PCPP1 CERTIFICATION / WHAT HAS CHANGED

A new chapter in your PCPP1™ certification



Your guide to PCPP1-32-101 vs. PCPP1-32-102

Same professional certification pathway. A stronger focus on modern Python software development.

PCPP1-32-102 builds on the advanced Python skills assessed by PCPP1-32-101 while reshaping the examination around the way professional Python applications are designed, developed, integrated, reviewed, and maintained. The new certification title, PCPP1™ – Certified Professional Python Programmer: Software Development, captures that focus.

The exam still contains 45 questions, but its scope has changed substantially. Discover what carries forward, which skills receive deeper coverage, which topics are newly assessed, and how to adjust your preparation without starting from zero.

SECTIONS	PASSING SCORE
5 → 7	**70% → 72%**
A dedicated sixth section	Updated passing standard

The Big Picture

Develop, Design & Maintain

Both versions belong to the PCPP1 professional certification pathway. The updated assessment takes a more focused approach to professional software development, reflecting the evolving demands of the industry and emerging trends in how software is designed, developed, and maintained. It places greater emphasis on building reusable Python components, processing application data, applying coding standards, integrating services, and making responsible engineering decisions. Importantly, it also recognizes the growing role of AI-assisted programming, validating candidates' ability to integrate AI tools into professional development workflows while ensuring code quality, security, reliability, and human oversight.

KEEP

Your advanced Python foundation

Your advanced Python foundation – Special methods, decorators, abstract classes, metaclasses, exceptions, copying, serialization, PEP 8/257, HTTP, REST, databases, CSV/XML, and logging remain relevant.

ADD & STRENGTHEN

Professional software development practices

Descriptors and dataclasses, exception groups, advanced data modeling, context managers, secure text processing, automated code quality, structured logging, Git workflows, technical risk management, and AI-assisted engineering are explicitly developed.

Already preparing for PCPP1-32-101? Keep the skills you have earned. Use this guide to bridge the gaps rather than repeating everything you have already studied.

The exams at a glance

A side-by-side comparison of the examination experience and credential terms.

EXAM DETAIL	PCPP1-32-101	PCPP1-32-102
Credential name	PCPP1™ – Certified Professional Python Programmer Level 1	PCPP1™ – Certified Professional Python Programmer: Software Development
Questions	45	45
Exam duration	65 minutes	65 minutes
Syllabus structure	5 blocks	7 sections
Passing score	70%	72%
Certification level	Professional	Professional
Prerequisites	No formal prerequisites; PCEP & PCAP knowledge recommended	No formal prerequisites; PCEP & PCAP knowledge recommended
Credential validity	Lifetime for credentials issued under lifetime terms	5 years
Credential awarded	PCPP1™	PCPP1™

What this means

The examination has not grown in length by question count. Instead, its assessment coverage has changed: subjects have been redistributed, several areas are now covered in greater depth, and new engineering competencies have been introduced. The updated passing requirement is 72%.

What does not change

Your PCPP1 certification and previously earned credentials

The release of PCPP1-32-102 does not, by itself, invalidate a PCPP1 certification previously earned through PCPP1-32-101. Credentials retain the validity terms under which they were issued. A lifetime credential remains a lifetime credential; a time-limited credential retains its original expiration date.

The new five-year validity period supports the ongoing relevance of professional credentials as Python, development practices, and software engineering tools evolve. Existing holders are not required to retake the examination merely because the syllabus has changed. However, we strongly recommend considering PCPP1-32-102 to refresh and validate current skills, particularly in software quality, secure development, collaborative engineering, and AI-assisted workflows.

Important dates

PCPP1-32-101 will be retired on November 30, 2027

PCPP1-32-101 will remain available during the transition period and will be officially **retired (sunsetted) on November 30, 2027**. After that date, candidates will no longer be able to take the PCPP1-32-101 exam.

Candidates currently preparing for PCPP1-32-101 can continue following its syllabus and take the exam before the retirement date. Those beginning their certification journey are encouraged to prepare for PCPP1-32-101, which reflects the updated and expanded assessment scope.

