As you arrive:

1. Start up your computer and plug it in
2. ***Log into Angel*** and go to CSSE 120
3. Do the ***Attendance Widget*** – the PIN is on the board
4. Don't need to go to the course ***Schedule Page*** *today*
5. ***Slides*** for today will be available on Angel after class
6. Check out today's project: **None until hw**

Plus in-class time working on and practicing these AND previous concepts.

Design, Simulation, Testing

- Designing a larger program
- Implementing a larger program
- Top-down design

Blackjack card game

- Top-level algorithm
- Design and implement using functional decomposition
- Practice using top-down design

Designing/implementing a larger program

- Until now, our programs have been small and simple
  - Possible exceptions: pizzPolyStar, speedReading
- For larger programs, we need a strategy to help us be organized
- One common strategy: **top-down design**
  - Break the problem into a few big pieces (functions)
  - Break each piece into smaller pieces
  - Eventually we get down to manageable pieces that do the details

Top-level algorithm for **Blackjack**

- Create initial card deck
- Deal initial cards
- Display game state
- Player plays until busted or chooses to stop
- Dealer plays until required to stop
- Report who wins

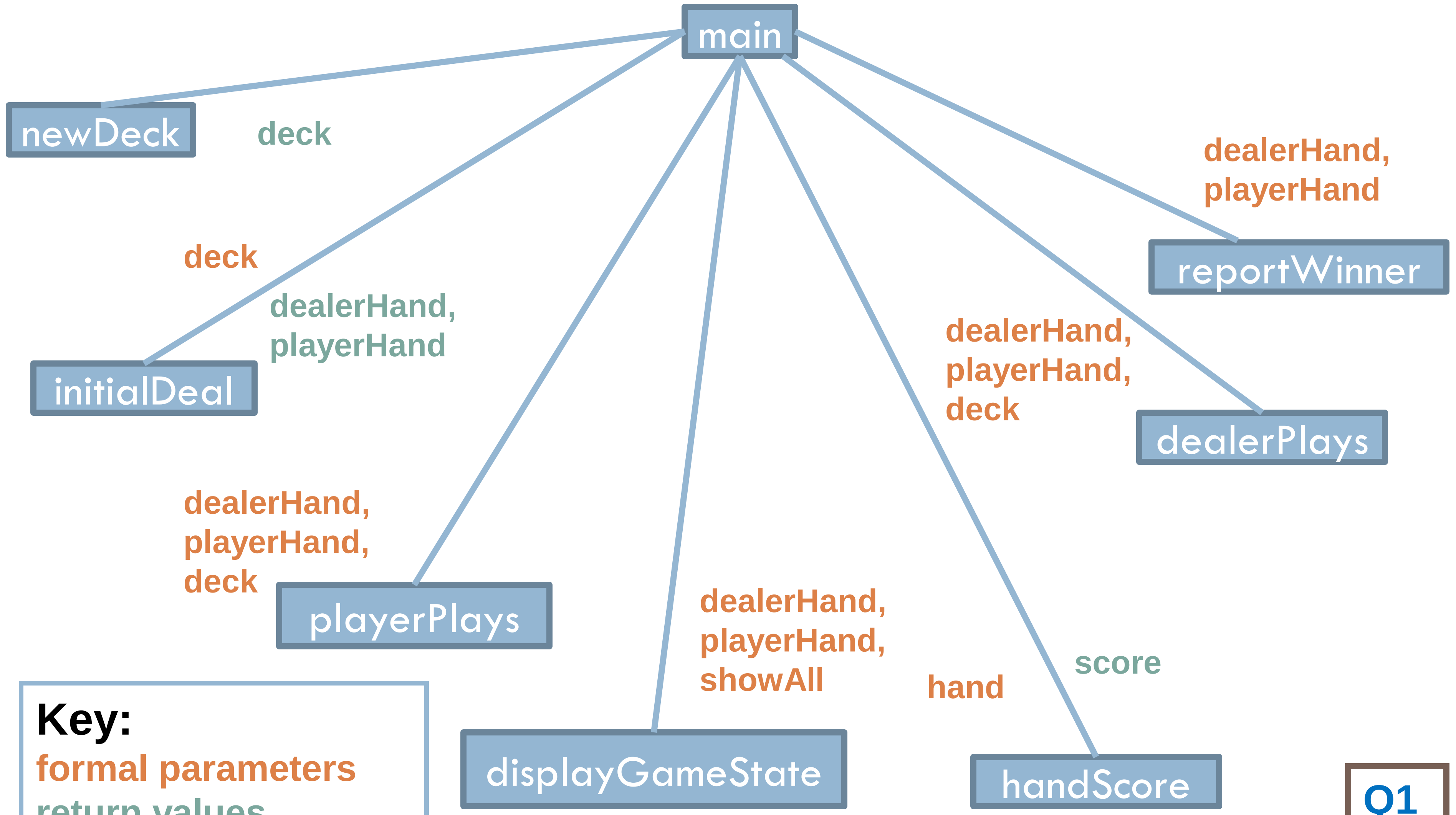
Top-level functions called by **main( )**

