Development of Online Modules for Teaching Blockchain

Oneal Douglin[1], Shuchen Liu[1], Dave Emdin[2], Alfonso D. Meraz[1], Yu-Chung Chang-Hou[1*], Alejandro Strachan[3]

[1]*Pasadena City College, Pasadena, CA 91106, USA*
[2]*Community College of Philadelphia, Philadelphia, PA 19130, USA*
[3]*Purdue University, West Lafayette, IN 47906, USA*
**YXCHANG@pasadena.edu*

Abstract:
Blockchain technology enables the creation of a distributed and tamper-proof ledger, even in the presence of untrusted agents. While much financial resources and attention are devoted to blockchain tools, the underlying technology is not well understood by the general population. This paper presents a newly developed online tool that allows users to learn and create their own blockchain, with a graphical user interface and code. The module is freely available on nanoHUB.org and describes all components of the blockchain, including the SHA256, Proof of Work, and other features that enable the blockchain to function as a tamper-proof ledger. This tool has been utilized to instruct students without prior knowledge of blockchain technology, and the survey of students' responses demonstrates that this tool is an effective way of teaching the general population about blockchain technology.

Keywords: Blockchain, SHA256, Proof of Work, online education, collaborative research

Introduction

Blockchain technology has garnered significant attention in recent years due to its decentralized and secure approach to recording and verifying transactions, revolutionizing various industries [1]. The importance of blockchain lies in its ability to enhance transparency through decentralization, ensure cryptographic security for immutability, enable the potential of cryptocurrencies like Bitcoin and Ethereum, and streamline transactions for greater speed and efficiency. It can speed up international transactions and can protect against inflation [2-4]. Walmart Canada applied blockchain to solve a common logistics nightmare [5].

To delve deeper into the technical aspects and original concepts behind Bitcoin and Ethereum, it is recommended to refer to their respective white papers. The Bitcoin white paper, titled "Bitcoin: A Peer-to-Peer Electronic Cash System," was published by Satoshi Nakamoto in 2008 [6]. The Ethereum white paper, titled "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," was authored by Vitalik Buterin in 2013 [7].

Comprehending blockchain technology can be challenging for the general population [8]. Despite conducting extensive searches, no accessible and interactive learning modules that effectively explain the fundamental mechanisms of blockchain, specifically tailored for the community college community, have been discovered.

To address this gap, an online instructional module has been developed to provide learners with hands-on experience in comprehending blockchain's inner workings. The module offers insights into data

structures, transaction verification, consensus mechanisms, and real-world applications beyond cryptocurrencies, empowering individuals to grasp the transformative potential of blockchain in diverse industries. By combining theoretical explanations with interactive coding examples, our objective is to foster a deeper understanding and broader adoption of blockchain in the modern world.

Demonstrating Blockchain Technology Using Python

Our blockchain has two main features: SHA256 and Proof of Work [9]. The SHA, which stands for Secure Hash Algorithms, was created by the National Security Agency (NSA) of the United States. It is a collection of cryptographic hash functions designed to ensure data security [8].

SHA256

The SHA256 function converts input data into a hash of 64 seemingly random numbers and letters. For example, "hi" converted by SHA256 will output "8f434346648f6b96df89dda901c5176b10a6d83961dd3c1ac88b59b2dc327aa4" and this output is affected by factors such as uppercase and lowercase letters, spaces, and other symbols in the input. However, the output hash results are not truly random because each specific input will always have its own unique hash. Crucially, while the SHA256 function easily computes a hash from a given input (See Figure 1), it is extremely challenging to reverse-engineer the original input based solely on the hash result [8]. There are various Secure Hash Algorithms: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 with different levels of security [8]. SHA256 was used to reproduce the demonstration in Blockchain 101 - A Visual Demo [7].

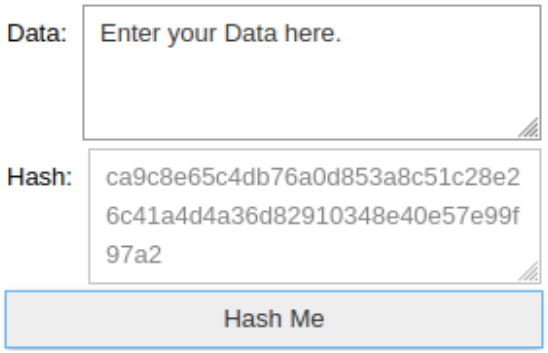


Fig. 1. SHA256 Graphic User Interface [9]

Proof of Work

The Proof of Work is a method of validating a block to be added to the blockchain. It is a consensus mechanism within blockchain and cryptocurrency ecosystems, facilitating transaction verification and block addition. In this process, miners or participants engage in solving intricate puzzles or mathematical problems to validate transactions and solidify them on the network [10, 11].

The blockchain learning module used a simple Proof of Work. A hash that starts with four zeros (0000) indicates that Proof of Work was done on a block. For this demonstration of a simple Proof of Work, the nonce and data are concatenated to generate the hash. The data never changes but the nonce is increment by one until the nonce causes the hash to begin with four zeros. (See Figure 2.)

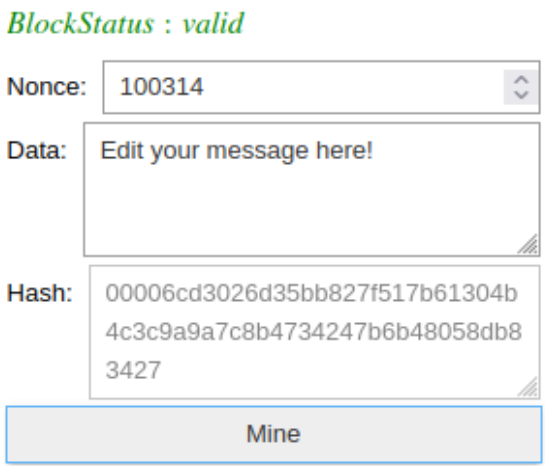


Fig. 2. Proof of Work Graphic User Interface [9]

A Block in a Blockchain

A blockchain consists of valid individual blocks that are linked together. An individual block stores a transaction, so each block has the following: index, nonce, data, previous hash, and hash. The index is the numerical position of the block in the blockchain. For example, the third block in the blockchain will have an index of 2 and not 3. This is because the first block index is 0. The nonce is an integer that changes when performing Proof of Work. The data stores the transaction details. The previous hash is the hash result of the previous block. Importantly, the previous hash makes the blockchain secure; it links the current block to the previous one. The hash is the SHA256 hash result of the concatenation of index, nonce, data, and previous hash.

