Classical Virtues in Python

SEPTEMBER 2025

11 min read • 2,454 words

Themes: Consciousness Programming Spiritual Contemplative

Growing up in a Calvinist household and attending a fundamental Southern Baptist Christian school through middle school gave me deep exposure to virtue-based thinking. While my relationship with institutional religion has evolved, the emphasis on character formation left a lasting impression. The classical seven virtues—four cardinal virtues from ancient philosophy and three theological virtues from Christian tradition—provide a framework for human flourishing that transcends any single religious system.

As I explored in Ram Dass Teachings in Python and Vedic Principles in Python, wisdom traditions across cultures identify similar patterns for character development. The classical virtues represent Western civilization's distillation of these patterns into systematic form—practical algorithms for living well.

Our increasingly disconnected digital age desperately needs these foundational virtues. Below are the seven classical virtues translated into Python—not to reduce the profound to the technical, but to reveal timeless patterns through contemporary metaphors.

Prudence

Prudentia - The Master Virtue

Prudence stands as the "master virtue" because it governs how all other virtues should be applied. It's not mere intelligence or book learning, but practical wisdom—the ability to discern the right course of action in specific, often complex circumstances. Without prudence, courage becomes recklessness, justice becomes rigid legalism, and temperance becomes joyless restriction.

Thomas Aquinas called prudence "auriga virtutum" (the charioteer of the virtues) because it directs when, how, and to what degree other virtues should be exercised. A prudent person doesn't just know abstract moral principles; they understand how to apply those principles wisely in the messy realities of daily life.

```python
from dataclasses import dataclass
from typing import Any, List, Optional

@dataclass
class Prudence:
    """Practical wisdom - the charioteer of the virtues."""
    experience: List[Any]
    reflection: callable
    good_judgment: bool = True

    def deliberate(self, situation: Any) -> Any:
        """Consider all relevant factors before acting."""

        # Gather information
        facts = self.examine_circumstances(situation)
        consequences = self.predict_outcomes(situation)
        principles = self.apply_universal_truths(situation)

        # Synthesize wisdom
        past_learning = self.consult_experience(self.experience)
        present_context = self.assess_current_reality(situation)
        future_impact = self.consider_long_term_effects(consequences)

        return self.choose_wisely(facts, principles, past_learning, present_context, fu

    def know_thyself(self) -> dict:
        """Self-knowledge as foundation of wisdom."""
        return {
            'strengths': self.honest_assessment_of_capabilities(),
            'weaknesses': self.acknowledge_limitations(),
            'biases': self.recognize_cognitive_distortions(),
            'values': self.clarify_core_principles()
        }

    def discern_timing(self, action: Any) -> bool:
        """Knowing when to act and when to wait."""
        if self.conditions_are_right(action) and self.resources_available(action):
            return True
        return False
```

Prudence serves as the operating system for ethical decision-making—without it, other virtues become rigid rules rather than flexible wisdom

> Thomas Aquinas called prudence "auriga virtutum" (the charioteer of the virtues) because it directs how all other virtues should be applied in specific situations.

.

Justice

Justitia - Giving Each Their Due

Justice forms the foundation of all healthy relationships and functional societies. It operates on the simple but profound principle of "giving each their due"—rendering to every person what they are owed by right, merit, or need. This isn't mere rule-following, but the active cultivation of fairness that considers both individual dignity and communal harmony.

Aristotle identified three types of justice: distributive (fair allocation of resources and responsibilities), corrective (addressing wrongs and restoring balance), and commutative (ensuring fair exchanges between equals). Modern legal systems still use these categories because they capture fundamental patterns of how humans organize themselves ethically.

```python
@dataclass
class Justice:
    """Giving to each what they are owed."""

    def render_what_is_owed(self, person: Any, context: Any) -> Any:
        """The fundamental algorithm of justice."""

        debt = self.calculate_what_is_owed(person, context)

        if debt.type == "distributive":
            # Fair allocation of resources/burdens
            return self.distribute_proportionally(debt)

        elif debt.type == "corrective":
            # Restoring balance after wrongdoing
            return self.restore_equilibrium(debt)

        elif debt.type == "commutative":
            # Fair exchange between equals
            return self.ensure_equal_exchange(debt)

    def uphold_rights(self, community: List[Any]) -> None:
        """Protect legitimate claims of all members."""

        for person in community:
            basic_rights = person.get_human_dignity_rights()
            earned_rights = person.get_merit_based_rights()

            self.protect_rights(basic_rights)
            self.honor_earned_rights(earned_rights)

    def establish_rule_of_law(self, society: Any) -> dict:
        """Systems that apply principles consistently."""

        return {
            'equal_treatment': self.apply_same_standards_to_all(),
            'due_process': self.ensure_fair_procedures(),
            'proportional_response': self.match_consequences_to_actions(),
            'impartial_judgment': self.remove_personal_bias()
        }
```

