TOSA

Your skills. Your advantage.

Tosa Python

Exam Blueprint

contact@isograd.com

Table of Contents

Table of Contents 2

Introduction to the Tosa® Exam Blueprint 3

- Tosa® (Test on Software Applications) 4
- Tosa® Exam Blueprint Objective 4
- Unique Tosa® Scoring 5
- About the Tosa® Python certification 6
- Tosa Sequence for Progressive Skills Development 7
- Isograd Learning Lab 8
- Tosa® Python Level Descriptions 9
- Business Applications 10
- Domain 1: Fundamentals and Applications 11
  - Sub-domain: Getting started with Python 11
- Domain 2: Language and Syntax 12
  - Sub-domain 1: Mastering basic syntax and control structures 12
  - Sub-domain 2: Defining and using functions 13
  - Sub-domain 3: Applying advanced programming concepts 14
- Domain 3: Data Structures and Objects 15
  - Sub-domain 1: Handling primitive and compound data types 15
  - Sub-domain 2: Using object-oriented programming 16
  - Sub-domain 3: Managing and optimizing data 17
- Domain 4: Modules and Packages 18
  - Sub-domain 1: Using and creating modules 18
  - Sub-domain 2: Developing and distributing packages 19
  - Sub-domain 3: Managing environments and dependencies 20
- Domain 5: Code Optimization 21
  - Sub-domain 1: Analyzing and profiling code 21
  - Sub-domain 2: Improving code efficiency 22
  - Sub-domain 3: Testing and debugging 23

All reproduction prohibited

Introduction

to the Tosa® Exam Blueprint

For Tosa® Assessment and Certification

All reproduction prohibited

Tosa® (Test on Software Applications)

Tosa® assessments and certifications are designed to determine a candidate's proficiency level by evaluating their skills in office software and digital tools commonly used in a professional or educational environment.

These tests are specifically developed to validate the professional competencies of candidates looking to enhance their employability (employees, students, job seekers, and individuals undergoing career transitions).

Tosa® assessments and certifications are adaptive tests, developed using scientific methodologies. The scoring is based on Item Response Theory (IRT). The test algorithm adapts to each candidate's response in real time, adjusting the difficulty level of subsequent questions until it precisely determines the candidate's skill level by calculating the upper limit of their competencies. As a result, the tests provide a detailed and unique diagnosis of each candidate's abilities.

The rigor and reliability of Tosa® tests stem from the combination of a mathematical model for analyzing question difficulty and the relevance of the questions selected for each candidate (IRT).

Tosa® Exam Blueprint Objective

This exam blueprint outlines all the skills assessed within the domains and sub-domains of the Tosa® assessment and certification tests for Python.

Tosa® assessment and certification solutions are designed to determine learners' proficiency levels using a single scoring scale—ranging from 0 to 1000 for certification—and divided into five levels, from "Beginner" to "Expert," for the assessment.

