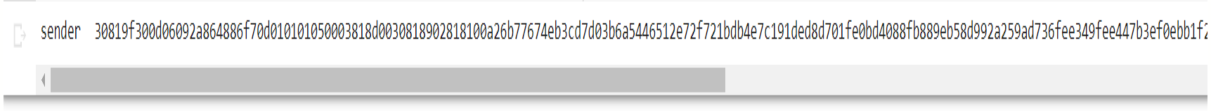
Practical – 1

A] A Simple Client class that generates private and public keys by using built in Python RSA algorithm.

Input:

```
# following imports are required by PKI
!pip install Crypto
#!pip install pycryptodome
import Crypto
import hashlib
import random
import string
import json
import binascii
import numpy as np
import pandas as pd
import pylab as pl
import logging
import datetime
import collections
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5

class Client:
  def __init__(self):
    random = Random.new().read
    self._private_key = RSA.generate(1024, random)
    self._public_key = self._private_key.publickey()
    self._signer = PKCS1_v1_5.new(self._private_key)
  @property
  def identity(self):
    return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')
Dinesh = Client()
print ("sender ",Dinesh.identity)
```

Output:-



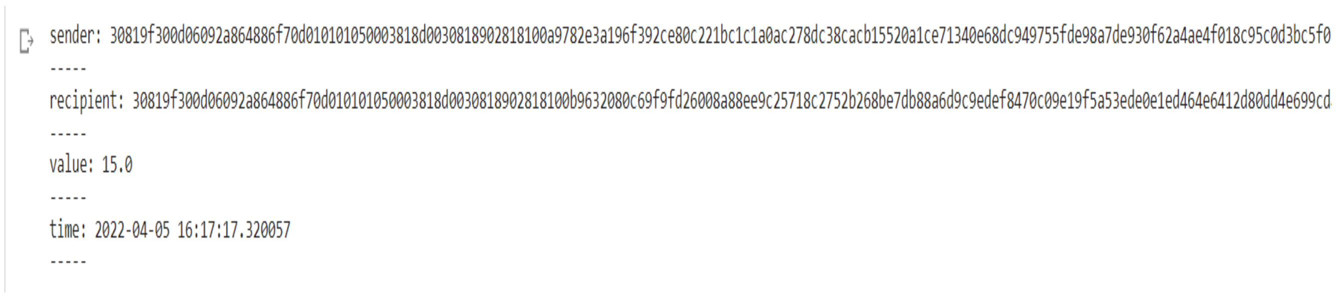
```
sender  30819f300d06092a864886f70d010101050003818d0030818902818100a26b77674eb3cd7d03b6a5446512e72f721bdb4e7c191ded8d701fe0bd4088fb889eb58d992a259ad736fee349fee447b3ef0ebb1f
```

B] A Transaction class to send and receive amount and use it.
# following imports are required by PKI

Input:

```
import hashlib
import random
import string
import json
import binascii
import numpy as np
import pandas as pd
import pylab as pl
import logging
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
class Client:
  def __init__(self):
    random = Random.new().read
    self._private_key = RSA.generate(1024, random)
    self._public_key = self._private_key.publickey()
    self._signer = PKCS1_v1_5.new(self._private_key)
  @property
  def identity(self):
    return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

class Transaction:
  def __init__(self, sender, recipient, value):
    self.sender = sender
    self.recipient = recipient
    self.value = value
    self.time = datetime.datetime.now()
  def to_dict(self):
    if self.sender == "Genesis":
        identity = "Genesis"
    else:
        identity = self.sender.identity

    return collections.OrderedDict({
        'sender': identity,
        'recipient': self.recipient,
        'value': self.value,
        'time' : self.time})

  def sign_transaction(self):
    private_key = self.sender._private_key
    signer = PKCS1_v1_5.new(private_key)
    h = SHA.new(str(self.to_dict()).encode('utf8'))
    return binascii.hexlify(signer.sign(h)).decode('ascii')

def display_transaction(transaction):
    #for transaction in transactions:
    dict = transaction.to_dict()
    print ("sender: " + dict['sender'])
    print ('-----')
    print ("recipient: " + dict['recipient'])
    print ('-----')
    print ("value: " + str(dict['value']))
    print ('-----')
    print ("time: " + str(dict['time']))
    print ('-----')
transactions = []

Dinesh = Client()
Ramesh = Client()

t1 = Transaction(
  Dinesh,
  Ramesh.identity,
  15.0
)
t1.sign_transaction()
display_transaction (t1)
```

Output:-



```
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100a9782e3a196f392ce80c221bc1c1a0ac278dc38cacb15520a1ce71340e68dc949755fde98a7de930f62a4ae4f018c95c0d3bc5f0
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b9632080c69f9fd26008a88ee9c25718c2752b268be7db88a6d9c9edef8470c09e19f5a53ede0e1ed464e6412d80dd4e699cd
-----
value: 15.0
-----
time: 2022-04-05 16:17:17.320057
-----
```

C] Create a blockchain genesis block and use it.

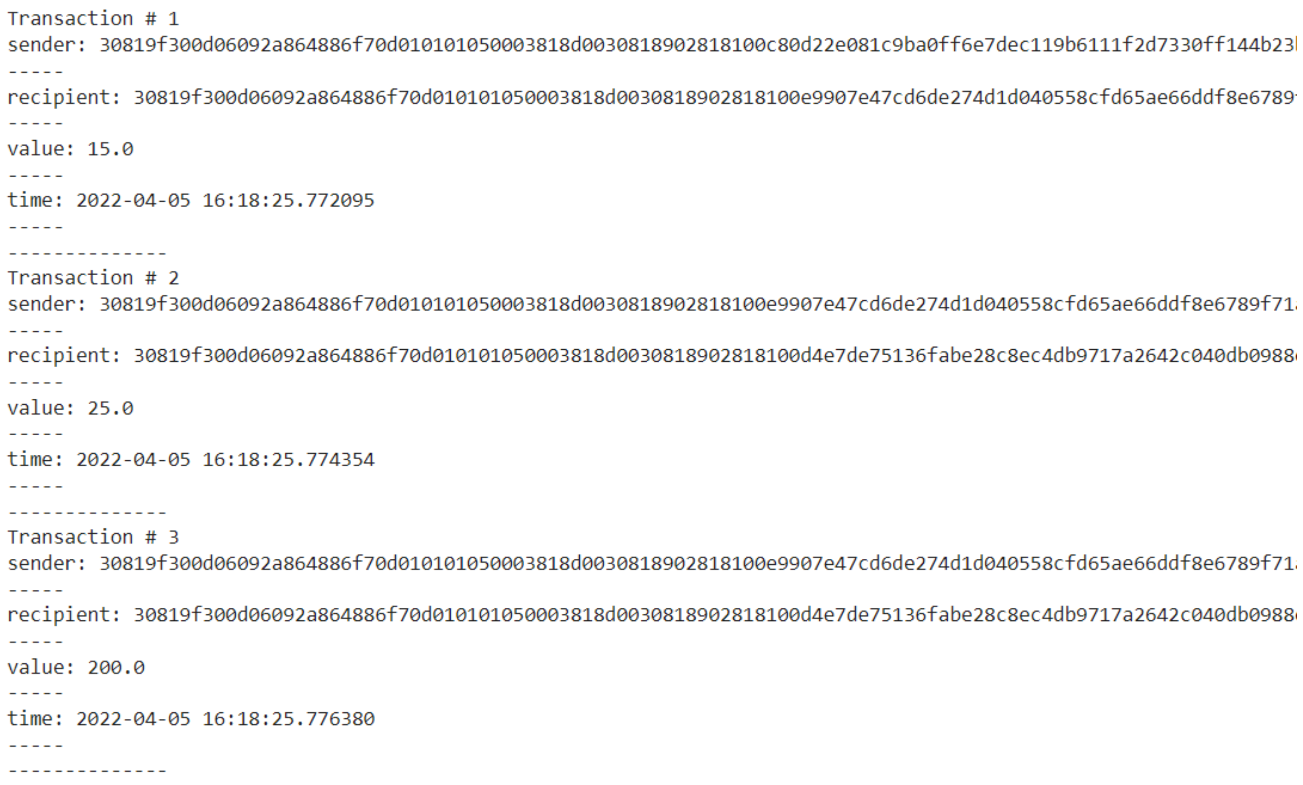
# following imports are required by PKI

Input:

```
import hashlib
import random
import string
import json
import binascii
import numpy as np
import pandas as pd
import pylab as pl
import logging
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
class Client:
    def __init__(self):
      random = Random.new().read
      self._private_key = RSA.generate(1024, random)
      self._public_key = self._private_key.publickey()
      self._signer = PKCS1_v1_5.new(self._private_key)
    @property
    def identity(self):
      return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

class Transaction:
    def __init__(self, sender, recipient, value):
       self.sender = sender
       self.recipient = recipient
       self.value = value
       self.time = datetime.datetime.now()

    def to_dict(self):
       if self.sender == "Genesis":
           identity = "Genesis"
       else:
           identity = self.sender.identity

       return collections.OrderedDict({
           'sender': identity,
           'recipient': self.recipient,
           'value': self.value,
           'time' : self.time})

    def sign_transaction(self):
       private_key = self.sender._private_key
       signer = PKCS1_v1_5.new(private_key)
       h = SHA.new(str(self.to_dict()).encode('utf8'))
       return binascii.hexlify(signer.sign(h)).decode('ascii')

def display_transaction(transaction):
       #for transaction in transactions:
       dict = transaction.to_dict()
       print ("sender: " + dict['sender'])
       print ('-----')
       print ("recipient: " + dict['recipient'])
       print ('-----')
       print ("value: " + str(dict['value']))
       print ('-----')
       print ("time: " + str(dict['time']))
       print ('-----')

transactions = []

Dinesh = Client()
Ramesh = Client()
Suresh = Client()

t1 = Transaction(
    Dinesh,
    Ramesh.identity,
    15.0
)

t1.sign_transaction()
transactions.append(t1)
```

```
t2 = Transaction(
  Ramesh,
  Suresh.identity,
  25.0
)
t2.sign_transaction()
transactions.append(t2)

t3 = Transaction(
  Ramesh,
  Suresh.identity,
  200.0
)
t3.sign_transaction()
transactions.append(t3)

tn=1
for t in transactions:
  print("Transaction #",tn)
  display_transaction (t)
  tn=tn+1
  print ('--------------')
```

Output:-



```
Transaction # 1
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c80d22e081c9ba0ff6e7dec119b6111f2d7330ff144b23
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100e9907e47cd6de274d1d040558cfd65ae66ddf8e6789
-----
value: 15.0
-----
time: 2022-04-05 16:18:25.772095
-----
--------------
Transaction # 2
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100e9907e47cd6de274d1d040558cfd65ae66ddf8e6789f71
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d4e7de75136fabe28c8ec4db9717a2642c040db0988
-----
value: 25.0
-----
time: 2022-04-05 16:18:25.774354
-----
--------------
Transaction # 3
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100e9907e47cd6de274d1d040558cfd65ae66ddf8e6789f71
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d4e7de75136fabe28c8ec4db9717a2642c040db0988
-----
value: 200.0
-----
time: 2022-04-05 16:18:25.776380
-----
--------------
```

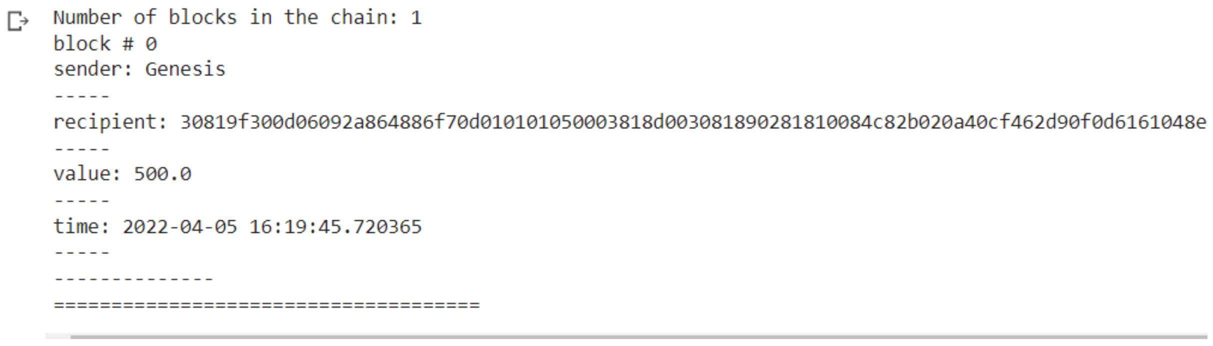
D] Create a block chain genesis block and use it.

```
# create a block chain genesis block and use it
# following imports are required by PKI
```

Input:

```
import hashlib
import random
import string
import json
import binascii
import numpy as np
import pandas as pd
import pylab as pl
import logging
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

class Client:
  def __init__(self):
    random = Random.new().read
    self._private_key = RSA.generate(1024, random)
    self._public_key = self._private_key.publickey()
    self._signer = PKCS1_v1_5.new(self._private_key)
  @property
  def identity(self):
    return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

class Transaction:
  def __init__(self, sender, recipient, value):
     self.sender = sender
     self.recipient = recipient
     self.value = value
     self.time = datetime.datetime.now()
  def to_dict(self):
     if self.sender == "Genesis":
       identity = "Genesis"
     else:
       identity = self.sender.identity

     return collections.OrderedDict({
       'sender': identity,
       'recipient': self.recipient,
       'value': self.value,
       'time' : self.time})

  def sign_transaction(self):
     private_key = self.sender._private_key
     signer = PKCS1_v1_5.new(private_key)
     h = SHA.new(str(self.to_dict()).encode('utf8'))
     return binascii.hexlify(signer.sign(h)).decode('ascii')

def display_transaction(transaction):
     #for transaction in transactions:
     dict = transaction.to_dict()
     print ("sender: " + dict['sender'])
     print ('-----')
     print ("recipient: " + dict['recipient'])
     print ('-----')
     print ("value: " + str(dict['value']))
     print ('-----')
     print ("time: " + str(dict['time']))
     print ('-----')

def dump_blockchain (self):
  print ("Number of blocks in the chain: " + str(len (self)))
  for x in range (len(TPCoins)):
    block_temp = TPCoins[x]
    print ("block # " + str(x))
    for transaction in block_temp.verified_transactions:
       display_transaction (transaction)
       print ('--------------')
    print ('=====================================')

class Block:
  def __init__(self):
        self.verified_transactions = []
        self.previous_block_hash = ""
        self.Nonce = ""

Dinesh = Client()

t0 = Transaction (
   "Genesis",
   Dinesh.identity,
   500.0
)

block0 = Block()
block0.previous_block_hash = None
Nonce = None
block0.verified_transactions.append (t0)
digest = hash (block0)
last_block_hash = digest

TPCoins = []
TPCoins.append (block0)

dump_blockchain(TPCoins)
```

Output:-



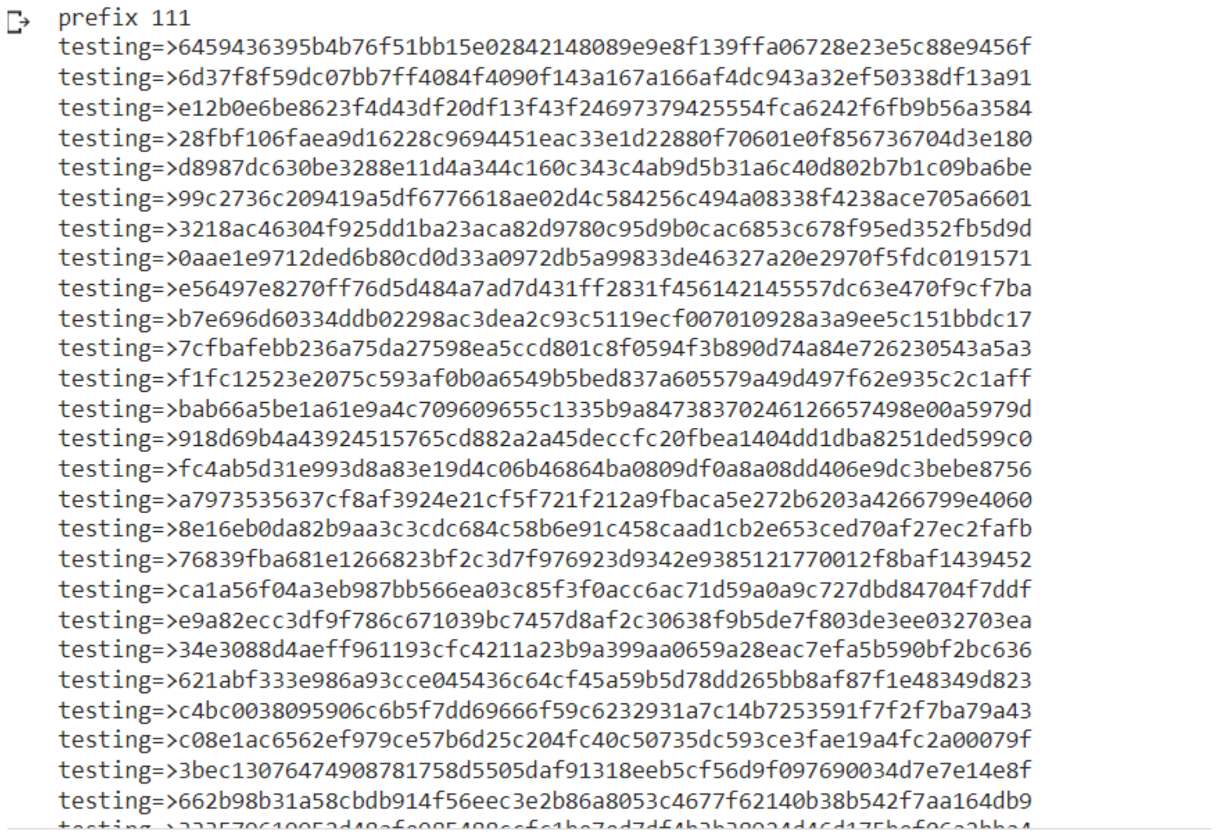
```
Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100084c82b020a40cf462d90f0d6161048e
-----
value: 500.0
-----
time: 2022-04-05 16:19:45.720365
-----
--------------
=====================================
```

E] Create a mining function and test it.

