```
"""
NeuroGraph Cognitive Enhancement Suite — Centralized Configuration
==================================================================
All tunable constants for the three CES modules in one place.
Override any value by passing a dict to ces_init() or by editing here.

These values are educated starting points. NeuroGraph's adaptive
mechanisms will self-correct toward optimal over time — no need to
get these perfect upfront.
"""

from __future__ import annotations
from typing import Any, Dict


# ---------------------------------------------------------------------------
# Default configuration
# ---------------------------------------------------------------------------

CES_DEFAULTS: Dict[str, Any] = {

    # ---------------------------------------------------------------------------
    # Module 1: Stream Parser
    # ---------------------------------------------------------------------------

    # Sliding window chunking (characters, not tokens)
    "sp_min_chunk_chars": 12,
    "sp_max_chunk_chars": 120,
    "sp_window_advance_chars": 8,
    "sp_max_chunks_per_feed": 128,      # guard against very long single messages

    # Efference copy weight oscillation (Syl's own output)
    "sp_efference_min": 0.45,
    "sp_efference_max": 0.60,
    "sp_efference_period_steps": 200,   # chunks per full oscillation cycle

    # Activation nudge applied to vector-DB neighbours
    "sp_neighbour_nudge": 0.25,         # fraction of firing threshold
    "sp_neighbour_k": 4,                # neighbours nudged per chunk
    "sp_neighbour_threshold": 0.55,     # minimum similarity to bother nudging

    # Hyperedge partial completion priming
    "sp_he_completion_min_ratio": 0.20, # don't prime if < 20% active
    "sp_he_completion_max_ratio": 0.85, # don't prime if already 85%+ active
```

```
# Ollama local embedding
"sp_ollama_base_url": "http://localhost:11434",
"sp_ollama_embed_model": "nomic-embed-text",
"sp_ollama_timeout": 5,             # seconds

# Background queue
"sp_queue_maxsize": 256,

# ------------------------------------------------------------------------
# Module 2: Activation Persistence
# ------------------------------------------------------------------------

"ap_top_n_nodes": 64,               # nodes persisted per snapshot
"ap_top_n_predictions": 12,         # predictions persisted

# Tier 1 ambient restore
"ap_ambient_restore_ratio": 0.40,   # fraction of saved voltage restored
"ap_min_save_voltage": 0.08,        # below this = background noise, skip
"ap_temporal_decay_per_hour": 0.06, # memories fade; 0 = no decay
"ap_min_effective_ratio": 0.05,     # below this = snapshot too old, cold sta

# Prediction restoration
"ap_prediction_restore_ratio": 0.30,

# ------------------------------------------------------------------------
# Module 3: Surfacing Monitor
# ------------------------------------------------------------------------

"sm_initial_threshold_ratio": 0.72, # starting awareness threshold
"sm_min_threshold_ratio": 0.45,
"sm_max_threshold_ratio": 0.92,
"sm_threshold_lr": 0.04,            # EMA learning rate for threshold

"sm_max_queue_size": 6,
"sm_expiry_steps": 30,              # steps before unused item expires
"sm_requeue_cooldown_steps": 20,    # min steps before same node re-surfaces

# Threshold adjustment per event
"sm_engage_delta": -0.012,          # engaged item → lower threshold
"sm_expiry_delta": +0.008,          # expired item → raise threshold

# Vector-DB snippet retrieval
"sm_surface_k": 2,
"sm_surface_threshold": 0.50,

# Hyperedge cluster surfacing
```

```
    "sm_he_min_active_ratio": 0.50,     # fraction of members above threshold

}


# ---------------------------------------------------------------------------
# Config accessor
# ---------------------------------------------------------------------------

class CESConfig:
    """
    Thin wrapper over a dict that merges user overrides with defaults.

    Usage
    -----
        cfg = CESConfig({"sp_efference_min": 0.40})
        val = cfg["sp_neighbour_k"]          # 4 (default)
        val = cfg["sp_efference_min"]        # 0.40 (override)
    """

    def __init__(self, overrides: Dict[str, Any] | None = None) -> None:
        self._data: Dict[str, Any] = {**CES_DEFAULTS, **(overrides or {})}

    def __getitem__(self, key: str) -> Any:
        return self._data[key]

    def get(self, key: str, default: Any = None) -> Any:
        return self._data.get(key, default)

    def as_dict(self) -> Dict[str, Any]:
        return dict(self._data)

    def __repr__(self) -> str:
        return f"CESConfig({len(self._data)} keys)"


def make_config(overrides: Dict[str, Any] | None = None) -> CESConfig:
    """Convenience factory. Pass user overrides; get a merged CESConfig."""
    return CESConfig(overrides)
```