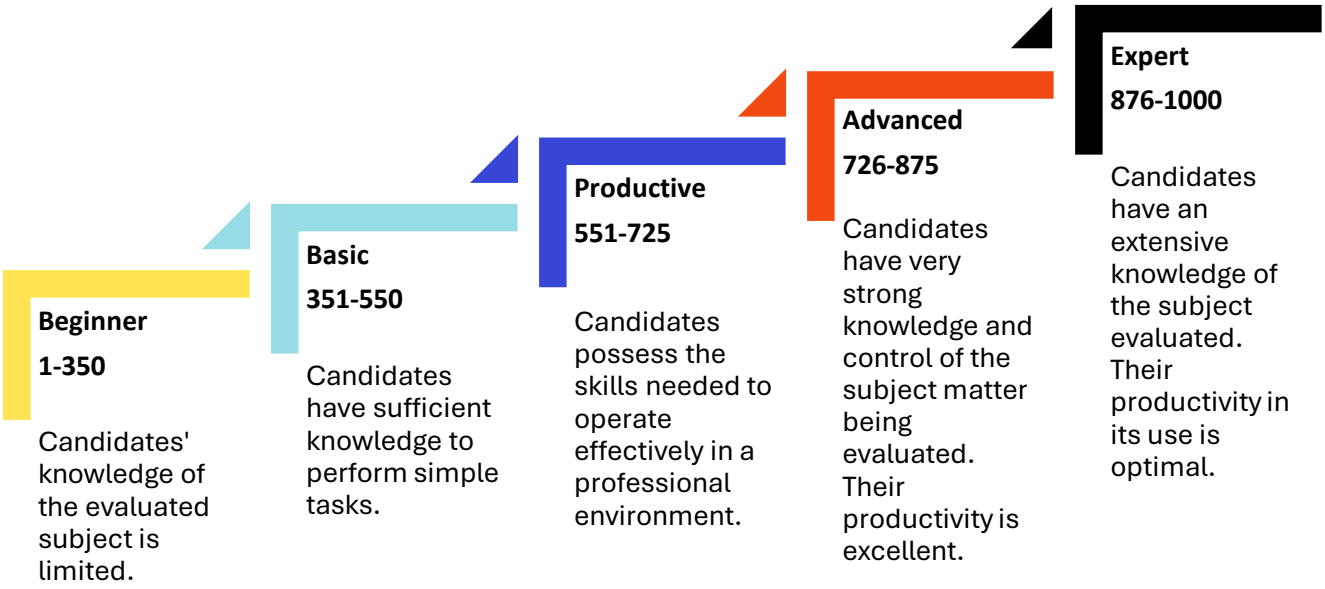
The purpose of this blueprint is to specify the technical knowledge expected at each level and within each of the four main skill categories of Python. It is intended to help identify the most appropriate teaching or training programs to match a learner's target score.

All reproduction prohibited

Unique Tosa® Scoring

The Tosa® assessments and certifications are based on a unique score, divided into five levels.

- Ranging from 1 to 1000 for the certification.
- Divided into five levels, from Beginner to Expert, for the assessment.



Certification failed	Certification failed	Certification earned	Certification earned	Certification earned
Certificate of completion issued	Certificate of completion issued	Diploma & Credly digital badge issued	Diploma & Credly digital badge issued	Diploma & Credly digital badge issued

All reproduction prohibited

About the Tosa® Python certification

The Tosa Python Certification relies on a database of around 135 questions. It is composed of 35 questions from the question database and lasts for 1 hour and 30 minutes. The algorithm adapts to each of the candidate's answers to adjust the difficulty level of the questions until reaching the candidate's skill limit. This ensures a precise and accurate result.

Since the test is adaptive, the series of questions that a candidate gets is unique for each test. This algorithm allows for a more accurate evaluation of the candidate's level. It also limits cheating and the memorization of questions on different passages.

Our platform allows individuals to take the certification in a classroom setting, an approved testing center, or remotely via our integrated asynchronous online proctoring solutions.

Our remote proctoring solutions provide added flexibility for both the administrator and the candidate, allowing the certification exam to be taken anywhere, at any time. The candidate only needs an internet connection and a computer equipped with a working webcam and microphone.

Candidates receive a numeric score out of 1000 points. This score corresponds to one of five proficiency levels. Candidates who score between 1 and 550 points do not earn the certification. They receive a certificate of completion in lieu of a diploma. Candidates who score 551 points or above earn the certification and will receive a diploma by email within five business days. If a candidate scores 551 points or above, they will also be eligible to receive a digital Credly badge.

There are no prerequisites to be eligible to take the exam, but to ensure a candidate is well prepared on exam day, we suggest they:

- Take at least one Tosa Python adaptive assessment to estimate their level and get familiar with the test format
- Use the free practice tests on our website for training
- Follow an e-learning or training course (average duration per level is between 10 and 15 hours per certification, so around 150 hours total)

Tosa certification diplomas are valid for three years from the date of issue. This three-year period has been set to ensure that our certifications are consistently accurate and relevant, taking into account software version updates as well as the natural evolution of a candidate's skills over time. Limiting the certification period also reflects the need for life-long learning and continual professional development.