Input:

```
import hashlib
def sha256(message):
    return hashlib.sha256(message.encode('ascii')).hexdigest()

def mine(message, difficulty=1):
  assert difficulty >= 1
  #if(difficulty <1):
  #       return
  #'1'*3=> '111'
  prefix = '1' * difficulty
  print("prefix",prefix)
  for i in range(1000):
    digest = sha256(str(hash(message)) + str(i))
    print("testing=>"+digest)
    if digest.startswith(prefix):
      print ("after " + str(i) + " iterations found nonce: "+ digest)
      return i #i= nonce value

mine ("test message",3)
```

Output:-



```
prefix 111
testing=>6459436395b4b76f51bb15e02842148089e9e8f139ffa06728e23e5c88e9456f
testing=>6d37f8f59dc07bb7ff4084f4090f143a167a166af4dc943a32ef50338df13a91
testing=>e12b0e6be8623f4d43df20df13f43f24697379425554fca6242f6fb9b56a3584
testing=>28fbf106faea9d16228c9694451eac33e1d22880f70601e0f856736704d3e180
testing=>d8987dc630be3288e11d4a344c160c343c4ab9d5b31a6c40d802b7b1c09ba6be
testing=>99c2736c209419a5df6776618ae02d4c584256c494a08338f4238ace705a6601
testing=>3218ac46304f925dd1ba23aca82d9780c95d9b0cac6853c678f95ed352fb5d9d
testing=>0aae1e9712ded6b80cd0d33a0972db5a99833de46327a20e2970f5fdc0191571
testing=>e56497e8270ff76d5d484a7ad7d431ff2831f456142145557dc63e470f9cf7ba
testing=>b7e696d60334ddb02298ac3dea2c93c5119ecf007010928a3a9ee5c151bbdc17
testing=>7cfbafebb236a75da27598ea5ccd801c8f0594f3b890d74a84e726230543a5a3
testing=>f1fc12523e2075c593af0b0a6549b5bed837a605579a49d497f62e935c2c1aff
testing=>bab66a5be1a61e9a4c709609655c1335b9a8473837024612665749 8e00a5979d
testing=>918d69b4a43924515765cd882a2a45deccfc20fbea1404dd1dba8251ded599c0
testing=>fc4ab5d31e993d8a83e19d4c06b46864ba0809df0a8a08dd406e9dc3bebe8756
testing=>a7973535637cf8af3924e21cf5f721f212a9fbaca5e272b6203a4266799e4060
testing=>8e16eb0da82b9aa3c3cdc684c58b6e91c458caad1cb2e653ced70af27ec2fafb
testing=>76839fba681e1266823bf2c3d7f976923d9342e9385121770012f8baf1439452
testing=>ca1a56f04a3eb987bb566ea03c85f3f0acc6ac71d59a0a9c727dbd84704f7ddf
testing=>e9a82ecc3df9f786c671039bc7457d8af2c30638f9b5de7f803de3ee032703ea
testing=>34e3088d4aeff961193cfc4211a23b9a399aa0659a28eac7efa5b590bf2bc636
testing=>621abf333e986a93cce045436c64cf45a59b5d78dd265bb8af87f1e48349d823
testing=>c4bc0038095906c6b5f7dd69666f59c6232931a7c14b7253591f7f2f7ba79a43
testing=>c08e1ac6562ef979ce57b6d25c204fc40c50735dc593ce3fae19a4fc2a00079f
testing=>3bec1307647490878175 8d5505daf91318eeb5cf56d9f097690034d7e7e14e8f
testing=>662b98b31a58cbdb914f56eec3e2b86a8053c4677f62140b38b542f7aa164db9
```

F] Add blocks to the miner and dump the blockchain.

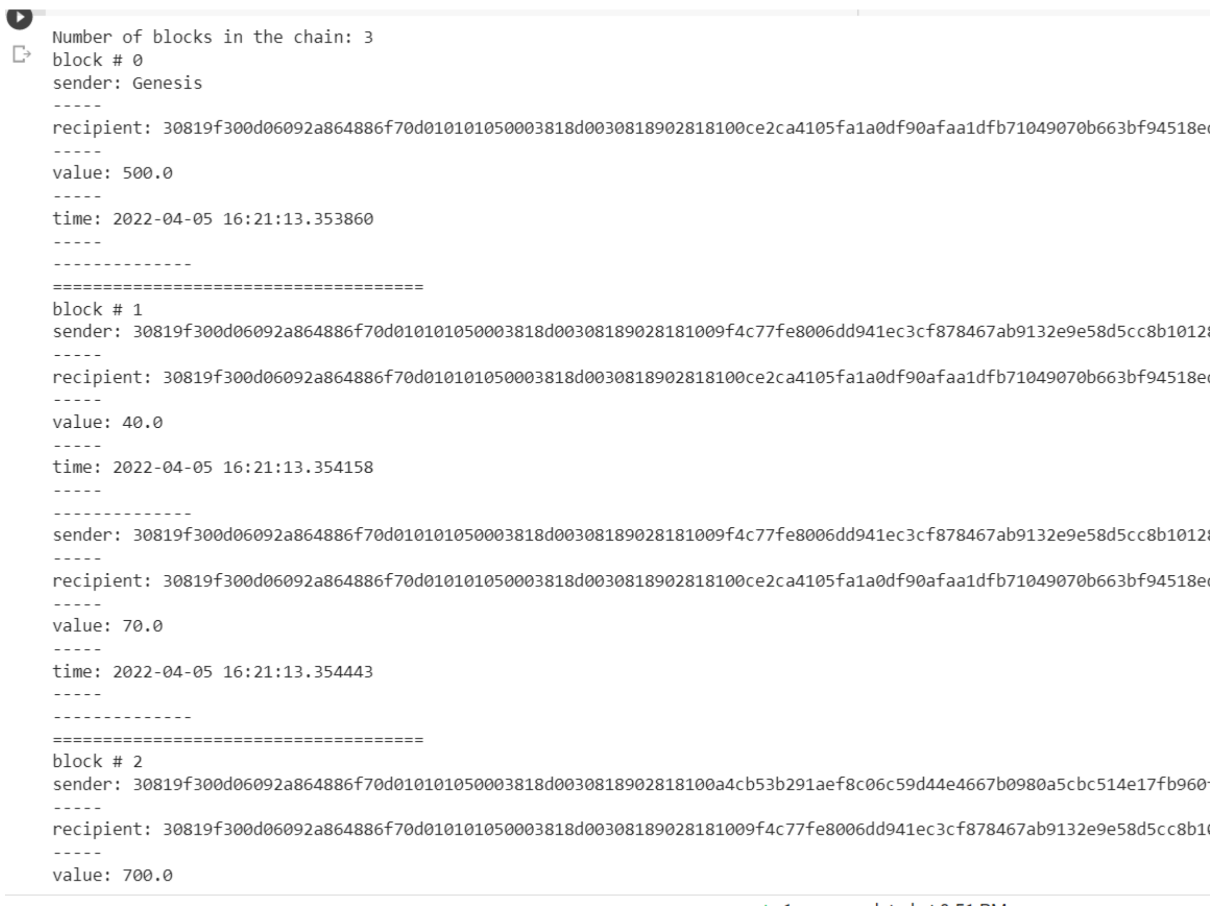
# following imports are required by PKI

Input:

```
import hashlib
import random
import string
import json
import binascii
import numpy as np
import pandas as pd
import pylab as pl
import logging
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

class Client:
  def __init__(self):
    random = Random.new().read
    self._private_key = RSA.generate(1024, random)
    self._public_key = self._private_key.publickey()
    self._signer = PKCS1_v1_5.new(self._private_key)
  @property
  def identity(self):
    return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

class Transaction:
  def __init__(self, sender, recipient, value):
     self.sender = sender
     self.recipient = recipient
     self.value = value
     self.time = datetime.datetime.now()
  def to_dict(self):
     if self.sender == "Genesis":
         identity = "Genesis"
     else:
         identity = self.sender.identity

     return collections.OrderedDict({
         'sender': identity,
         'recipient': self.recipient,
         'value': self.value,
         'time' : self.time})

  def sign_transaction(self):
     private_key = self.sender._private_key
     signer = PKCS1_v1_5.new(private_key)
     h = SHA.new(str(self.to_dict()).encode('utf8'))
     return binascii.hexlify(signer.sign(h)).decode('ascii')

def display_transaction(transaction):
     #for transaction in transactions:
     dict = transaction.to_dict()
     print ("sender: " + dict['sender'])
     print ('-----')
     print ("recipient: " + dict['recipient'])
     print ('-----')
     print ("value: " + str(dict['value']))
     print ('-----')
     print ("time: " + str(dict['time']))
     print ('-----')

def dump_blockchain (self):
  print ("Number of blocks in the chain: " + str(len (self)))
  for x in range (len(TPCoins)):
    block_temp = TPCoins[x]
    print ("block # " + str(x))
    for transaction in block_temp.verified_transactions:
         display_transaction (transaction)
         print ('--------------')
    print ('=====================================')

class Block:
  def __init__(self):
    self.verified_transactions = []
    self.previous_block_hash = ""
    self.Nonce = ""

def sha256(message):
    return hashlib.sha256(message.encode('ascii')).hexdigest()

def mine(message, difficulty=1):
  assert difficulty >= 1
  #if(difficulty <1):
  #       return
  #'1'*3=> '111'
  prefix = '1' * difficulty
  for i in range(1000):
    digest = sha256(str(hash(message)) + str(i))
    if digest.startswith(prefix):
      return i #i= nonce value


Dinesh = Client()
Ramesh =Client()
Vikas =Client()
t0 = Transaction (
  "Genesis",
  Dinesh.identity,
  500.0
)

t1 = Transaction (
  Ramesh,
  Dinesh.identity,
  40.0
)
t2 = Transaction (
  Ramesh,
  Dinesh.identity,
  70.0
)
t3 = Transaction (
  Vikas,
  Ramesh.identity,
  700.0
)
#blockchain
TPCoins = []
```

```
block0 = Block()
block0.previous_block_hash = None
Nonce = None
block0.verified_transactions.append (t0)
digest = hash (block0)
last_block_hash = digest #last_block_hash it is hash of block0
TPCoins.append (block0)

block1 = Block()
block1.previous_block_hash = last_block_hash
block1.verified_transactions.append (t1)
block1.verified_transactions.append (t2)
block1.Nonce=mine (block1, 2)
digest = hash (block1)
last_block_hash = digest
TPCoins.append (block1)


block2 = Block()
block2.previous_block_hash = last_block_hash
block2.verified_transactions.append (t3)
Nonce = mine (block2, 2)
block2.Nonce=mine (block2, 2)
digest = hash (block2)
last_block_hash = digest
TPCoins.append (block2)


dump_blockchain(TPCoins)
```

Output:-



```
Number of blocks in the chain: 3
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100ce2ca4105fa1a0df90afaa1dfb71049070b663bf94518e
-----
value: 500.0
-----
time: 2022-04-05 16:21:13.353860
-----
--------------
=====================================
block # 1
sender: 30819f300d06092a864886f70d010101050003818d00308189028181009f4c77fe8006dd941ec3cf878467ab9132e9e58d5cc8b1012
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100ce2ca4105fa1a0df90afaa1dfb71049070b663bf94518e
-----
value: 40.0
-----
time: 2022-04-05 16:21:13.354158
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d00308189028181009f4c77fe8006dd941ec3cf878467ab9132e9e58d5cc8b1012
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100ce2ca4105fa1a0df90afaa1dfb71049070b663bf94518e
-----
value: 70.0
-----
time: 2022-04-05 16:21:13.354443
-----
--------------
=====================================
block # 2
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100a4cb53b291aef8c06c59d44e4667b0980a5cbc514e17fb960
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181009f4c77fe8006dd941ec3cf878467ab9132e9e58d5cc8b1
-----
value: 700.0
```

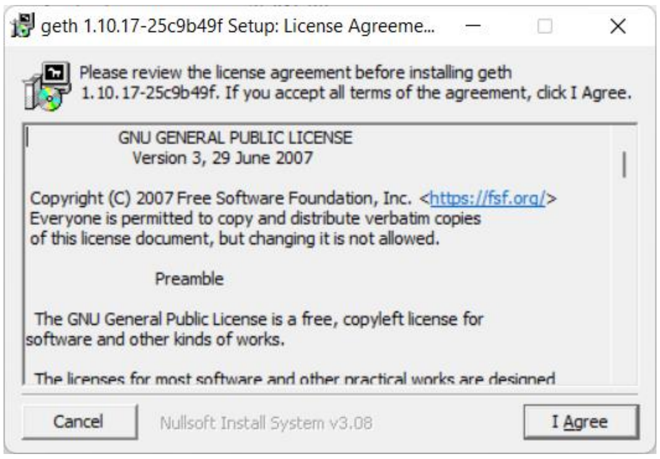
Practical – 2

Q. Install and configure Go Ethereum and Mist Browser

Let's get started with the first tool we need to know and to install, that's Geth.! Go to the Link Below and download Geth -> https://geth.ethereum.org/downloads/

Install Geth:

Step 1: Click on I agree Button



geth 1.10.17-25c9b49f Setup: License Agreeme...

Please review the license agreement before installing geth 1.10.17-25c9b49f. If you accept all terms of the agreement, click I Agree.

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

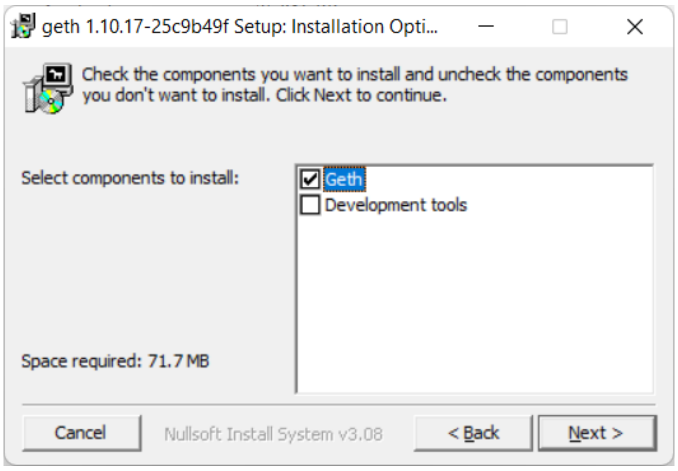
Preamble

The GNU General Public License is a free, copyleft license for
software and other kinds of works.

The licenses for most software and other practical works are designed

Cancel | Nullsoft Install System v3.08 | I Agree

Step 2:



geth 1.10.17-25c9b49f Setup: Installation Opti...

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

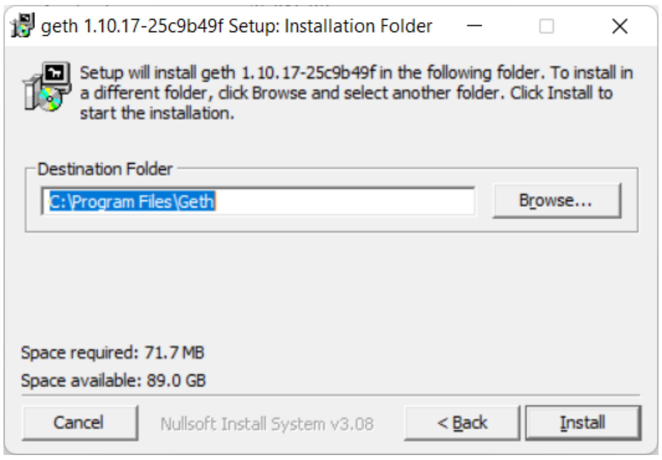
Select components to install:

- [x] Geth
- [ ] Development tools

Space required: 71.7 MB

Cancel | Nullsoft Install System v3.08 | < Back | Next >

Step 3:



geth 1.10.17-25c9b49f Setup: Installation Folder

Setup will install geth 1.10.17-25c9b49f in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

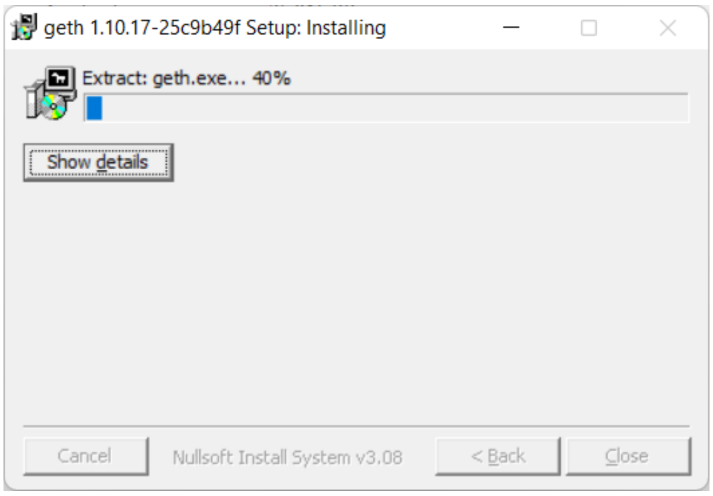
Destination Folder

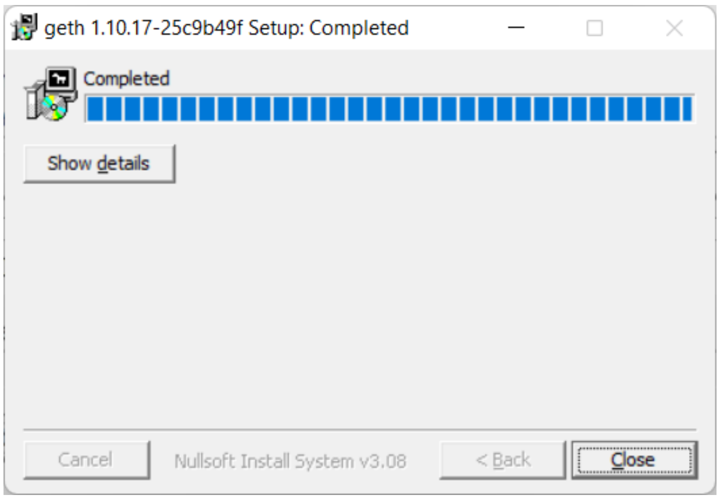
C:\Program Files\Geth | Browse...

