Python Pro Tips

Modern Python Syntax

- **f-strings (3.6+)**: Most efficient way to format strings. `print(f"Value: {val:.2f}")`
- **Walrus Operator (3.8+)**: Assignment within an expression. `if (n := len(data)) > 10: print(f"Too long: {n}")`
- **Type Hinting (3.5+)**: Improve code clarity and catch errors early with `mypy`. `def greet(name: str) -> str: return f"Hi {name}"`
- **Structural Pattern Matching (3.10+)**: Modern `switch-case`.

```
match status:
    case 200: return "OK"
    case 404: return "Not Found"
    case _: return "Error"
```

List & Dictionary Power Moves

- **Comprehensions**: Concise creation of collections.
  - `[x**2 for x in range(10) if x % 2 == 0]`
  - `{x: x**2 for x in range(5)}`
- **Unpacking**:
  - `a, b, *rest = [1, 2, 3, 4, 5]`
  - `combined = {**dict1, **dict2}`
- **Defaultdict**: Avoid `KeyError` effortlessly. `from collections import defaultdict; d = defaultdict(list)`

Built-in Functionality

- `enumerate()`: Get both index and value in a loop. `for i, val in enumerate(my_list): ...`
- `zip()`: Iterate over multiple sequences in parallel. `for name, age in zip(names, ages): ...`
- `any()` & `all()`: Check boolean conditions across an iterable.
- `itertools`: Advanced iteration tools (product, permutations, cycle).
- `pathlib`: Modern, object-oriented file path manipulation.

Performance & Memory

- **Generators**: Use `()` instead of `[]` to save memory for large datasets. `sum(x*x for x in range(1_000_000))`
- `__slots__`: Reduce memory footprint of class instances.
- **cProfile**: Built-in profiler to find bottlenecks.
- **join()**: Use `''.join(list_of_strings)` instead of `+=` for string concatenation.

Clean Code & Decorators

- **Decorators**: Wrap functions to add functionality.

```
@timer
def heavy_task(): ...
```

- **Context Managers**: Handle resources safely with `with`.
- **Property**: Use `@property` for getters and setters without breaking the API.
- **Dataclasses (3.7+)**: Automatically generate `__init__`, `__repr__`, etc.

```
from dataclasses import dataclass
@dataclass
class User: id: int; name: str
```

Testing & Debugging

- `pytest`: The standard for modern Python testing.
- `breakpoint()`: Fast way to trigger a debugger (PDB).
- `logging`: Use the `logging` module instead of `print` for production tracing.

Pro Tips

Zen of Python

Run `import this` in a Python shell to read the guiding principles of the language. “Simple is better than complex.”

Virtual Environments

Always use a virtual environment (`venv`, `poetry`, `uv`, `pixi`) to isolate project dependencies.

PIP Compile

Use `pip-tools` or `uv` to pin your dependencies into a lock file for reproducible environments.