The Genesis block is the first block in the blockchain, and it has an index of 0. Because there is no block before the Genesis block, the previous hash of the Genesis block is defined to be a sequence of 64 zeros. The system automatically generates the Proof of Work when the Genesis block was created, making it a valid block with the nonce value 58259.

Suppose two transactions will be added to the blockchain: Robert paid John $1000, and Mary paid Tom $1688. Assume the blockchain currently only has the Genesis block created. The following process is performed when adding the transaction "Robert paid John $1000!" to the blockchain as a new block. The new block index is 1, the nonce starts with 1, data is "Robert paid John $1000!" the previous hash has the hash result of the Genesis block, "000045a66e07b1edd4fb65cc5eaf31eca1e2955befbd43975bff6a392164e76f", and hash of the current block is the SHA256 hash result of "11Robert paid John $1000!", which is "1ba761f1dbcf91e0d29e06d14d15d813b3f6d7f54c758c5f617fb289060b3ad3", which may appear random but is actually not random. (See Figure 3.)

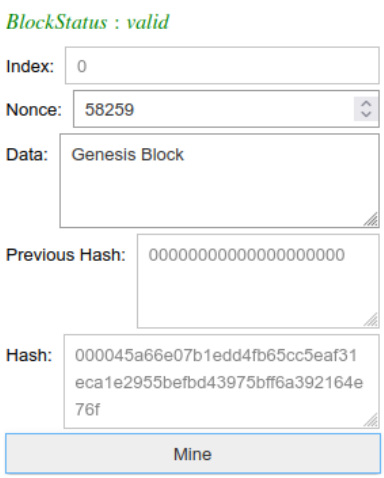


Fig. 3-1. Genesis block

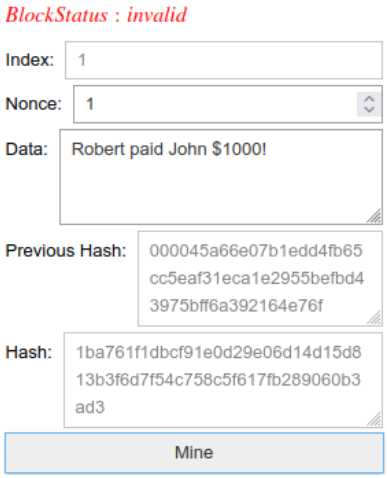


Fig. 3-2. Invalid block-1

To validate this transaction, a nonce value must be found, so the hash result will start with four zeros. The nonce is incremented one by one. It will take a long time to find the hash result beginning with four zeros. The probability of getting four zeros at the beginning of the hash result is very small, P = (1/16)^4 because each digit of the hash result could be a number from 0 to 9, or lower-case character from a to f; in total, there are 16 possibilities [8]. In the Python code, a while loop was used to speed up the search. The nonce “48871” was found for this block-1 to become valid, with the hash value “0000a606f7bc866285273e52ffcfc56049fb60c7dd71a398296f929fcf5fa9bc”.
(See Figure 4.)

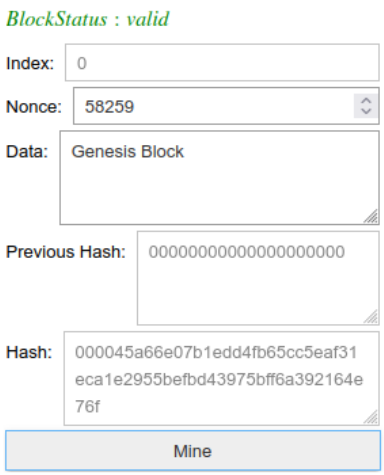


Fig. 4-1. Genesis block

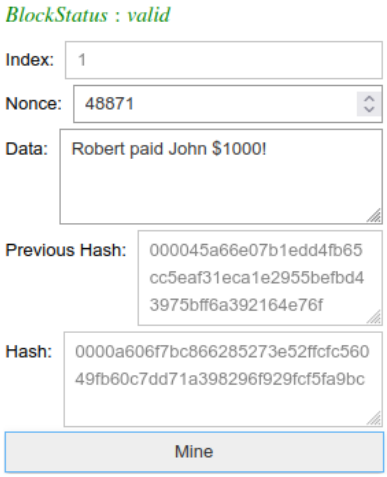


Fig. 4-2. Valid block-1

To add the second transaction, "Mary paid Tom $1688!" to the blockchain, as a new block, the following process is performed. The new block index is 2, the nonce starts with 1, data is "Mary paid Tom $1688!" the previous hash has the hash result of the block 1, "0000a606f7bc866285273e52ffcfc56049fb60c7dd71a398296f929fcf5fa9bc", and hash of the current block is the hash result of "21Mary paid Tom $1688!", which is "fa34602246b194672f6728ab696f044d79ebbf1e8336f35450ed33dce5cd7030". (See Figure 5.)