Space required: 71.7 MB
Space available: 89.0 GB

Cancel | Nullsoft Install System v3.08 | < Back | Install

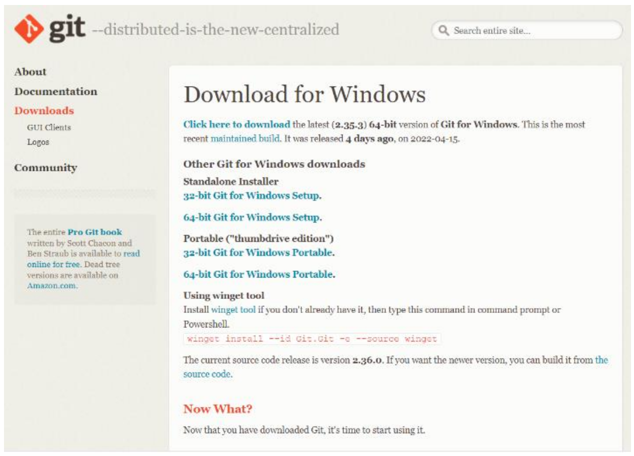
Step 4:


geth 1.10.17-25c9b49f Setup: Installing
Extract: geth.exe... 40%
Show details
Cancel
Nullsoft Install System v3.08
< Back
Close


geth 1.10.17-25c9b49f Setup: Completed
Completed
Show details
Cancel
Nullsoft Install System v3.08
< Back
Close

Download and install Gitbash -> https://git-scm.com/downloads

step 2: click on window



The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Download for Windows

Click here to download the latest (2.35.3) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 4 days ago, on 2022-04-15.

Other Git for Windows downloads

Standalone Installer
32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Portable ("thumbdrive edition")
32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

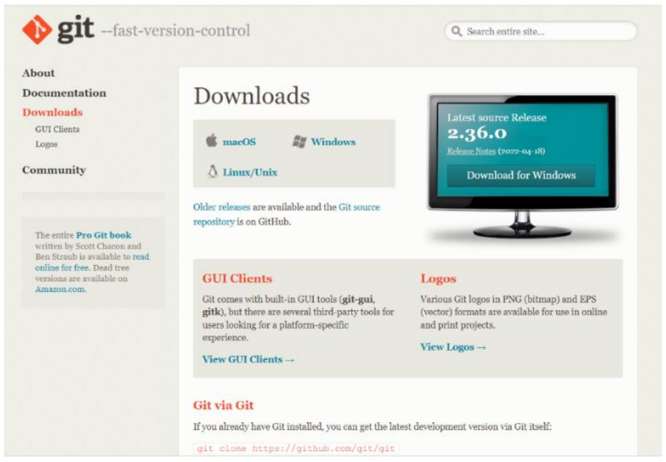
Using winget tool
Install winget tool if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version 2.36.0. If you want the newer version, you can build it from the source code.

Now What?

Now that you have downloaded Git, it's time to start using it.



Downloads

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Older releases are available and the Git source repository is on GitHub.

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →

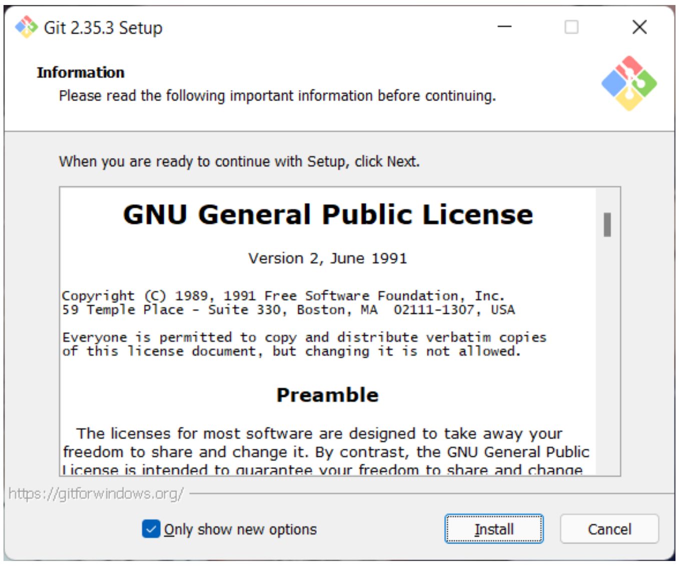
Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

Installation Steps:

1.



Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

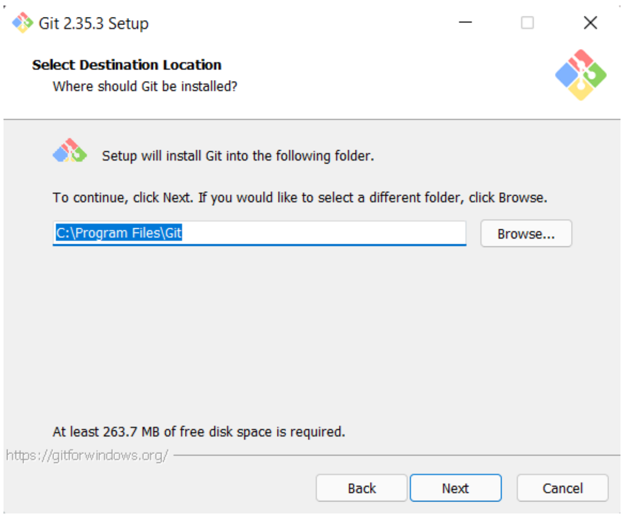
```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://gitforwindows.org/

2.



Select Destination Location

Where should Git be installed?

Setup will install Git into the following folder.

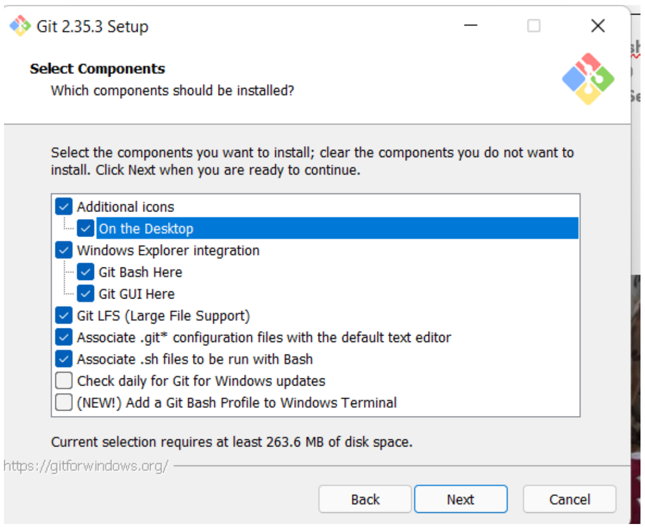
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 263.7 MB of free disk space is required.

https://gitforwindows.org/

3.


Git 2.35.3 Setup

Select Components

Which components should be installed?

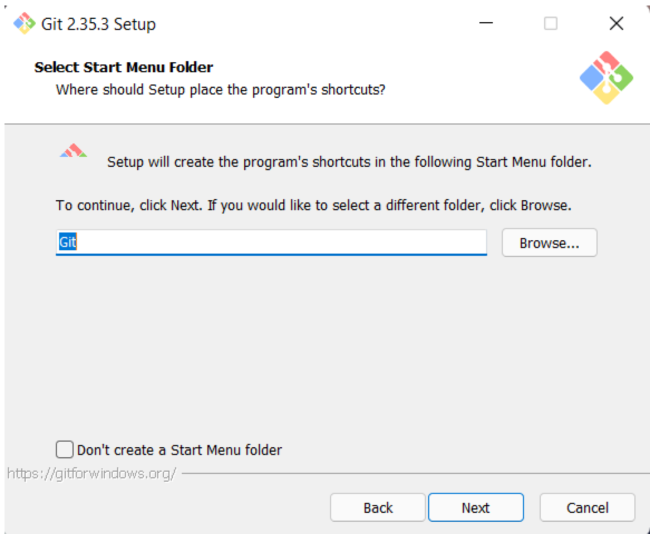
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Additional icons
  - [✓] On the Desktop
- [✓] Windows Explorer integration
  - [✓] Git Bash Here
  - [✓] Git GUI Here
- [✓] Git LFS (Large File Support)
- [✓] Associate .git* configuration files with the default text editor
- [✓] Associate .sh files to be run with Bash
- [ ] Check daily for Git for Windows updates
- [ ] (NEW!) Add a Git Bash Profile to Windows Terminal

Current selection requires at least 263.6 MB of disk space.

https://gitforwindows.org/

4.


Git 2.35.3 Setup

Select Start Menu Folder

Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

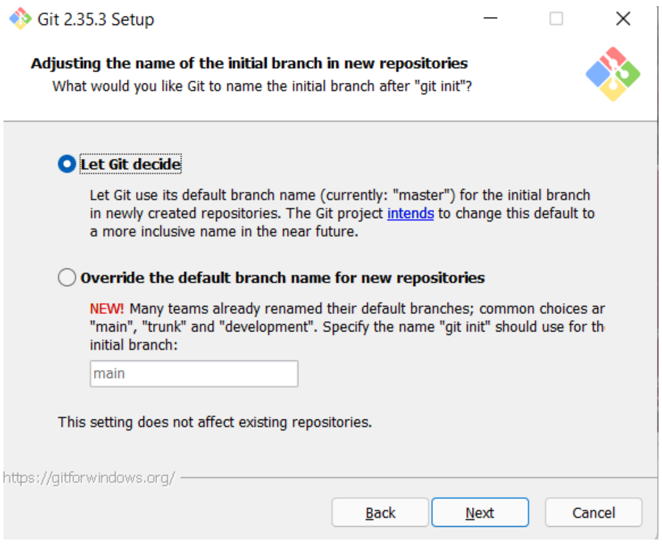
To continue, click Next. If you would like to select a different folder, click Browse.

Git

[ ] Don't create a Start Menu folder

https://gitforwindows.org/

5.


Git 2.35.3 Setup

Adjusting the name of the initial branch in new repositories

What would you like Git to name the initial branch after "git init"?

(●) **Let Git decide**

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project intends to change this default to a more inclusive name in the near future.

( ) **Override the default branch name for new repositories**

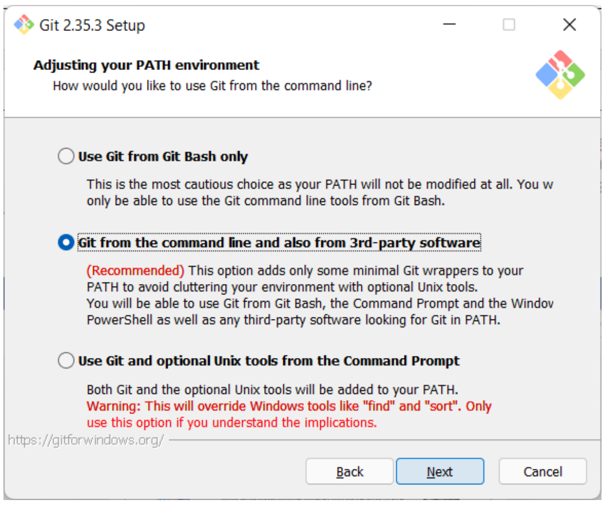
NEW! Many teams already renamed their default branches; common choices ar "main", "trunk" and "development". Specify the name "git init" should use for th initial branch:

main

This setting does not affect existing repositories.

https://gitforwindows.org/

6.


Git 2.35.3 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

This is the most cautious choice as your PATH will not be modified at all. You w
only be able to use the Git command line tools from Git Bash.

● **Git from the command line and also from 3rd-party software**

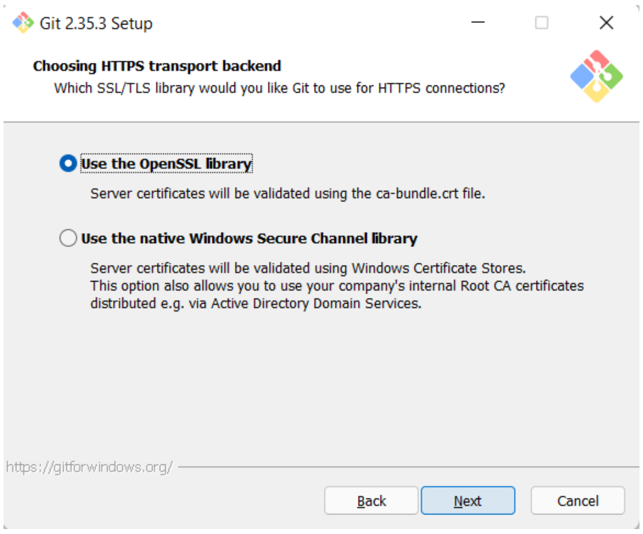
(Recommended) This option adds only some minimal Git wrappers to your
PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windov
PowerShell as well as any third-party software looking for Git in PATH.

○ **Use Git and optional Unix tools from the Command Prompt**

Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only
use this option if you understand the implications.

https://gitforwindows.org/

7.


Git 2.35.3 Setup

Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?

● **Use the OpenSSL library**

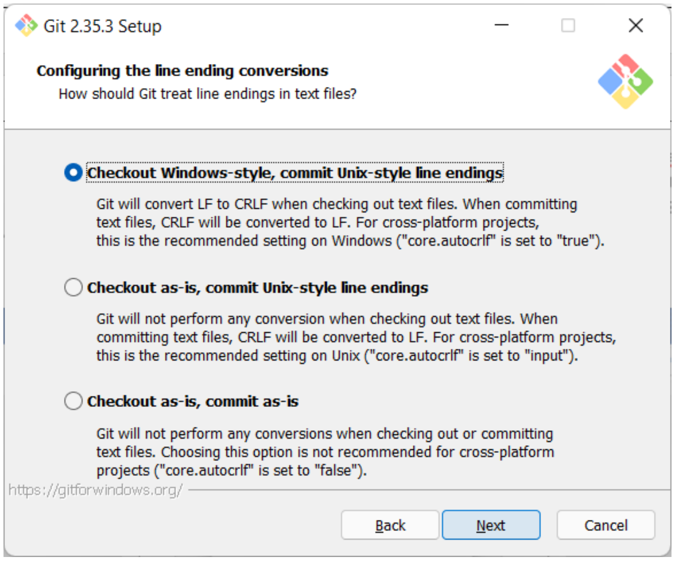
Server certificates will be validated using the ca-bundle.crt file.

○ **Use the native Windows Secure Channel library**

Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates
distributed e.g. via Active Directory Domain Services.

https://gitforwindows.org/

8.


Git 2.35.3 Setup

Configuring the line ending conversions

How should Git treat line endings in text files?

● **Checkout Windows-style, commit Unix-style line endings**

Git will convert LF to CRLF when checking out text files. When committing
text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Windows ("core.autocrlf" is set to "true").

○ **Checkout as-is, commit Unix-style line endings**

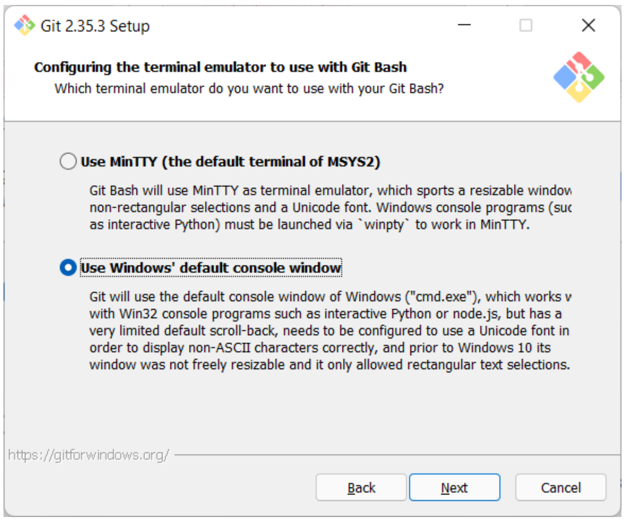
Git will not perform any conversion when checking out text files. When
committing text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Unix ("core.autocrlf" is set to "input").

○ **Checkout as-is, commit as-is**

Git will not perform any conversions when checking out or committing
text files. Choosing this option is not recommended for cross-platform
projects ("core.autocrlf" is set to "false").

https://gitforwindows.org/

9.



Git 2.35.3 Setup

Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?

( ) **Use MinTTY (the default terminal of MSYS2)**

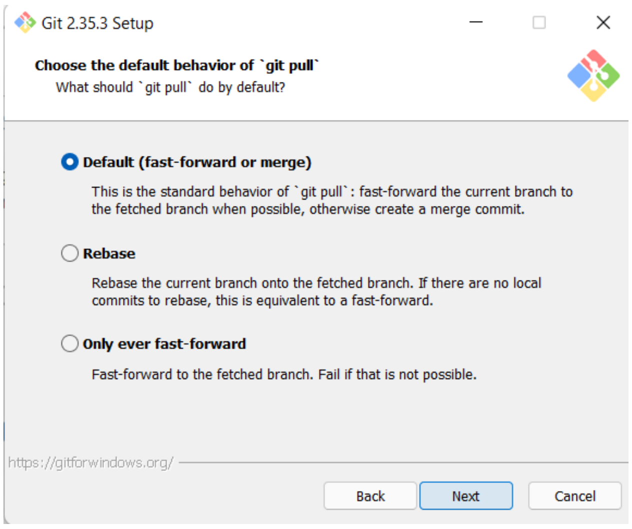
Git Bash will use MinTTY as terminal emulator, which sports a resizable window non-rectangular selections and a Unicode font. Windows console programs (suc as interactive Python) must be launched via `winpty` to work in MinTTY.

(●) **Use Windows' default console window**

Git will use the default console window of Windows ("cmd.exe"), which works v with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

https://gitforwindows.org/

10.



Git 2.35.3 Setup

Choose the default behavior of `git pull`

What should `git pull` do by default?

(●) **Default (fast-forward or merge)**

This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

( ) **Rebase**

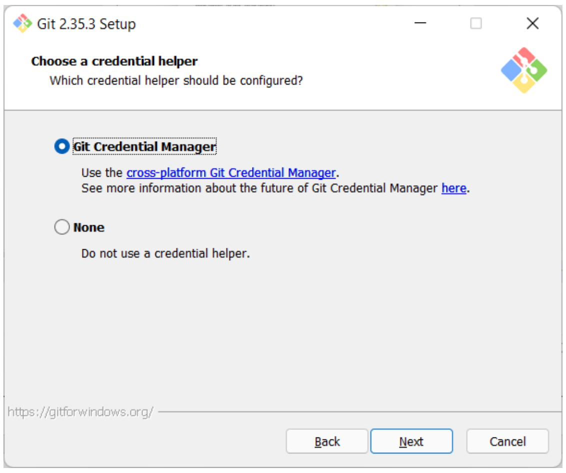
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

( ) **Only ever fast-forward**

Fast-forward to the fetched branch. Fail if that is not possible.

https://gitforwindows.org/

11.



Git 2.35.3 Setup

Choose a credential helper

Which credential helper should be configured?

(●) **Git Credential Manager**

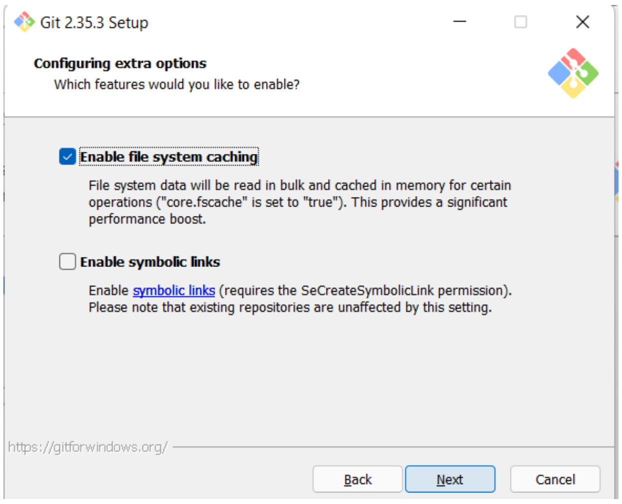
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.

( ) **None**

Do not use a credential helper.

https://gitforwindows.org/

12.



Git 2.35.3 Setup

Configuring extra options

Which features would you like to enable?

[✓] **Enable file system caching**

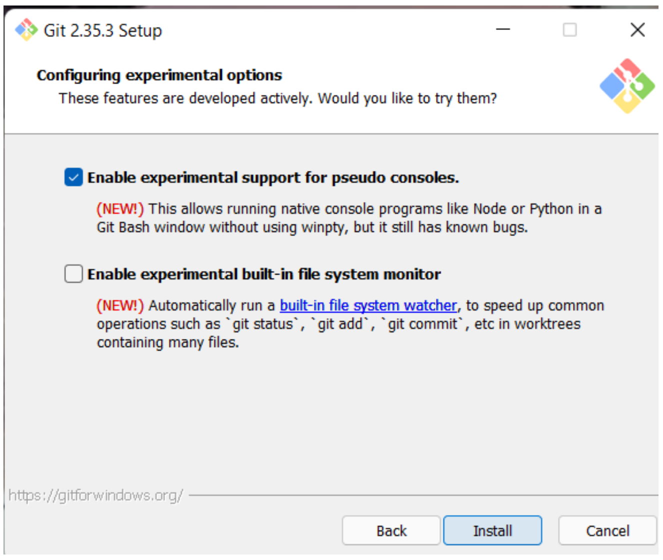
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

[ ] **Enable symbolic links**

Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.

https://gitforwindows.org/

13.



Git 2.35.3 Setup

Configuring experimental options

These features are developed actively. Would you like to try them?

[✓] **Enable experimental support for pseudo consoles.**

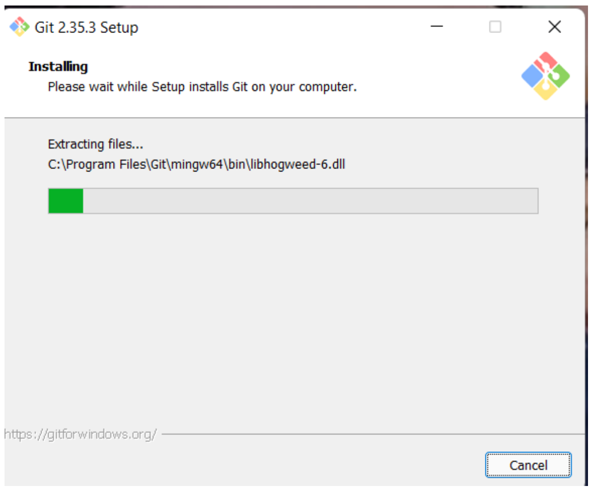
(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.

[ ] **Enable experimental built-in file system monitor**

(NEW!) Automatically run a built-in file system watcher, to speed up common operations such as `git status`, `git add`, `git commit`, etc in worktrees containing many files.

https://gitforwindows.org/

14.



Git 2.35.3 Setup

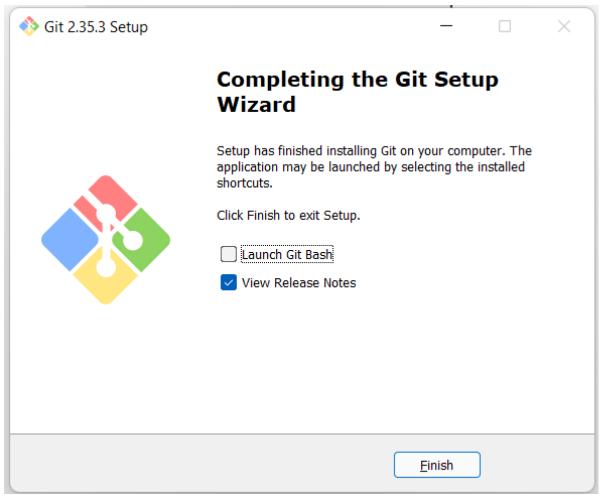
Installing

Please wait while Setup installs Git on your computer.

Extracting files...
C:\Program Files\Git\mingw64\bin\libhogweed-6.dll

https://gitforwindows.org/

15.


Git 2.35.3 Setup

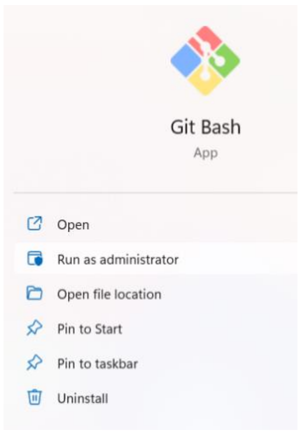
Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

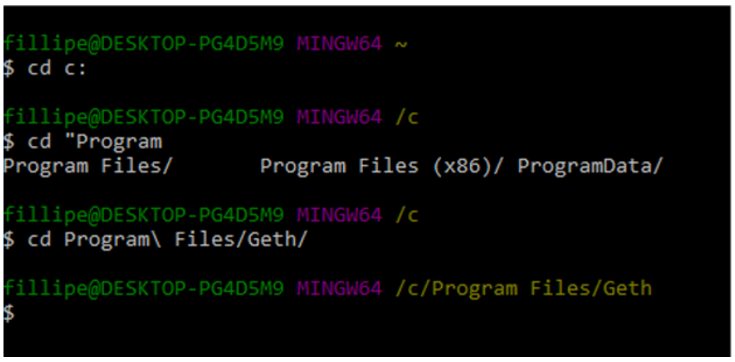
Click Finish to exit Setup.

[ ] Launch Git Bash
[✓] View Release Notes

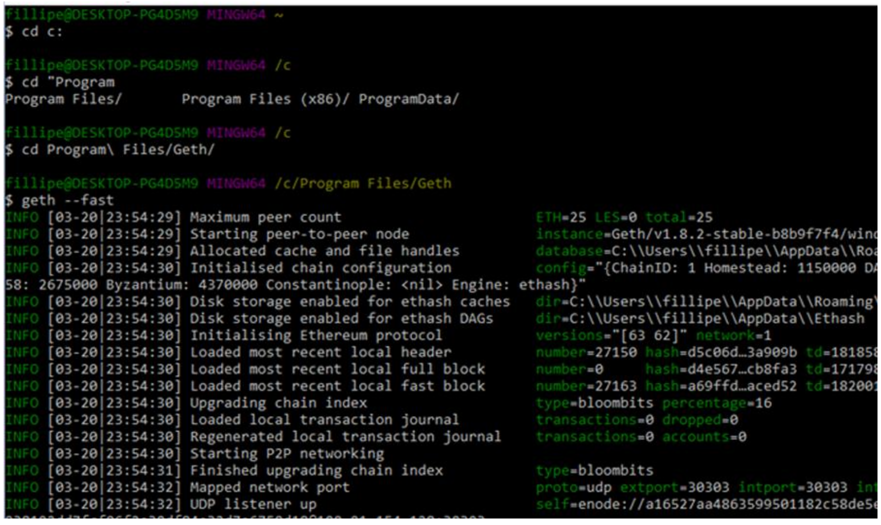
16.



Go to Geth installation Folder:

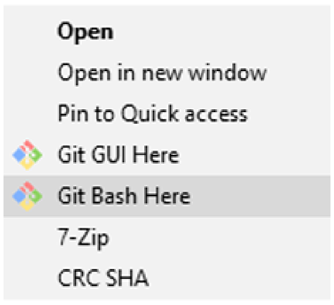

