LESSON 16: SMART CONTRACTS AND DECENTRALIZED APPLICATIONS

A Prison Professors Self-Directed Course

Estimated time: 70–105 minutes

Educational and Safety Boundary:

> *This lesson provides general education about smart contracts, decentralized applications, and related risks. It does not provide individualized financial, investment, legal, tax, or programming advice. No activity requires internet access, an account, a wallet, a digital asset, programming, or a transaction.*

OPENING GUIDANCE

While serving 26 years in federal prison, I relied on self-directed learning because I could not access the changing technology outside. I learned by reading, developing a vocabulary, writing about new ideas, and documenting my progress. That process taught me to examine how a system works instead of being impressed by a label.

The term smart contract can sound more powerful than it is. The word smart may suggest that the code can think. The word contract may suggest that every program creates a legally enforceable agreement. Neither conclusion is reliable. A smart contract is computer code deployed on a blockchain. It follows the instructions written into it when a person, another program, or an outside service triggers the relevant function.

This lesson connects the earlier lessons on blockchains, tokens, wallets, transactions, and safety. A token on BNB Smart Chain or Ethereum is often created and managed through smart-contract code. A decentralized application may give a person a website or other interface for interacting with that code. The visible screen is only one part of the system.

The independent PP token community gives us a recurring example. Neither Prison Professors nor I created, organized, managed, or controlled that token project. Community members used a smart contract on BNB Smart Chain to create a token with programmed fee behavior. That code can perform defined calculations and record results. It cannot decide whether a mission deserves support, whether a public claim is complete, or whether a human decision is wise.

I encourage you to ask two questions throughout this lesson: What can the code do, and what still depends on people? Those questions will help you evaluate any proposed blockchain application without accepting advertising language as proof.

PURPOSE AND ESSENTIAL QUESTION

Lesson 16 explains how code can carry out defined actions on a blockchain and how people may interact with that code through decentralized applications. It also shows why automated execution does not remove coding defects, faulty data, administrative control, confusing interfaces, legal uncertainty, or the need for human judgment.

Essential Question:

- What can smart-contract code automate, and which decisions still require human judgment, evidence, governance, and accountability?

LEARNING OBJECTIVES

After completing this lesson, you should be able to:

- Define smart contract and decentralized application in plain language.
- Explain why a smart contract is code rather than an intelligent decision-maker or automatic legal agreement.
- Trace a simple input–rule–output process and identify the role of an oracle when outside information is needed.
- Identify possible benefits of transparent, repeatable execution.
- Evaluate coding, data, upgrade, governance, usability, financial, and legal risks.

KEY TERMS

SMART CONTRACT

Definition: A program deployed at an address on a blockchain. It stores data and carries out defined functions when a transaction or another contract calls those functions.

Sample sentence: The smart contract applied the written rule after an authorized transaction supplied the required input.

DECENTRALIZED APPLICATION

Definition: An application, often called a dapp, that uses one or more smart contracts on a decentralized network and usually includes a user interface and other supporting components.

Sample sentence: The decentralized application gave users a readable screen for requesting an action from the underlying smart contract.

CODE

Definition: Instructions written in a programming language so that a computer can perform defined operations.

Sample sentence: The code checked whether the recorded conditions were satisfied before producing an output.

ORACLE

Definition: A service or system that provides a blockchain application with information from outside the blockchain, such as a price, weather result, or verified event.

Sample sentence: The contract relied on an oracle to receive information that the blockchain could not observe by itself.

AUDIT

Definition: A structured technical review of code, design, or controls intended to identify defects and security weaknesses within a stated scope.

Sample sentence: The audit identified a weakness, but it did not guarantee that every future problem had been found.

VULNERABILITY

Definition: A weakness in code, design, configuration, data, or operations that could cause failure or allow misuse.

Sample sentence: A missing authorization check created a vulnerability that could allow an unapproved action.

MAIN LESSON

A SMART CONTRACT IS A PROGRAM ON A BLOCKCHAIN

A conventional program may run on a company server, personal computer, or telephone. A smart contract runs within a blockchain network's computing environment. Once deployed, it has an address. It can store information, receive calls to its functions, check conditions, update its state, and create records under the network's rules.

