includes importing standard and third-party libraries, creating virtual environments to isolate dependencies, and understanding how Python resolves module imports.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits a strong and comprehensive understanding of Python modules, packages, and environment management, including importing standard and third-party libraries, creating and managing virtual environments, and understanding Python's module resolution process. They are well-equipped to independently design, organize, and maintain robust Python project structures and dependency configurations.

Interview Guide

Walk me through how you would set up a virtual environment for a new Python project, install dependencies, and ensure another developer can replicate your setup.

1	2	3	4	5
Unfamiliar with virtual environments or cannot describe how to create and activate one.		Correctly describes creating a virtual environment with venv, installing packages with pip, and generating a requirements.txt file.		Describes the full workflow including venv creation, activation, pip install, requirements.txt generation, and discusses why isolation matters for reproducibility.

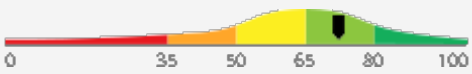
What is the difference between importing an entire module and importing a specific function from a module, and when might you prefer one approach over the other?

1	2	3	4	5
Cannot distinguish between 'import module' and 'from module import function' or explain when to use each.		Correctly describes both import styles and gives a basic reason for preferring one, such as readability or namespace management.		Explains both styles, discusses namespace clarity, performance considerations, and gives practical examples of each in context.

Detail

Object-Oriented Programming (OOP)

Score: 72



Description:

Covers the design and use of classes and objects to organize code. Includes defining classes with attributes and methods, using inheritance to share behavior between classes, and applying encapsulation to manage how data is accessed and modified.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid and proficient understanding of object-oriented programming in Python. They are capable of designing and working with classes and objects, applying inheritance to share behavior, and using encapsulation to appropriately manage data access and modification.

Interview Guide

Describe how inheritance works in Python and give an example of how a child class can extend or override behavior from a parent class.

1	2	3	4	5
Vague understanding of inheritance; cannot explain how a child class relates to a parent class.		Correctly explains inheritance and demonstrates a basic override using a child class.		Explains inheritance clearly, uses super() correctly, and gives a practical example with meaningful method overriding.

Can you explain what a class is in Python and describe the purpose of the '__init__' method?

1	2	3	4	5
Cannot clearly define a class or does not know the role of __init__ in setting up an object.		Correctly explains that a class is a blueprint and that __init__ initializes object attributes.		Clearly explains class structure, __init__, self, and gives a practical example showing attribute assignment and method use.

IT Coding Tasks

During the assessment, the candidate was asked to write one or more programs or scripts. Their responses are included below for review.

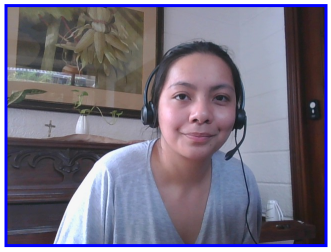
Question or Task	Response

Complete the provided partial C program by filling in the missing sections marked with TODO comments. Your completion must use standard C keywords and libraries.
 	
The program declares a function called duplicate_array that:
1. Takes a const int pointer to a source array and its length as parameters.
2. Uses calloc to allocate a new int array of the same length.
3. Returns NULL if calloc fails.
4. Copies each element from the source array into the new array using pointer arithmetic (not array subscript notation).
5. Returns the pointer to the newly allocated copy.
 	
In main, the program:
1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35.
2. Calls duplicate_array to create a heap-allocated copy.
3. Checks for NULL and prints an error and returns 1 if the call failed.
4. Prints each element of the duplicate using a loop.
5. Frees the duplicate array.
 	
Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.
	#include <stdio.h>
	#include <stdlib.h>
	
	int *duplicate_array(const int *src, int length) {
	/* TODO: Use calloc to allocate a new array of 'length' integers, return
	NULL if calloc fails, copy elements from src using pointer arithmetic,
	and return the new pointer. */
	calloc(303);
	}
	
	int main(void) {
	/* TODO: Declare and initialize a stack array of 4 integers: 5, 15, 25, 35,
	then call duplicate_array and store the result. Check for NULL and
	print an error message returning 1 if it failed. */
	array[4]={5,15,25,35};
	
	int i;
	
	/* Print each element of the duplicate */
	for (i = 0; i < 4; i++) {
	printf("duplicate[%d] = %d\n", i, *(duplicate + i));
	}
	
	/* Free the duplicate array */
	free(duplicate);
	return 0;
	}
	
	Comments (AI): The code segment has several syntax errors and incomplete implementation. The duplicate_array function does not correctly allocate memory or copy elements. The main function has syntax errors and does not properly call the duplicate_array function. However, the structure and intent of the code are somewhat clear, and the code attempts to follow the requirements.

Identity Confirmation Photos

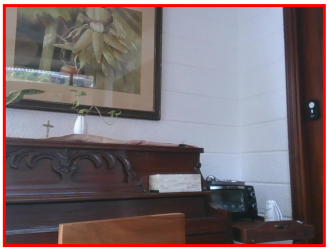
The following photos of the candidate and any identification were uploaded during the assessment session.