The retirement of PCPP1-32-101 does not affect the validity of certifications already earned.

Important for exam day

Check the exact exam code

Make sure your syllabus, learning resources, and preparation products correspond to the exam version you intend to take. If you have already purchased a voucher, confirm its exam version, validity, and redemption terms in your candidate account or with your voucher provider.

Do not assume that the old and new examinations, vouchers, or preparation products are interchangeable. Check your exact exam code, the version associated with your voucher, the applicable delivery arrangements, and the latest testing policies.How the syllabus has changed

The original five-section syllabus has been reorganized into seven sections.

This is not a one-to-one renaming: advanced OOP subjects have been redistributed, networking and data processing have been restructured, and new professional engineering topics have replaced the former dedicated GUI area.

Where the old sections appear in the updated exam

PCPP1-32-101 – OLD EXAM BLOCK	WHAT HAPPENS IN PCPP1-32-102
Block 1: Advanced Object-Oriented Programming 15 questions · 35%	Distributed mainly across **Section 1** (advanced OOP and core syntax), **Section 2** (abstract classes, metaclasses, exception groups), and **Section 3** (object mechanics and data modeling). Existing foundations are retained, with deeper coverage of descriptors, exceptions, and dataclasses.
Block 2: Coding Conventions, Best Practices, and Standardization 7 questions · 12%	Reorganized as **Section 5: Code Development and Maintenance**. PEPs, style, and documentation remain; modern linters, formatters, and code coverage receive explicit attention.
Block 3: GUI Programming 8 questions · 20%	**No dedicated GUI section** in PCPP1-32-102. GUI toolkits, Tkinter widgets, geometry managers, and GUI event handling are no longer explicitly listed among its detailed objectives.
Block 4: Network Programming 8 questions · 18%	HTTP and REST integration continue in **Section 6: Communication with Services and Logging**. JSON and XML are addressed in **Section 3**. Low-level socket programming is no longer explicitly listed in the new detailed objectives.
Block 5: File Processing and Communicating with a Program's Environment 7 questions · 15%	Split mainly between **Section 3** (CSV, JSON, XML, serialization, and data modeling) and Section 6 (SQLite/database operations and logging). `ConfigParser`/INI-file processing is no longer explicitly named.

The old percentages describe the original scoring weights; the new percentages on the next page describe the updated blueprint. Because topics have moved across sections, old and new row totals are not directly interchangeable.

The new seven-section structure

PCPP1-32-102 SECTION	QUESTIONS	WEIGHT
1. Advanced Object-Oriented Programming and Core Syntax	7	15%
2. Abstract Classes, Metaclasses, and Exception Groups	5	12%
3. Data Modeling, Object Mechanics, and Type Systems	9	19%
4. Advanced Text Processing, String Formatting, and Security	5	12%
5. Code Development and Maintenance: Conventions, Best Practices, and Standardization	7	15%
6. Communication with Services and Logging	7	15%
7. Collaborative Engineering and AI-Assisted Development	5	12%
TOTAL	**45**	**100%**

What Stays Familiar

Five foundations carry forward

- **Advanced object-oriented programming**: Special methods, decorators, class/static methods, abstract classes, metaclasses, subclassing built-ins, and object behavior remain essential.
- **Exceptions and object mechanics**: Exception objects and chaining, traceback analysis, identity, copying, and object persistence continue to be assessed.
- **Coding standards**: Python philosophy, PEP 8, PEP 257, style conventions, and documentation remain fundamental to professional code quality.
- **Application integration**: Network concepts, HTTP, REST APIs, SQLite/database operations, and structured data formats remain relevant, although they are organized differently.
- **Logging and maintenance**: Logging levels, handlers, formatters, and maintainable application code remain important; the new exam goes further into structured logging and professional workflow practices.

You already have a substantial head start if you have studied these topics for PCPP1-32-101.

The Most Significant Change

From GUI programming to professional software engineering

PCPP1-32-101 devoted **8 questions and 20%** of its assessment to GUI programming, including Tkinter, widgets, layout and geometry managers, and event-driven interactions. PCPP1-32-102 no longer includes a dedicated GUI section.