On Ethereum and compatible networks, a person normally interacts with a smart contract by submitting a transaction that calls a function. Validators process the transaction. Each participating node follows the same network rules and should reach the same result from the same valid starting state and input. The result may update a balance, record a vote, issue a token, transfer an asset, or reject the request.

The transaction requires a network fee. The fee pays for computation and record-keeping. More complex operations may require more computation than a simple transfer. A failed request may still consume network resources and create a fee even when the intended state change does not occur.

A smart contract does not normally begin an action merely because time passes or an event happens in the world. A transaction, another contract, or an authorized automation service must call the relevant function. The code then evaluates the information available to it.

THINK IN TERMS OF INPUT, RULE, STATE, AND OUTPUT

A beginner can understand smart-contract logic without writing code. Use four questions:

- Input: What information or request enters the program?
- Rule: What written conditions and calculations will the program apply?
- State: What information has the program already recorded?
- Output: What result will the program record or attempt to produce?

Imagine a simple program for issuing one digital completion badge. The input is a learner identifier and an authorized completion report. The rule says that a badge may be issued only when the completion report is valid and no badge has already been issued for that identifier. The state records whether the identifier already received a badge. The output is either a new badge record or a rejected request.

This model helps a learner locate weaknesses. What if the completion report is wrong? What if two people receive the same identifier? What if the authorized reporter loses control of a signing credential? What if the rules contain no correction process? The program may follow its instructions exactly and still produce an unfair or unintended human result.

"SMART" DOES NOT MEAN INTELLIGENT

A smart contract does not understand purpose, fairness, hardship, or context unless people translate a limited part of those ideas into explicit conditions. It cannot

recognize an exception that the designers failed to include. It does not reconsider a decision because the outcome feels wrong.

The code may be complex, but complexity is not judgment. A program can calculate a fee precisely while applying the wrong fee rate. It can reject an unauthorized request while failing to recognize a newly authorized person. It can produce a consistent result from inaccurate input.

Artificial intelligence and smart contracts are different concepts. Some systems may combine them, but a basic smart contract is not an artificial-intelligence model. It executes programmed logic within the blockchain environment.

"CONTRACT" DOES NOT AUTOMATICALLY MEAN LEGAL AGREEMENT

Some smart contracts help carry out obligations that people have described in a separate agreement. Others create tokens, record votes, operate games, manage access, or perform calculations without serving as a legal contract.

The label does not determine whether an arrangement is legally enforceable. Legal effect may depend on the parties, consent, disclosures, jurisdiction, subject, surrounding documents, and applicable law. Code may perform an action even when people later dispute authority, ownership, fraud, error, or legal responsibility. This course provides education rather than legal advice.

A DECENTRALIZED APPLICATION HAS SEVERAL LAYERS

A decentralized application usually combines a user-facing interface with one or more smart contracts. The interface may be a website, mobile application, or other screen. It helps a person choose an action and prepare a transaction. The smart contract performs the on-chain logic after the request reaches the network.

OTHER COMPONENTS MAY INCLUDE:

- A wallet or signing tool that authorizes a transaction.
- A conventional web server that delivers the interface.
- A data-indexing service that organizes blockchain records for faster display.
- An oracle that supplies outside information.
- An administrator or governance process that changes settings or upgrades code.
- Customer-support, legal, accounting, and operational teams.

These components can have different levels of decentralization. A smart contract may be public while the website is controlled by one company. The contract may be difficult to change while an administrator can change selected settings. The in-

terface may display information from a centralized server. Calling the entire system decentralized does not explain who controls each layer.

When evaluating a dapp, ask who controls the interface, code, administrative keys, upgrades, data sources, treasury, and public communications. Also ask what happens if one component fails.

ORACLES CONNECT CODE WITH OUTSIDE INFORMATION

A blockchain can verify information already available under its own rules. It cannot independently observe the weather, a court decision, a sports result, the value of an off-chain asset, or whether a learner completed a course. An oracle brings selected outside information into the blockchain environment.