- newDeck()
  - Creates and returns a complete deck of cards
- initialDeal(deck)
  - deals cards from the deck to each player, returns the hands
- displayGameState(playerHand, dealerHand, showAll)
  - shows visible cards and player's scores.   showAll is boolean
- playerPlays(playerHand, dealerHand, deck)
  - Allows player to choose hit or stay
- dealerPlays(playerHand, dealerHand, deck)
  - Dealer does hit or stay, based on the rules
- reportWinner(playerHand, dealerHand)
  - Determines and displays who wins.

Complete code for main()

```
def main():
    deck = newDeck()
    player, dealer = initialDeal(deck)
    displayGameState(player, dealer, False)
    playerPlays(player, dealer, deck)
    if handScore(player) > winningScore:
        print("BUSTED!  You lose.")
    else:
        print("Now Dealer will play ...")
        dealerPlays(player, dealer, deck)
        reportWinner(player, dealer)
    displayGameState(player, dealer, True)
```

Top-level Structure Diagram



Key:
formal parameters
return values

Q1

Some preliminary data values

```
# Define some constants used by many functions
suits = ['Clubs', 'Diamonds', 'Hearts', 'Spades']
cardNames = ['Ace', 'Deuce', '3', '4', '5',
             '6', '7', '8', '9', '10',
             'Jack', 'Queen', 'King']
winningScore = 21
dealerMustHoldScore = 16

# Card is represented by a list: [cardName, suit]
# Examples: ['Ace','Clubs'] or ['7','Diamonds']
# A hand or a deck is a list of cards.
```

A List of lists

Q2

Designing **newDeck()**

- Write steps of **newDeck()** in English
- Write the code
- Refer to:
  - Data values on handout
  - Structure diagram on handout

newDeck()—returns a complete deck

- start with an empty list
- for each cardName/suit pair
  - generate a card with that name and suit
  - add card to list
- Return the list

```python
# Create an entire deck of cards
def newDeck():
    deckList = []
    for s in suits:
        for c in cardNames:
            deckList.append([c, s])
    return deckList
```