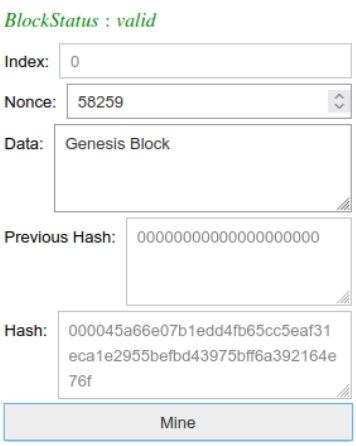


Fig. 5-1. Genesis block

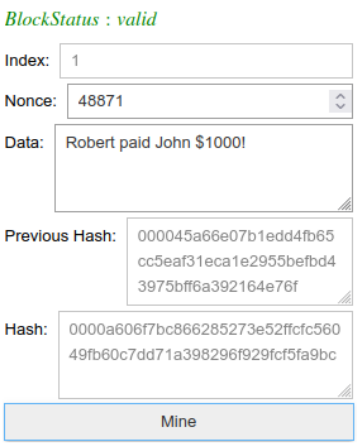


Fig. 5-2. Valid block-1

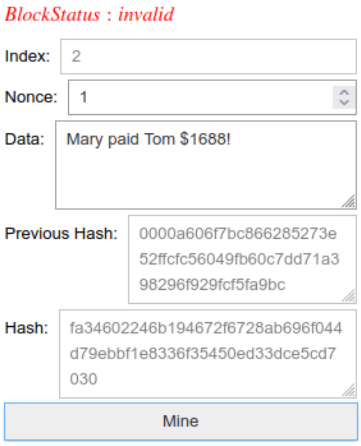


Fig. 5-3. Invalid block-2

To validate this transaction, a nonce value needs to be found, so that the hash result will start with four zeros. The nonce is incremented one by one. The nonce "171666" was found for this block-2 to become valid, with the hash value "0000cb8e1e9d0b1b24cc7abdb862dc27d5047fa3fc0620bb6c1204cd21dbf180". (See Figure 6.)

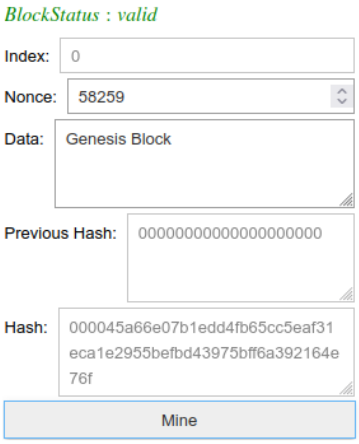


Fig. 6-1. Genesis block

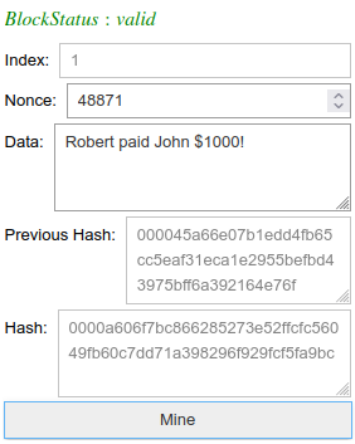


Fig. 6-2. Valid block-1

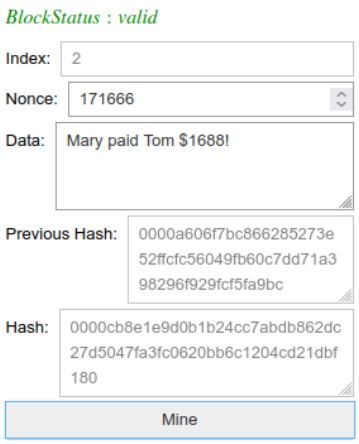
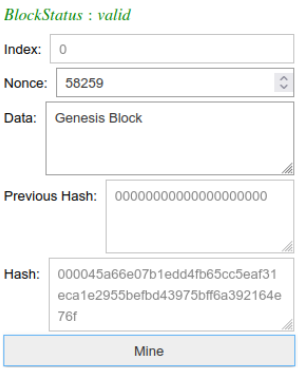


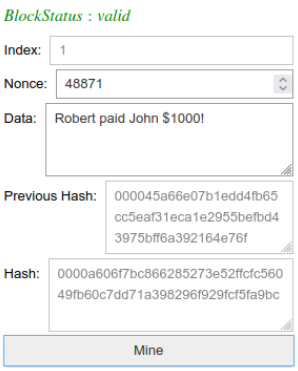
Fig. 6-3. Valid block-2

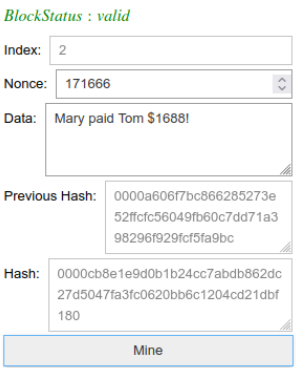
Similarly, more blocks can be added to the blockchain, with a lot of computation time and energy.

Why is the Blockchain Tamper-resistant?

The unique property of SHA256 provides the integrity of tamper resistance for the blockchain. To illustrate this, refer to the blockchain with four valid blocks in Figure 7.







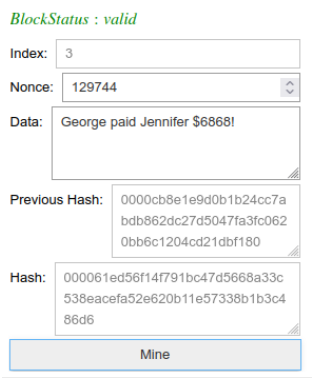
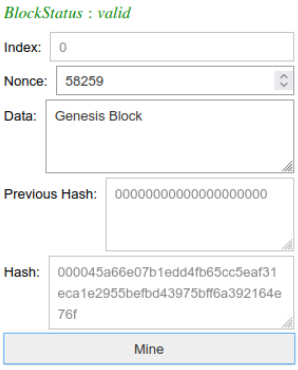
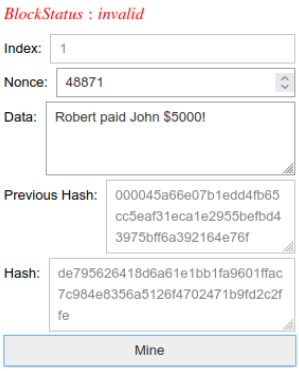
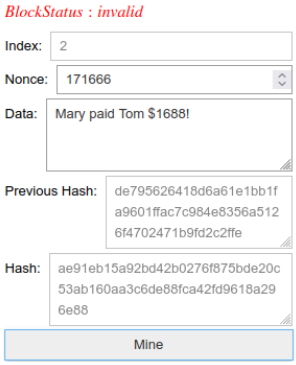


