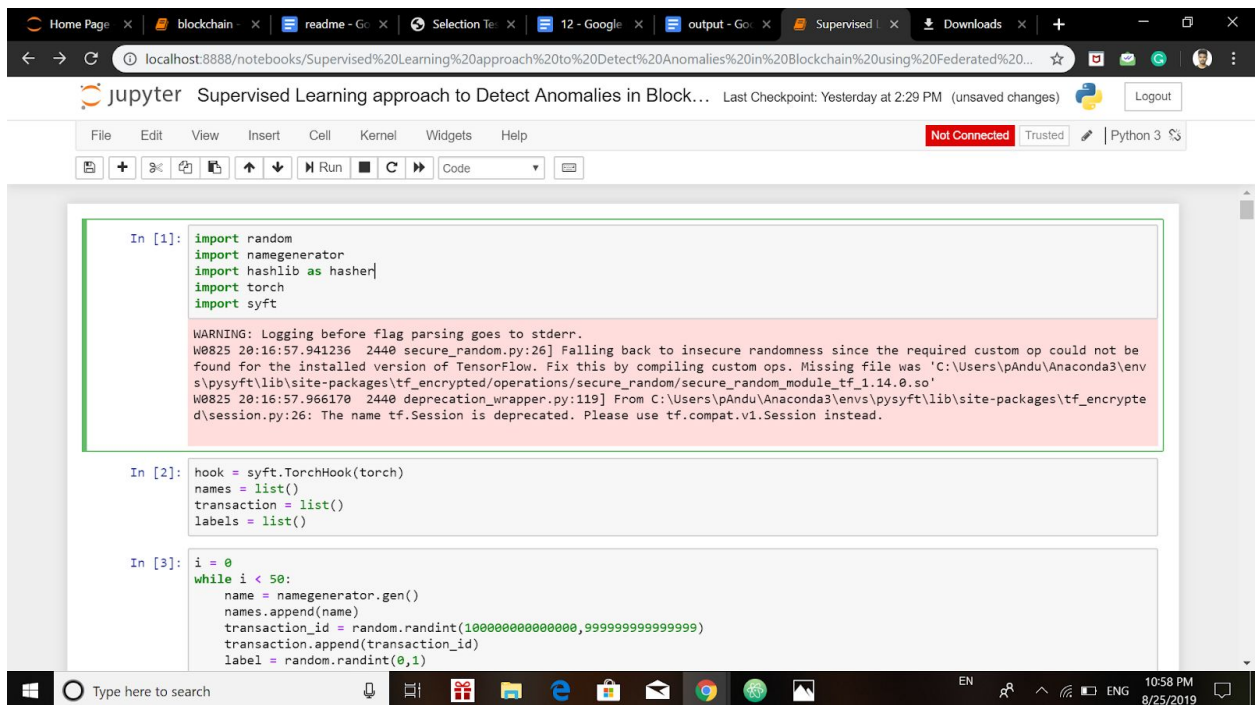


1. I had used a random name generator by using namegenerator for unique for each transaction
2. I created transaction\_id for each transaction
3. I had used a supervised learning algorithm to solve this problem. I had assigned either 0 or 1 to each transaction i.e.,

0 → Anomaly  
1 → Normal data



The screenshot shows a Jupyter Notebook titled "Supervised Learning approach to Detect Anomalies in Block...". The notebook contains three code cells. The first cell imports necessary libraries: random, namegenerator, hashlib as hasher, torch, and syft. The second cell initializes syft hooks for torch. The third cell is a loop that generates 50 transactions, each with a unique name, a random transaction\_id, and a random label (0 or 1). The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar at the bottom showing the current kernel is Python 3.