The oracle expands what a smart contract can do, but it also creates a new dependency. If the source is wrong, delayed, manipulated, unavailable, or misunderstood, the contract may execute correctly from bad data. Multiple data providers or verification methods can reduce some risks, but they do not turn an uncertain outside event into perfect information.

An oracle also does not decide whether a rule is fair. It reports data according to its design. People remain responsible for choosing sources, resolving disputes, designing correction procedures, and deciding whether automated action is appropriate.

APPLIED CASE: THE INDEPENDENT PP TOKEN COMMUNITY

The independent PP token community provides a real example of smart-contract logic on BNB Smart Chain. The community website clearly states that the token is not an official Prison Professors project or endorsement. Neither CZ nor I created the token, and Prison Professors did not organize, manage, or control the project.

The public community website describes a 3 percent charge on buy or sell trades. It says that the smart contract collects the charge and directs resources toward a public mission wallet. A public BscScan page identifies the PP token contract and displays source information for code containing transaction-fee logic.

We can organize the example with the four-question model:

- Input: A trade involving the token and the recognized trading pair.
- Rule: Calculate the programmed fee under the contract's conditions and allocate it according to configured percentages.
- State: Track balances, fee settings, recorded allocations, and other contract data.
- Output: Update token balances and process the fee according to the code.

That description is simplified. The actual code contains additional functions, roles, conditions, and dependencies. A public source display can help a qualified reviewer examine the logic. It does not prove that every user understands the code, that the code has no vulnerability, that every website statement is complete, or that every off-chain action will follow a mission.

The case also separates several components that a learner may otherwise combine. The token contract contains on-chain rules. BNB Smart Chain processes transactions. BscScan displays public records and source information. The community website communicates the project's description. A public treasury address shows selected on-chain activity. Prison Professors carries out its educational mission through people, policies, institutions, and programs outside the contract.

Future lessons will use this same case to examine decentralized finance and governance. The purpose is education. No participant is being asked to acquire the token, visit the website, connect a wallet, approve a contract, or conduct a transaction.

TRANSPARENT AND REPEATABLE EXECUTION CAN OFFER BENEFITS

Smart contracts can support useful designs when the rules are suitable for automation.

- The same valid input can produce the same result under the same recorded state.
- Public code and transaction records may allow independent inspection.
- Automated calculations can reduce some manual steps and delays.
- Shared rules can coordinate people or organizations that use the same network.
- One contract may interact with another, allowing developers to combine existing components.
- Records can show when a function was called and what on-chain result followed.

These benefits are conditional. Public code helps only when someone can understand it. Repeatability reproduces a defect as consistently as it reproduces a correct rule. Composability can spread a failure from one component into another. Automation can reduce one human bottleneck while creating a different dependency.

CODING AND DESIGN RISKS

Code is written by people. Developers can misunderstand a requirement, omit a condition, use an unsafe pattern, or connect components incorrectly. Common classes of smart-contract weakness include broken access controls, faulty business

logic, unvalidated inputs, unsafe external calls, calculation errors, oracle manipulation, and upgrade problems.

A small defect can have a large effect when code controls valuable assets or a widely used application. Public deployment may allow many people and other contracts to interact with the same weakness before maintainers can respond.

Testing and review reduce risk but cannot prove perfection. A contract may behave correctly during expected tests and fail under an unusual sequence of actions. A dependency may change. A new interaction may expose a weakness that was not included in the original review.

UPGRADE AND GOVERNANCE RISKS

Some contracts are designed to be difficult or impossible to change. That can protect users from unexpected changes, but it can also preserve defects. Other systems use upgrade mechanisms or administrative settings. Those features can support repairs, but they give selected people or governance processes power over future behavior.

ASK:

- Can the code be upgraded or replaced?
- Who can change settings, fees, permissions, or destinations?
- Is a delay, vote, or multiple-signature approval required?
- Can users observe a proposed change before it takes effect?
- What happens if an administrator loses a key or abuses authority?

