**Tasty Test Framework**

Emulator based testing

checkPredicate :: String - > TracePredicate -> EmulatorTrace () -> TestTree

TracePredicate is some sort of condition to use to run your test by

Can use not, (.&&.) for logical combinators for combing predicates

walletsFundChange :: Wallet -> Value -> TracePredicate

Check that funds in wallet have changed by given amount, excluding fees

**Brief intro to Optics and lens**

Optics are about reaching deeply into hierarchical data types

When dealing with lens, conventional to use proceeding \_ when talking about fields

Optics are like . operators in java or python

Provides way to zoom in on nested data structures

Can also compose these differents lens

Lens are field in data type

**Property based testing Example**

Will have two different wallets running the token sale

**Property Based Testing**

Quick check checks hundreds of different of scenarios from a given property to be tested

How to right property based tests with for plutus

--Start with a model

addr1qxjxcmumeqvc3j4vtsf5dtdtpw58xcdaytmewx0eyl59nlvpcqj6g8ms3llsy98edyh24uc37p89vu94q2tr2lf5c0ksgd9952

Apply the same action to both the model and real world situation

--compare the resulting states to see if they are as expected

-- shrinking in this case comes in if there is a problem, it while try to drop actions to see if the bug still persists

Example

Create a TS model with 2 wallets