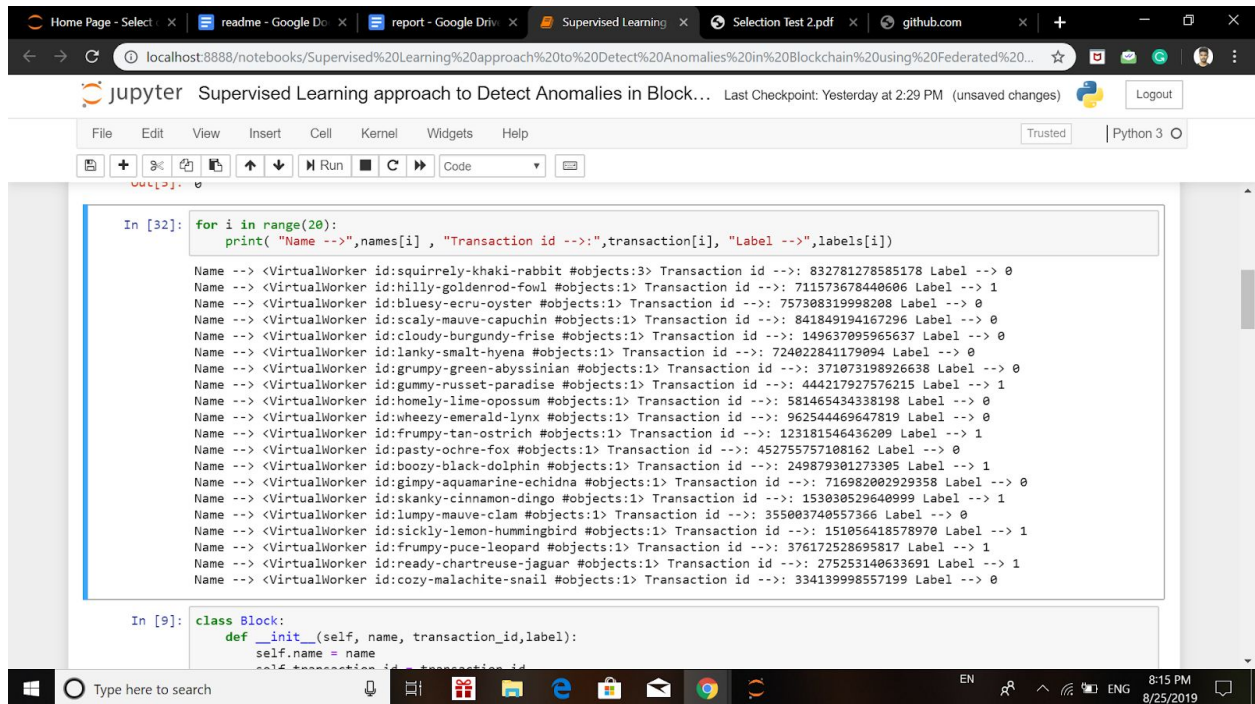
```
In [1]: import random
import namegenerator
import hashlib as hasher
import torch
import syft

WARNING: Logging before flag parsing goes to stderr.
W0825 20:16:57.941236 2440 secure_random.py:26] Falling back to insecure randomness since the required custom op could not be
found for the installed version of TensorFlow. Fix this by compiling custom ops. Missing file was 'C:\Users\pAndu\Anaconda3\env
s\pysyft\lib\site-packages\tf_encrypted\operations\secure_random\secure_random_module_tf_1.14.0.so'
W0825 20:16:57.966170 2440 deprecation_wrapper.py:119] From C:\Users\pAndu\Anaconda3\envs\pysyft\lib\site-packages\tf_encrypte
d\session.py:26: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

In [2]: hook = syft.TorchHook(torch)
names = list()
transaction = list()
labels = list()

In [3]: i = 0
while i < 50:
    name = namegenerator.gen()
    names.append(name)
    transaction_id = random.randint(1000000000000000, 9999999999999999)
    transaction.append(transaction_id)
    label = random.randint(0,1)
```

4. Below picture shows name → transaction\_id → label for the first 20 transactions



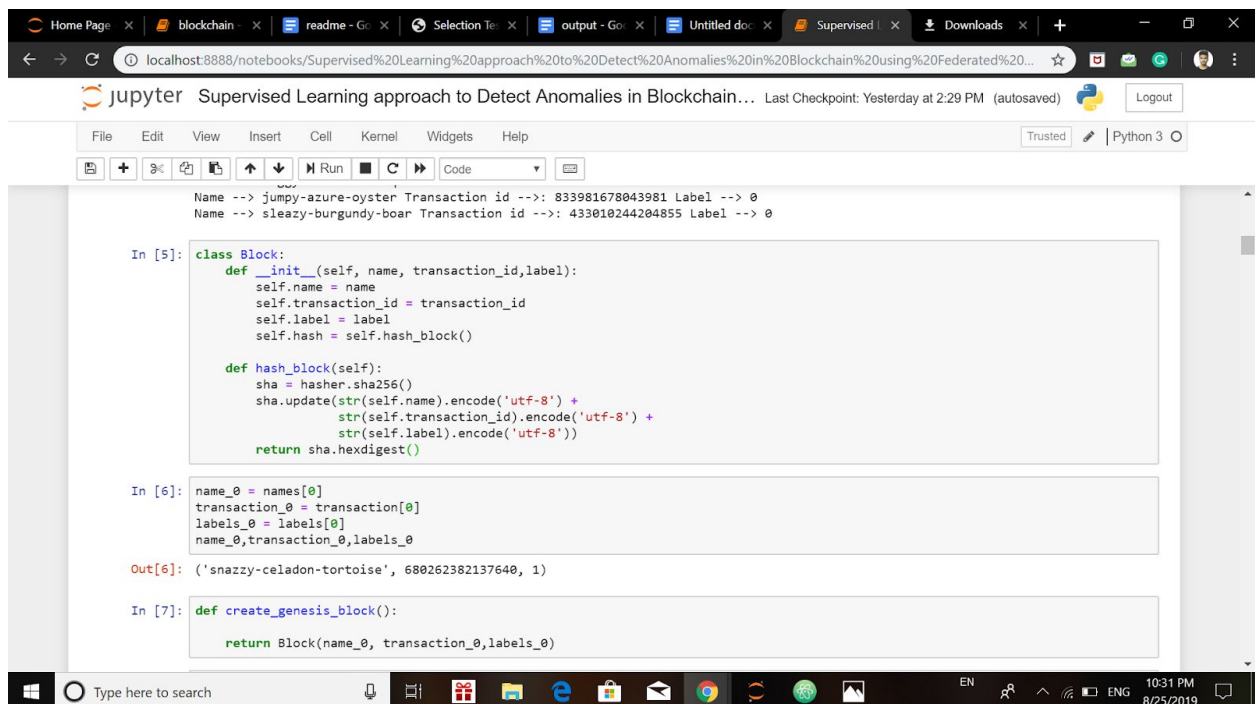
The screenshot shows a Jupyter Notebook titled "Supervised Learning approach to Detect Anomalies in Blockchain...". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a code editor. The code in the notebook is as follows:

```
In [32]: for i in range(20):
          print("Name -->", names[i], "Transaction id -->", transaction[i], "Label -->", labels[i])

Name --> <VirtualWorker id:squirrelly-khaki-rabbit #objects:3> Transaction id -->: 832781278585178 Label --> 0
Name --> <VirtualWorker id:hilly-goldenrod-fowl #objects:1> Transaction id -->: 711573678448606 Label --> 1
Name --> <VirtualWorker id:bluesy-ecru-oyster #objects:1> Transaction id -->: 757308319998208 Label --> 0
Name --> <VirtualWorker id:scaly-mauve-capuchin #objects:1> Transaction id -->: 841849194167296 Label --> 0
Name --> <VirtualWorker id:cloudy-burgundy-frise #objects:1> Transaction id -->: 149637095965637 Label --> 0
Name --> <VirtualWorker id:lanky-smalt-hyena #objects:1> Transaction id -->: 724022841179894 Label --> 0
Name --> <VirtualWorker id:grumpy-green-abyssinian #objects:1> Transaction id -->: 371073198926638 Label --> 0
Name --> <VirtualWorker id:gummy-russet-paradise #objects:1> Transaction id -->: 444217927576215 Label --> 1
Name --> <VirtualWorker id:homely-lime-opossum #objects:1> Transaction id -->: 581465434338198 Label --> 0
Name --> <VirtualWorker id:wheezy-emerald-lynx #objects:1> Transaction id -->: 962544469647819 Label --> 0
Name --> <VirtualWorker id:frumpy-tan-ostrich #objects:1> Transaction id -->: 123181546436209 Label --> 1
Name --> <VirtualWorker id:pasty-ochre-fox #objects:1> Transaction id -->: 452755757108162 Label --> 0
Name --> <VirtualWorker id:boozy-black-dolphin #objects:1> Transaction id -->: 249879301273305 Label --> 1
Name --> <VirtualWorker id:gimpy-aquamarine-echidna #objects:1> Transaction id -->: 716982002929358 Label --> 0
Name --> <VirtualWorker id:skanky-cinnamon-dingo #objects:1> Transaction id -->: 153030529640099 Label --> 1
Name --> <VirtualWorker id:lumpy-mauve-clam #objects:1> Transaction id -->: 355003740557366 Label --> 0
Name --> <VirtualWorker id:sickly-lemon-hummingbird #objects:1> Transaction id -->: 151056418578970 Label --> 1
Name --> <VirtualWorker id:frumpy-puce-leopard #objects:1> Transaction id -->: 376172528695817 Label --> 1
Name --> <VirtualWorker id:ready-chartreuse-jaguar #objects:1> Transaction id -->: 275253140633691 Label --> 1
Name --> <VirtualWorker id:cozy-malachite-snail #objects:1> Transaction id -->: 334139998557199 Label --> 0

In [9]: class Block:
          def __init__(self, name, transaction_id, label):
              self.name = name
              self.transaction_id = transaction_id
              self.label = label
              self.hash = self.hash_block()
```