```
fillipe@DESKTOP-PG4D5M9 MINGW64 ~
$ cd c:

fillipe@DESKTOP-PG4D5M9 MINGW64 /c
$ cd "Program
Program Files/       Program Files (x86)/ ProgramData/

fillipe@DESKTOP-PG4D5M9 MINGW64 /c
$ cd Program\ Files/Geth/

fillipe@DESKTOP-PG4D5M9 MINGW64 /c/Program Files/Geth
$
```

Sync with Ethereum Network with the command get --fast. Take a time to relax, it will take a while, some GBs needs to be downloaded.

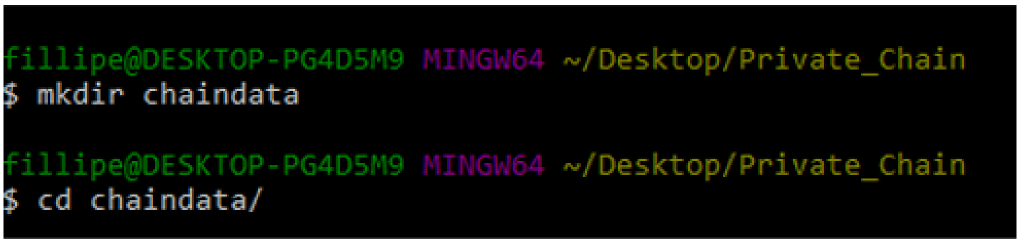


```
fillipe@DESKTOP-PG4D5M9 MINGW64 ~
$ cd c:

fillipe@DESKTOP-PG4D5M9 MINGW64 /c
$ cd "Program
Program Files/       Program Files (x86)/ ProgramData/

fillipe@DESKTOP-PG4D5M9 MINGW64 /c
$ cd Program\ Files/Geth/

fillipe@DESKTOP-PG4D5M9 MINGW64 /c/Program Files/Geth
$ geth --fast
INFO [03-20|23:54:29] Maximum peer count                       ETH=25 LES=0 total=25
INFO [03-20|23:54:29] Starting peer-to-peer node               instance=Geth/v1.8.2-stable-b8b9f7f4/win
INFO [03-20|23:54:29] Allocated cache and file handles         database=C:\\Users\\fillipe\\AppData\\Ro
INFO [03-20|23:54:30] Initialised chain configuration          config="{ChainID: 1 Homestead: 1150000 D
58: 2675000 Byzantium: 4370000 Constantinople: <nil> Engine: ethash}"
INFO [03-20|23:54:30] Disk storage enabled for ethash caches   dir=C:\\Users\\fillipe\\AppData\\Roaming
INFO [03-20|23:54:30] Disk storage enabled for ethash DAGs     dir=C:\\Users\\fillipe\\AppData\\Ethash
INFO [03-20|23:54:30] Initialising Ethereum protocol           versions="[63 62]" network=1
INFO [03-20|23:54:30] Loaded most recent local header          number=27150 hash=d5c06d…3a909b td=18185
INFO [03-20|23:54:30] Loaded most recent local full block      number=0     hash=d4e567…cb8fa3 td=17179
INFO [03-20|23:54:30] Loaded most recent local fast block      number=27163 hash=a69ffd…aced52 td=18200
INFO [03-20|23:54:30] Upgrading chain index                    type=bloombits percentage=16
INFO [03-20|23:54:30] Loaded local transaction journal         transactions=0 dropped=0
INFO [03-20|23:54:30] Regenerated local transaction journal    transactions=0 accounts=0
INFO [03-20|23:54:30] Starting P2P networking
INFO [03-20|23:54:31] Finished upgrading chain index           type=bloombits
INFO [03-20|23:54:32] Mapped network port                      proto=udp extport=30303 intport=30303 in
INFO [03-20|23:54:32] UDP listener up                          self=enode://a16527aa4863599501182c58de5
```

Once synced with Ethereum network, create a new folder on your desktop called Private_Chain where save your new Blockchain and open a Gitbash terminal from there:



Inside the folder Private_Chain create a new folder to store the Blockchain blocks called chaindata.

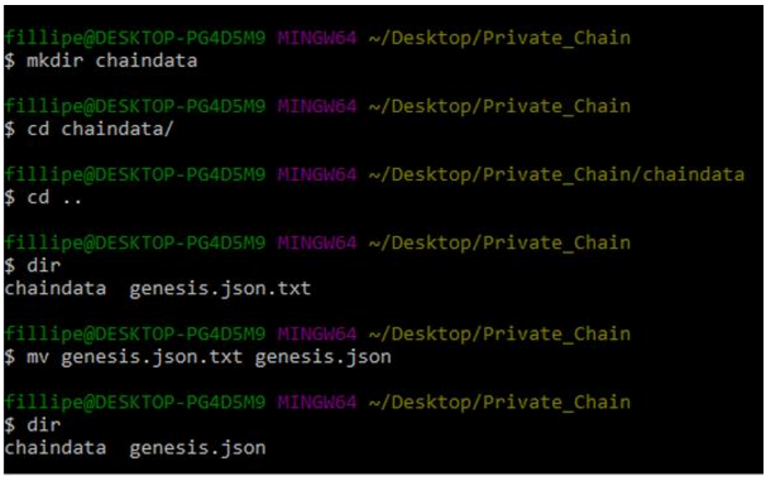


```
fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ mkdir chaindata

fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ cd chaindata/
```

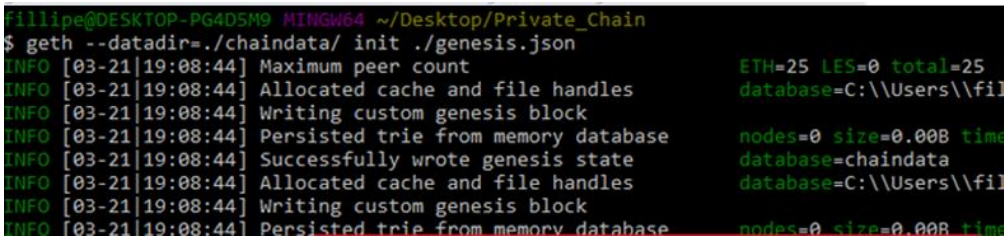
Now we're ready to deploy our own blockchain.