Designing **initialDeal(deck)**

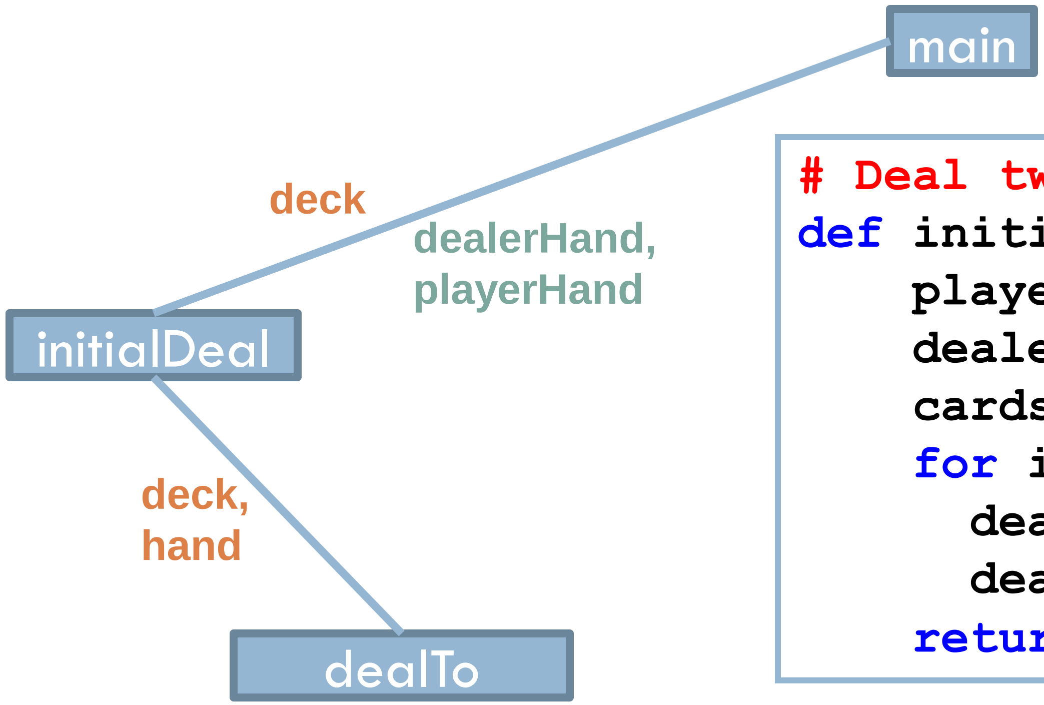
- Work in groups of 4 at a whiteboard
- Write steps for **initialDeal(deck)** function in English
- Write the code
- Take about 10 minutes
- Refer to:
  - Data values on handout
  - Structure diagram on handout
  - **Do you need new functions? Add them to your structure chart**

initialDeal(deck)-returns two hands

- start with two empty hands
- deal two cards to each hand
- return the two hands

```
# Deal two cards to each player.
def initialDeal(deck):
    playerHand = []
    dealerHand = []
    cardsToDeal = 2
    for i in range(cardsToDeal):
      dealTo(playerHand, deck)
      dealTo(dealerHand, deck)
    return playerHand, dealerHand
```

initialDeal Structure Diagram



```
# Deal two cards to each player.
def initialDeal(deck):
    playerHand = []
    dealerHand = []
    cardsToDeal = 2
    for i in range(cardsToDeal):
      dealTo(playerHand, deck)
      dealTo(dealerHand, deck)
    return playerHand, dealerHand
```