## 5. Creating a Block class and hash\_block function



The screenshot shows a Jupyter Notebook titled "Supervised Learning approach to Detect Anomalies in Blockchain...". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a code editor. The code in the notebook is as follows:

```
Name --> jumpy-azure-oyster Transaction id -->: 833981678843981 Label --> 0
Name --> sleazy-burgundy-boar Transaction id -->: 433010244204855 Label --> 0

In [5]: class Block:
          def __init__(self, name, transaction_id, label):
              self.name = name
              self.transaction_id = transaction_id
              self.label = label
              self.hash = self.hash_block()

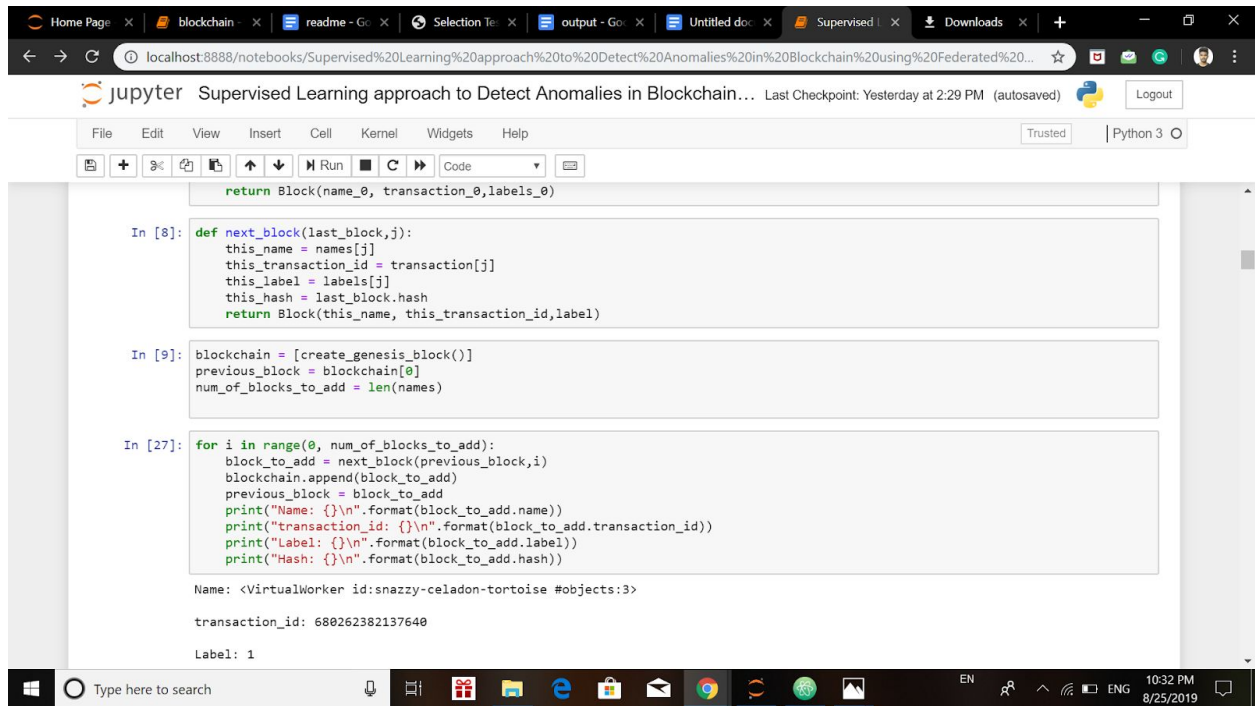
          def hash_block(self):
              sha = hasher.sha256()
              sha.update(str(self.name).encode('utf-8') +
                         str(self.transaction_id).encode('utf-8') +
                         str(self.label).encode('utf-8'))
              return sha.hexdigest()

In [6]: name_0 = names[0]
          transaction_0 = transaction[0]
          labels_0 = labels[0]
          name_0, transaction_0, labels_0

Out[6]: ('sleazy-burgundy-boar', 433010244204855, 0)

In [7]: def create_genesis_block():
          return Block(name_0, transaction_0, labels_0)
```