Photo Analysis Results	
- Risk:	Medium risk of cheating based on image inconsistencies
- Percent match among processed faces	100%
- Total images processed	17
- Total images with valid faces	14 (82%)
- Total pairs of faces compared	13
- Pairs in which faces matched	13 (100%)

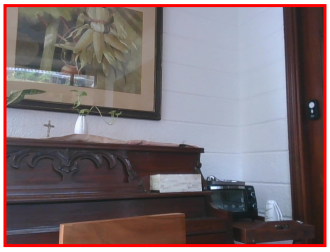


Pre/Post-Test Photo

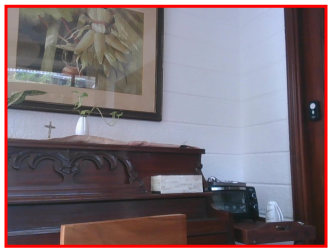
ID Photo



In-Test Error Detected (No Face Detected)

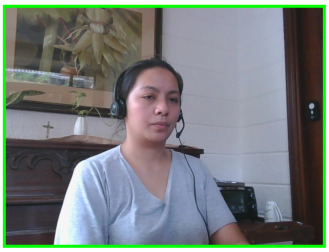


In-Test Error Detected (No Face Detected)

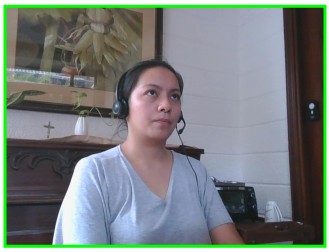


In-Test Error Detected (No Face Detected)

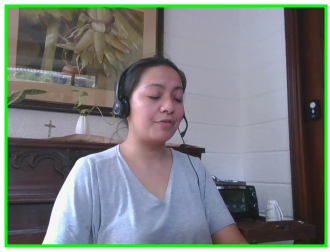
In-Test Photo



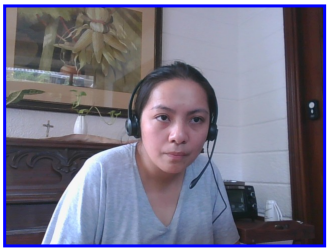
In-Test Photo



In-Test Photo



In-Test Photo



Pre/Post-Test Photo

Resume or CV

Summary Updated on

Motivated career professional with extensive experience in office administration and management. Proven track record of improving efficiency, reducing costs, and enhancing office operations through strategic initiatives and technology implementation.

Objective

I am seeking a role where I can use my many skills and my exceptional judgment and empathy for customers to make a difference to a growing company.

Education

- Associate of Applied Science in Office Administration, Portland Community College, 2020

Experience

- General Office Clerk, Paramount Office Management, 09/2023 – Present
- Administrative Assistant, Global Enterprises Inc., 04/2021 – 08/2023
- Administrative Assistant, Innovative Business Solutions Ltd., 07/2019 – 03/2021

Other Qualifications

- Microsoft Office Specialist (MOS) Certification
- Certified Administrative Professional (CAP)
- International Association of Administrative Professionals (IAAP) Certification

Report Preparation Notes

- Hiring decisions should never be based on a single source of information. The most effective use of this assessment report is as a part of a multi-faceted program of candidate evaluation that includes resume review, interviews, and reference checks.
- Overall vs Percentiles Scores: The overall score reflects the success in the test, based on the mean (average) and standard deviation of the test scores. The percentile score reflects the percentage of test-takers who scored equal or below this overall score. We recommend you use the Overall Score as your primary evaluation criteria. However, percentile scores can often be useful in comparing specific candidates against one another and with a group, such as for test takers in a certain organization or within a certain account.
- Note that comparison information is calculated based on completed instances of this assessment at that time the assessment is scored. As additional instances are completed, the comparative data may change. You can always update a report to the current values by clicking on 'Recalculate Percentiles' within the online results viewing pages at www.hravatar.com.
- Most competency scores are norm-based, which means that they can be interpreted in terms of their distance from the average or mean score. For all scales, a score equal to the mean receives a score of 65 and scores above and below this value are set so that a score change of 15 equals one standard deviation.
- For linear competencies, higher is better across the entire scale. For these scales a score between 65 and 80 (light green) represents 0 to 1 standard deviation above the mean and a score above 80 (dark green) represents more than one standard deviation above the mean. Similarly, a score of 50 - 65 (yellow) represents 0 to 1 standard deviation below the mean, while a score of 35 - 50 (orange) equates to 1 to 2 standard deviations below the mean, and a score below 35 represents more than 2 standard deviations below the mean.
- Sim ID: 20435-1, Key: 0-0, Rpt: 68, Prd: 9520, Created: 2026-06-27 13:26 EDT
- UA: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; Touch; rv:11.0) like Gecko

Score Calculation Detail

The following table provides a summary of how the overall score was calculated from each of the individual competency scores. First, all competency scores are calculated on a scale of 0-100. Note that some competencies use their color category rather than their actual numeric score in the overall calculation. For these, a standard score associated with the assigned color category is used in the overall score calculation rather than the actual numeric score. This is reflected in the ''Score Value Used'' column. Next, a weighted average of scores is computed using individual competency weights, typically set using job analysis data provided by the US Government Occupational Information Network (O*Net).

Competency	Score	How applied to overall	Score Value Used	Weight (%)
Control Flow and Functions	67.0698	Numeric Score	67.0698	12.5000
Control Flow and Functions (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Core Data Structures	64.6750	Numeric Score	64.6750	12.5000
Core Data Structures (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Error Handling and Debugging	73.4752	Numeric Score	73.4752	12.5000
File Handling and Data Processing	81.1012	Numeric Score	81.1012	12.5000
Modules, Packages, and Environments	81.7484	Numeric Score	81.7484	12.5000
Object-Oriented Programming (OOP)	72.6469	Numeric Score	72.6469	12.5000
Weighted Average:				70.8342
Final Overall Score:				70

Notes

(This area is intentionally blank - it's reserved as space for your notes.)