The word decentralized does not answer these questions. Control may be distributed in one part of the system and concentrated in another.

INTERFACE AND USABILITY RISKS

Most users do not read source code. They rely on an interface that translates a human choice into transaction data. A confusing or dishonest interface can describe one action while preparing another. A copied website can imitate a legitimate application. A wallet screen may display technical permissions that a beginner cannot interpret.

The underlying contract can also be genuine while the user selects the wrong network, address, function, or amount. Technical validity does not establish that the action matches the user's purpose.

No learner should connect to an application merely to understand this lesson. Paper analysis can build the same foundational judgment: identify the components, locate the decision points, and ask what evidence supports each claim.

FINANCIAL, OPERATIONAL, AND LEGAL RISKS

Smart-contract use may involve transaction fees, changing asset values, limited liquidity, network congestion, irreversible errors, and dependence on other contracts. An application may fail even when its own code works because an oracle, interface, administrator, market, network, or external institution fails.

Legal and operational responsibilities also continue. A public record does not automatically establish identity, lawful authority, ownership, compliance, accounting treatment, tax treatment, consumer protection, or enforceability. Organizations still need human policies for authorization, custody, records, privacy, dispute handling, and accountability.

AN AUDIT IS EVIDENCE, NOT A GUARANTEE

An audit can identify weaknesses within the code version, assumptions, and scope reviewed. A useful audit report should allow a reader to understand what was examined, when the review occurred, which findings were identified, and whether reported changes were rechecked.

An audit does not guarantee that the code is free from defects. It may exclude the website, oracle, governance process, administrative keys, later upgrades, or connected contracts. A logo stating “audited” is not enough to evaluate scope or quality.

The same disciplined approach applies here as in earlier lessons: separate a claim from the evidence supporting it. Ask what the evidence shows, what it leaves unresolved, and what could change after the review.

HUMAN JUDGMENT REMAINS PART OF THE SYSTEM

Smart contracts can automate narrow rules. People still decide which problem to address, which rules to write, which data to trust, who can change the system, how to communicate risks, and how to respond when the output harms someone.

Good design does not try to automate every decision. It identifies where consistency helps and where discretion, appeal, investigation, or compassion is required. A correction process may be as valuable as the original rule.

Self-directed learning works in a similar way. A study schedule can create structure, but the schedule cannot judge whether you understood the lesson. You must review your work, identify gaps, seek feedback, and adjust. Code can support a process. It cannot replace the person responsible for the outcome.

APPLIED SCENARIO: THE COMPLETION BADGE DISPUTE

The following scenario is fictional and requires no device or account.

An education organization creates a dapp that issues digital course-completion badges. An approved educator sends a completion result to an oracle. The oracle places that result on the blockchain. A smart contract then issues one badge to the learner identifier when all recorded conditions are satisfied.

THE CODE CHECKS FOUR ITEMS:

- The report came through the approved oracle.
- The report says "completed."
- The learner identifier has not received a badge before.
- The request arrived before the program deadline.

A learner completed the course, but the educator accidentally submitted "not completed." The oracle accurately delivered that incorrect report. The smart contract rejected the badge request. The program contains no appeal, correction, pause, or replacement process.

The code followed its instructions. The oracle delivered the data it received. The human result was still wrong.

Use the scenario to separate responsibilities:

- The educator created the original information.
- The oracle transported the information into the blockchain environment.
- The smart contract applied the written conditions.
- The dapp interface displayed the result.
- The organization designed the policy and omitted a correction path.

The organization should not describe the result as correct merely because the code executed successfully. A better design could include a documented review process, correction authority, time delay, second source, or appeal before a permanent badge record is issued.

BENEFITS, LIMITATIONS, RISKS, AND MISCONCEPTIONS

POSSIBLE BENEFITS

- Defined rules can execute consistently across valid requests.
- Public records may support inspection of selected actions and results.
- Automation can reduce some repetitive manual work.

- Shared code can coordinate participants across organizations.
- Smart contracts can interact with other contracts to build larger systems.
- A clear input–rule–output model can improve process design even when a blockchain is not used.