## 6. Adding each block to the chain



The screenshot shows a Jupyter Notebook running in a web browser. The browser's address bar shows the URL: `localhost:8888/notebooks/Supervised%20Learning%20approach%20to%20Detect%20Anomalies%20in%20Blockchain%20using%20Federated%20...`. The notebook's title bar reads "Supervised Learning approach to Detect Anomalies in Blockchain...". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and code execution. The code is written in Python and is organized into three input cells. The first cell defines a `Block` class and a `next_block` function. The second cell initializes a `blockchain` list and sets the `previous_block` to the first block in the chain. The third cell is a loop that generates blocks based on the number of blocks to add, printing the details of each block. The output of the third cell shows the details of the first block: Name, transaction\_id, Label, and Hash.

```
return Block(name_0, transaction_0, labels_0)

In [8]: def next_block(last_block,j):
        this_name = names[j]
        this_transaction_id = transaction[j]
        this_label = labels[j]
        this_hash = last_block.hash
        return Block(this_name, this_transaction_id,label)

In [9]: blockchain = [create_genesis_block()]
        previous_block = blockchain[0]
        num_of_blocks_to_add = len(names)

In [27]: for i in range(0, num_of_blocks_to_add):
        block_to_add = next_block(previous_block,i)
        blockchain.append(block_to_add)
        previous_block = block_to_add
        print("Name: {}".format(block_to_add.name))
        print("transaction_id: {}".format(block_to_add.transaction_id))
        print("Label: {}".format(block_to_add.label))
        print("Hash: {}".format(block_to_add.hash))

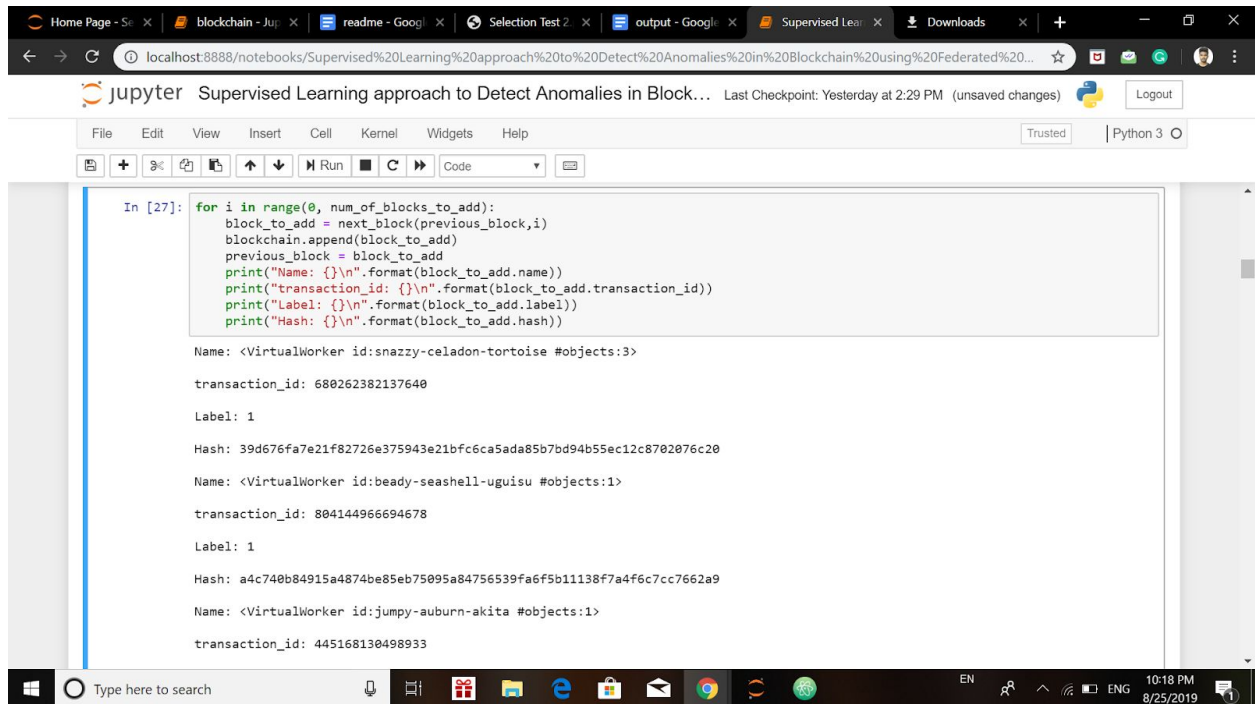
Name: <VirtualWorker id:snazzy-celadon-tortoise #objects:3>

transaction_id: 680262382137640

Label: 1
```