```
{
  "coinbase"   : "0x0000000000000000000000000000000000000001",
  "difficulty" : "0x20000",
  "extraData"  : "",
  "gasLimit"   : "0x2fefd8",
  "nonce"      : "0x0000000000000042",
  "mixhash"    :
"0x00000000000000000000000000000000000000000000000000000000000
00000",
  "parentHash" :
"0x00000000000000000000000000000000000000000000000000000000000
00000",
  "timestamp"  : "0x00",
  "alloc": {},
  "config": {
        "chainId": 15,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    }
}
```

After that rename it using Gitbash since it probably **has .txt extension**.



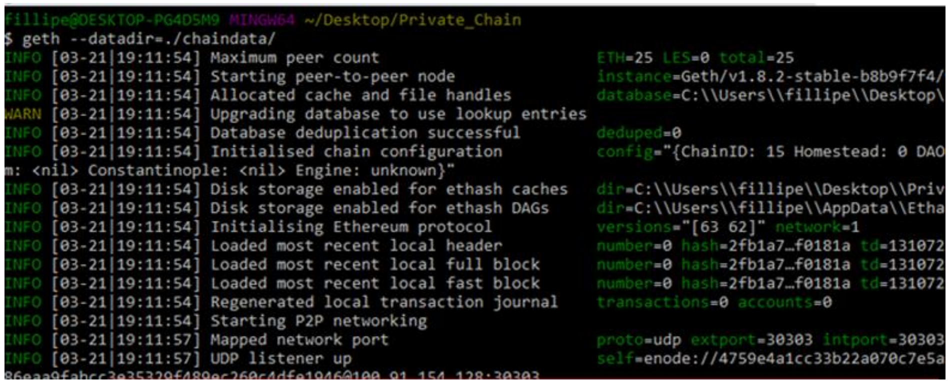
```
fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ mkdir chaindata

fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ cd chaindata/

fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain/chaindata
$ cd ..

fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ dir
chaindata  genesis.json.txt

fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ mv genesis.json.txt genesis.json

fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ dir
chaindata  genesis.json
```



```
fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ geth --datadir=./chaindata/ init ./genesis.json
INFO [03-21|19:08:44] Maximum peer count                       ETH=25 LES=0 total=25
INFO [03-21|19:08:44] Allocated cache and file handles         database=C:\\Users\\fil
INFO [03-21|19:08:44] Writing custom genesis block
INFO [03-21|19:08:44] Persisted trie from memory database      nodes=0 size=0.00B time
INFO [03-21|19:08:44] Successfully wrote genesis state         database=chaindata
INFO [03-21|19:08:44] Allocated cache and file handles         database=C:\\Users\\fil
INFO [03-21|19:08:44] Writing custom genesis block
INFO [03-21|19:08:44] Persisted trie from memory database      nodes=0 size=0.00B time
```

Here, successfully wrote genesis state.

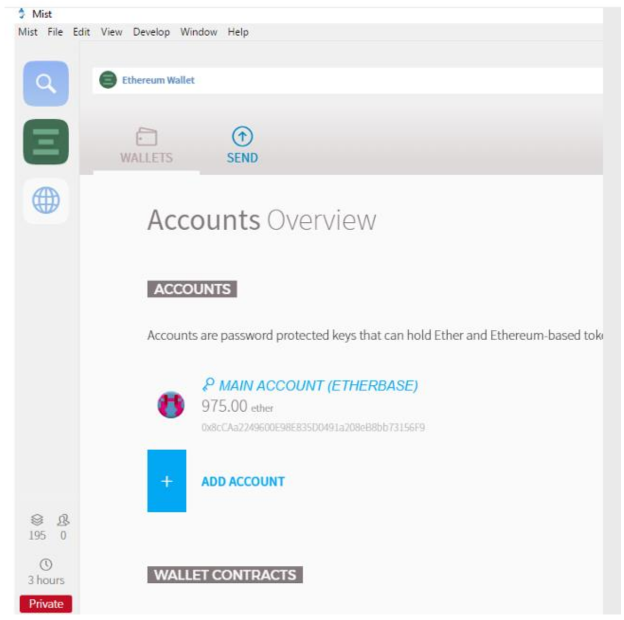
If you get the above output "Successfully wrote genesis state" means that our first block is written on our private Blockchain..



```
fillipe@DESKTOP-PG4D5M9 MINGW64 ~/Desktop/Private_Chain
$ geth --datadir=./chaindata/
INFO [03-21|19:11:54] Maximum peer count                       ETH=25 LES=0 total=25
INFO [03-21|19:11:54] Starting peer-to-peer node               instance=Geth/v1.8.2-stable-b8b9f7f4/
INFO [03-21|19:11:54] Allocated cache and file handles         database=C:\\Users\\fillipe\\Desktop\
WARN [03-21|19:11:54] Upgrading database to use lookup entries
INFO [03-21|19:11:54] Database deduplication successful        deduped=0
INFO [03-21|19:11:54] Initialised chain configuration          config="{ChainID: 15 Homestead: 0 DAO
m: <nil> Constantinople: <nil> Engine: unknown}"
INFO [03-21|19:11:54] Disk storage enabled for ethash caches   dir=C:\\Users\\fillipe\\Desktop\\Priv
INFO [03-21|19:11:54] Disk storage enabled for ethash DAGs     dir=C:\\Users\\fillipe\\AppData\\Etha
INFO [03-21|19:11:54] Initialising Ethereum protocol           versions="[63 62]" network=1
INFO [03-21|19:11:54] Loaded most recent local header          number=0 hash=2fb1a7…f0181a td=131072
INFO [03-21|19:11:54] Loaded most recent local full block      number=0 hash=2fb1a7…f0181a td=131072
INFO [03-21|19:11:54] Loaded most recent local fast block      number=0 hash=2fb1a7…f0181a td=131072
INFO [03-21|19:11:54] Regenerated local transaction journal    transactions=0 accounts=0
INFO [03-21|19:11:54] Starting P2P networking
INFO [03-21|19:11:57] Mapped network port                      proto=udp extport=30303 intport=30303
INFO [03-21|19:11:57] UDP listener up                          self=enode://4759e4a1cc33b22a070c7e5a
86eaa9fabcc3e35329f489ec260c4dfe1946@100.91.154.128:30303
```

Mist Browser/Wallet

Open mist application:



Accounts Overview

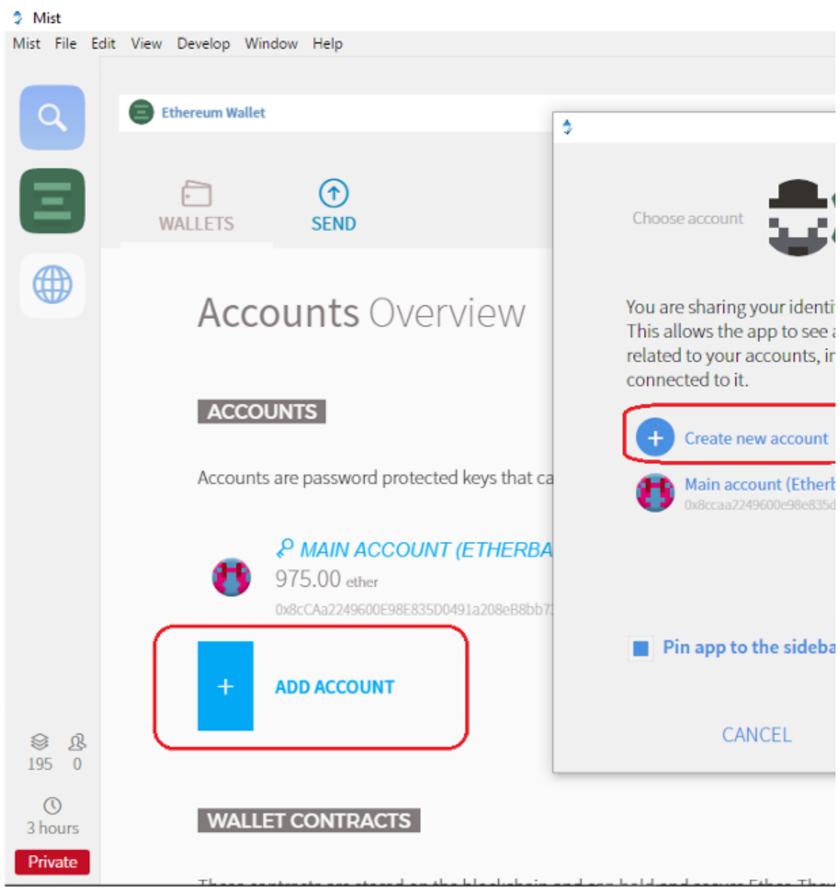
ACCOUNTS

Accounts are password protected keys that can hold Ether and Ethereum-based tok

MAIN ACCOUNT (ETHERBASE)
975.00 ether
0x8cCAa2249600E98E835D0491a208eB8bb73156F9

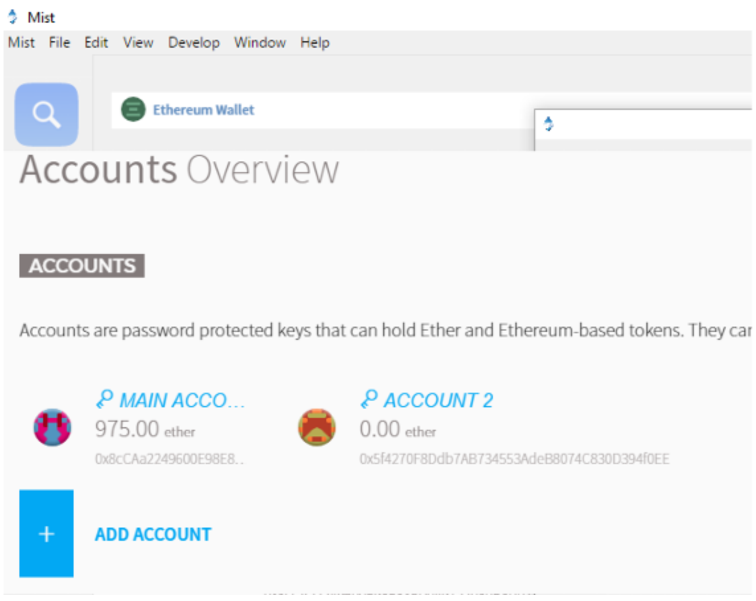
ADD ACCOUNT

WALLET CONTRACTS

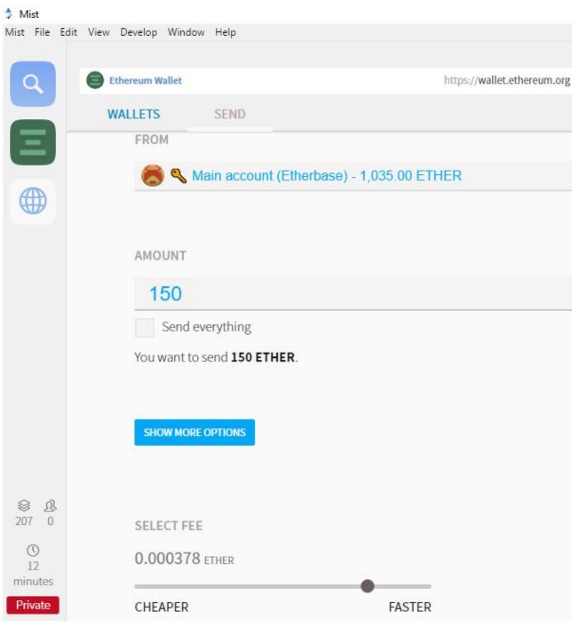


Accounts are password protected keys that ca

You are sharing your identi
This allows the app to see a
related to your accounts, ir
connected to it.



Accounts are password protected keys that can hold Ether and Ethereum-based tokens. They car



FROM

Main account (Etherbase) - 1,035.00 ETHER

AMOUNT

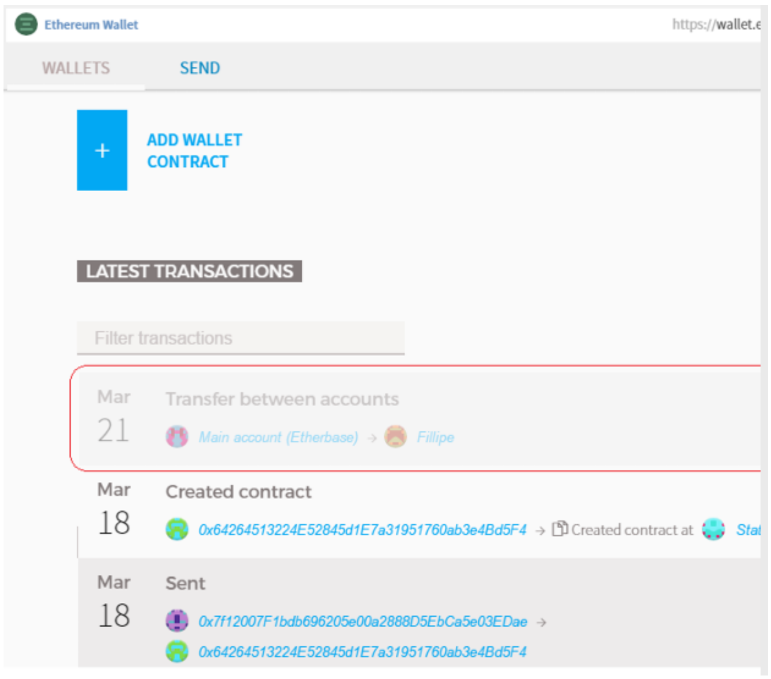
150

Send everything

You want to send **150 ETHER**.

SELECT FEE

0.000378 ETHER



Mar 21 Transfer between accounts
Main account (Etherbase) → Fillipe

Mar 18 Created contract
0x64264513224E52845d1E7a31951760ab3e4Bd5F4 → Created contract at Stat

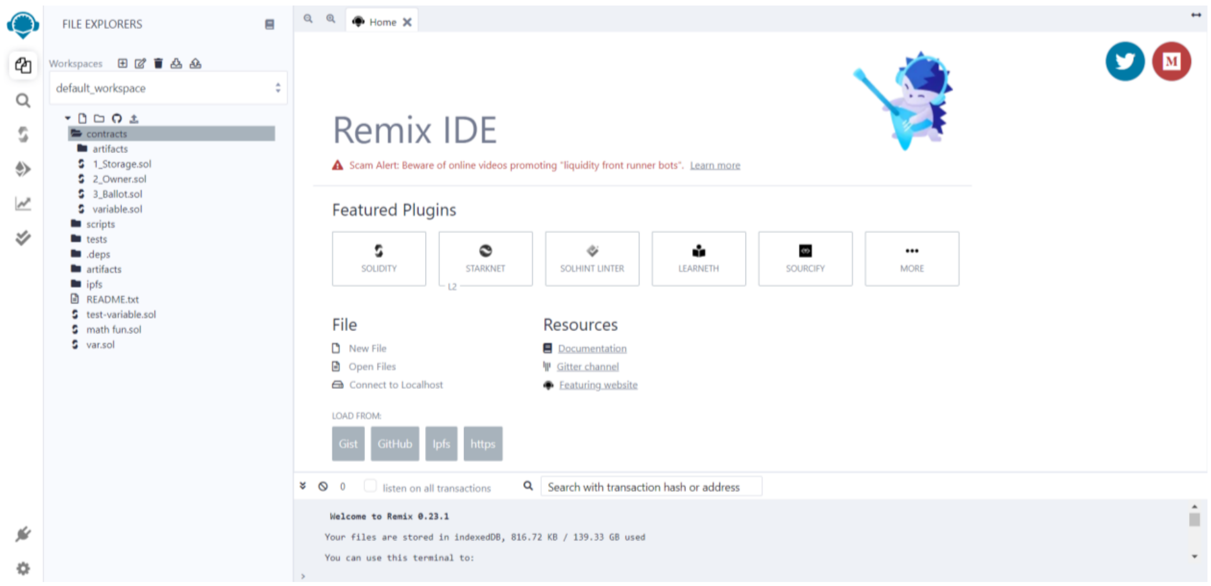
Mar 18 Sent
0x7f12007F1bdb696205e00a2888D5EbCa5e03EDae →
0x64264513224E52845d1E7a31951760ab3e4Bd5F4

Practical – 3

Q. Implement and demonstrate the use of following in solidity.

All solidity program performs on Remix – Ethereum IDE.

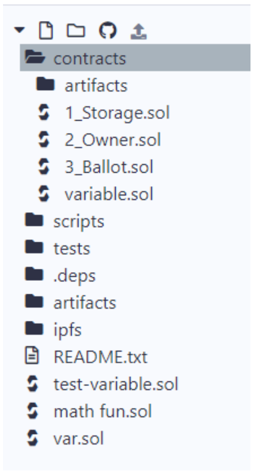
Link - https://remix.ethereum.org/



