CERTLABZ CERTIFICATION BOARD

Certified Python Application Developer (CPAD)

Ten Sample Questions

These ten questions are written at the difficulty of the live exam. Six carry a figure, a data table, a configuration or a code listing; four are plain scenario questions, as on the live paper. Every one is a scenario a working Python application developer would meet, and none of them appears in the live question bank.

The live exam is multistage adaptive: how you score on one stage selects the difficulty of the next, so no two candidates sit the same paper. It is a performance-based exam: each form can carry up to 5 to 12 performance-based questions alongside the multiple-choice questions, which means you build, sequence or complete something rather than only choosing from a list. Answers and full explanations follow each question here; the live exam does not disclose item-level answers.

Question 1

Domain 1: Python Language Core

DATA

Emma Wilson matches each Python expression to what it evaluates.

Expression	Meaning
a is b	(to choose)
a == b	(to choose)
len(seq)	(to choose)
seq[-1]	(to choose)

Table 1.1 Expressions and their meanings.

Which set of meanings is correct?

A. Value equality, identity, last item and item count, so a is b compares values while a == b tests whether two names point at one object.

B. Identity, value equality, item count and last item, matching each expression to what it actually evaluates for the given sequence here.

C. Item count, last item, identity and value equality, so len returns the final element and indexing with minus one returns the length here.

D. Last item, item count, value equality and identity, so seq[-1] returns how many items the sequence holds rather than the final element.

Correct answer: B
The is operator tests identity, == tests value equality, len returns the item count, and seq[-1] returns the last item. Only option B pairs each expression with what it actually evaluates, in the right order.

Question 2

Domain 2: Data Structures, OOP and Modules

CONFIG

James Carter reviews a function that keeps old values between separate calls.

```
def add_item(x, bucket=[]):
    bucket.append(x)
    return bucket
# a later call still sees the old items
```

Listing 2.1 The current function.

What is the soundest fix?

A. Call the function only once per program run, since a shared default list stays safe as long as no second call is ever made to it at all.

B. Raise the recursion limit before each call, on the belief that the shared list is caused by Python running out of stack depth partway in.

C. Default the argument to None and build a fresh list inside the function when none is passed, so each call begins with its own empty list.

D. Make the list a module global and clear it by hand after each call, trusting every caller to remember to reset the shared state each time.

Correct answer: C
A default argument is evaluated once at definition, so a mutable default list is shared across calls. Defaulting to None and creating a new list inside the function gives each call its own list. Calling once, changing the recursion limit, or relying on manual resets does not fix the shared default.

Question 3

Domain 3: Standard Library and Ecosystem

DATA

Olivia Davis matches each standard library module to its job.

Module	Job
json	(to choose)
pathlib	(to choose)
datetime	(to choose)
collections	(to choose)

Table 3.1 Modules and their jobs.

Which set of jobs is correct?

A. Encode and decode JSON, work with filesystem paths, handle dates and times, and supply extra container types, pairing each module here.

B. Handle dates and times, encode JSON, supply container types and work with paths, so json is what builds filesystem paths across systems.

C. Work with paths, supply container types, encode JSON and handle dates, so datetime is what parses and writes the JSON text in this set.

D. Supply container types, handle dates, work with paths and encode JSON, so collections is what reads and writes the JSON documents here.

Correct answer: A
The json module encodes and decodes JSON, pathlib works with filesystem paths, datetime handles dates and times, and collections supplies extra container types. Only option A pairs each module with its real job.

Question 4

Domain 4: Building Applications

CONFIG

Liam Anderson reviews how an application loads its database password.

```
PASSWORD = "hunter2"  # hard-coded
conn = connect(pw=PASSWORD)
committed into the shared repo
same value in every environment
```

Listing 4.1 The current setup.

What is the soundest change?

A. Leave the password in the source but add a comment asking teammates not to read that one line when they open the module during a review.

B. Rename the constant to something vague so the value stays in the repo yet is harder for a casual reader to recognise as a real secret now.

C. Encode the password with base64 in the source, on the belief that an encoded string held in version control can no longer be read at all.

D. Read the password from an environment variable or secret store at run time and keep it out of source, with a per-environment value.

Correct answer: D
Secrets do not belong in source that is committed to a shared repository. Reading the password from an environment variable or secret store at run time keeps it out of version control and lets each environment use its own value. Comments, vague names, and base64 leave the readable secret in the repo.

Question 5

Domain 5: Testing, Quality and Packaging

DATA

Ava Thompson matches each testing problem to the practice that fixes it.

Problem	Practice
Tests hit the real payment API	(to choose)
One test breaks many others	(to choose)
Only the happy path is covered	(to choose)
Coverage is never measured	(to choose)

Table 5.1 Problems and practices.

Which set of practices is correct?

A. Add edge-case tests, run a coverage tool, mock the payment API and isolate tests, so mocking is what measures how much code the tests run.

B. Mock the payment API, isolate each test, add edge-case tests and run a coverage tool, matching each problem to the practice that fixes it.