Fig. 7-1. Genesis block Fig. 7-2. Valid block-1 Fig. 7-3. Valid block-2 Fig. 7-4. Valid block-3

If someone wants to tamper the block-1 by changing "1000" to "5000", immediately, the block-1 will become invalid. (See Figure 8.) To make block-1 valid, one must work hard to find the nonce to make it valid. Even then, the new hash result will be passed down to the next block, which makes the following block invalid; similarly, all subsequent blocks become invalid. Unless the person has more than 50% of the world's computing power, it is impossible to make all subsequent blocks valid [10, 11].







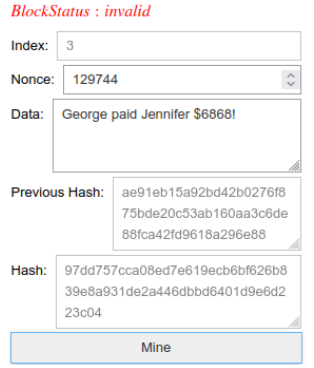


Fig. 8-1. Genesis block Fig. 8-2. Invalid block-1 Fig. 8-3. Invalid block-2 Fig. 8-4. Invalid block-3

Code Development of a Blockchain Learning Module Using Python

The development of our project requires us to research what a blockchain is, how it works, and identify the necessary features needed for a simple blockchain to work. It starts by reviewing Blockchain 101 - A Visual Demo [1][7]. In this Visual Demo, Brownworth presents a graphical user interface for each main concept of a blockchain; SHA256, Proof of Work, and implements them to create an interactive blockchain.

Our project was carried out through remote collaboration. As a result, online tools were necessary for coding, file sharing, and communication. Our team scheduled communication meetings via Cisco Webex and Zoom. The coding was written in Python, and Jupyter Notebook was used to run the scripts. nanoHUB.org is where all the files were hosted, shared, and executed. Our blockchain module is one of many tools available to use on nanoHUB.org.

The flowchart in Figure 9 demonstrates how the blockchain operates and presents the logic of the module, which includes SHA256, Proof of Work, and the complete blockchain.

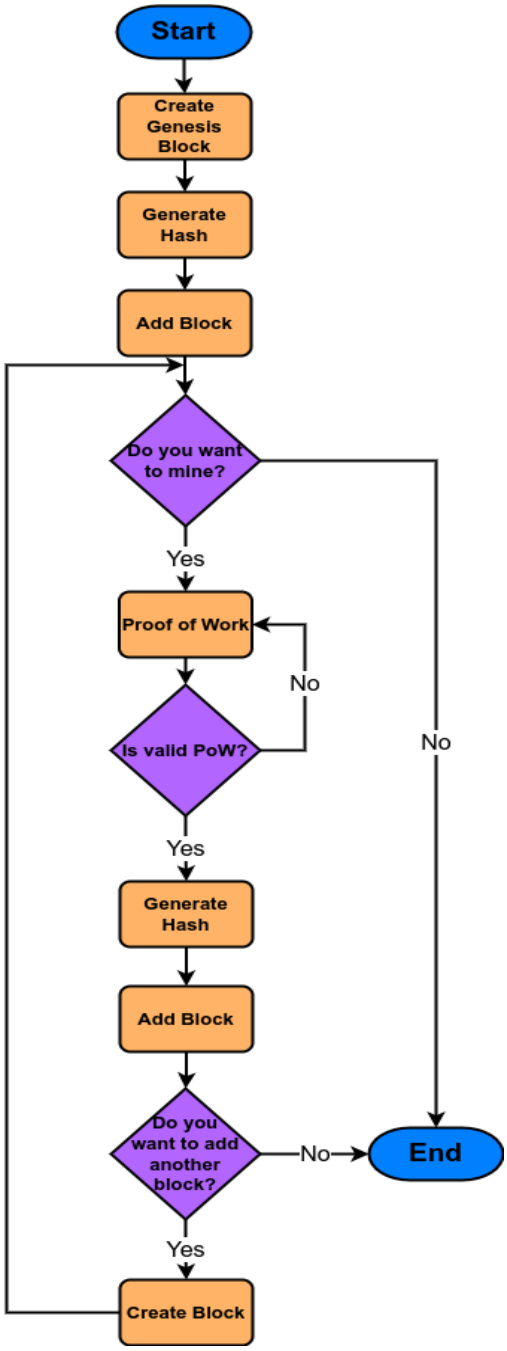


Fig. 9. Flowchart

Create a Blockchain in Python

Naming conventions are rules for how a programmer names the various parts of a computer code to make it easy to read [12]. The authors used the lowerCamelCase naming convention [13]. The lowerCamelCase forms a compound word by uppercase the first letter of each word except the first word and removing the space between each word. For example, naming the previous hash variable as previousHash, or generate hash function as generateHash.

Note: The previousHash is a variable name used in the code to represent the previous hash mentioned in this paper.

The blockchain module uses the string and hashlib libraries:

```
import string
from hashlib import sha256
```

A Blockchain is a chain of linked blocks that were validated by Proof of Work. For this learning module, a valid Proof of Work is when the hash value starts with four zeros; for example, "0000a606f7bc866285273e52ffcfc56049fb60c7dd71a398296f929fcf5fa9bc" is a valid hash. The four zeros for the Proof of Work were chosen because it is manually difficult to find, yet quick for a computer to generate.

