

Welcome

Today We Learn [Cont.]



Opening & Closing Files

Reading & Writing Data

Using with for Saver Access

Saving Python Data to Files with json or
pickle

Today We Learn [New]



Debugging vs Error Handling

Use Debugging Tools

Handle Errors Correctly

Classes & Objects

Blockchain [Continue]



▶ Store Blockchain Locally

NEW in Blockchain

▶ Ensure Save File Access

▶ Cleaner Structure with Objects

pickle vs json

pickle

Works for all Python Data Types

Serializes Data in Binary Format

Printable and
non-printable
Characters

json

Only works for a Limited Set of
Python Data Types

Manually Convert
Data to "Known
Data Types"

Converts Data to Text

:Which Error We Should Handle:

"Handle All Errors!"

```
try:  
    something()  
except Error:  
    handle_error()
```

The Most Generic Error
Python Knows

Error Handling

~~Handle All Errors~~

```
try:  
    something()  
except Error:   
    handle_error()
```

The Most Generic Error
Python Knows

Only Handle Errors That You Can't Predict/ Avoid!

IOError (File Access), OSError (OS-related Error), ...

Applying OOP's to Our Blockchain [Task]

Procedural

```
blockchain = []
```

```
def load_data():
```

```
    load_data():
```

```
    ...
```

```
while running:
```

Execute Steps Sequentially

Code is Relatively
"Unstructured"

Object-Oriented

```
class Blockchain:
```

```
    blockchain = Blockchain()
```

```
    blockchain.load_data()
```

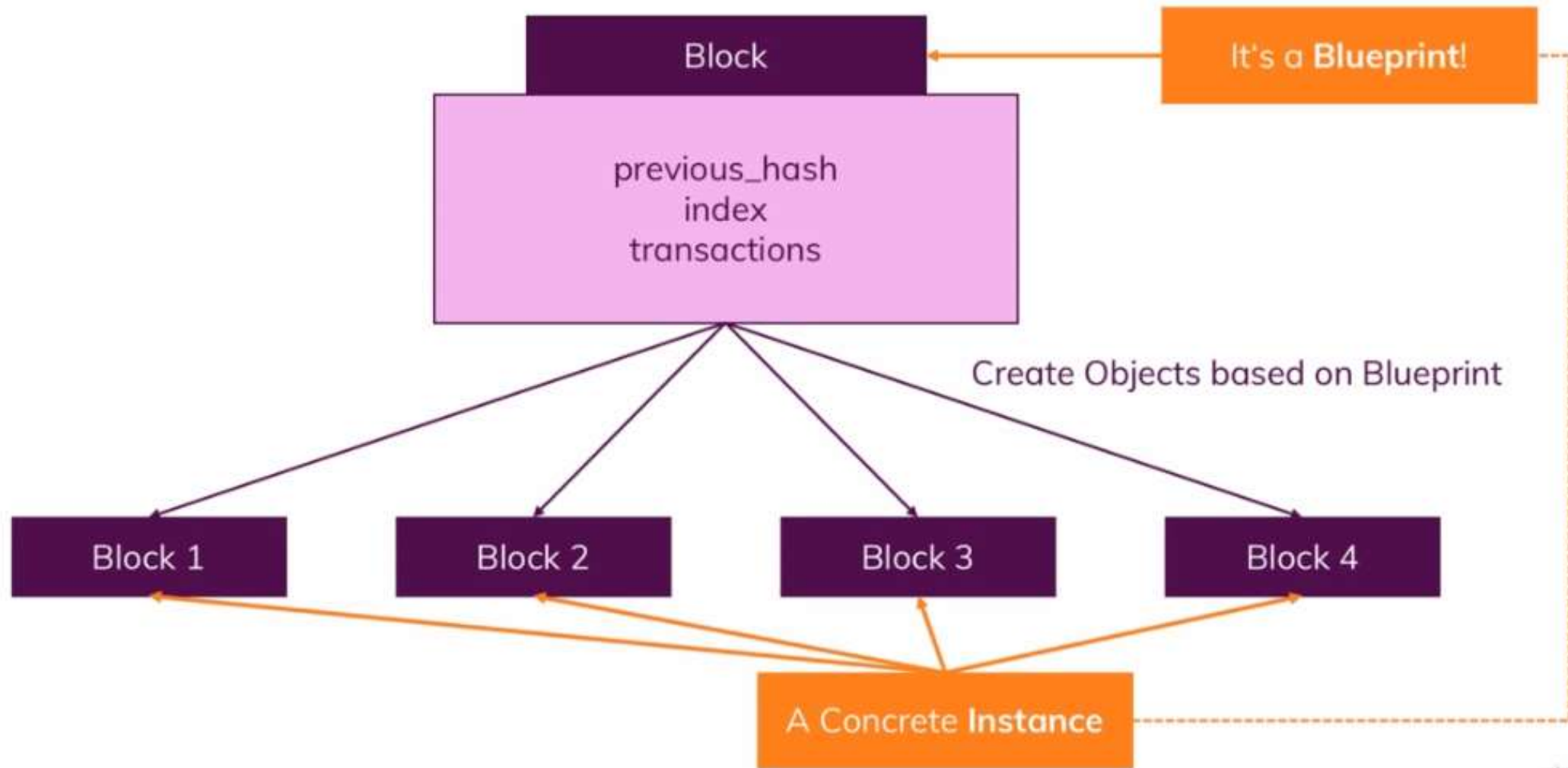
```
    ...
```

```
    user_interface = UI()
```