Key:
formal parameters
return values

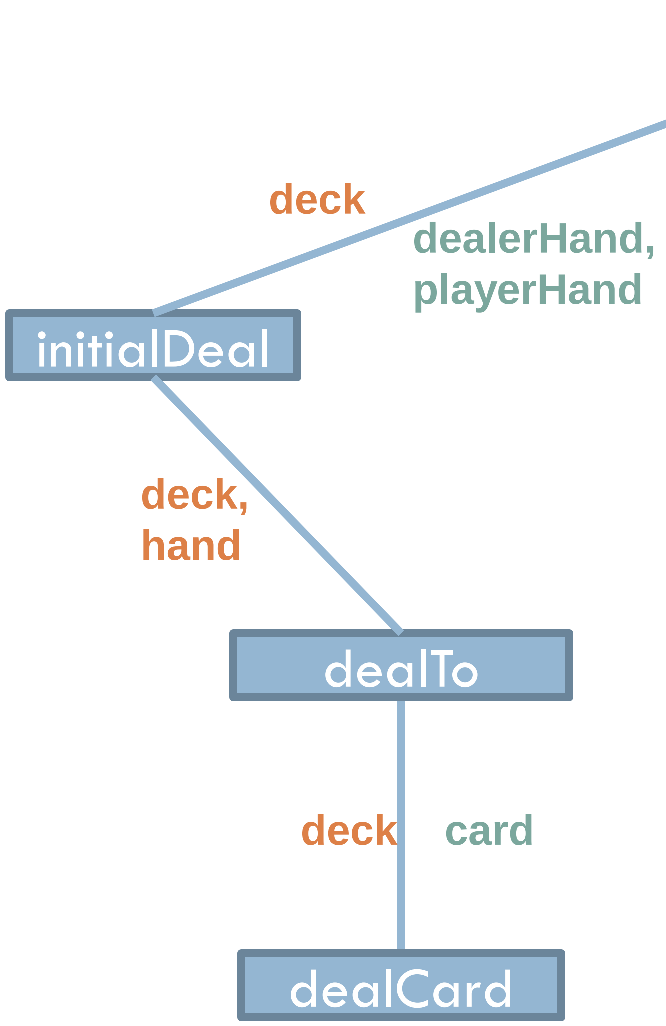
Q3-4

dealTo(hand, deck)

- Pick a random card from the deck and move it to the hand

```
# deal a card from this deck and place it in this hand.
def dealTo(hand, deck):
    hand.append(dealCard(deck))