LIMITATIONS AND RISKS

- Code can contain vulnerabilities, missing conditions, and misunderstood requirements.
- Correct code can produce a harmful result from inaccurate or manipulated data.
- Oracles introduce outside sources, operators, availability needs, and trust assumptions.
- Upgrade rights and administrative settings can concentrate power.
- Immutable code can preserve a defect, while upgradeable code creates change-control risk.
- Interfaces may be centralized, misleading, unavailable, or compromised.
- Users may not understand the permissions or transaction data presented to them.
- Network fees, congestion, market conditions, and connected contracts can alter practical outcomes.
- Audits have a scope and date; they do not guarantee future safety.
- On-chain execution does not resolve every legal, ethical, accounting, privacy, or governance question.

COMMON MISCONCEPTIONS

“Smart contracts think.” They execute programmed logic. They do not understand intent, fairness, or context.

“Every smart contract is a legal contract.” Some code supports an agreement, while other code manages tokens, votes, games, or records. Legal effect depends on facts and applicable law.

“Code removes trust.” A system may shift trust toward developers, administrators, oracle operators, interface providers, auditors, governance participants, or data sources.

“A successful transaction proves a good outcome.” The network may process the instructions correctly even when the input was wrong or the human purpose was not achieved.

“A dapp is decentralized in every component.” The smart contract may be public while the website, data service, administrator, or communication channel remains centrally controlled.

“An audit proves the contract is safe.” An audit is a bounded review. Defects, excluded components, later changes, or new interactions may remain.

“Public code is easy for everyone to verify.” Availability is not the same as understanding. Skilled review, documentation, and clear communication remain necessary.

“Automation eliminates governance.” People still set rules, assign authority, approve upgrades, select data sources, and respond to failures.

OFFLINE EXERCISE: TEST THE MISSING CONDITION

Estimated time: 25–30 minutes

Materials: Separate paper and a pencil.

You will simulate a fictional smart contract for a learning-credit program. The credits have no cash value and exist only for this exercise.

The paper program has four inputs:

- Participant code
- Completion report: completed or not completed
- Report source: authorized or unauthorized
- Previous credit recorded: yes or no

Apply these written rules in order:

1. If the report source is unauthorized, write REJECT.
2. If the report says not completed, write REJECT.
3. If a previous credit is recorded, write REJECT.
4. If the source is authorized, the report says completed, and no previous credit exists, write APPROVE ONE CREDIT.

Test the rules against four cases:

CASE A: EXPECTED APPROVAL

Participant code P-101 has an authorized completion report and no previous credit. Record the output and identify the rule that produced it.

CASE B: UNAUTHORIZED SOURCE

Participant code P-102 has a completed report from an unauthorized source and no previous credit. Record the output and explain why the claimed completion alone is insufficient.

CASE C: INCORRECT INPUT

Participant code P-103 completed the course, but an authorized source mistakenly reports not completed. Record the output required by the code. Then explain why correct execution produced an incorrect human result.

CASE D: DUPLICATE IDENTIFIER

Two participants were mistakenly assigned code P-104. The first participant received a credit. The second participant later submits an authorized completion report under the same code. Record the output and explain the identity problem.

After completing all four cases, answer these questions:

- Which inputs came from people or outside systems?
- Which outputs followed the written rules correctly?
- What correction, appeal, or identity condition did the designers omit?
- What role could a second reviewer play before the output becomes permanent?
- Would every part of this process benefit from automatic execution? Explain your answer.
- Write one revised rule that improves the process without claiming to solve every possible error.

The exercise simplifies real smart-contract development. Actual code requires programming, testing, security review, network deployment, and operational controls. The purpose here is to practice identifying assumptions and missing conditions.