## 7. Details of each transaction

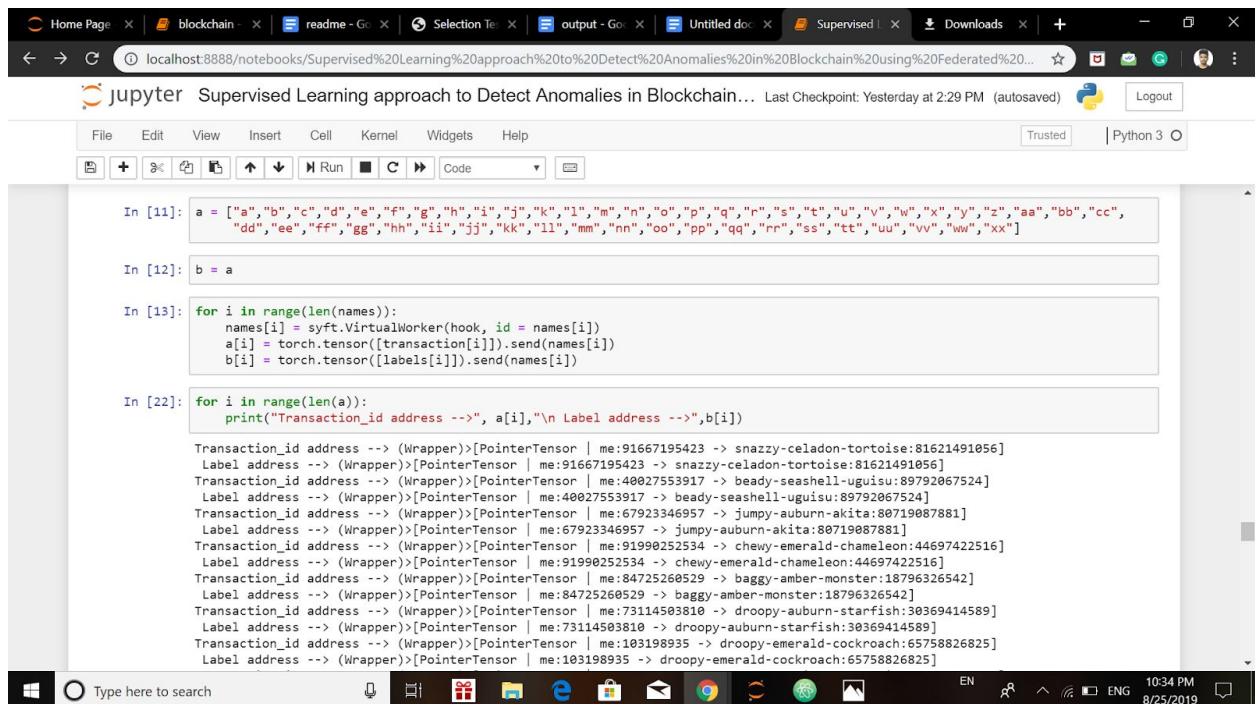
- Name
- Transaction\_id
- Label
- Hash



```
In [27]: for i in range(0, num_of_blocks_to_add):
        block_to_add = next_block(previous_block,i)
        blockchain.append(block_to_add)
        previous_block = block_to_add
        print("Name: {}".format(block_to_add.name))
        print("transaction_id: {}".format(block_to_add.transaction_id))
        print("Label: {}".format(block_to_add.label))
        print("Hash: {}".format(block_to_add.hash))

Name: <VirtualWorker id:snazzy-celadon-tortoise #objects:3>
transaction_id: 680262382137640
Label: 1
Hash: 39d676fa7e21f82726e375943e21bfc6ca5ada85b7bd94b55ec12c8702076c20
Name: <VirtualWorker id:beady-seashell-uguisu #objects:1>
transaction_id: 804144966694678
Label: 1
Hash: a4c740b84915a4874be85eb75095a84756539fa6f5b1138f7a4f6c7cc7662a9
Name: <VirtualWorker id:jumpy-auburn-akita #objects:1>
transaction_id: 445168130498933
```