```

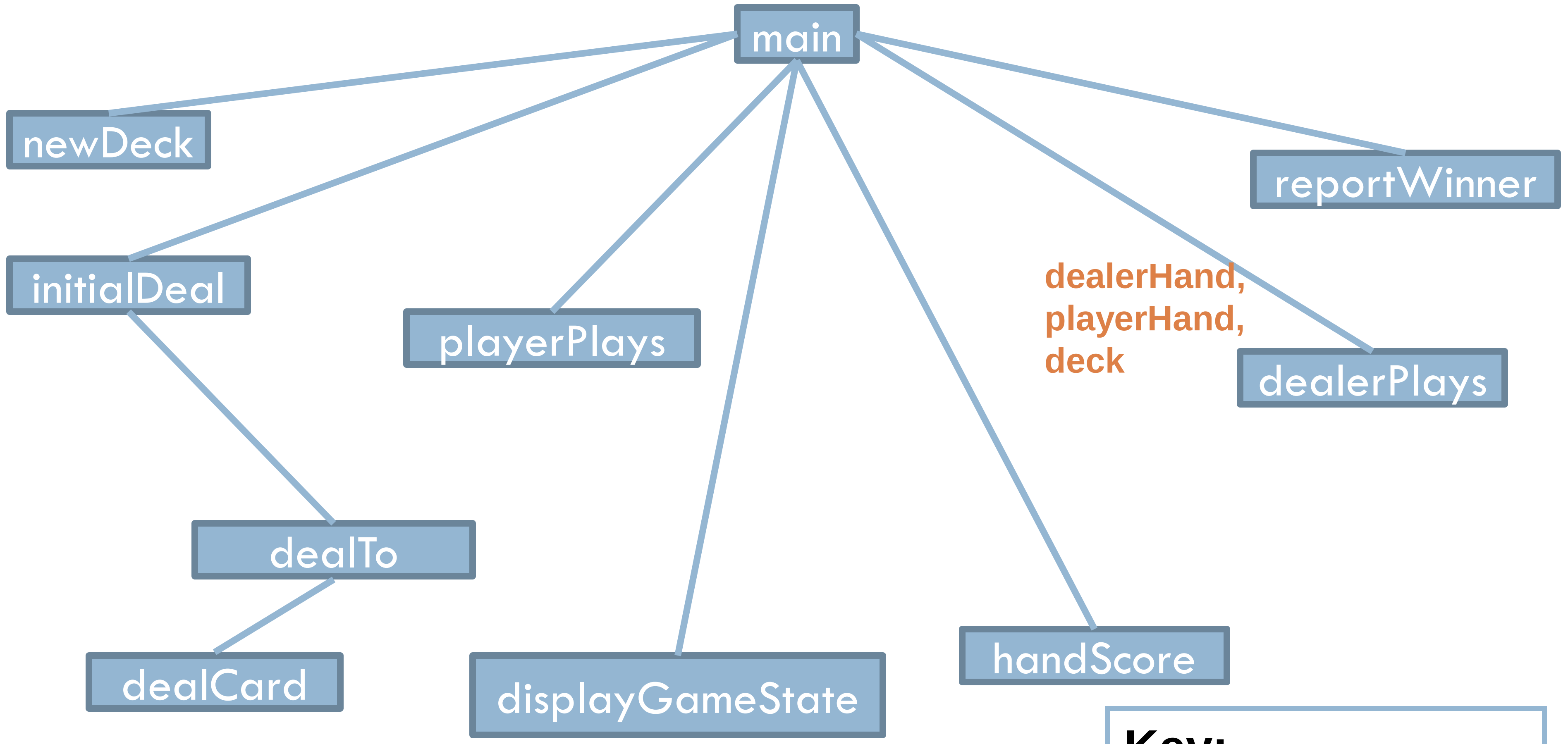
initialDeal Structure Diagram



```
# Remove  a random card from
# the deck and return it
def dealCard(deck):
    pos = randrange(len(deck))
    card = deck[pos]
    deck.remove(card)
    return card
```