Instead, the revised blueprint introduces **Section 7: Collaborative Engineering and AI-Assisted Development** (5 questions, 12%) and expands several other areas: data modeling and type systems; security-conscious text processing; automated quality tooling; structured logging; and advanced exception management.

The change reflects a more explicit focus on the skills involved in maintaining professional Python applications throughout their development lifecycle. **This does not mean GUI programming is obsolete** – only that it is no longer individually identified in the new assessment objectives.

What to add to your study plan

New and significantly expanded coverage

The following topics are new, more explicit, or materially more advanced in the PCPP1-32-102 assessment objectives. Use each *self-check* to decide where you need extra practice.

TOPIC	WHAT TO STUDY & SELF-CHECK
DESCRIPTOR PROTOCOL	Analyze `__get__()`, `__set__()`, `__delete__()`, and `__set_name__()`. Distinguish data from non-data descriptors and trace lookup precedence through instance and class namespaces. **SELF-CHECK:** Can you explain why a descriptor takes precedence over a value stored in an instance's `__dict__`?
ADVANCED DECORATORS	Build parameterized decorators and callable classes; use `functools.wraps`, trace decorator stacking, and preserve metadata. **SELF-CHECK:** Can you predict the order of execution when two decorators wrap one function?

EXCEPTION GROUPS AND DIAGNOSTICS	Work with `ExceptionGroup`, `except*`, partial subgroup handling, `add_note()`, and context suppression with `raise ... from None`. **SELF-CHECK:** Can you determine which exceptions remain unhandled after an `except*` branch processes a group?
TYPE ANNOTATIONS AND DATA MODELING	Apply type annotations and understand that they are not enforced automatically at runtime. Compare type-annotated dataclasses, `NamedTuple`, and Pydantic models for different requirements. **SELF-CHECK:** Can you explain what a type checker might flag that ordinary execution would not?
ADVANCED DATACLASSES	Use `@dataclass`, `__post_init__()`, keyword-only fields, `slots=True`, `frozen=True`, and field metadata. Analyze initialization and mutability trade-offs. **SELF-CHECK:** Can you explain when to use `__post_init__()` and what changes when a dataclass is frozen?
CONTEXT MANAGERS	Implement class-based and generator-based context managers using `__enter__()`, `__exit__()`, and `contextlib.contextmanager`. Trace cleanup during exceptions. **SELF-CHECK:** Can you design a context manager that reliably releases a resource when the managed operation fails?
ADVANCED REGULAR EXPRESSIONS	Use `re.compile()`, `search()`, `match()`, `findall()`, `finditer()`, `sub()`, and `split()`. Apply capturing, named, and non-capturing groups. **SELF-CHECK:** Can you extract named values from structured text and replace selected matches correctly?
ADVANCED FORMATTING AND TEMPLATING	Construct complex f-strings, use the `=` debugging specifier and format specifications, compare formatting approaches, and recognize risks from untrusted dynamic content.

	SELF-CHECK: Can you diagnose an f-string error and explain why interpolation alone does not make untrusted content safe?
IMMUTABLE DATA MODELS	Use `frozenset`, understand immutable mapping concepts such as `frozendict`, and compare frozen dataclasses with custom immutable objects. **SELF-CHECK:** Can you explain the difference between an immutable container and an object that merely restricts attribute reassignment?
AUTOMATED CODE QUALITY	Differentiate linters, formatters, fixers, and type checkers. Understand Flake8, Pylint, Ruff, Black, isort, and appropriate diagnostic suppression. **SELF-CHECK:** Can you decide whether a reported issue should be reformatted, corrected, or explicitly suppressed?
CODE COVERAGE AND DEAD CODE	Read coverage information to identify unexecuted lines and branches; distinguish low coverage from proof that code is unnecessary. **SELF-CHECK:** Can you interpret a coverage result and identify which behavior needs additional testing?
STRUCTURED APPLICATION LOGGING	Analyze logger hierarchies and propagation; use `LogRecord` attributes, handlers, formatters, and structured JSON logging with `structlog`. **SELF-CHECK:** Can you explain why a log message might appear twice and how to attach a request identifier to a log record?
ADVANCED GIT WORKFLOWS	Compare trunk-based development and branching strategies; understand rebase, cherry-pick, tags, release history, and feature flags. **SELF-CHECK:** Can you choose a branching strategy for a team maintaining frequent releases?