8. Created a new list for assigning details to the 50 members in the Blockchain. It will create TensorPointers



```
In [11]: a = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","aa","bb","cc","dd","ee","ff","gg","hh","ii","jj","kk","ll","mm","nn","oo","pp","qq","rr","ss","tt","uu","vv","ww","xx"]

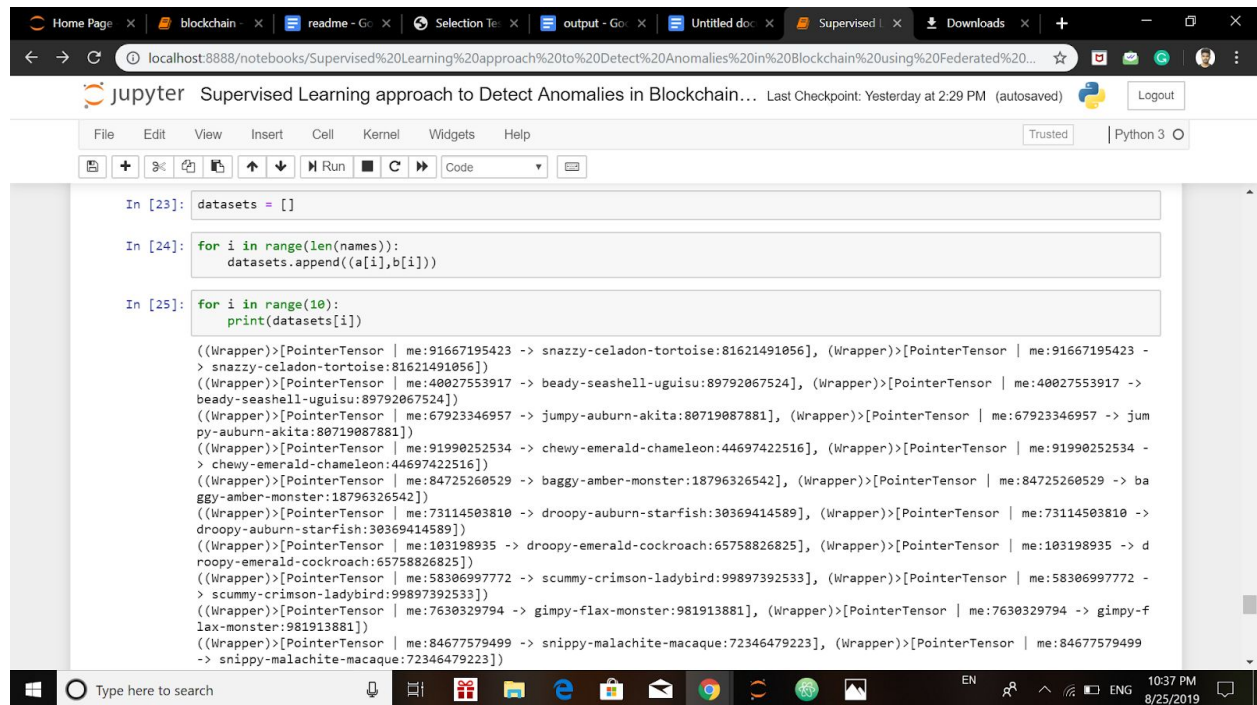
In [12]: b = a

In [13]: for i in range(len(names)):
        names[i] = syft.VirtualWorker(hook, id = names[i])
        a[i] = torch.tensor([transaction[i]]).send(names[i])
        b[i] = torch.tensor([labels[i]]).send(names[i])

In [22]: for i in range(len(a)):
        print("Transaction_id address -->", a[i], "\n Label address -->", b[i])

Transaction_id address --> (Wrapper)>[PointerTensor | me:91667195423 -> snazzy-celadon-tortoise:81621491056]
Label address --> (Wrapper)>[PointerTensor | me:91667195423 -> snazzy-celadon-tortoise:81621491056]
Transaction_id address --> (Wrapper)>[PointerTensor | me:40027553917 -> beady-seashell-uguisu:89792067524]
Label address --> (Wrapper)>[PointerTensor | me:40027553917 -> beady-seashell-uguisu:89792067524]
Transaction_id address --> (Wrapper)>[PointerTensor | me:67923346957 -> jumpy-auburn-akita:80719087881]
Label address --> (Wrapper)>[PointerTensor | me:67923346957 -> jumpy-auburn-akita:80719087881]
Transaction_id address --> (Wrapper)>[PointerTensor | me:91990252534 -> chewy-emerald-chameleon:44697422516]
Label address --> (Wrapper)>[PointerTensor | me:91990252534 -> chewy-emerald-chameleon:44697422516]
Transaction_id address --> (Wrapper)>[PointerTensor | me:84725260529 -> baggy-amber-monster:18796326542]
Label address --> (Wrapper)>[PointerTensor | me:84725260529 -> baggy-amber-monster:18796326542]
Transaction_id address --> (Wrapper)>[PointerTensor | me:73114503810 -> droopy-auburn-starfish:30369414589]
Label address --> (Wrapper)>[PointerTensor | me:73114503810 -> droopy-auburn-starfish:30369414589]
Transaction_id address --> (Wrapper)>[PointerTensor | me:103198935 -> droopy-emerald-cockroach:65758826825]
Label address --> (Wrapper)>[PointerTensor | me:103198935 -> droopy-emerald-cockroach:65758826825]
```