Key:
formal parameters
return values

Let's skip ahead to dealerPlays( )



Key:
formal parameters
return values

Designing **dealerPlays()**

- Work in groups of 4 at a whiteboard
- Write steps of **dealerPlays()** in English
- Write the code:
  - **Do you need new functions? Add them to your structure chart**
- Take about 10 minutes

dealerPlays

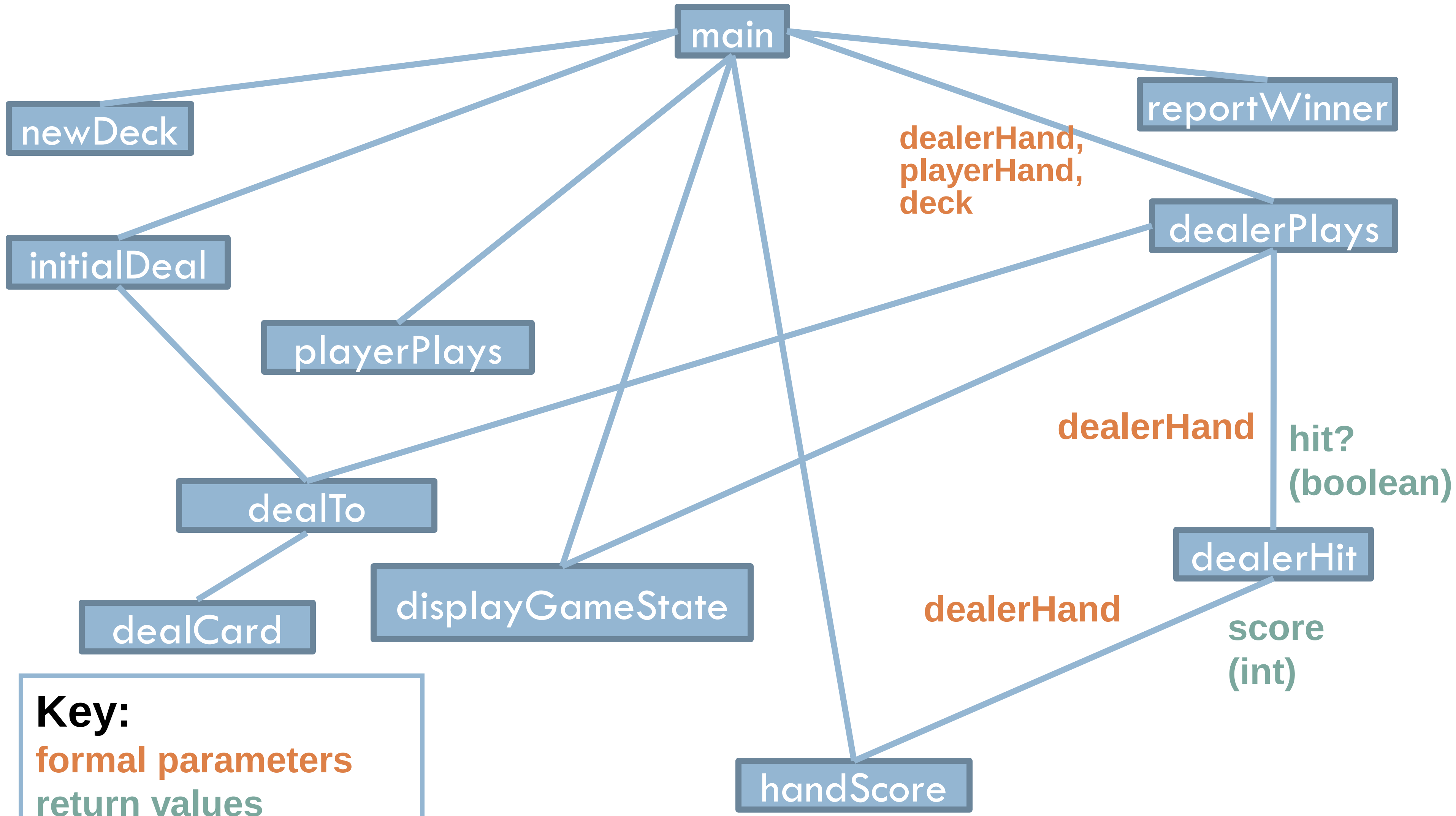
- while dealerMustTakeaHit
  - deal a card to Dealer's hand

```
# Dealer takes hits until no more hits allowed.
def dealerPlays(player, dealer, deck):
    displayGameState(player, dealer, True)
    while dealerHit(dealer):
        sleep(3)
        print("Dealer takes a hit")
        dealTo(dealer, deck)
        displayGameState (player, dealer, True)
```