KNOWLEDGE CHECK

MULTIPLE CHOICE

1. Which statement best describes a smart contract?
   a. An intelligent system that can judge fairness and intent
   b. Code deployed on a blockchain that carries out defined functions when called
   c. A legal agreement that is automatically enforceable in every jurisdiction

d. A customer-service process that can reverse every transaction

2. Which description best fits a decentralized application?
   a. A smart contract with no interface, supporting service, or human control
   b. Any website that uses the word blockchain
   c. An application that uses smart contracts on a decentralized network and may also rely on interfaces, wallets, data services, or administrators
   d. A program that cannot contain centralized components
3. Why might a smart contract use an oracle?
   a. To obtain selected information from outside the blockchain
   b. To guarantee that all outside information is true
   c. To replace every human decision with artificial intelligence
   d. To eliminate the need for a transaction or function call

TRUE OR FALSE

» A completed audit guarantees that a smart contract, its interface, its data sources, and all future upgrades are free from vulnerabilities.

SHORT EXPLANATION

» In four or five sentences, use the independent PP token community case or the completion-badge scenario to explain one action code can automate and two forms of human judgment or accountability that code does not replace.

SUMMARY AND PRACTICAL TAKEAWAYS

» A smart contract is code deployed at a blockchain address.
» A transaction, another contract, or an authorized service normally triggers a function.
» Inputs, rules, stored state, and outputs provide a useful model for analyzing the code.
» A smart contract is not automatically intelligent or legally enforceable.
» A dapp usually combines smart contracts with an interface and other supporting components.
» Different parts of a dapp can have different levels of decentralization and control.
» An oracle supplies selected outside information that a blockchain cannot observe by itself.
» Correct code can produce an unintended result when the input or rule is wrong.

- Public source information can support inspection without guaranteeing safety or complete understanding.
- Audits can identify weaknesses within a defined scope and time; they cannot prove perfection.
- Upgradeability supports correction but creates authority and governance risks.
- Immutable code limits changes but can preserve defects.
- Interfaces, administrators, data sources, connected contracts, and human policies remain part of the system.
- The independent PP token community shows how code can apply fee logic while people remain responsible for mission, governance, communication, and use of resources.
- No course activity requires a wallet, asset, application connection, programming, or transaction.

The practical takeaway is direct: evaluate both the automated rule and the human system surrounding it. Ask what the code can do, what information it depends on, who can change it, and what happens when the output is wrong.

PROFILE JOURNAL ASSIGNMENT

Write one coherent journal entry of approximately 300–400 words. Write in the first person, as part of the Profile you are building. Do not submit a disconnected list of answers. Use the prompts below to organize a clear explanation:

- Define a smart contract and decentralized application in your own words.
- Explain one task that code can perform consistently.
- Explain why correct execution does not guarantee a fair, safe, legal, or useful human outcome.
- Use the independent PP token community case or the completion-badge scenario to identify the input, rule, state, and output.
- Describe one missing condition or inaccurate input that could create a harmful result.
- Explain the role of an oracle and one risk that accompanies outside data.
- Describe what an audit can show and what it cannot guarantee.
- Identify one decision that should remain subject to human review, appeal, or accountability.

» Connect the lesson to your own self-directed learning. Explain how written routines can support your progress without replacing reflection, feedback, and adjustment.

Your response should demonstrate technical understanding, critical thinking, and personal responsibility. Do not visit a website, connect a wallet, create an account, write deployable code, acquire an asset, or perform a transaction. Do not include any real password, private key, seed phrase, authentication code, wallet address, balance, transaction identifier, or financial-account information.

SUBMISSION REMINDER

Write your response on separate paper or in an approved institutional messaging system. Include your name or approved Profile identifier, Lesson 16: Smart Contracts and Decentralized Applications, and the date you completed the entry.

Use one of the established Prison Professors Profile methods, subject to your facility's rules:

» Send it by institutional email to Playbook@PrisonProfessors.org. Suggested subject: Web3 Lesson 16 — [date completed].

» Send it by postal mail to: Prison Professors, 1205 BMC Drive, Suite 706, Cedar Park, TX 78613.

» Send it to an approved family member or supporter who can enter it on your Profile at PrisonProfessors.org.

Keep a copy when circumstances permit. Never include passwords, private keys, seed phrases, authentication codes, account numbers, real wallet addresses, balances, transaction identifiers, transaction details, or other sensitive information in a Profile response.