9. These values are assigned to a tuple called datasets. Below shows datasets values



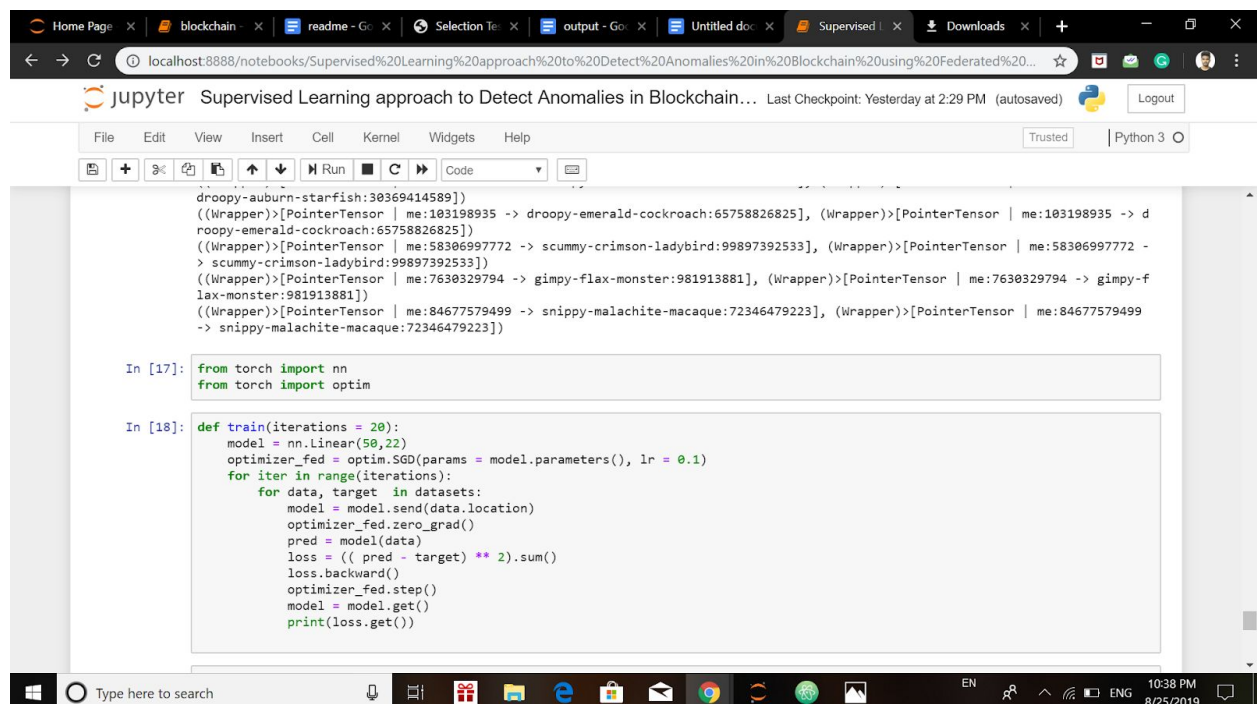
```
In [23]: datasets = []

In [24]: for i in range(len(names)):
          datasets.append((a[i],b[i]))

In [25]: for i in range(10):
          print(datasets[i])

((Wrapper)>[PointerTensor | me:91667195423 -> snazzy-celadon-tortoise:81621491056], (Wrapper)>[PointerTensor | me:91667195423 ->
> snazzy-celadon-tortoise:81621491056])
((Wrapper)>[PointerTensor | me:40027553917 -> beady-seashell-uguisu:89792067524], (Wrapper)>[PointerTensor | me:40027553917 ->
beady-seashell-uguisu:89792067524])
((Wrapper)>[PointerTensor | me:67923346957 -> jumpy-auburn-akita:80719087881], (Wrapper)>[PointerTensor | me:67923346957 -> jum
py-auburn-akita:80719087881])
((Wrapper)>[PointerTensor | me:91990252534 -> chevy-emerald-chameleon:44697422516], (Wrapper)>[PointerTensor | me:91990252534 ->
chevy-emerald-chameleon:44697422516])
((Wrapper)>[PointerTensor | me:84725260529 -> baggy-amber-monster:18796326542], (Wrapper)>[PointerTensor | me:84725260529 -> ba
ggy-amber-monster:18796326542])
((Wrapper)>[PointerTensor | me:73114503810 -> droopy-auburn-starfish:30369414589], (Wrapper)>[PointerTensor | me:73114503810 ->
droopy-auburn-starfish:30369414589])
((Wrapper)>[PointerTensor | me:103198935 -> droopy-emerald-cockroach:65758826825], (Wrapper)>[PointerTensor | me:103198935 -> d
roopy-emerald-cockroach:65758826825])
((Wrapper)>[PointerTensor | me:58306997772 -> scummy-crimson-ladybird:99897392533], (Wrapper)>[PointerTensor | me:58306997772 ->
scummy-crimson-ladybird:99897392533])
((Wrapper)>[PointerTensor | me:7630329794 -> gimpy-flax-monster:981913881], (Wrapper)>[PointerTensor | me:7630329794 -> gimpy-f
lax-monster:981913881])
((Wrapper)>[PointerTensor | me:84677579499 -> snippy-malachite-macaque:72346479223], (Wrapper)>[PointerTensor | me:84677579499
-> snippy-malachite-macaque:72346479223])
```