Each block has the following variables to store the block details: index, nonce, data, previousHash, and hash. The index and nonce variables are an integer, while the data, previousHash, and hash variables are string.

Note: A string variable stores any character that can be typed from the keyboard.

Here is a code snippet to create a block object:

```
# create new block object to be added to the blockchain
class __block:
    def __init__(self):
        self.index = 0
        self.nonce = 1
        self.data = “”
        self.previousHash = “”
        self.hash = “”
```

The hash is generated by using sha256 from the hashlib library. Here is a code snippet to generate the hash:

```
 # generate hash from block properties: index, nonce, data, previousHash
def generateHash(self, index, nonce, data, previousHash):
    return sha256((str(index) + str(nonce) + str(data) + str(previousHash)).encode()).hexdigest()
```

The Proof of Work function is a simple function that uses a Python while loop: it increments the nonce by one until the sha256 hash result starts with four zeros.

Note: A loop is repeatedly running the same set of code until a specified condition is met.

Here is a code snippet for the Proof of Work:

```
# generate proof of work by computing a hash that begins with mining zero length (four zeros '0000')
def proofOfWork(self, index, nonce, data, previousHash):
    # lending length of zero of hash proof of work
    MINING_ZERO_LENGTH = 4
```

```
    # set nonce to start from 1
    nonce = 1

    # variable to indicate that proof of work was done
    Check = False

    while check == False:

        # generate proof of work hash by calling generateHash function
        powHash = self.generateHash(index, nonce, data, previousHash)

        # validate the proof of work
        if powHash.find('0' *  MINING_ZERO_LENGTH) == 0:
            check = True
        else:
            # increment nonce linearly, nonce is only variable that can be changed to computer proof of work
            nonce = nonce + 1

    # return proof of work hash and update nonce
    return powHash, nonce
```

Each valid block object is added to a Python list. Here is the code snippet to create a Python list and add block object to it:

```
# blockchain list to store each block
chain = []

# create a new block
newBlock = self.__block()

# update a new block object
…
newBlock.nonce = 1
newBlock.data = "enter your transaction here"
…

# add new block to the chain
self.chain.append(newBlock)
```

Note: When creating a new block, it needs the last index and previousHash from the chain. The hash is generated from the concatenated index, nonce, data, and previousHash.

The content of the blockchain should be immutable (cannot be changed). Here is how to display the content of the blockchain:

```
# display blocks in the blockchain
for x in range(0, 3):
    print(f"Index: {chain[x].index}")
    print(f"Nonce: {chain[x].nonce}")
    print(f"Data: {chain[x].data}")
    print(f"Previous Hash: {chain[x].previousHash}")
    print(f"hash: {chain[x].hash}")
```

```
  print()
```

Note: For a Python list, the first index starts from zero.

To create the Graphical User Interface (GUI) in Jupyter Notebook, the following libraries must be called before creating and displaying the widgets:

```
# import GUI libraries
import functools
from IPython.display import display
from ipywidgets import Layout, Label, Text, Textarea, Button, IntText
```

Here is how to create those widgets:

```
# set width of description
styleDescription = {'description_width':'initial'}

# set width of widgets (Label, Text, IntText, Button)
layoutWidgetWidth = Layout(width="275px")

# set width for Textarea
layoutTextareaWidth = Layout(width="70px")

# label
iAmLabel = Label(
    value = 'enter message here',
    style = styleDescription,
    layout = layoutWidgetWidth)

# change label value
iAmLabel.value = 'new message here'

# textbox
iAmTextbox = Text(
  description = 'enter input name here:',
  style = styleDescription,
  layout = layoutWidgetWidth,
  disabled = True)

# change textbox value
iAmTextbox.value = '5'

# intText, to store only integer
iAmIntText = IntText(
  description = 'enter input name here:',
  style = styleDescription,
  layout = layoutWidgetWidth)

# change inttext value
IAmIntText.value = '1'

# textarea
```

```
iAmTextarea = Textarea(
  description = 'enter input name here:',
  style = styleDescription,
  layout = layoutTextareaWidth,
  disabled = False)

# change textarea value
iAmTextarea = 'new content'

# button
iAmButton = Button(
  description = 'enter button name here',
  style = styleDescription,
  layout = layoutWidgetWidth)

# display widgets
display(iAmLabel, iAmTextbox, iAmIntText, iAmTextarea, iAmButton)
```

Live Code Examples from the Blockchain Learning Module

To make an effective learning module, two separate functions were created that focus on understanding the two key concepts of SHA256 and Proof of Work, with explanations and live code. The live code allows the user to modify and execute online directly. The code is straightforward, as well as easy to understand and run. (See Figures 10 and 11.)

```
1  ################################################
2  # import module
3  from hashlib import sha256
4  ################################################
5
6  ################################################
7  ## generate hash from user input
8  ################################################
9  def generateHash(data):
10     # The encode() method returns an encoded version of the given string.
11     # The hexdigest() method returns a string object of double length, containing only hexadecimal digits.
12     return sha256((data).encode()).hexdigest()
13
14 # type your message below in red.
15 input_data = "Enter your Data here."
16
17 # this generates the hash from your input_data
18 hash_result = generateHash(input_data)
19
20 # display output
21 print(f"This is your input data: {input_data}")
22 print(f"This is your sha256 hash result: {hash_result}")
23
24
```

```
This is your input data: Enter your Data here.
This is your sha256 hash result: ca9c8e65c4db76a0d853a8c51c28e26c41a4d4a36d82910348e40e57e99f97a2
```