Tosa certifications can be retaken when they expire. Earners willing to improve their score and proficiency level can retake the exam at any time.

All reproduction prohibited

Tosa Sequence for Progressive Skills Development

Students and professionals can tailor their certification journey with Tosa through vertical progression, demonstrating increasing proficiency in Python. Starting at a productive level, users can advance to advanced and expert levels as their skills grow. This clear path encourages continuous improvement and validates each stage of their development. Tosa's structure makes it easy to track progress and showcase evolving expertise.

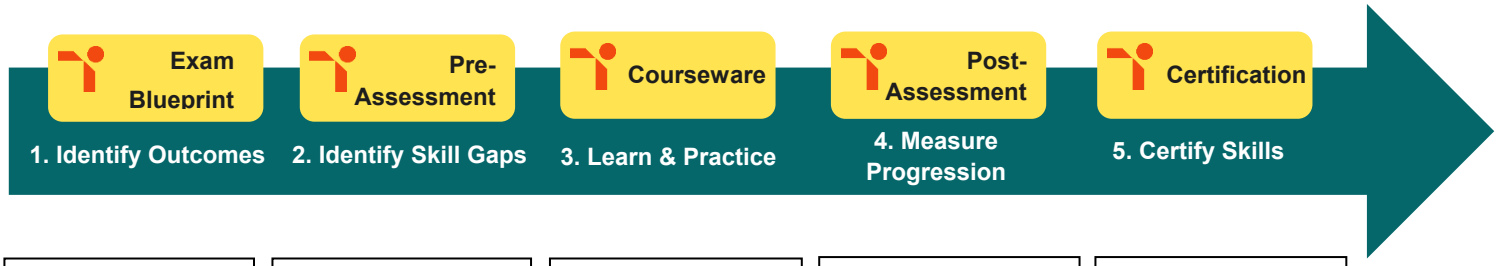
All reproduction prohibited

Isograd Learning Lab

The Isograd Learning Lab is a multifaceted and adaptable courseware solution designed to help learners prepare for Tosa certification exams. It offers personalized, self-paced, and fully interactive learning experiences, equipping candidates with the essential digital skills that employers seek. These skills span a wide and varied range, applicable to learners just starting their career journey to those wishing to advance on their path.

The browser-based platform supports all learning styles with a wide array of features, including in-application exercises. These in-application capabilities allow learners to experience real-world examples of specific tasks within a given software environment. The lab includes inclusive learning resources, along with extension activities and project-based learning challenges that foster creativity and critical thinking.

All course content on the Isograd Learning Lab is aligned to a Tosa exam blueprint, which is the foundation of any Tosa certification exam. By aligning course content to an exam blueprint learners will be prepared to take the exam once they complete the courseware.



1. Identify Outcomes	2. Identify Skill Gaps	3. Learn & Practice	4. Measure Progression	5. Certify Skills
Using a Tosa® exam bluprint, instructors integrate Tosa® certifications into their existing content or curriculum and set define outcomes.	Students begin with a formative pre-course assessment to identify their current skills level. Instructors gather data and adjust lesson plans based on students' results.	Instructors match courseware in the Isograd Learning Lab to student's proficiency levels and intended course objectives.	Once the courseware is completed, students take a summative assessment to verify their skills match the intended proficiency level. If skill gaps still exist, instructors can assign additional course content.	When instructors are satisfied with students' progress, certification exams are given.

All reproduction prohibited

Tosa® Python Level Descriptions

At each level, candidates can:

Beginner

- Define Python and identify the apropriate audience.
- Demonstrate foundational knowledge of Python, such as basic syntax, use of variables, and simple control structures (loops and conditions).
- Write small scripts to automate simple tasks, but may need assistance with more complex tasks or debugging code.

Basic

- Identify the first steps in program developement and Python's features.
- Apply basic knowledge of Python such as data processing or automating simple tasks.
- Demonstrate knowledge of how to use functions, handle exceptions, and work with files.
- Solve common problems or execute clearly defined tasks independently.