Use Classes as Blueprints of
Objects (Data Structures)

Code is Structured in
Objects

What one can do?



Help!

```
# Block Chain Object
class Blockchain:
    def __init__(self):
        self.chain = [self.createGenesisBlock()]

    def createGenesisBlock(self):
        return Block(0, "10/01/2017", "Genesis Block", "0")

    def getLatestBlock(self):
        return self.chain[len(self.chain)-1]

    def addBlock(self, newBlock):
        newBlock.previousHash = self.getLatestBlock().hash
        newBlock.hash = newBlock.calculateHash()
        self.chain.append(newBlock)

    def isChainValid(self):
        for i in range(1, len(self.chain)):
            currentBlock = self.chain[i]
            previousBlock = self.chain[i-1]
            # checks whether data has been tampered with
            if currentBlock.hash != currentBlock.calculateHash():
                return False
            if currentBlock.previousHash != previousBlock.hash:
                return False
        return True

    def printBlockchain(self):
        for i in range(1, len(self.chain)):
            self.chain[i].printBlock()
```

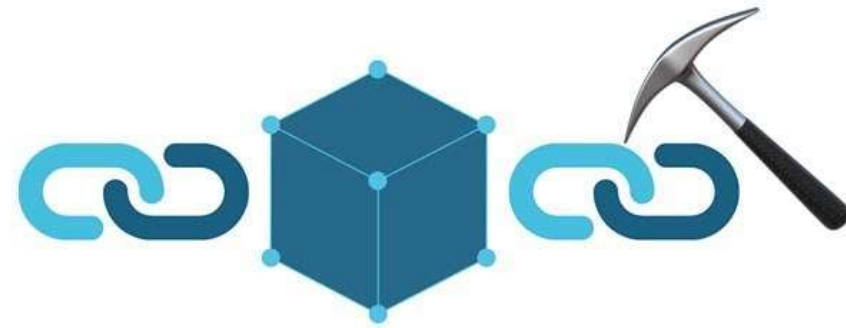

The background features a network diagram with circular nodes containing icons of a person at a laptop or a smartphone, connected by lines. Faint binary code (0s and 1s) is scattered across the entire background.

Help! [main()]

```
def main():  
    annaCoin = Blockchain()  
    annaCoin.addBlock(Block(1, "10/10/2017", {"account": "Anna", "amount": 25, "action": "buy"}))  
    annaCoin.addBlock(Block(2, "11/01/2017", {"account": "Joe", "amount": 10, "action": "buy"}))  
    annaCoin.addBlock(Block(3, "12/01/2017", {"account": "Katie", "amount": 20, "action": "buy"}))  
    annaCoin.addBlock(Block(4, "12/07/2017", {"account": "Ethan", "amount": 4, "action": "buy"}))  
    annaCoin.printBlockchain()  
    # no tampering in our block chain yet so should be true here  
    print "Chain valid? " + str(annaCoin.isChainValid())  
    # now lets tamper the block chain and see what happens  
    annaCoin.chain[1].data = {"account": "Anna", "amount": 100, "action": "buy"}  
    print "Chain valid? " + str(annaCoin.isChainValid())
```


:Summing All Up:

python



blockchain

Thank
you