Fig. 10. Python code for SHA256 [9]

```python
from hashlib import sha256

# generate hash from block properties: nonce, data
def generateHash(data, nonce):
    return sha256((str(nonce) + str(data)).encode()).hexdigest()

# type your message below in red
data = "Edit your message here!"

# initialize nonce
nonce = 1

# leading length of zeros of the hash for proof-of-work
MINING_ZERO_LENGTH = 4

# creating an amount of zeros that satisfies computing the proof of work
mine = '0' * MINING_ZERO_LENGTH

# variable to indicate that proof of work was done
is_valid = False

# process of mining
while is_valid == False:
    # new hash for the added block
    hash_result = generateHash(data, nonce)

    # validate the proof of work
    # using string method to search for the MINING_ZERO_LENGTH in beginning of the hash
    if hash_result.find(mine) == 0:
        is_valid = True
    else:
        # increment nonce. nonce is only variable that can be change to compute proof of work
        nonce = nonce + 1

# outputs
print(f'This is your input data: {data}')
print(f'This is your obtained nonce for Proof of Work: {nonce}')
print(f'This is your sha256 result: {hash_result}')
print(f'Is the status valid?: {is_valid}')
```

```
This is your input data: Edit your message here!
This is your obtained nonce for Proof of Work: 100314
This is your sha256 result: 00006cd3026d35bb827f517b61304b4c3c9a9a7c8b4734247b6b48058db83427
Is the status valid?: True
```

Fig. 11. Python code for Proof of Work [9]

The Implementation of the Blockchain Development

Ten students from a College Algebra class (Math 3) and eighteen students from a Calculus class (Math 5A) at Pasadena City College and thirty-three students from a Calculus class (Math 190) at El Camino College participated in reviewing the blockchain learning module and then filling out the survey. As a guest speaker, an instructor briefly introduced students to the blockchain and then showed the blockchain video to the class. She encouraged students to participate in interactive activities for testing the tool. The total time spent working on the module was 40 minutes on average.

A sixteen-question survey was designed to capture learning outcomes and experiences. Eighty-one individuals, including instructors and students, completed the survey, and the feedback overall was positive. Figures 12-1, 12-2, 12-3 and 12-4 presents four images of students practicing the developed module. Students were asked to rank their experiences completing the module from 1 to 5, with 5 being the best experience.

Fig. 12-1. Practicing blockchain learning module in a Calculus class at Pasadena City College

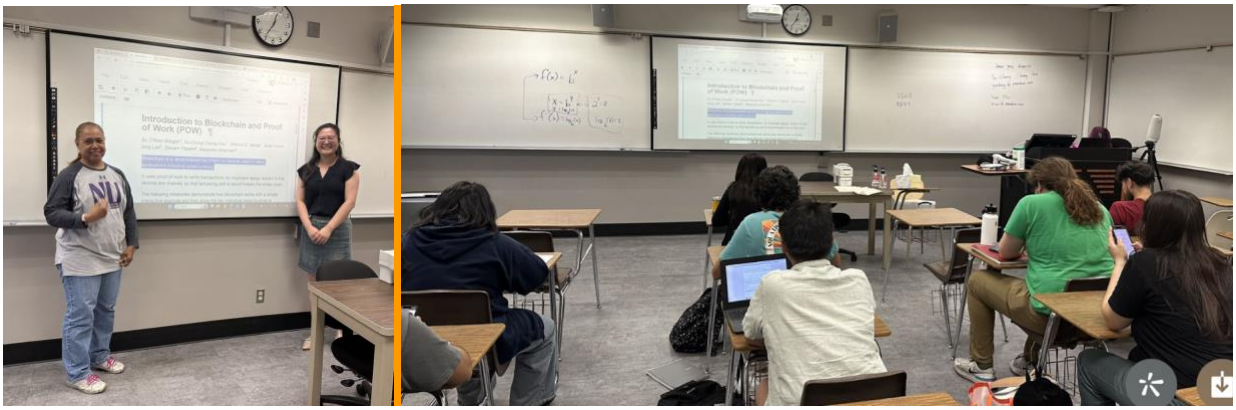


Fig. 12-2. and 12-3. Practicing blockchain learning module in a College Algebra class at Pasadena City College

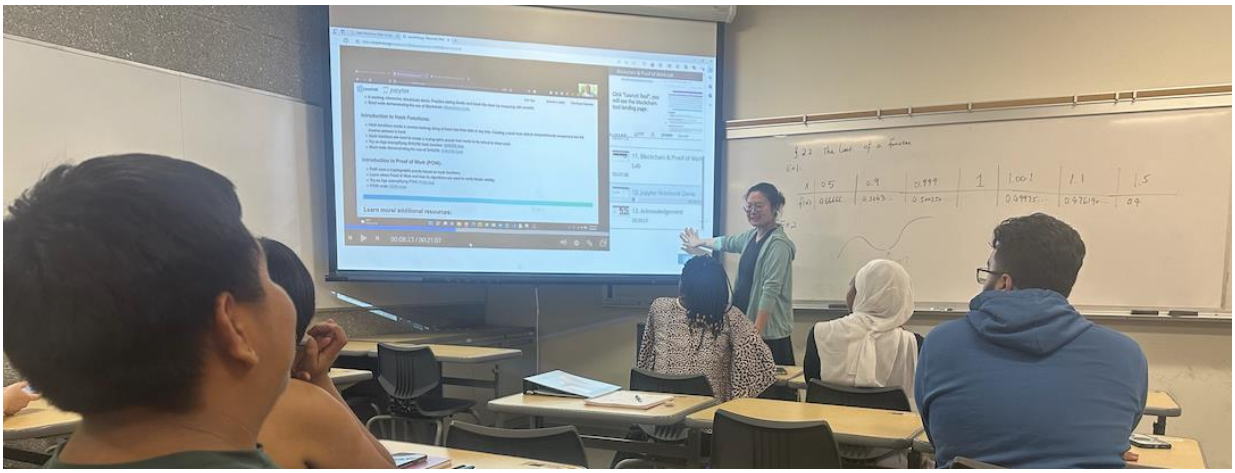


Fig. 12-4. Practicing blockchain learning module in a Calculus class at El Camino College