```
Welcome to Remix 0.23.1
Your files are stored in indexedDB, 816.72 KB / 139.33 GB used
You can use this terminal to:
```

Steps

1. **Open Remix – Ethereum IDE.**
2. **In that create new file in contracts folder i.e show in left hand side**



3. **Compile program**

4. **Deploy the program**

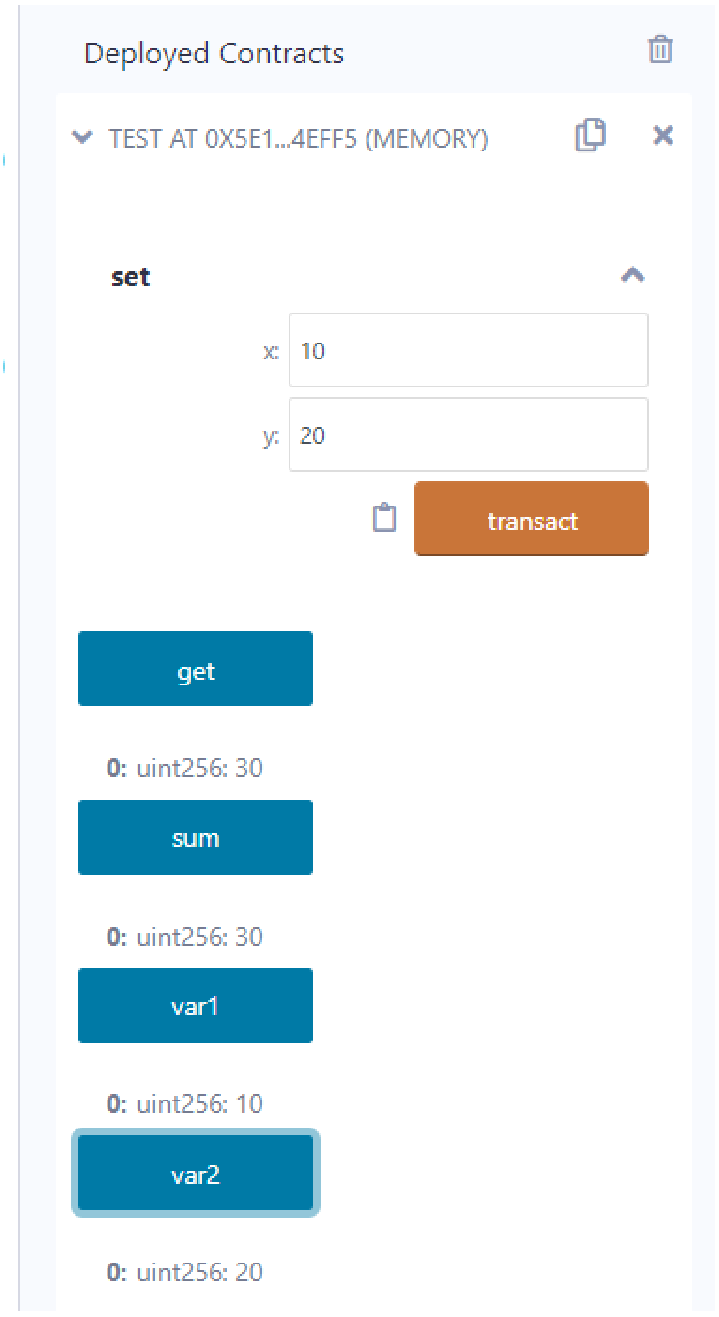
A]

1. <u>Variables:</u>

Code:

```
// Solidity program to  // demonstrate how to // use of variables
//SPDX-License-Identifier: GPL-3.0
pragma solidity >= 0.4.16 < 0.7.0;
// Defining a contract
contract Test
{
    // Declaring state variables
    uint public var1;
    uint public var2;
    uint public sum;
    // Defining public function
    // that sets the value of
    // the state variable
    function set(uint x, uint y) public
    {
        var1 =x;
        var2=y;
        sum=var1+var2;
    }
    // Defining function to
    // print the sum of
    // state variables
    function get(
    ) public view returns (uint) {
        return sum;
    }
}
```

<u>Output:</u>



x: 10
y: 20

0: uint256: 30

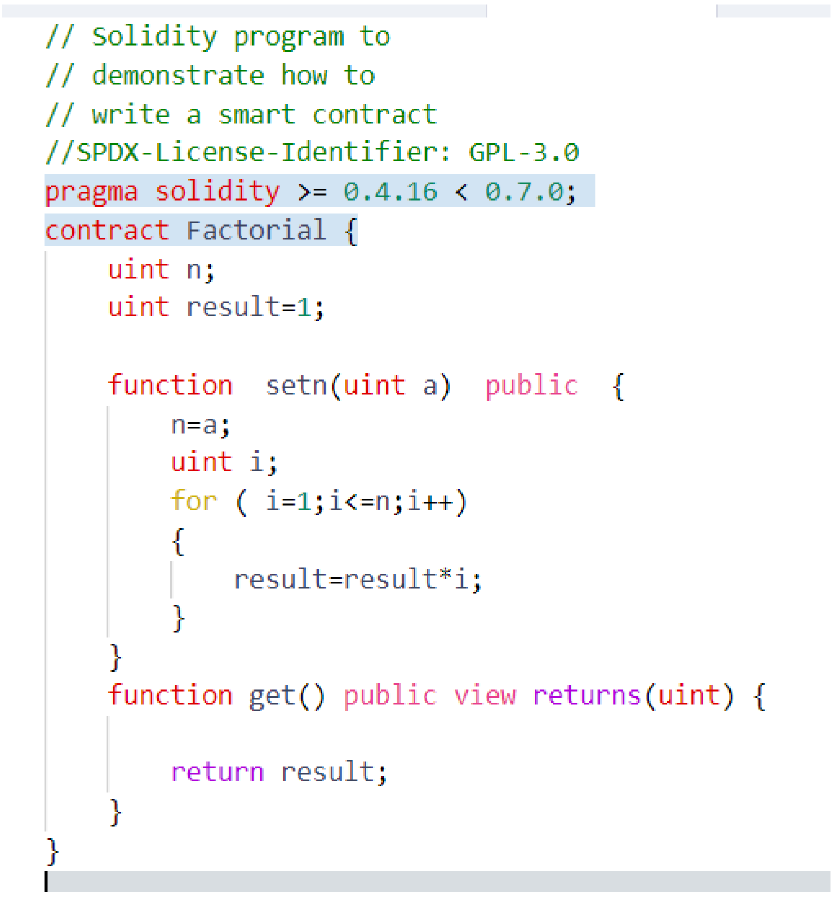
0: uint256: 30

0: uint256: 10

0: uint256: 20

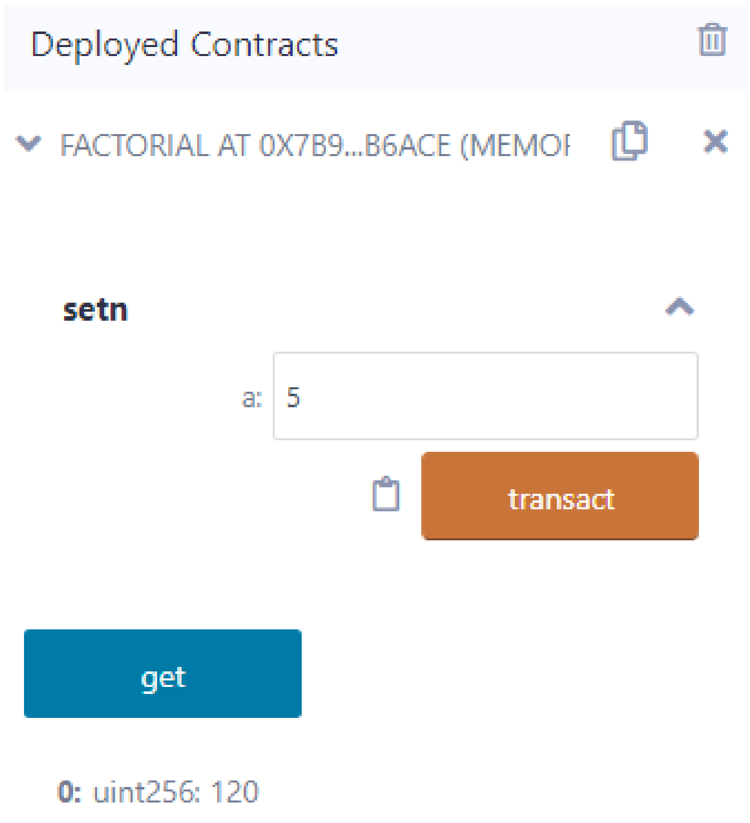
2. Loops:

Code:



```
// Solidity program to
// demonstrate how to
// write a smart contract
//SPDX-License-Identifier: GPL-3.0
pragma solidity >= 0.4.16 < 0.7.0;
contract Factorial {
    uint n;
    uint result=1;

    function  setn(uint a)  public  {
        n=a;
        uint i;
        for ( i=1;i<=n;i++)
        {
            result=result*i;
        }
    }
    function get() public view returns(uint) {

        return result;
    }
}
```

Output:



Deployed Contracts

FACTORIAL AT 0X7B9...B6ACE (MEMOI

setn

a: 5

transact

get

0: uint256: 120

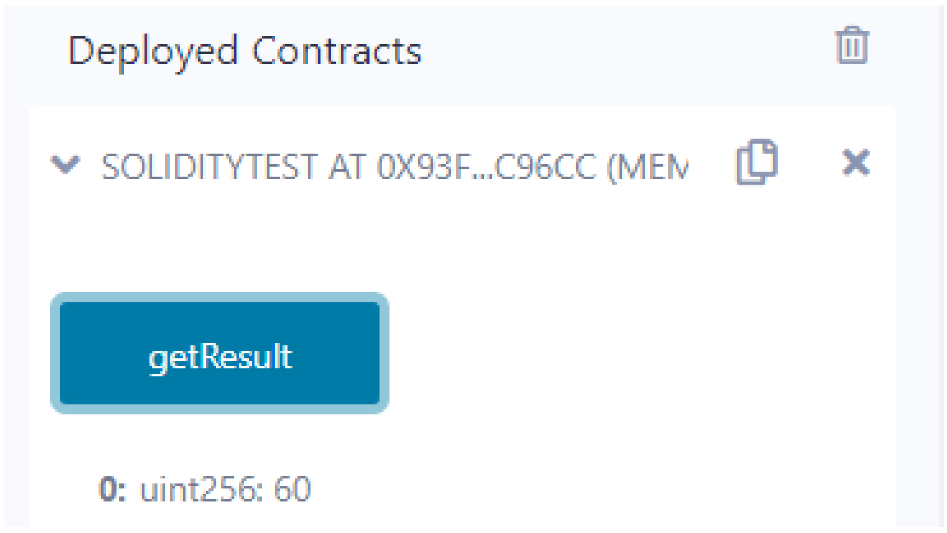
3. Decision Making:

If_else statement:

Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity >0.5.0;
contract SolidityTest {
    constructor() public {

    }

    function getResult() public pure returns (uint) {
        uint a=100;
        uint b=40;
        uint result;
        if (a > b)
        {
            result=a-b;
        }
        else
        {
            result=b-a;
        }
        return result;
    }

}
```

Output:



```
0: uint256: 60
```

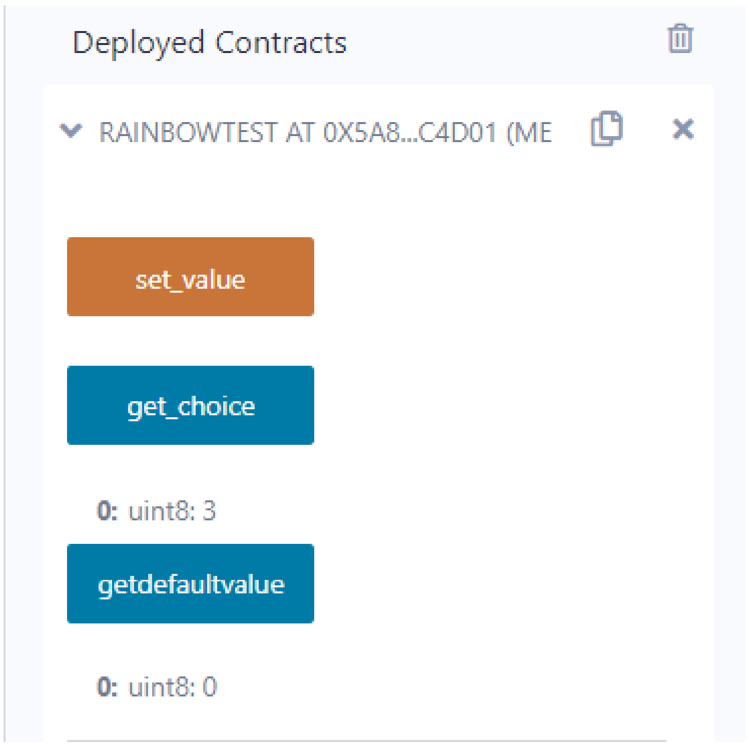
4. Enum:

Code:

```
//Solidity program to demonstrate
// how to use 'enumerator'
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

// Creating a contract
contract rainbowtest {

   // Creating an enumerator
   enum rainb
   {
     Violet,
     Indigo,
     Blue,
     Green,
     Yellow,
     Orange,
     Red
    }

   // Declaring variables of
   // type enumerator
   rainb r1;

   rainb choice;

   // Setting a default value
   rainb constant default_value
     = rainb.Violet;

   // Defining a function to
   // set value of choice
   function set_value() public {
     choice = rainb.Green;
   }

   // Defining a function to
   // return value of choice
   function get_choice(
   ) public view returns (rainb) {
     return choice;
   }

   // Defining function to
   //  return default value
```

```
  function getdefaultvalue(
  ) public pure returns(rainb) {
     return default_value;
  }
}
```

Output:



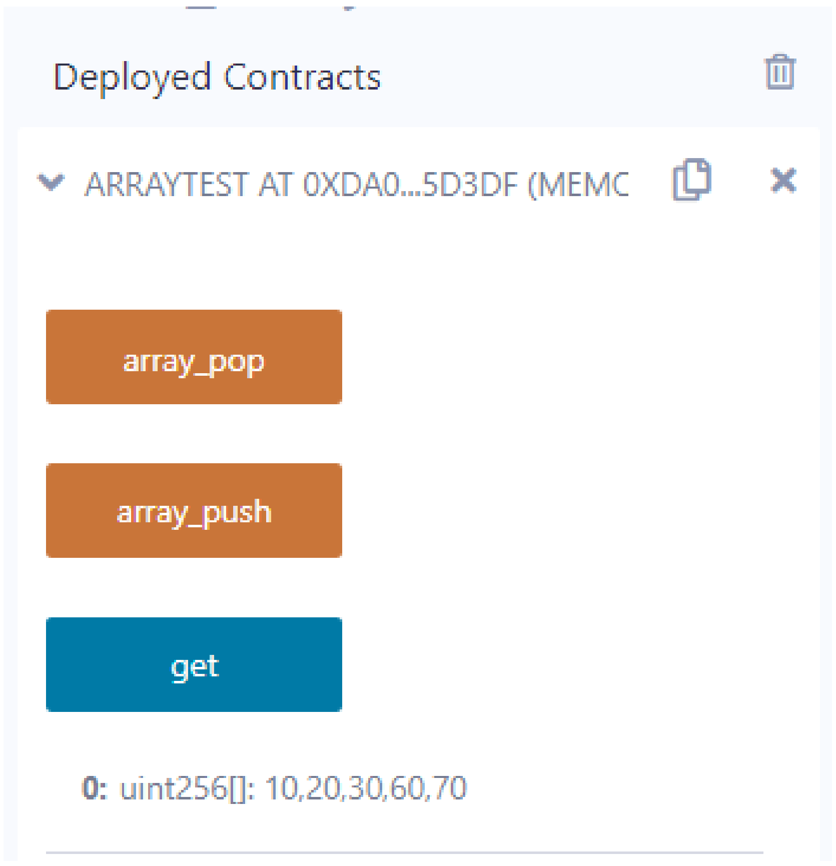
```
Deployed Contracts
RAINBOWTEST AT 0X5A8...C4D01 (ME
set_value
get_choice
0: uint8: 3
getdefaultvalue
0: uint8: 0
```

5. Array:

Code:

```
 //program to
// demonstrate how to
// write a smart contract

pragma solidity >= 0.4.16 < 0.7.0;

// Creating a contract
contract arraytest {

    // Defining the array
    uint[] data = [10, 20, 30, 40,50];
    function array_push() public  {
        data.push(60);
        data.push(70);

    }

    // Defining the function to push
    // values to the array
    function get (
    ) public  view returns(uint[] memory  ){

        return data;


    }
      function array_pop(
    ) public   returns(uint[] memory){
        data.pop();
        return data;
    }
}
```

Output:



```
0: uint256[]: 10,20,30,60,70
```

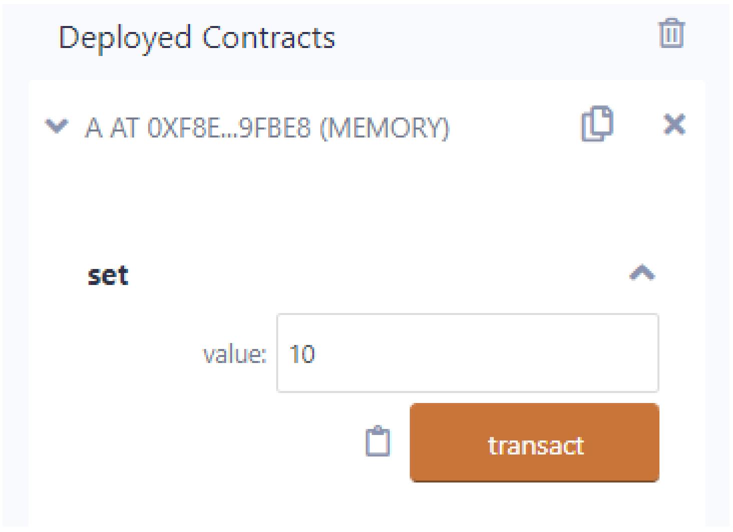
B]