Justice creates the framework within which all other virtues can flourish—without fair systems, individual virtue becomes meaningless in social contexts

> Aristotle distinguished between distributive justice (fair allocation), corrective justice (addressing wrongs), and commutative justice (fair exchange)—categories still used in legal philosophy today.

.

Fortitude

Fortitudo - Strength in Adversity

Fortitude bridges the critical gap between knowing what's right and actually doing it. It's the virtue that enables action when action is difficult, dangerous, or costly. Without fortitude, moral knowledge remains merely theoretical—we may know what we should do, but lack the strength to do it when it matters most.

C.S. Lewis observed that courage is not simply one virtue among others, but "the form of every virtue at the testing point." You cannot consistently practice prudence, justice, temperance, faith, hope, or love without the courage to act on these virtues when they're challenged by fear, social pressure, or personal cost.

```python
@dataclass
class Fortitude:
    """Courage to do right despite opposition."""

    def face_adversity(self, challenge: Any) -> Any:
        """Stand firm when doing right is difficult."""

        if challenge.threatens_safety():
            physical_courage = self.act_despite_bodily_harm()
            return physical_courage.proceed_with_caution()

        elif challenge.threatens_reputation():
            moral_courage = self.act_despite_social_consequences()
            return moral_courage.speak_truth_anyway()

        elif challenge.threatens_comfort():
            spiritual_courage = self.act_despite_personal_cost()
            return spiritual_courage.sacrifice_for_greater_good()

    def endure_suffering(self, hardship: Any) -> str:
        """Patience under trial without losing hope."""

        while hardship.persists():
            # Maintain dignity and purpose
            self.refuse_to_compromise_principles()
            self.find_meaning_in_suffering()
            self.support_others_in_similar_trials()

        return "Character refined through fire"

    def take_calculated_risks(self, opportunity: Any) -> bool:
        """Courage is not absence of fear but right action despite it."""

        if self.prudence.deems_worthy(opportunity):
            fear_level = self.assess_legitimate_concerns(opportunity)
            potential_good = self.evaluate_benefits(opportunity)

            if potential_good > fear_level:
                return self.act_courageously()
        return self.wait_for_better_opportunity()
```

Fortitude bridges the gap between knowing what's right and actually doing it—courage transforms moral knowledge into moral action

> C.S. Lewis wrote that courage is not simply one of the virtues, but the form of every virtue at the testing point—you cannot practice any virtue consistently without courage.

.

Temperance

Temperantia - Moderation and Self-Control

Temperance is perhaps the most misunderstood virtue, often confused with joyless asceticism or rigid self-denial. True temperance is actually about freedom—the liberation that comes from being master of your desires rather than their slave. It enables you to enjoy legitimate pleasures without being controlled by them.