TECHNICAL DECISIONS AND RISKS	Track blockers, dependencies, assumptions, and risks. Record technical decisions and define appropriate escalation paths. **SELF-CHECK:** Can you write a brief decision record that explains a technical choice and its trade-offs?
AI-ASSISTED ENGINEERING	Evaluate coding assistants, retrieval-augmented and agent-based tools; use AI to analyze codebases, generate tests, investigate defects, and suggest refactorings. **SELF-CHECK:** Can you plan a multi-step AI-assisted task while retaining independent verification at each decision point?
AI GOVERNANCE AND ACCOUNTABILITY	Assess hallucinated APIs, insecure code, hidden dependencies, licensing, provenance, and confidential-data exposure. Require human review and approval for higher-risk changes. **SELF-CHECK:** Can you identify what should be verified before AI-generated code is committed or deployed?

Retained skills and coverage

These skills remain in the updated syllabus. Some have been reassigned to different sections or developed with more precise assessment requirements.

TOPIC	WHAT TO STUDY & SELF-CHECK
SPECIAL METHODS	Customize comparison, numeric, representation, attribute, and container behavior using Python special methods. **SELF-CHECK:** Can you predict which special method Python invokes for a particular operation?
CLASS AND STATIC METHODS	Define `@classmethod` and `@staticmethod`; understand `cls`, class state, and alternative constructors. **SELF-CHECK:** Can you choose the appropriate method type for a given design requirement?
ABSTRACT CLASSES AND METACLASSES	Implement abstract contracts and explain how metaclasses and `type()` influence class creation and inspection. **SELF-CHECK:** Can you explain why a class with an unimplemented abstract method cannot be instantiated?
SUBCLASSING BUILT-INS	Extend a built-in class without unexpectedly breaking inherited behavior. **SELF-CHECK:** Can you identify when a method should call its superclass implementation?
EXCEPTION CHAINING AND TRACEBACKS	Analyze `__context__`, `__cause__`, `__traceback__`, and exception objects. **SELF-CHECK:** Can you explain how one error caused another in a chained traceback?
OBJECT IDENTITY AND COPYING	Distinguish identity, equality, shallow copy, and deep copy.

	SELF-CHECK: Can you trace which nested objects are shared after a shallow copy?
PICKLE AND SHELVE	Explain persistence and serialization using `pickle` and `shelve`, including appropriate usage boundaries. **SELF-CHECK:** Can you identify why untrusted serialized input requires special care?
CSV, JSON, AND XML	Choose the right interchange format; read/write CSV, serialize JSON, and parse or construct XML. **SELF-CHECK:** Can you explain the trade-offs of using CSV, JSON, or XML for the same dataset?
PYTHON CODING STANDARDS	Apply PEP 1, PEP 8, PEP 20, and PEP 257, including naming, layout, imports, comments, and docstrings. **SELF-CHECK:** Can you identify and correct code that violates a relevant style or documentation convention?
HTTP AND REST APIS	Understand domains, ports, URIs, URLs, HTTP methods and status categories, REST constraints, and CRUD operations. **SELF-CHECK:** Can you select the correct HTTP method for an API update and interpret its response?
DATABASE OPERATIONS	Use `sqlite3`, SQL CRUD statements, cursor methods, transactions, and parameterized queries. **SELF-CHECK:** Can you explain why a parameterized SQL statement is safer than string-building a query from user input?
APPLICATION LOGGING	Use logging levels, custom handlers, formatters, and records. **SELF-CHECK:** Can you choose an appropriate logging level and describe what information should be recorded for troubleshooting?

Topics No Longer Explicitly Assessed

Some topics were named directly in the PCPP1-32-101 syllabus but are not individually specified in the detailed PCPP1-32-102 assessment objectives.