Students were asked: "**Describe your overall experience using the blockchain tools.**" and **"How well did the tool help with understanding general blockchain terminology?"** Most respondents (91%) said they better understood the blockchain technology (rank 3, 4, or 5). Only one student selected rank 1, indicating that almost all respondents (99%) had a positive experience. The responses were encouraging, including **"**I believe that I have a better understanding of blockchain technology and terminology" and **"**I feel significantly better being I have had no prior experience with blockchain tools, and now I have a better understanding of the mechanism of blockchain and why it can be used for financial transactions.**"** **"**I feel, after this tool, my understanding definitely got better, especially about the technical implementation part. I like the process of playing with GUI first, and then look[ing] at the code. This learning experience builds a separation of abstraction, help[ed] me to learn the content step-by-step.**"** **"**Never heard of blockchain, I like the justification of how secure the blockchain is. I feel the system is thought out well.**"** **"**I really enjoyed this module! Thank you for organizing this tool and presentation. I've used cryptocurrency years ago, but never really understood how it worked. Now I understand how powerful and secure block chains can be and why people have so much trust in the value of cryptocurrency.**"** **"**I had no understanding of how blockchain worked beforehand, but this lesson very simply summarized it and explained its functionality using hands on resources. Thank you very much!**"**

To survey the content usability, students were additionally asked: "**For the content of this tool, please kindly give us your comments and/or suggestions to improve it.**" Most respondents (90%) said the content was clear (rank 3, 4, or 5). One respondent emphasized the usefulness of watching the blockchain video before or while testing the learning module, saying: "Once I watched the video [...] everything [was] so clear." Another respondent said, "Put the blockchain videos before [the] introduction." Others provided suggestions to improve users' experience in learning the content. The following are three examples.

> "For the video, perhaps subtitles can be included to help understand what speakers are saying."
> "I think that having multiple tabs may make it a bit harder to navigate, especially for those without a larger monitor as they wouldn't be able to see both the presentation and module at the same time."
> "I feel that this tool is an overall good tutorial for technical concepts. If I have any suggestions, I would say that some content, such as the formula for calculating the hash, was explicitly mentioned in the presentation video … but not in the notebook. Personally, I believe it would be helpful to improve the documents by including some information from the video. This will help users review the concepts and follow the flow better. Additionally, I think it would be beneficial to attach a quick, visually appealing, and entertaining YouTube video that talks about blockchain applications and why it works before presenting the entire tool. Through this, users will be more inclined to learn more about blockchain and, as a result, take a look at this technical tool."

In a continued effort to survey usability, students were asked: "**For the layout or structure of this tool, please kindly give us your comments and/or suggestions to improve it.**" One respondent suggested including more pictures, and one respondent recommended the nanoHUB.org administration consider using Jupyter Lab instead of Jupyter Notebook so that there would be a table of contents on the left side which would allow users to navigate among different sessions easily. Additionally, some respondents suggested improving the layout so that it would be easy for users to access the video while using the tool.

> **"**I strongly believe, with no prior experience, the interface and layout for the tutorial are very approachable. Everything is Clear and Concise as well as the video helps. My only advice is putting the video along with the tool."
> "Possibly having the presentation video and tools be usable on the same page."

Most people found the layout reasonably easy to follow and understand, saying:

> "I think it is good because it breaks down to the basic SHA256 and uses it to explain [Proof of Work], so that I can easily understand the whole mechanism of blockchain."
> "The layout is fairly easy to understand. The fact that it opens a new window for each notebook or section makes it easier to navigate back to the main page by just exiting out of that one notebook.