Productive

- Identify tasks and projects relevant for Python.
- Master standard libraries, use the math and random packages.
- Structure larger projects, use version control systems, and deploy applications.
- Demonstrate autonomy and initiative in solving more complex problems.

Advanced

- Differentiate Python from other programming languages.
- Master Python's core syntax, variables, printing, mutability, and basic classes.
- Use and reuse standard library packages, data structures, and functions.
- Optimize code for efficiency and performance.
- Work independently or as part of a team on projects.

Expert

- Explain use cases for Python.
- Handle errors effectively.
- Use decorators, generators, and class methods to structure advanced Python code.
- Work with files and system tools using different formats and the sys and os modules.
- Optimize code and choose the right data structure for each use case.

All reproduction prohibited

Business Applications

Achievement of Beginner score defines little or limited knowledge of Python, including the languages' basic functions and features, highlighting the candidate's inability to perform in a professional environment.	*Administrative assistant, customer service representative, office support employee, or any position requiring the use of Python to perform basic data manipulation, automate repetitive tasks, or interact with structured data files (e.g., CSV, Excel).*	*Operations assistant, project coordinator, junior data analyst, IT support technician, or any role requiring the regular use of Python for structured data analysis, report generation, and the development of reusable scripts to support daily business operations.*	*Financial controller, data analyst, project manager, management consultant, or any position requiring advanced use of Python for managing and analyzing complex datasets, creating dashboards, and developing automated workflows to inform strategic decisions.*	*Senior data analyst, senior project manager, chief financial officer, business intelligence consultant, or any role requiring expert-level proficiency in Python for designing and deploying scalable data solutions, performing predictive analytics, and supporting high-level decision-making through advanced modeling and automation.*

All reproduction prohibited

Domain 1: Fundamentals and Applications

Sub-domain: Getting started with Python

Includes identifying when and for whom to use Python, and matching Python to real-world use cases.

Beginner

- Describe what Python is and what it is used for.
- Identify audiences an scenarios where Python is a good choice.
- Recognize basic programming terms and their meanings.

Basic

- Distinguish between types of programming languages.
- Identify the first steps in program development.
- Recognize language-specific features of Python.

Productive

- Identify tasks that Python can be used to accomplish.
- Explain Python's value in applied context.
- Identify projects well-suited for Python implementation.

Advanced

- Differentiate Python from other programming languages.

Expert

- Explain optimal and sub-optimal use cases for Python.

All reproduction prohibited

Domain 2: Language and Syntax

Sub-domain 1: Mastering basic syntax and control structures

Includes basic Python syntax, loops, conditionals, and error handling.

Beginner

- Identify basic data types such as integers, floats, strings and Booleans.
- Identify simple arithmetic operators to perform basic calculations.

Basic

- Apply basic control structures such as "if," "for," and "while" statements to control program flow.
- Manipulate strings and lists for simple tasks such as adding, deleting or searching for elements.

Productive

- Handle common exceptions using "try" and "except" to prevent errors during code execution.
- Implement simple functions with arguments and return values to organize code and improve reusability.

Advanced

- Use list comprehensions to create lists more concisely and efficiently.
- Integrates asynchronous flow control to manage I/O operations without blocking program execution.

Expert

- Optimize control structures to maximize code efficiency and clarity, including refining conditions and improving loop logic.
- Develop complex control structures incorporating multiple levels of conditions, nested loops, and advanced error handling for large-scale applications.

All reproduction prohibited

Sub-domain 2: Defining and using functions

Covers function creation, variable scope, function arguments and returns, and lambda functions.

Beginner

- Read and understand simple functions to automate repetitive tasks and structure code.
- Use basic parameters to customize function behavior.

Basic

- Include function arguments with default parameters and named arguments.
- Create functions that return values to integrate calculated results into other parts of the code.

Productive

- Design lambda functions for short, specific operations without the need for a formal function definition.
- Use functions as first-class objects, passing them as arguments or using them as return values.

Advanced

- Develop decorators to modify or extend function behavior without changing the source code.
- Implement recursive functions to handle complex data structures such as nested lists or trees.

