Format Python Code Blocks - Original

Slides 3, 5, and 7 contain unstyled Python code.

Utilities: Fibonacci and Point

```
from __future__ import annotations
from dataclasses import dataclass

def fib(n: int) -> int:
    """Return nth Fibonacci number using iterative DP."""
    if n < 2:
        return n
    a, b = 0, 1
    for _ in range(2, n + 1):
        a, b = b, a + b
    return b

@dataclass
class Point:
    x: float
    y: float

if __name__ == "__main__":
    p = Point(3.0, 4.0)
    print(f"fib(10) = {fib(10)}; |p|^2 = {p.x**2 + p.y**2}")
```

Async I/O example

```
import asyncio
import time

async def fetch(n: int) -> str:
    await asyncio.sleep(0.1 * n)
    return f"done {n}"

async def main():
    t0 = time.perf_counter()
    results = await asyncio.gather(*(fetch(i) for i in range(5)))
    dt = time.perf_counter() - t0
    for r in results:
        print(r)
    print(f"{len(results)} tasks in {dt:.3f}s")

if __name__ == "__main__":
    asyncio.run(main())
```

Decorators and context manager

```
from contextlib import contextmanager

def deco(fn):
    def wrapper(*args, **kwargs):
        print(f"calling {fn.__name__}")
        return fn(*args, **kwargs)
    return wrapper

@deco
def square(x: int) -> int:
    return x * x

@contextmanager
def timer():
    import time
    t0 = time.perf_counter()
    try:
        yield
    finally:
        dt = time.perf_counter() - t0
        print(f"elapsed: {dt:.6f}s")
```