Python Functions, Generators & Decorators — Q&A; (Set 14)

Q1. Is += only for show? Can it be faster?

Not just for show. For mutable types (like lists), += calls __iadd__, modifying in place and avoiding allocation. For immutable types (str, tuple), += creates a new object, though CPython may optimize in some cases.

Q2. Fewest statements to replace a, b = a + b, a in most languages

Typically 3 statements with a temp:
t = a + b;
b = a;
a = t;

Q3. Most effective way to set a list of 100 integers to 0

xs = [0] * 100

Q4. Initialize length-99 list repeating 1,2,3

lst = ([1,2,3] * 33)[:99]
Alternative: [(i % 3) + 1 for i in range(99)]

Q5. Print a multidimensional list efficiently in IDLE

Use pprint for readability:
from pprint import pprint
pprint(matrix, width=100, compact=True)

Q6. List comprehension with a string?

Yes. Iterate characters, optionally rejoin:
s = 'A1b2C3'
letters_upper = [c.upper() for c in s if c.isalpha()]
digits = ''.join([c for c in s if c.isdigit()])

Q7. Get help for a user module (CLI & IDLE)

Command line: python -m pydoc yourmodule or python yourscript.py --help
In IDLE: import yourmodule; help(yourmodule) or enter help() interactively.

Q8. First-class functions in Python vs C/C++

Python functions can be assigned, passed, returned, nested, capture closures, decorated, and even carry attributes. C/C++ has function pointers and lambdas but not the same dynamic, object-like treatment.

Q9. Wrapper vs wrapped feature vs decorator

- Wrapped feature: the original function/class.
- Decorator: a callable that takes and returns a function/class.
- Wrapper: the new callable that adds behavior and calls the original.

Q10. What does a generator function return?

A generator iterator—an object implementing __iter__ and __next__, not a list.

Q11. Single change to make a function a generator

Include yield (or yield from) in the function body.

Q12. One benefit of generators

Lazy, memory-efficient iteration: produce values on demand. Useful for large/streaming/infinite data, pipelines, and stateful iteration.