The temperate person doesn't avoid pleasure but approaches it wisely. They can enjoy food without gluttony, success without pride, rest without sloth, and material goods without greed. This self-mastery creates space for higher pursuits by ensuring that lower appetites don't dominate consciousness and decision-making.

```python
@dataclass
class Temperance:
    """Self-mastery over appetites and desires."""

    def moderate_pleasures(self, desire: Any) -> Any:
        """Enjoy good things without being controlled by them."""

        if desire.is_legitimate_good():
            appropriate_amount = self.calculate_proper_portion(desire)
            appropriate_timing = self.determine_right_moment(desire)
            appropriate_context = self.ensure_suitable_setting(desire)

            if all([appropriate_amount, appropriate_timing, appropriate_context]):
                return self.enjoy_responsibly(desire)

        return self.abstain_wisely(desire)

    def develop_self_control(self, impulse: Any) -> str:
        """Freedom through discipline."""

        # First, understand the impulse
        root_need = self.identify_underlying_need(impulse)
        trigger_pattern = self.recognize_activation_conditions(impulse)

        # Then, create healthy response
        if root_need.is_legitimate():
            healthy_fulfillment = self.find_virtuous_satisfaction(root_need)
            return self.redirect_impulse(impulse, healthy_fulfillment)
        else:
            return self.transform_disordered_desire(impulse)

    def practice_delayed_gratification(self, temptation: Any) -> Any:
        """Present discipline for future freedom."""

        immediate_pleasure = temptation.short_term_benefit
        future_consequence = temptation.long_term_cost

        if future_consequence > immediate_pleasure:
            return self.choose_long_term_good()
        else:
            return self.enjoy_harmless_pleasure()
```

Temperance creates space for higher pursuits by ensuring lower appetites don't dominate consciousness—freedom from compulsion enables freedom for purpose

> The Stoics understood temperance not as joyless asceticism but as the skill of enjoying pleasure without being enslaved by it—mastery rather than suppression.

.

Faith

Fides - Confidence in Ultimate Reality

Faith is often misunderstood as believing things without evidence, but classical virtue ethics sees it differently. Faith involves trust in realities that transcend empirical verification while still being grounded in reasonable evidence. It's the confidence that allows us to make commitments and live according to principles that can't be fully proven through immediate experience.

Faith doesn't eliminate reason but transcends its limitations. It enables us to act on our deepest convictions about reality's nature, meaning, and purpose even when these convictions extend beyond what can be definitively proven. This trust becomes the foundation for hope and love by establishing confidence in ultimate realities that make both meaningful.

```
@dataclass
class Faith:
    """Trust in realities beyond empirical verification."""

    def trust_beyond_proof(self, uncertain_reality: Any) -> bool:
        """Confidence based on sufficient but not conclusive evidence."""

        available_evidence = uncertain_reality.gather_supporting_data()
        rational_analysis = uncertain_reality.logical_coherence()
        experiential_confirmation = uncertain_reality.lived_verification()
        authoritative_testimony = uncertain_reality.credible_witnesses()

        total_support = sum([available_evidence, rational_analysis,
                           experiential_confirmation, authoritative_testimony])

        if total_support >= self.minimum_threshold_for_reasonable_belief():
            return self.commit_despite_remaining_uncertainty()
        else:
            return self.maintain_healthy_skepticism()

    def live_according_to_unseen_realities(self, daily_choices: List[Any]) -> List[Any]
        """Let invisible truths shape visible actions."""

        transformed_choices = []
        for choice in daily_choices:
            if self.eternal_perspective_changes_calculation(choice):
                better_choice = self.choose_based_on_ultimate_reality(choice)
                transformed_choices.append(better_choice)
            else:
                transformed_choices.append(choice)  # Some choices are morally neutral

        return transformed_choices

    def maintain_trust_through_difficulty(self, trial: Any) -> str:
        """Faith tested becomes faith strengthened."""

        if trial.challenges_belief():
            deeper_understanding = self.wrestle_with_questions(trial)
            refined_trust = self.emerge_with_mature_faith(deeper_understanding)
            return refined_trust
        else:
            return self.continue_trusting()
```

Faith doesn't eliminate reason but transcends its limitations—providing foundation for commitments that evidence alone cannot fully justify

> Aquinas argued that faith and reason are complementary rather than contradictory—faith accepts truths beyond reason's reach but not against reason's findings.

.

Hope

Spes - Trust in Ultimate Good

Hope differs fundamentally from mere optimism or wishful thinking. While optimism is often temperamental—a sunny disposition that expects good outcomes—hope is metaphysical. It's grounded in convictions about reality's ultimate structure and direction, enabling confident action even in the face of temporary defeats or long delays.