TOPIC	WHAT HAS CHANGED
GUI PROGRAMMING AND TKINTER	The old dedicated GUI section—including `tkinter`, `Tk()`, widgets, event callbacks, geometry managers, and canvas operations—has no corresponding dedicated section in the new blueprint.
LOW-LEVEL SOCKET PROGRAMMING	Creating sockets and using `send()`/`recv()` are no longer individually listed. Network concepts, HTTP, and REST remain part of the new syllabus.
CONFIGURATION FILES WITH CONFIGPARSER	The old explicit coverage of `.ini` files, `ConfigParser`, and configuration-value interpolation is no longer individually listed.
CERTAIN OOP FOUNDATIONS AS STANDALONE OBJECTIVES	Basic class terminology, general inheritance/composition comparisons, duck typing, and `*args`/`**kwargs` no longer appear as separate advanced assessment objectives. These remain useful prerequisite knowledge.
GUI EVENT-DRIVEN DESIGN	Event binding and UI-specific event loops are no longer explicit. This does not mean callbacks or event-driven programming have disappeared from Python or the broader professional landscape.

Where Your Study Priorities Should Shift

Some topics were named directly in the PCPP1-32-101 syllabus but are not individually specified in the detailed PCPP1-32-102 assessment objectives.

NEW SECTION	WEIGHT	STUDY EMPHASIS
3. Data Modeling, Object Mechanics, and Type Systems	**19%**	Highest-weighted area: CSV/JSON/XML, copying, persistence, annotations, dataclasses, and context managers.
1. Advanced Object-Oriented Programming and Core Syntax	**15%**	Special methods, decorators, class/static methods, descriptors, and subclassing built-ins.
5. Code Development and Maintenance	**15%**	PEP conventions, docstrings, automated linting/formatting, and coverage analysis.
6. Communication with Services and Logging	**15%**	HTTP, REST, SQLite/database programming, and structured logging.
2. Abstract Classes, Metaclasses, and Exception Groups	**12%**	Abstract classes, class creation, exception chaining, and `ExceptionGroup`.
4. Advanced Text Processing, String Formatting, and Security	**12%**	Regular expressions, immutable data, formatting, and secure templating.
7. Collaborative Engineering and AI-Assisted Development	**12%**	Git workflow decisions, technical risk, AI tooling, verification, and governance.

Which exam version should you choose?

Both exam versions belong to the PCPP1 professional certification pathway. The best choice depends on your readiness and the availability of the version for which you have prepared.

YOUR SITUATION	OUR RECOMMENDATION
I'm beginning my professional-level Python studies.	Prepare for PCPP1-32-102. It reflects the updated Software Development scope and current professional engineering practices.
I've already been studying for PCPP1-32-101.	Keep your advanced Python foundation. If you are nearly exam-ready and the old version is still available, confirm your schedule and voucher terms. Otherwise, use the tables in this guide to transition efficiently.
I've completed a GUI-focused PCPP1 course.	Retain the skills as useful professional knowledge, but do not rely on that course alone. Study the new objectives for dataclasses, descriptors, exception groups, code quality, security, Git, and AI-assisted development.
I've purchased a PCPP1-32-101 voucher.	Confirm the voucher's version, permitted delivery channel, expiration, and exam availability before scheduling. Do not assume automatic transfer to PCPP1-32-102.
I'm already PCPP1 certified.	Your existing certification retains its original validity. Retaking the exam is not required, but **we strongly recommend PCPP1-32-102** as an opportunity to validate current Software Development competencies and demonstrate continuous learning.

Our recommendation: If you are beginning your professional-level certification journey now, choose PCPP1-32-102. If you have already invested substantial time in PCPP1-32-101 preparation, assess your readiness and examination timeline before deciding whether to transition.

Reference Syllabi

- PCPP1-32-101 Examination Syllabus: https://pythoninstitute.org/pcpp1-32-101-exam-syllabus
- PCPP1-32-102 Examination Syllabus: https://pythoninstitute.org/pcpp1-exam-syllabus

This guide is a version-transition resource. The applicable official exam syllabus, testing policies, and voucher terms govern assessment and scheduling.