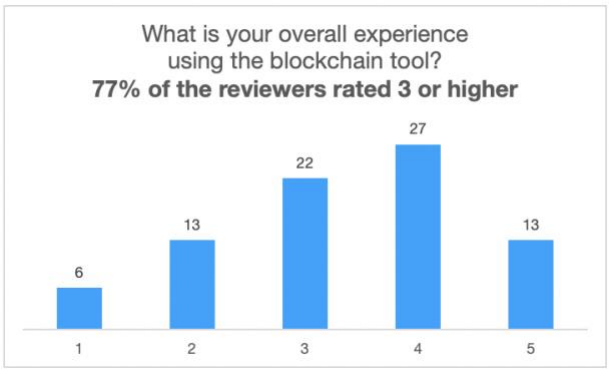


Fig. 13-1. Survey results with sample size 81

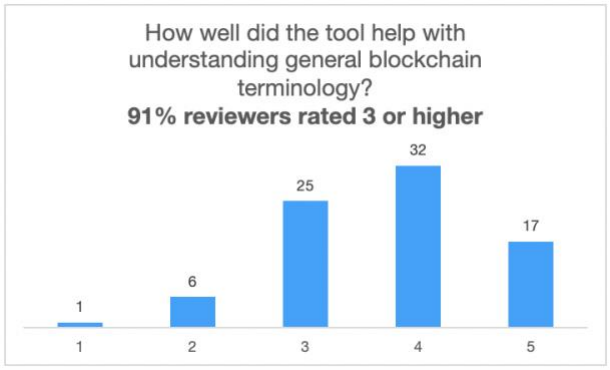


Fig. 13-2. Survey results with sample size 81

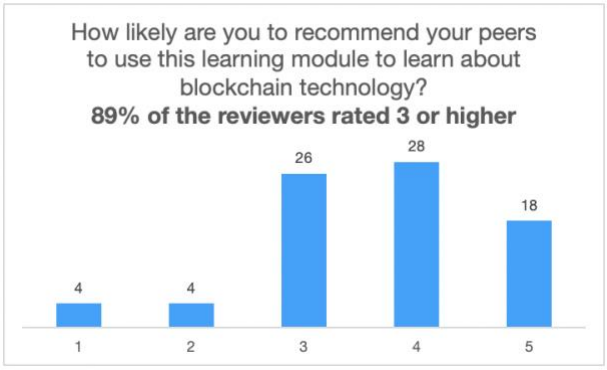


Fig. 13-3. Survey results with sample size 81

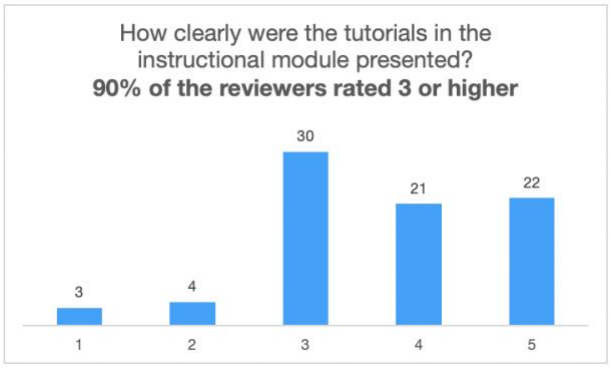


Fig. 13-4. Survey results with sample size 81

Conclusion

This paper presents an original online tool that instructs the mechanism and code behind blockchain technology. The online tool is freely available, and the instructions provide sufficient information to learn and apply blockchain technology. The general reaction from instructors and students was that the modules helped them understand blockchain. It is expected that the tool will be expanded and improved according to the feedback and suggestions and more systematically disseminated to various groups of people beyond students at community colleges, including local middle and high school students, as well as people at work.

Acknowledgments

This work was supported by National Science Foundation under DUE ATE #2000281, the Micro Nano Technology Collaborative Undergraduate Research Network (MNT-CURN), and the Network for Computational Nanotechnology (NCN) - home of nanoHUB.org. The authors would like to express special thanks to Professor Jared Ashcroft at Pasadena City College, Dr. Brian Hyun-jong Lee, Dr. Shivam Tripathi, Dr. Tanya Faltens, and Juan Carlos Verduzco at Purdue University for the valuable discussions and guidance over the past two years. The authors would also like to thank Professor Jie Zhong at Cal State Los Angeles, Professor Valerie Carr, Professor Morris Jones and Professor Wendy Lee from San Jose State University, Professor Kun Niu from El Camino College, Professor Michelle

Guo, Professor Michelle Ingram, Professor Fendi He, Professor Dave Smith, Professor Erin Shaw, Professor Thomas Thoen from Pasadena City College, and student Kevin D. Ethridge from Community College of Philadelphia, student Melody Huang from Cornell University, student Janet Tang from MIT, student Sophia Barber from UC San Diego, student Melinda Wu from UC Riverside, students Anthony Ko, Gayvalin Tammy Sujaritchai, Jingchao Zhong, David Tao, Jan Poster, Jocelyn Zhu, Jasmine Lai, Joya Stewart, Thet Paing Da Na, Pete Chayapirad, a class of Calculus from El Camino College, a class of College Algebra and a class of Calculus from Pasadena City College, student Daniel Weiss from Peninsula High School, students John Xie, Ben Yeh, Sydney Hsu, and Eric Qiu from Arcadia High School for their suggestions and feedback and/or conducting the practicing of this learning tool in their classes or groups.

Disclosure
The authors declare no conflicts of interest.

References

[1] G. Habib, S. Sharma, S. Ibrahim, I. Ahmad, S. Qureshi, & M. Ishfaq, (2022). Blockchain technology: Benefits, challenges, applications, and integration of Blockchain technology with cloud computing. Future Internet, 14(11), 341. https://doi.org/10.3390/fi14110341 https://www.proquest.com/docview/2748280995?pq-origsite=summon.

[2] S. Chouhan, "Blockchain Features May Control Inflation", International Journal of Science and Research (IJSR), Volume 12 Issue 7, July 2023, pp. 1315-1317, https://www.ijsr.net/getabstract.php?paperid=SR23717225333.

[3] I. Ioannou, G. Demirel, Blockchain and supply chain finance: a critical literature review at the intersection of operations, finance and law. *J BANK FINANC TECHNOL* 6, 83–107 (2022). https://doi.org/10.1007/s42786-022-00040-1.

[4] P. Sharma, R. Jindal, & M.D. Borah, A Review of Blockchain-Based Applications and Challenges. Wireless Pers Commun 123, 1201–1243 (2022). https://doi.org/10.1007/s11277-021-09176-7.

[5] Vitasek, K., Bayliss, J., Owen, L., and Srivastava, N., Harvard Business Review, (2022, May). How Walmart Canada Uses Blockchain to Solve Supply-Chain Challenges.

[6] S. Nakamoto, (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf.

[7] A. Brownworth, Introductory video by Anders Brownworth: Blockchain 101 - A Visual Demo.

[8] FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS) chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[9] O. Douglin, Y. Chang-Hou, A. Morez, B. Lee, S. Tripathi, A. Strachan, (2022, March). nanoHUB, Blockchain and Proof of Work Lab, https://nanohub.org/resources/35369/about#citethis.

[10] Banafa, Ahmed. Blockchain Technology and Applications. 1st ed., River Publishers, 2020. https://caccl-pcc.primo.exlibrisgroup.com/permalink/01CACCL_PCC/1hls13v/alma991001327569605270.

[11] P. R. Nair and D. R. Dorai, "Evaluation of Performance and Security of Proof of Work and Proof of Stake using Blockchain," *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Tirunelveli, India, 2021, pp. 279-283, doi: 10.1109/ICICV50876.2021.9388487.

[12] "Naming Conventions." In *Principles of Programming and Coding*, edited by Donald R. Franceschetti, 200-203. Ipswich, MA: Salem Press/Grey House, 2018. *Gale eBooks* (accessed August 28, 2023). https://link-gale-

com.ezp.pasadena.edu/apps/doc/CX7384400104/GVRL?u=pasa19871&sid=bookmark-GVRL&xid=8b0f7751.
[13] “CamelCase.” 2007. *New Scientist* 196 (2627): 58. https://search-ebscohost-com.ezp.pasadena.edu/login.aspx?direct=true&db=f6h&AN=27506966&site=ehost-live.