Hope prevents despair by maintaining vision of ultimate good even when circumstances seem discouraging. It sustains effort toward goals that may not be achieved in this lifetime, trusting that worthwhile endeavors serve purposes larger than immediate success. This virtue becomes especially crucial when working for justice, truth, or other goods that require generational effort.

```
@dataclass
class Hope:
    """Confident expectation of ultimate good."""

    def maintain_vision(self, setback: Any) -> Any:
        """Keep sight of the goal despite temporary defeats."""

        if setback.seems_permanent():
            longer_perspective = self.expand_time_horizon(setback)
            return self.trust_in_eventual_vindication(longer_perspective)

        elif setback.seems_insurmountable():
            alternative_paths = self.find_different_routes_to_goal(setback)
            return self.adapt_strategy_while_keeping_destination(alternative_paths)

        else:
            return self.persevere_through_temporary_difficulty(setback)

    def inspire_others(self, community: List[Any]) -> List[Any]:
        """Hope is contagious when genuine."""

        hopeful_community = []
        for person in community:
            if person.is_discouraged():
                shared_vision = self.communicate_realistic_optimism(person)
                renewed_person = person.receive_hope(shared_vision)
                hopeful_community.append(renewed_person)
            else:
                hopeful_community.append(person)

        return hopeful_community

    def transform_suffering(self, pain: Any) -> Any:
        """Hope doesn't eliminate suffering but gives it meaning."""

        if pain.serves_greater_purpose():
            meaningful_suffering = self.find_redemptive_value(pain)
            return meaningful_suffering.bear_with_purpose()
        else:
            return self.work_to_eliminate_pointless_suffering(pain)
```

Hope prevents despair by maintaining confidence that present struggles serve ultimate purpose—it's optimism grounded in conviction about reality's fundamental nature

> Hope differs from mere optimism by being anchored in metaphysical convictions about reality's ultimate structure rather than temperamental disposition or wishful thinking.

.

Love

Caritas - Love That Seeks the Good of Others

Love serves as both the highest virtue and the form that gives meaning to all other virtues. In classical virtue ethics, love (caritas) isn't primarily an emotion but a choice—the deliberate willing of another's good for their own sake, not for what they can do for us. This love provides the ultimate motivation that prevents virtue from becoming mere self-improvement or social posturing.

Without love, even impressive virtuous actions become what St. Paul called "sounding brass or tinkling cymbal"—technically correct but ultimately hollow. Love transforms prudence from cleverness into wisdom, justice from rule-following into genuine care for others' dignity, courage from bravado into service, and all virtues from performance into authentic goodness.

```
@dataclass
class Love:
    """Willing the good of another for their own sake."""

    def seek_others_good(self, person: Any) -> Any:
        """Love as choice rather than feeling."""

        their_authentic_good = self.discern_what_truly_benefits(person)
        my_emotional_state = self.current_feelings_toward(person)

        # Love acts regardless of emotional state
        if their_authentic_good.requires_action():
            return self.choose_their_benefit_over_my_comfort(their_authentic_good)
        else:
            return self.respect_their_autonomy()

    def practice_self_sacrifice(self, loved_ones: List[Any], self_interest: Any) -> Any
        """Love gives rather than takes."""

        legitimate_needs_of_others = sum([person.essential_needs() for person in loved_
        my_legitimate_needs = self_interest.essential_needs()

        if legitimate_needs_of_others > my_legitimate_needs:
            return self.give_preferentially_to_others()
        else:
            return self.maintain_healthy_self_care()  # Can't give what you don't have

    def love_enemies(self, opponent: Any) -> Any:
        """The highest expression of love."""

        if opponent.has_wronged_me():
            # Forgiveness doesn't mean enabling harmful behavior
            appropriate_boundaries = self.establish_protective_limits(opponent)
            genuine_care = self.wish_their_authentic_transformation(opponent)

            return self.combine_boundaries_with_care(appropriate_boundaries, genuine_ca
        else:
            return self.treat_as_fellow_human_being(opponent)
    def integrate_all_virtues(self, moral_situation: Any) -> Any:
        """Love provides the why for all other virtues."""

        prudent_analysis = Prudence().deliberate(moral_situation)
        just_distribution = Justice().render_what_is_owed(moral_situation)
        courageous_action = Fortitude().face_adversity(moral_situation)
        temperate_moderation = Temperance().moderate_pleasures(moral_situation)
        faithful_trust = Faith().trust_beyond_proof(moral_situation)
        hopeful_perseverance = Hope().maintain_vision(moral_situation)

        # Love motivates and harmonizes all other virtues
        return self.unite_virtues_in_service_of_good(
            prudent_analysis, just_distribution, courageous_action,
            temperate_moderation, faithful_trust, hopeful_perseverance
        )
```