10. I had imported nn module and optimizer from torch.



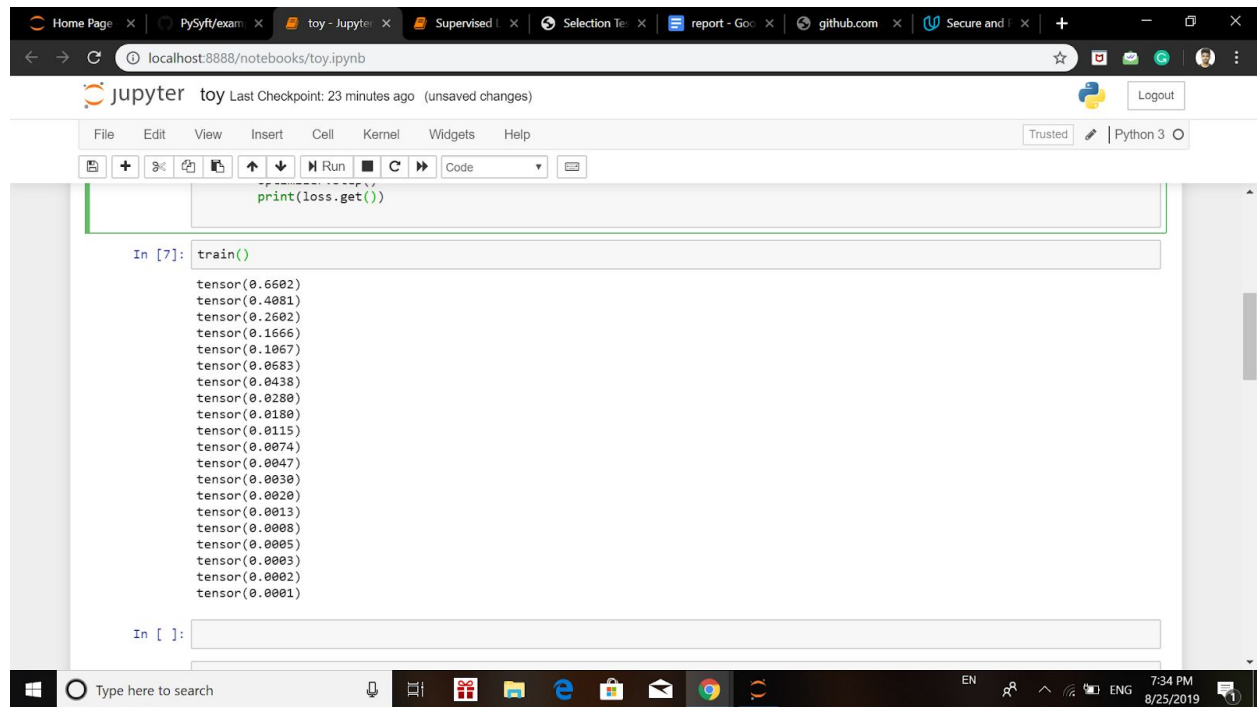
```
droopy-auburn-starfish:30369414589])
((Wrapper)>[PointerTensor | me:103198935 -> droopy-emerald-cockroach:65758826825], (Wrapper)>[PointerTensor | me:103198935 -> d
roopy-emerald-cockroach:65758826825])
((Wrapper)>[PointerTensor | me:58306997772 -> scummy-crimson-ladybird:99897392533], (Wrapper)>[PointerTensor | me:58306997772 ->
scummy-crimson-ladybird:99897392533])
((Wrapper)>[PointerTensor | me:7630329794 -> gimpy-flax-monster:981913881], (Wrapper)>[PointerTensor | me:7630329794 -> gimpy-f
lax-monster:981913881])
((Wrapper)>[PointerTensor | me:84677579499 -> snippy-malachite-macaque:72346479223], (Wrapper)>[PointerTensor | me:84677579499
-> snippy-malachite-macaque:72346479223])

In [17]: from torch import nn
          from torch import optim

In [18]: def train(iterations = 20):
          model = nn.Linear(50,22)
          optimizer_fed = optim.SGD(params = model.parameters(), lr = 0.1)
          for iter in range(iterations):
              for data, target in datasets:
                  model = model.send(data.location)
                  optimizer_fed.zero_grad()
                  pred = model(data)
                  loss = ((pred - target) ** 2).sum()
                  loss.backward()
                  optimizer_fed.step()
                  model = model.get()
                  print(loss.get())
```



11. I had trained my model over 20 iterations. Below shows that decreasing of loss every iteration



The screenshot shows a Jupyter Notebook interface in a web browser. The browser's address bar displays 'localhost:8888/notebooks/toy.ipynb'. The Jupyter interface includes a top bar with the 'jupyter' logo and a 'Logout' button. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A toolbar contains icons for file operations and a 'Run' button. The main area shows a code cell with the following content:

```
print(loss.get())
```

```
In [7]: train()
```

```
tensor(0.6602)
```

```
tensor(0.4081)
```

```
tensor(0.2602)
```

```
tensor(0.1666)
```

```
tensor(0.1067)
```

```
tensor(0.0683)
```

```
tensor(0.0438)
```

```
tensor(0.0280)
```

```
tensor(0.0180)
```

```
tensor(0.0115)
```

```
tensor(0.0074)
```

```
tensor(0.0047)
```

```
tensor(0.0030)
```

```
tensor(0.0020)
```

```
tensor(0.0013)
```

```
tensor(0.0008)
```

```
tensor(0.0005)
```

```
tensor(0.0003)
```

```
tensor(0.0002)
```

```
tensor(0.0001)
```

The output shows a series of tensor values representing the loss at each iteration, which decreases from 0.6602 to 0.0001 over 20 iterations. The Jupyter interface also shows a 'Trusted' status and 'Python 3' as the selected kernel.