Note use of dealTo() function

```
# Determine whether dealer "takes a hit" (gets another card).
def dealerHit(dealerHand):
    dealerScore = handScore(dealerHand)
    return dealerScore < dealerMustHoldScore
```

Design so far



Key:
formal parameters
return values

Code for handScore( )

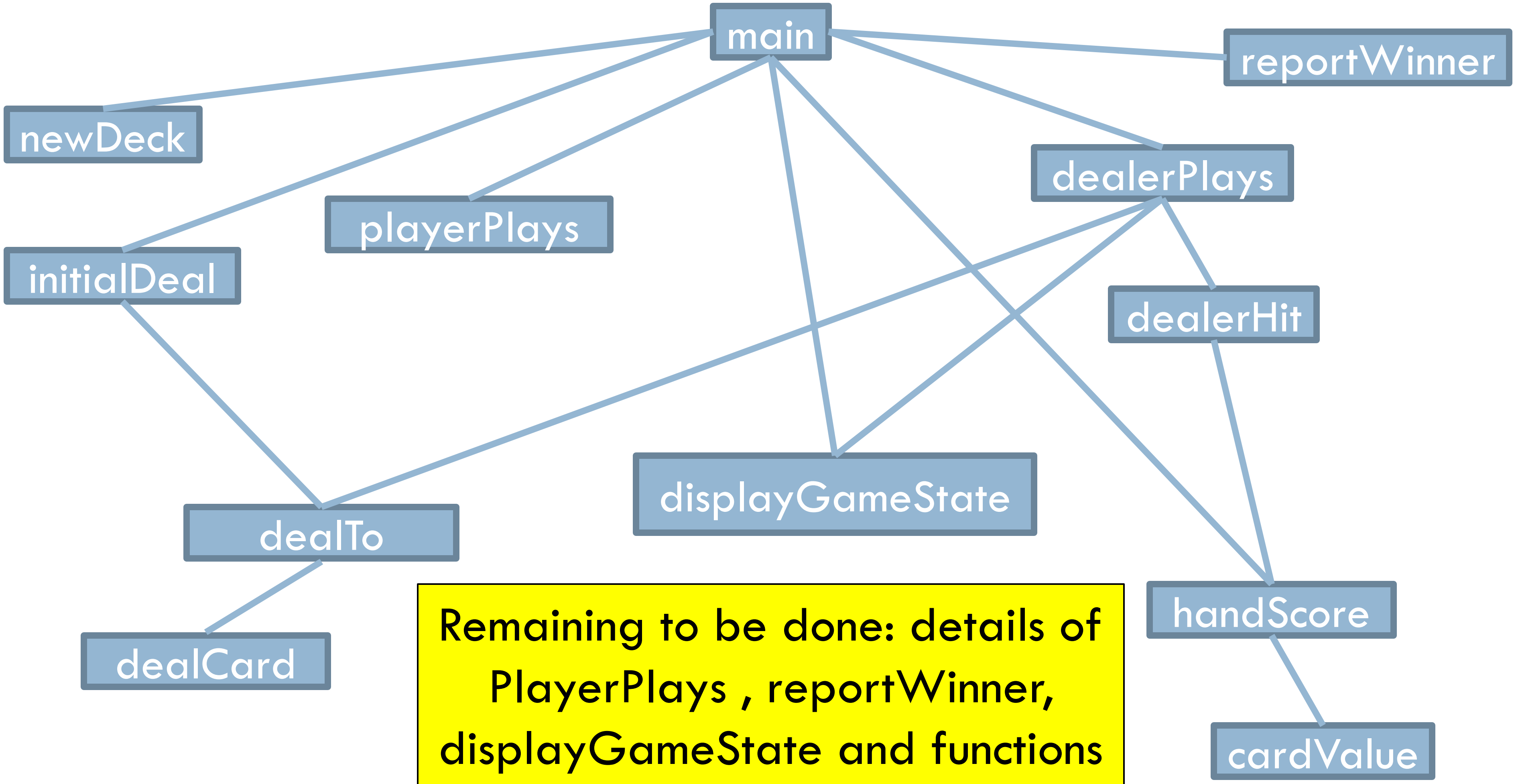
```
# Calculate the score for the whole hand.
def handScore(hand):
    score = 0
    hasAce = False
    for card in hand:
        val = cardValue(card)
        score += val
        if val == 1:
            hasAce = True
    if score <= winningScore - 10 and hasAce:
        score = score + 10
    return score
```

What if they have two or more aces?

Code for cardValue( )

```
# calculate how many points this card is worth.
# Face cards count 10.
# Ace Counts 1 (or 11, but that adjustment is
#                made at the handScore level).
def cardValue(card):
    name = card[0]
    pos = cardNames.index(name)
    if pos < 10:  # if not a face card.
        return pos + 1
    return 10
```