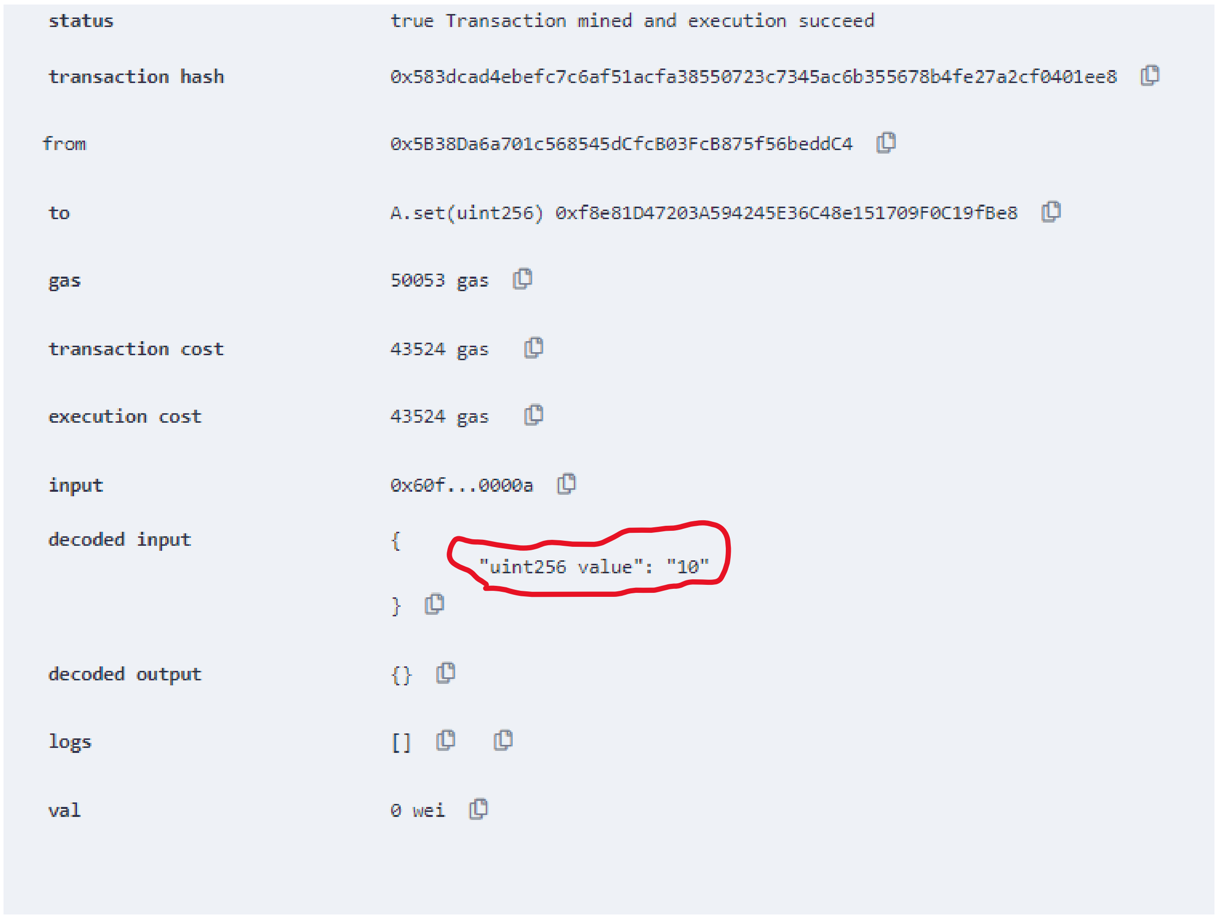
1. **Fallback function:**

Code:

```
pragma solidity ^0.5.12;

// contract with fallback function
contract A {
  uint n;
  function set(uint value) external {
    n = value;
  }

  function() external payable {
    n = 0;
  }
}

// contract to interact with contract A
contract example {
  function callA(A a) public returns (bool) {
     // calling a non-existing function
      (bool success,) = address(a).call(abi.encodeWithSignature("setter()"));
      require(success);

      // sending ether to A
      address payable payableA = address(uint160(address(a)));
      return (payableA.send(2 ether));
   }
}
```

Output:





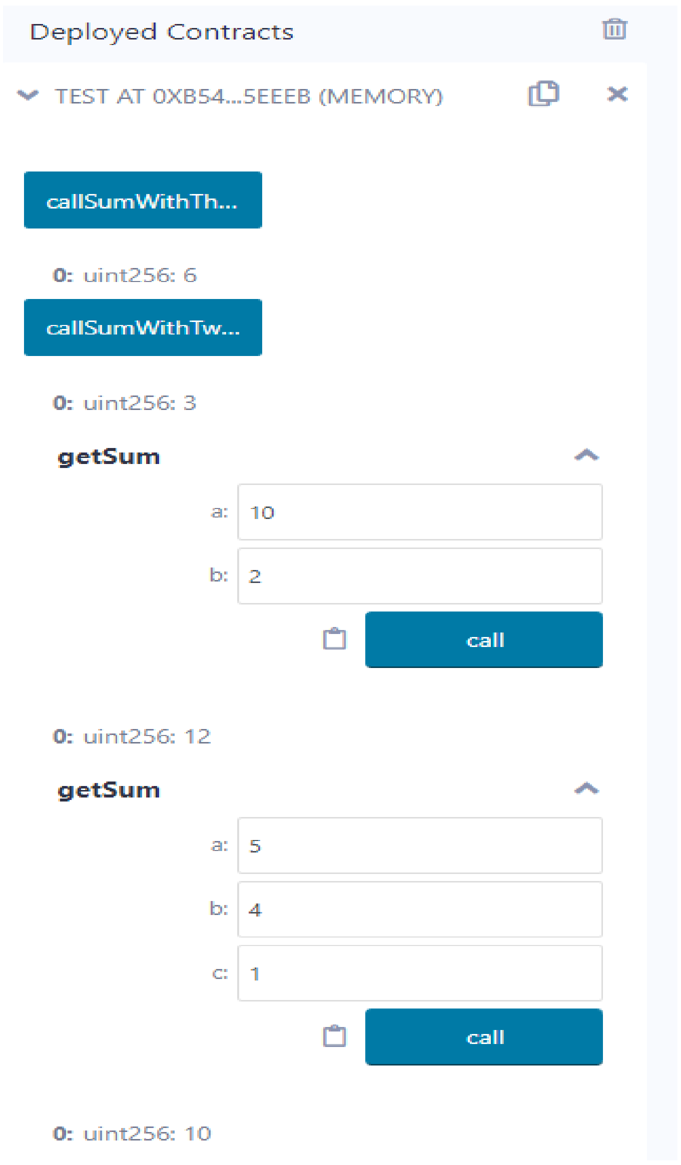
```
status                  true Transaction mined and execution succeed

transaction hash        0x583dcad4ebefc7c6af51acfa38550723c7345ac6b355678b4fe27a2cf0401ee8

from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to                      A.set(uint256) 0xf8e81D47203A594245E36C48e151709F0C19fBe8

gas                     50053 gas

transaction cost        43524 gas

execution cost          43524 gas

input                   0x60f...0000a

decoded input           {
                            "uint256 value": "10"
                        }

decoded output          {}

logs                    []

val                     0 wei
```

2. Function Overloading:

Code:

```
pragma solidity ^0.5.0;

contract Test {
   function getSum(uint a, uint b) public pure returns(uint){
      return a + b;
   }
   function getSum(uint a, uint b, uint c) public pure returns(uint){
      return a + b + c;
   }
   function callSumWithTwoArguments() public pure returns(uint){
      return getSum(1,2);
   }
   function callSumWithThreeArguments() public pure returns(uint){
      return getSum(1,2,3);
   }
}
```

Output:



```
0: uint256: 6
0: uint256: 3
```

getSum: a: 10, b: 2

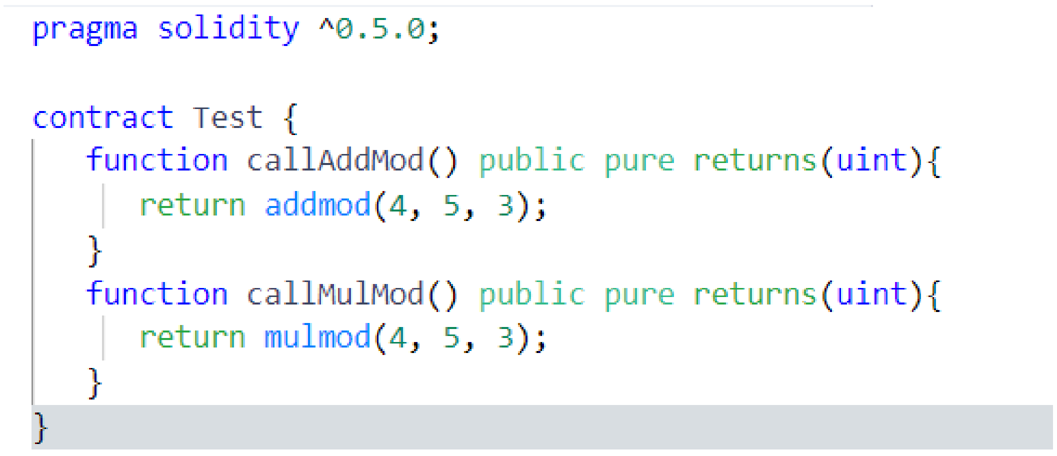
```
0: uint256: 12
```

getSum: a: 5, b: 4, c: 1

```
0: uint256: 10
```

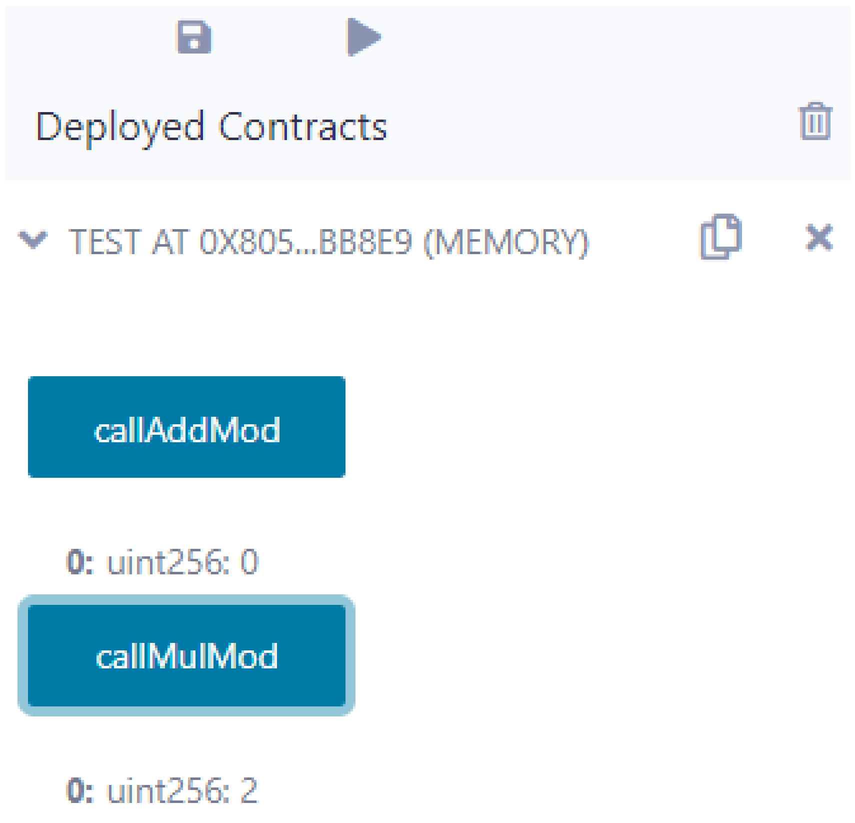
3. Mathematical Function:

Code:



```
pragma solidity ^0.5.0;

contract Test {
   function callAddMod() public pure returns(uint){
      return addmod(4, 5, 3);
   }
   function callMulMod() public pure returns(uint){
      return mulmod(4, 5, 3);
   }
}
```

Output:



Deployed Contracts

TEST AT 0X805...BB8E9 (MEMORY)

callAddMod

```
0: uint256: 0
```

callMulMod

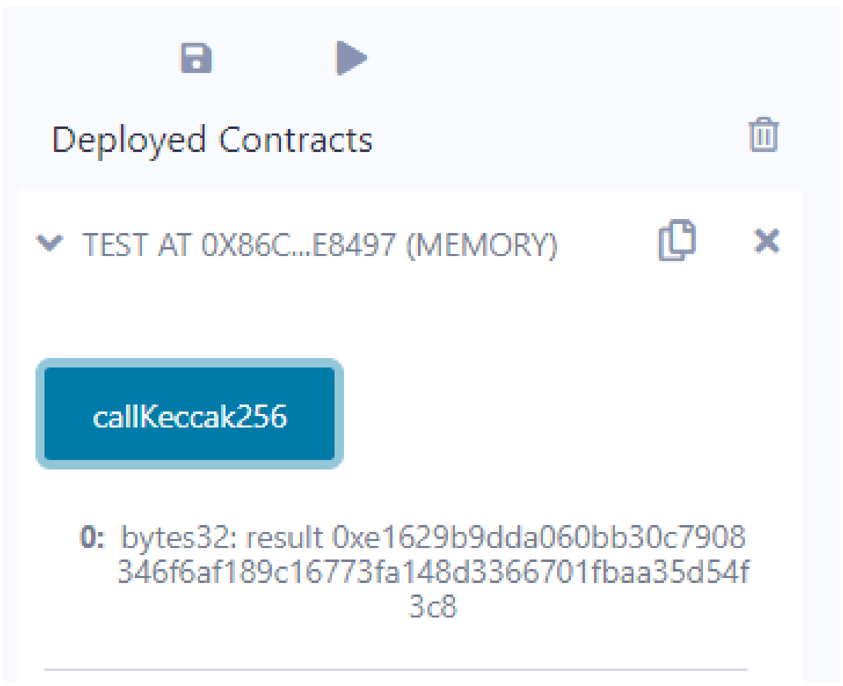
```
0: uint256: 2
```

4. Cryptographic Function:

Code:

```
pragma solidity ^0.5.0;

contract Test {
    function callKeccak256() public pure returns(bytes32 result){
        return keccak256("ABC");
    }
}
```

Output:



Deployed Contracts

TEST AT 0X86C...E8497 (MEMORY)

```
0: bytes32: result 0xe1629b9dda060bb30c7908
   346f6af189c16773fa148d3366701fbaa35d54f
   3c8
```

Practical – 4

Q. Implement and demonstrate the use of following in solidity.

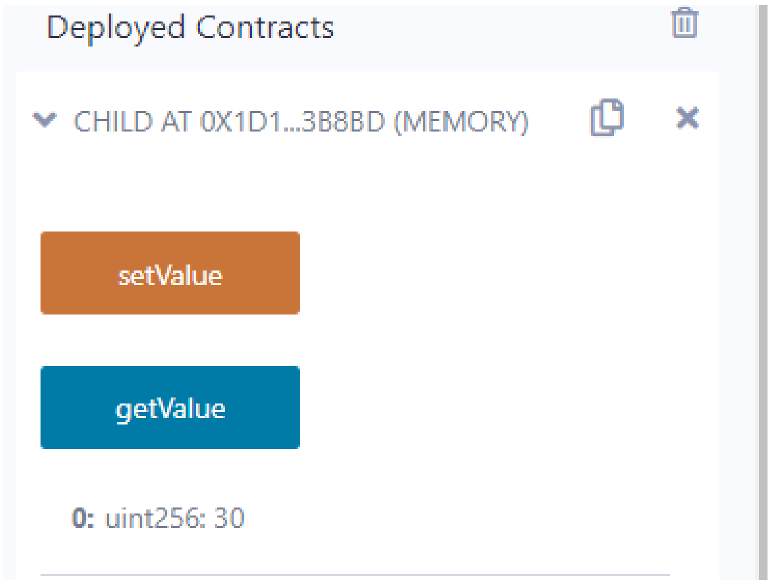
- **Inheritance**

  1. **Single Inheritance:**

Code:

```
// Solidity program to
// demonstrate
// Single Inheritance
pragma solidity >=0.4.22 <0.6.0;

// Defining contract
contract parent{

   // Declaring internal
   // state varaiable
   uint internal sum;

   // Defining external function
   // to set value of internal
   // state variable sum
   function setValue() external {
      uint a = 10;
      uint b = 20;
      sum = a + b;
   }
}

// Defining child contract
contract child is parent{

   // Defining external function
   // to return value of
   // internal state variable sum
   function getValue(
   ) external view returns(uint) {
      return sum;
   }
}
```

Output:



```
0: uint256: 30
```

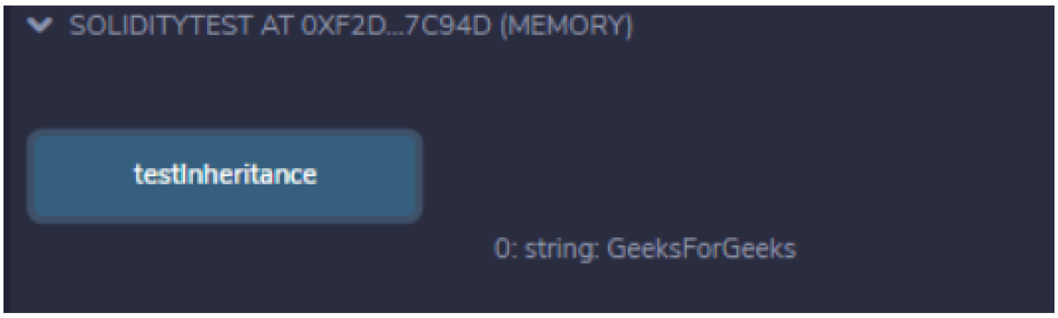
2. <u>Multi-level Inheritance:</u>

Code:

```
_//Solidity program to

// demonstrate Multi-Level
// Inheritance
pragma solidity >=0.4.22 <0.6.0;

// Defining parent contract A
contract A {

   // Declaring state variables
   string internal x;
   string a = "Geeks" ;
   string b = "For";

   // Defining external function
   // to return concatenated string
   function getA() external{
      x = string(abi.encodePacked(a, b));
   }
}

// Defining child contract B
// inheriting parent contract A
contract B is A {

   // Declaring state variables
   // of child contract B
```

```
    string public y;
    string c = "Geeks";

    // Defining external function to
    // return concatenated string
    function getB() external payable returns(
    string memory){
        y = string(abi.encodePacked(x, c));
    }
}

// Defining child contract C
// inheriting parent contract A
contract C is B {

    // Defining external function
    // returning concatenated string
    // generated in child contract B
    function getC() external view returns(
    string memory){
        return y;
    }
}

// Defining calling contract
contract caller {

    // Creating object of child C
    C cc = new C();

    // Defining public function to
    // return final concatenated string
    function testInheritance(
    ) public returns (
    string memory) {
        cc.getA();
        cc.getB();
        return cc.getC();
    }
}
```

Output:



```
SOLIDITYTEST AT 0XF2D...7C94D (MEMORY)
0: string: GeeksForGeeks
```

Practical – 5

Q. Install Hyperledger fabric and composer. Deploy and execute application.

Install Hyperledger Fabric:

Step 1: download VMware Player and Intsall in your PC.