C. Isolate each test, add edge cases, run coverage and mock the API, so isolating a test is what stops it calling the real payment service now.

D. Run a coverage tool, mock the API, isolate tests and add edge cases, so coverage tooling is what keeps a failing test from breaking others.

Correct answer: B
Mocking the payment API stops tests calling a real service, isolating each test keeps one failure from cascading, edge-case tests cover more than the happy path, and a coverage tool measures what is exercised. Only option B pairs each problem with the practice that resolves it.

Question 6

Domain 6: Performance, Concurrency and Deployment

CONFIG

An engineer adds many threads to a CPU-bound task and sees no speedup at all.

```
task     : pure CPU number crunch
approach : 8 threads, one process
result   : no speedup, GIL bound
goal     : use all CPU cores
```

Listing 6.1 The current approach.

What is the soundest change?

A. Use multiple processes for the CPU-bound work so the tasks run on separate cores at once instead of contending for one interpreter lock.

B. Add many more threads to the single process, on the belief that enough threads will eventually saturate every core despite the shared lock.

C. Raise the thread priority of every worker and keep one process, trusting the operating system to spread the locked threads across the cores.

D. Lower the thread count to one and accept single-core speed, since concurrency of any kind cannot help a task that is bound by the CPU here.

Correct answer: A
A single interpreter lock stops threads running Python bytecode on several cores at once, so threads do not speed up CPU-bound work. Multiple processes run on separate cores in parallel and use the whole machine. More threads, higher priority, or a single thread do not reach the other cores.

Question 7 Domain 1: Python Language Core

A script wraps a large block in a broad "except Exception: pass" and now hides real bugs. What is the soundest practice?

A. Keep catching every exception and passing, because a program that never shows an error to a user always looks more polished in the end.

B. Wrap even more code in the same broad handler, on the belief that the more errors are silenced the more reliable the whole script becomes.

C. Catch only the specific exceptions you expect and handle each, letting an unexpected error surface so a real bug is seen and fixed early.

D. Replace it with a bare except that also swallows keyboard interrupts, so the running script can never be stopped once it has been started.

Correct answer: C
A broad except that passes silences real errors and hides bugs. Catching only the specific exceptions you expect, and letting the rest surface, means unexpected failures are seen and fixed. Silencing everything, or a bare except that swallows interrupts, makes the program harder to debug and to stop.

Question 8 Domain 5: Testing, Quality and Packaging

A team argues over code style in every review and formatting churn fills the diffs. What practice best fixes this?

A. Ask each reviewer to reformat the other people code by hand to their own taste, so every file slowly drifts toward one preferred style.

B. Adopt an automatic formatter and a linter in the pipeline, so style is applied the same way and reviews focus on behavior over layout.

C. Ban all discussion of formatting and merge whatever arrives, since ignoring style entirely is the only way to keep reviews moving fast.

D. Rewrite the whole codebase in one large commit each week to a new style, trusting that frequent sweeping reformats will settle the debate.

Correct answer: B
An automatic formatter and a linter in the pipeline apply style consistently, so diffs stay small and reviews focus on behavior rather than layout. Manual reformatting, banning the topic, or weekly sweeping rewrites either continue the churn or leave style unaddressed.

Question 9 Domain 6: Performance, Concurrency and Deployment

An app works on one machine but breaks on another because each installs different library versions. What is required?

A. Tell every developer to install whatever versions they like, since matching library versions across machines has no effect on the code.

B. Copy the working machine site-packages folder around by hand, trusting that shipping installed files between systems will always behave.

C. Always install the very newest version of every dependency everywhere, on the belief that the latest release can never introduce a change.

D. Pin dependency versions in a lock file and install from it everywhere, so every machine builds the same set of library versions.

Correct answer: D
Unpinned dependencies let each machine resolve different versions, which is why the app breaks in one place and not another. Pinning versions in a lock file and installing from it makes every environment build the same set. Free-for-all installs, hand-copied folders, or always-latest do not give reproducible builds.

Question 10 Domain 2: Data Structures, OOP and Modules

A program checks membership millions of times against a large list and runs slowly. What is the soundest change?

A. Store the items in a set and test membership against it, so each lookup is roughly constant time instead of scanning the whole list.

B. Sort the list once before the program starts and keep scanning it from the front, trusting that ordering alone makes each linear scan fast.

C. Copy the list into a second list of the same items, on the belief that checking two lists in turn is quicker than checking one longer list.

D. Convert the list to a string and search for each item as text, since matching inside one long string is always faster than any list scan.

Correct answer: A
Membership testing against a list scans it item by item, which is slow when repeated millions of times. Storing the items in a set makes each lookup roughly constant time through hashing. Sorting, duplicating the list, or converting to a string does not give the fast membership test a set provides.

Published by the CertLabz Certification Board, a division of CertLabz LLC. Companion to objectives document CPAD-001. https://certlabz.com/certifications/certified-python-application-developer.html