Expert

- Optimize the use of functions in production environments, by evaluating their performance and improving their efficiency.
- Design software architectures using advanced functional programming strategies to improve code modularity and testability.

All reproduction prohibited

Sub-domain 3: Applying advanced programming concepts

Includes decorators, generators, list comprehension and asynchrony.

Beginner

- Recognize the usefulness of list comprehensions to simplify the creation of lists from existing sequences.
- Identifiy where the use of lambda expressions can make code clearer and more succinct.

Basic

- Use generators to create iterators with reduced memory usage.
- Apply regular expressions for string manipulation and data validation.

Productive

- Integrate asynchronous control into applications to improve performance by handling I/O in a non-blocking way.
- Manipulate metaclasses to change the behavior of classes when they are created.

Advanced

- Design context managers to efficiently manage resources such as files and network connections.
- Develop asynchronous programming protocols for complex web or network applications.

Expert

- Innovate by using advanced language features to create elegant solutions to complex problems.
- Contribute to open source projects by proposing improvements to the Python language or its main libraries.

All reproduction prohibited

Domain 3: Data Structures and Objects

Sub-domain 1: Handling primitive and compound data types

Includes the use of lists, tuples, dictionaries, sets, and common operations on these structures.

Beginner

- N/A

Basic

- Manipulate simple data such as lists and dictionaries to store and access information.
- Use basic operations such as adding, deleting, and accessing elements in these structures.

Productive

- Optimize data access using techniques such as list slicing and dictionary mapping.
- Create classes to encapsulate complex data structures and their associated operations.

Advanced

- Develop customized data structures to solve specific problems using advanced classes and methods.
- Integrate complex algorithms to optimize data processing and manipulation.

Expert

- Design systems to manage and analyze large quantities of data efficiently.
- Innovate in the creation of new data structures that significantly improve application performance.

All reproduction prohibited

Sub-domain 2: Using object-oriented programming

Covers class creation, inheritance, polymorphism, and advanced design principles such as abstract classes and interfaces.

Beginner

- Understand the importance of encapsulation to protect data and internal object behavior.

Basic

- Apply inheritance to extend or modify the functionality of existing classes.

Productive

- Implement polymorphism to use common interfaces for different object types.
- Design abstract classes to define interfaces that other classes must implement.

Advanced

- Develop software architectures that integrate multiple object-oriented design patterns to solve complex problems.
- Use advanced design patterns to improve code flexibility and maintainability.

Expert

- Innovate in the design of object-oriented architectures for large-scale systems, integrating advanced design practices and optimizing object interactions.
- Conduct code reviews and mentoring sessions to improve object-oriented design skills within a team.

All reproduction prohibited

Sub-domain 3: Managing and optimizing data

Includes advanced data manipulation with libraries such as pandas, and optimization of data structures for performance.

Beginner

- Understand the importance of organizing code into modules for better maintainability.

Basic

- Organize code into functions and classes within modules for greater clarity.

Productive

- Package modules for distribution and reuse in other projects.
- Document modules to facilitate their use by other developers.

Advanced

- Develop complex packages that include several interdependent modules.
- Uses advanced techniques for namespace management and dependency resolution within packages.

Expert

- Design modular frameworks used by a large community of developers.
- Leadssoftware development projects based on modular architectures to promote scalability and flexibility.

All reproduction prohibited

Domain 4: Modules and Packages

Sub-domain 1: Using and creating modules

Includes importing existing modules, creating new modules, and structuring code into reusable modules.

Beginner

- Understand the importance of organizing code into modules for better maintainability.

Basic

- Use standard modules to add functionality to programs without reinventing the wheel.
- Organize code into functions and classes within modules for greater clarity.

Productive

- Package modules for distribution and reuse in other projects.
- Document modules to facilitate their use by other developers.

Advanced

- Develop complex packages that include several interdependent modules.
- Use advanced techniques for namespace management and dependency resolution within packages.

Expert

- Design modular frameworks used by a large community of developers.
- Lead software development projects based on modular architectures to promote scalability and flexibility.