- Download Link : https://www.vmware.com/in/products/workstation-player/workstation-player-evaluation.html
- Installation Steps Link: https://youtu.be/Y-lyHf1Uq3U

Step 2: Download Ubuntu ISO image.

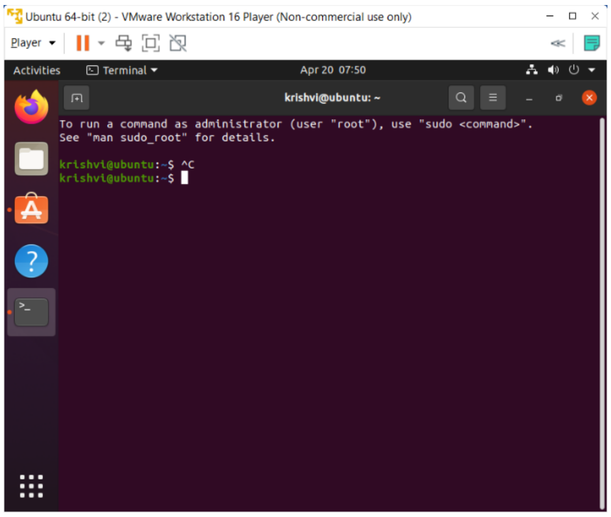
- Download link: https://ubuntu.com/download/desktop/thankyou?version=20.04.4&architecture=amd64

Step 3: To Create a New Virtual Machine

- Steps link: https://youtu.be/9rUhGWijf9U

Step 4: how to use Ubuntu Terminal.

- Steps link:
- Ubuntu terminal screen.

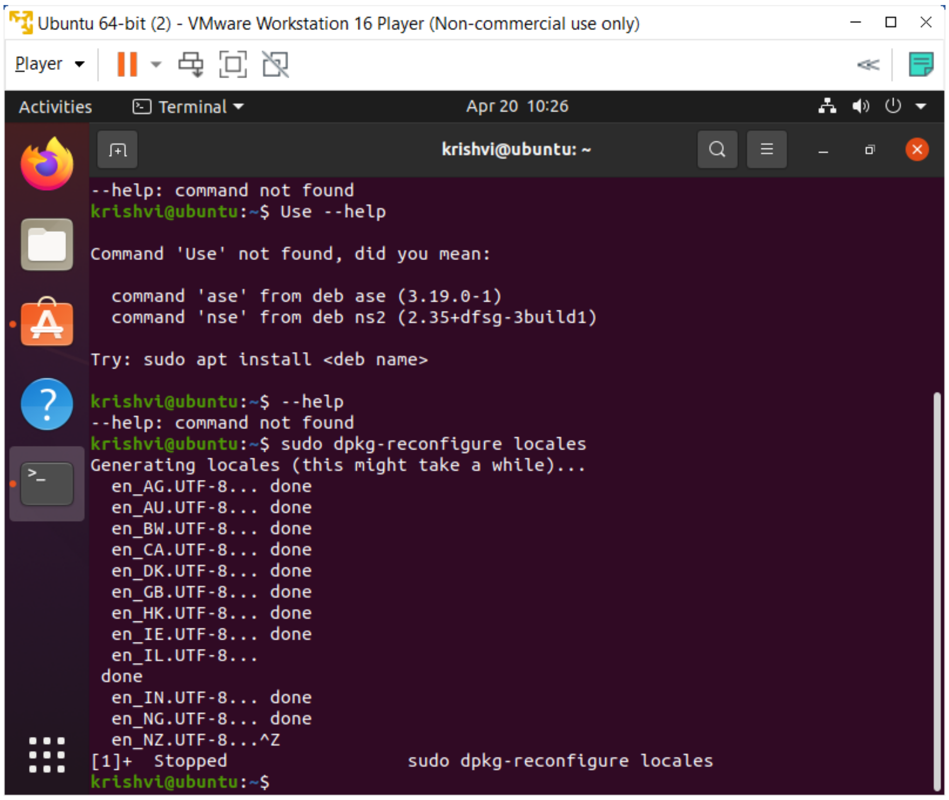

Ubuntu 64-bit (2) - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Apr 20 07:50
krishvi@ubuntu: ~

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

krishvi@ubuntu:~$ ^C
krishvi@ubuntu:~$ 
```

$ sudo dpkg-reconfigure locales

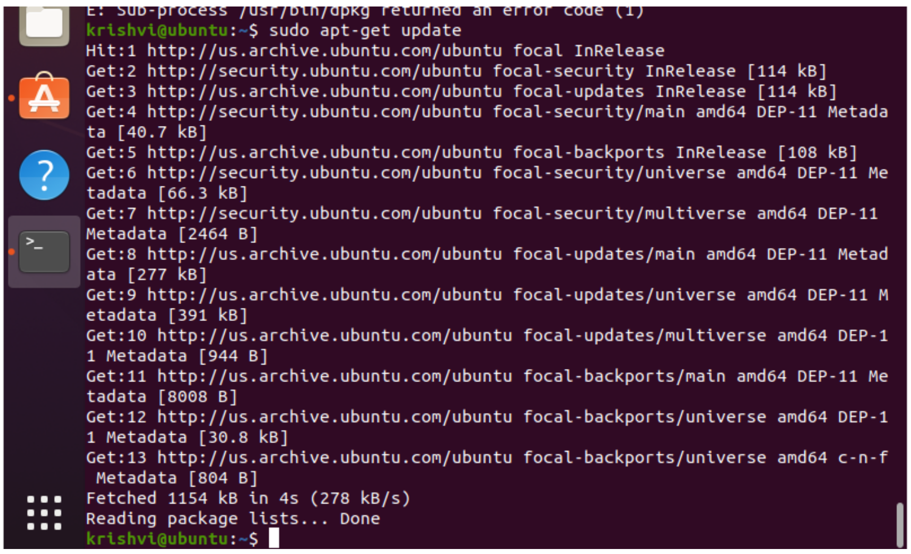
Write above command and press enter key. Second window open in that choose en_US.UTF-8 UTF-8 than click on ok button.

Output:


Ubuntu 64-bit (2) - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Apr 20 10:26
krishvi@ubuntu: ~

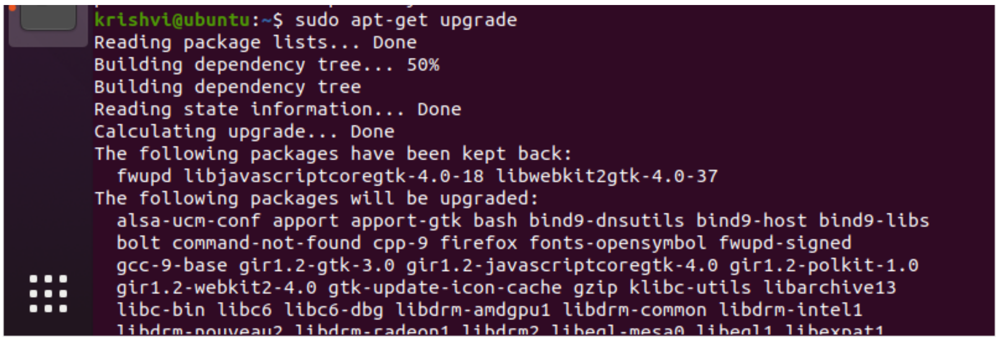
```
--help: command not found
krishvi@ubuntu:~$ Use --help

Command 'Use' not found, did you mean:

  command 'ase' from deb ase (3.19.0-1)
  command 'nse' from deb ns2 (2.35+dfsg-3build1)

Try: sudo apt install <deb name>

krishvi@ubuntu:~$ --help
--help: command not found
krishvi@ubuntu:~$ sudo dpkg-reconfigure locales
Generating locales (this might take a while)...
  en_AG.UTF-8... done
  en_AU.UTF-8... done
  en_BW.UTF-8... done
  en_CA.UTF-8... done
  en_DK.UTF-8... done
  en_GB.UTF-8... done
  en_HK.UTF-8... done
  en_IE.UTF-8... done
  en_IL.UTF-8...
 done
  en_IN.UTF-8... done
  en_NG.UTF-8... done
  en_NZ.UTF-8...^Z
[1]+  Stopped                 sudo dpkg-reconfigure locales
krishvi@ubuntu:~$
```

$ sudo apt-get update



```
E: Sub-process /usr/bin/dpkg returned an error code (1)
krishvi@ubuntu:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metada
ta [40.7 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Me
tadata [66.3 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11
Metadata [2464 B]
Get:8 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metad
ata [277 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 M
etadata [391 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 DEP-1
1 Metadata [944 B]
Get:11 http://us.archive.ubuntu.com/ubuntu focal-backports/main amd64 DEP-11 Me
tadata [8008 B]
Get:12 http://us.archive.ubuntu.com/ubuntu focal-backports/universe amd64 DEP-1
1 Metadata [30.8 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu focal-backports/universe amd64 c-n-f
 Metadata [804 B]
Fetched 1154 kB in 4s (278 kB/s)
Reading package lists... Done
krishvi@ubuntu:~$
```

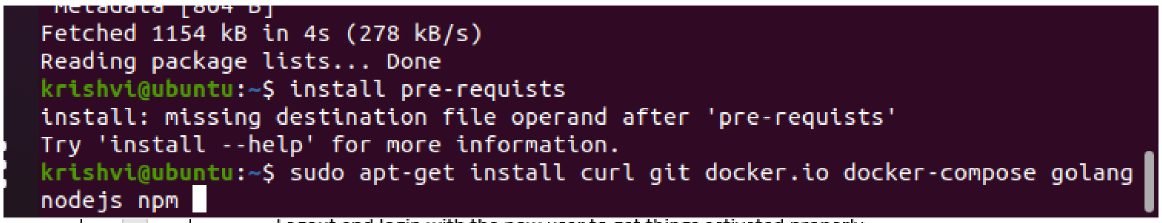
$ sudo apt-get upgrade



```
krishvi@ubuntu:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree... 50%
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  fwupd libjavascriptcoregtk-4.0-18 libwebkit2gtk-4.0-37
The following packages will be upgraded:
  alsa-ucm-conf apport apport-gtk bash bind9-dnsutils bind9-host bind9-libs
  bolt command-not-found cpp-9 firefox fonts-opensymbol fwupd-signed
  gcc-9-base gir1.2-gtk-3.0 gir1.2-javascriptcoregtk-4.0 gir1.2-polkit-1.0
  gir1.2-webkit2-4.0 gtk-update-icon-cache gzip klibc-utils libarchive13
  libc-bin libc6 libc6-dbg libdrm-amdgpu1 libdrm-common libdrm-intel1
  libdrm-nouveau2 libdrm-radeon1 libdrm2 libegl-mesa0 libegl1 libexpat1
```

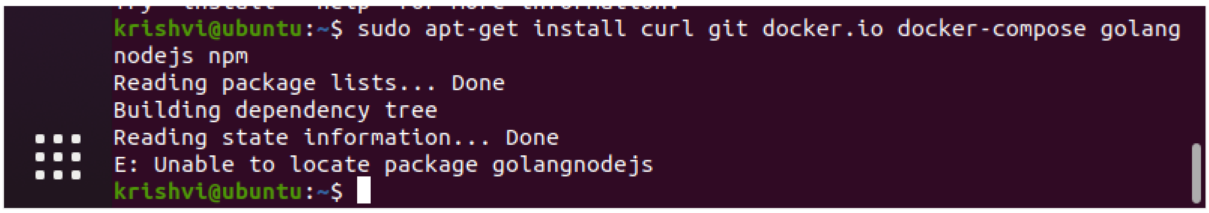
Install pre-requists

$ sudo apt-get install curl git docker.io docker-compose golang nodejs npm



```
Fetched 1154 kB in 4s (278 kB/s)
Reading package lists... Done
krishvi@ubuntu:~$ install pre-requists
install: missing destination file operand after 'pre-requists'
Try 'install --help' for more information.
krishvi@ubuntu:~$ sudo apt-get install curl git docker.io docker-compose golang
nodejs npm
```

Logout and login with the new user to get things activated properly



```
krishvi@ubuntu:~$ sudo apt-get install curl git docker.io docker-compose golang
nodejs npm
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package golangnodejs
krishvi@ubuntu:~$
```

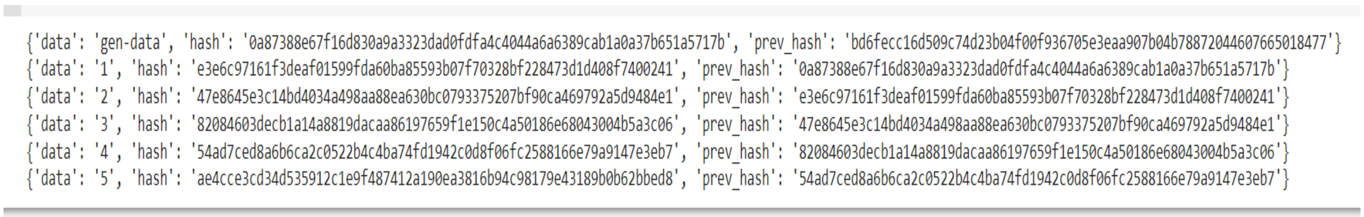
Practical – 9

Q. Create your own blockchain and demonstrate its use.

Code:

```
import hashlib

def hashGenerator(data):
  result=hashlib.sha256(data.encode())
  return result.hexdigest()

class Block:
  def __init__(self,data,hash,prev_hash):
      self.data=data
      self.hash=hash
      self.prev_hash=prev_hash

class Blockchain:
  def __init__(self):
    hashLast=hashGenerator('gen_last')
    hashStart=hashGenerator('gen_hash')
    genesis=Block('gen-data',hashStart,hashLast)
    self.chain=[genesis]

  def add_block(self,data):
      prev_hash=self.chain[-1].hash
      hash=hashGenerator(data+prev_hash)
      block=Block(data,hash,prev_hash)
      self.chain.append(block)

bc=Blockchain()
bc.add_block('1')
bc.add_block('2')
bc.add_block('3')
bc.add_block('4')
bc.add_block('5')

for block in bc.chain:
  print(block.__dict__)
```

Output:



```
{'data': 'gen-data', 'hash': '0a87388e67f16d830a9a3323dad0fdfa4c4044a6a6389cab1a0a37b651a5717b', 'prev_hash': 'bd6fecc16d509c74d23b04f00f936705e3eaa907b04b78872044607665018477'}
{'data': '1', 'hash': 'e3e6c97161f3deaf01599fda60ba85593b07f70328bf228473d1d408f7400241', 'prev_hash': '0a87388e67f16d830a9a3323dad0fdfa4c4044a6a6389cab1a0a37b651a5717b'}
{'data': '2', 'hash': '47e8645e3c14bd4034a498aa88ea630bc0793375207bf90ca469792a5d9484e1', 'prev_hash': 'e3e6c97161f3deaf01599fda60ba85593b07f70328bf228473d1d408f7400241'}
{'data': '3', 'hash': '82084603decb1a14a8819dacaa86197659f1e150c4a50186e68043004b5a3c06', 'prev_hash': '47e8645e3c14bd4034a498aa88ea630bc0793375207bf90ca469792a5d9484e1'}
{'data': '4', 'hash': '54ad7ced8a6b6ca2c0522b4c4ba74fd1942c0d8f06fc2588166e79a9147e3eb7', 'prev_hash': '82084603decb1a14a8819dacaa86197659f1e150c4a50186e68043004b5a3c06'}
{'data': '5', 'hash': 'ae4cce3cd34d535912c1e9f487412a190ea3816b94c98179e43189b0b62bbed8', 'prev_hash': '54ad7ced8a6b6ca2c0522b4c4ba74fd1942c0d8f06fc2588166e79a9147e3eb7'}
```