Love serves as the form and purpose of all other virtues—without love, virtues become self-serving excellence rather than genuine goodness

> As St. Paul wrote in 1 Corinthians 13, without love, even the most impressive virtuous actions become "sounding brass or tinkling cymbal"—impressive but ultimately empty.

.

Integration: The Virtue Operating System

The seven classical virtues work together as an integrated system for human flourishing:

```python
class VirtuousLife:
    """The complete virtue stack."""

    def __init__(self):
        # Cardinal virtues - natural human excellence
        self.prudence = Prudence()
        self.justice = Justice()
        self.fortitude = Fortitude()
        self.temperance = Temperance()

        # Theological virtues - divine gifts
        self.faith = Faith()
        self.hope = Hope()
        self.love = Love()

    def live_virtuously(self, situation: Any) -> Any:
        """Integrated virtue response."""

        # Prudence evaluates the situation
        wise_assessment = self.prudence.deliberate(situation)

        # Justice determines what's owed
        fair_response = self.justice.render_what_is_owed(situation)

        # Fortitude provides courage to act
        courageous_execution = self.fortitude.face_adversity(fair_response)

        # Temperance ensures moderation
        temperate_action = self.temperance.moderate_pleasures(courageous_execution)

        # Faith grounds action in ultimate reality
        faithful_commitment = self.faith.trust_beyond_proof(temperate_action)

        # Hope sustains long-term effort
        hopeful_perseverance = self.hope.maintain_vision(faithful_commitment)

        # Love provides the motivation and form
        loving_completion = self.love.seek_others_good(hopeful_perseverance)
        return loving_completion

def main():
    """The program of human flourishing."""

    virtuous_person = VirtuousLife()

    while life.continues():
        current_situation = life.present_challenge()
        virtuous_response = virtuous_person.live_virtuously(current_situation)

        # Virtue creates more virtue
        character = character.strengthen_through_practice(virtuous_response)
        community = community.improve_through_good_example(virtuous_response)

    return "A life well lived"

if __name__ == "__main__":
    main()
```

Why These Virtues Matter Now

Growing up with virtue-based formation created appreciation for character that purely secular frameworks struggle to motivate. The classical virtues offer several advantages over modern approaches to ethics:

Systematic Integration: Rather than isolated rules, virtues work together as a coherent system for human development.

Practical Wisdom: Prudence provides the flexibility that rigid moral codes lack—virtue adapts to circumstances while maintaining core principles.

Character Formation: Virtues focus on becoming the kind of person who naturally acts well, rather than just following external rules.

Transcendent Purpose: The theological virtues connect individual flourishing with ultimate meaning and community good.

Time-Tested Durability: These patterns have guided human development across cultures and centuries because they align with how humans actually flourish.

Our digital age desperately needs this foundation. As I explored in The Algorithm Eats Virtue, social media algorithms optimize for engagement over virtue. Political discourse rewards tribal loyalty over justice. Economic systems prioritize efficiency over human dignity. The classical virtues provide stable ground for building lives and communities that serve human flourishing rather than exploit human weakness.

The pseudo-Python is metaphor. The character it describes is what we need to cultivate in ourselves and our institutions.

This essay translates the seven classical virtues into Python code to reveal timeless patterns for human flourishing through contemporary metaphors. It connects to Ram Dass Teachings in Python on Eastern wisdom, Vedic Principles in Python on Sanskrit wisdom, The Algorithm Eats Virtue on virtue destruction, and Programming as Spiritual Practice on contemplative coding.

For philosophical foundations, see Aristotle's Nicomachean Ethics, Thomas Aquinas's Summa Theologica, and Alasdair MacIntyre's After Virtue.

"Virtue is not a feeling or emotion, but a disposition to act in ways that promote human flourishing." - Aristotle

"The code we live by shapes the life we live. Choose your algorithms wisely."