All reproduction prohibited

Sub-domain 2: Developing and distributing packages

Covers package creation, use of setup tools for package configuration, and distribution

Beginner

- N/A

Basic

- Publish packages to local repositories for sharing within a team.

Productive

- Manage package versions to ensure compatibility and stability across releases.
- Automate package testing and deployment to guarantee quality before distribution.

Advanced

- Develop strategies for dependency management and conflict resolution in complex environments.
- Integrate CI/CD tools to automate package release and maintenance.

Expert

- Lead initiatives to standardize package development and distribution practices within large developer communities.
- Innovate by creating tools and methodologies to improve large-scale package distribution and management.

All reproduction prohibited

Sub-domain 3: Managing environments and dependencies

Covers the use of virtual environments, managing dependencies with pip, and automating environments with tools like Docker.

Beginner

- Understand the importance of virtual environments for isolating project dependencies.

Basic

- Use virtual environments to manage dependencies in isolation and avoid conflicts between projects.
- Set up requirements.txt files to document and install necessary dependencies.

Productive

- Use tools like Docker to create reproducible and isolated development and production environments.
- Automate environment configuration via scripts or configuration management tools.

Advanced

- Integrate configuration management systems to automate and optimize the creation and maintenance of development and test environments.
- Use advanced tools like Kubernetes to manage large-scale environments in cloud deployments.

Expert

- Lead projects to standardize development environments across entire organizations, using DevOps practices and container management tools.
- Develop strategies to ensure the consistency, security, and efficiency of development and production environments in multi-project and multi-team contexts.

All reproduction prohibited

Domain 5: Code Optimization

Sub-domain 1: Analyzing and profiling code

Includes the use of tools to measure code performance, identify bottlenecks, and apply methodologies for performance optimization.

Beginner

- Identify time-consuming or memory-intensive operations in simple scripts.

Basic

- Use simple tools like timeit to measure the execution time of small pieces of code.
- Analyzes profiling results to identify specific bottlenecks.

Productive

- Optimize code by refactoring inefficient parts identified during profiling.
- Use online profiling tools to monitor application performance in real time.

Advanced

- Integrate profiling and optimization practices into the software development lifecycle to maintain performance throughout production.
- Advise on profiling and optimization best practices within his/her team or organization.

Expert

- Develop customized tools to analyze and improve the performance of complex systems.
- Lead initiatives to integrate performance optimization into standard development practices.

All reproduction prohibited

Sub-domain 2: Improving code efficiency

Includes techniques for reducing code complexity, optimizing loops, and efficient use of resources.

Beginner

- N/A

Basic

- N/A

Productive

- Improve execution speed by using more efficient algorithms for common tasks.
- Apply optimization techniques such as caching and memorization to reduce the cost of repetitive calculations.
- Replace inefficient loops with more efficient high-level operations, such as list comprehensions.

Advanced

- Develop strategies to minimize latency and maximize throughput in mission-critical applications.
- Optimize multi-threaded or multi-process systems to take full advantage of hardware resources.

Expert

- Lead system redesign projects to integrate efficiency and performance best practices.
- Develop frameworks and libraries that improve the efficiency of development operations within the industry.

All reproduction prohibited

Sub-domain 3: Testing and debugging

Covers the writing of unit and integration tests, the use of test frameworks such as ‘pytest’, and advanced debugging strategies.

Beginner

- N/A

Basic

- Use debugging tools to identify and resolve simple errors in code.
- Write unit tests for its simplest functions to ensure that they work as intended.

Productive

- Implement integration tests to check that the different parts of an application work together correctly.
- Integrate automated tests into the development process to ensure that code remains robust in the face of change.

Advanced

- Design test suites to cover critical use scenarios and guarantee the reliability of the application under various conditions.
- Optimize the testing process to reduce execution time while maximizing code coverage.

Expert

- Lead software quality initiatives, integrating advanced testing practices and static code analysis to ensure application robustness.
- May train other developers in debugging techniques and testing methodologies to improve code quality within an organization.

All reproduction prohibited

TOSA

Your skills. Your advantage.

contact@isograd.com

www.tosa.org