What we have developed so far

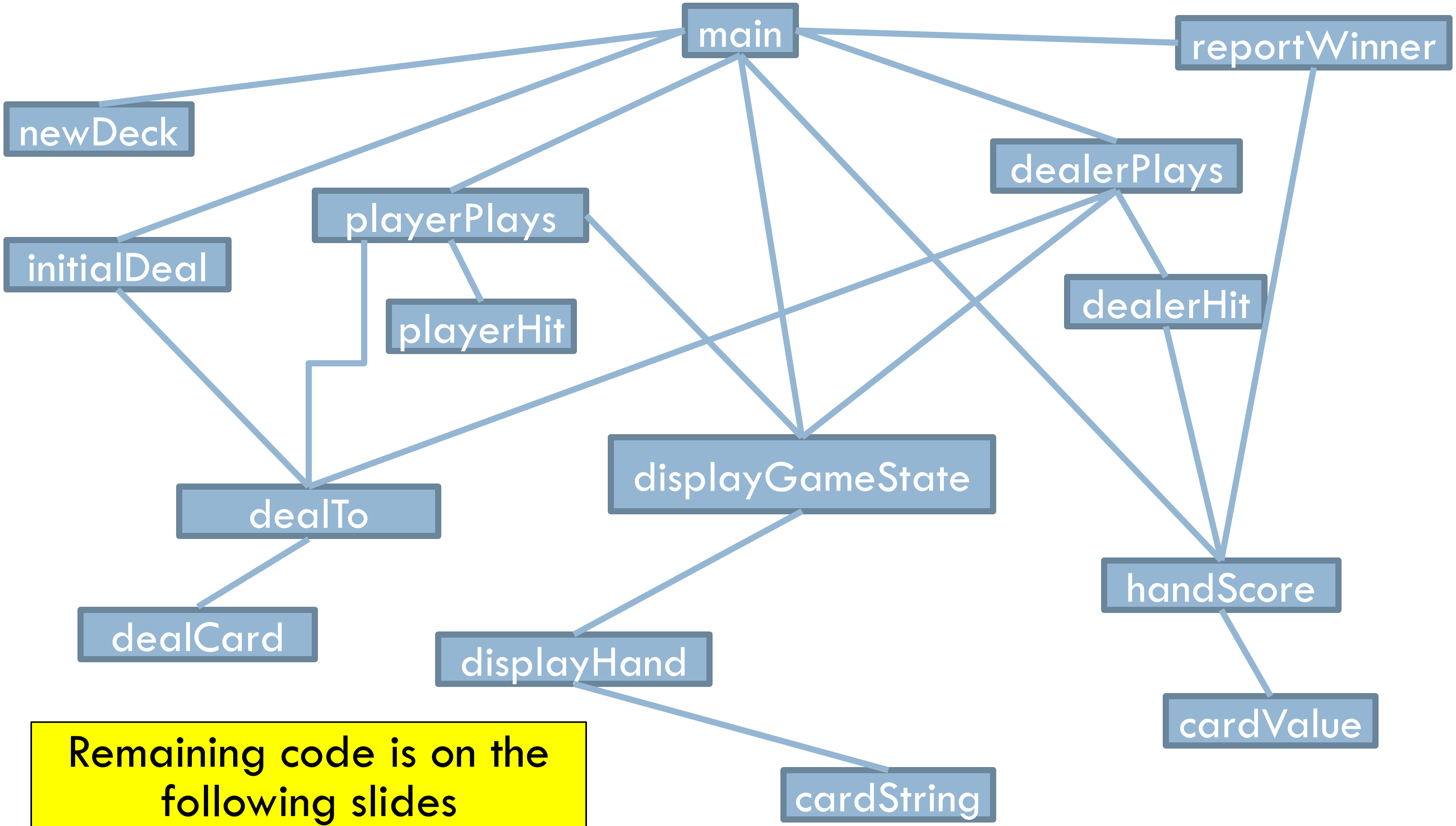


Remaining to be done: details of PlayerPlays , reportWinner, displayGameState and functions that they call

Bottom-up Testing

- If we wrote all of this code and tried to run it all together, there would probably be so many errors that it would be very hard to track down their causes
- So instead of testing the whole program at once, we want to test each function individually.
- To do this, we want to start with functions at the bottom of the structure chart, because they do not depend on other functions
- Tests of individual functions are called **Unit Tests**

Complete Structure Diagram



Remaining code is on the following slides

The display functions

```
# Show the contents of both players' hands.
def displayGameState(playerHand, dealerHand, gameOver):
    displayHand('Dealer', dealerHand, gameOver)
    displayHand('Player', playerHand, True)

# print out the contents of this hand. If the hand is the dealer's
# and the player hasn't played yet, showAll will be False.
def displayHand(name, hand, showAll):
    print(name + "'s hand:", end= " ")
    if showAll:
        print("(score is {})".format(handScore(hand)))
        print cardString(hand[0])
    else:
        print()
        print('    Face Down')
    # print the rest of the hand.
    for i in range(1, len(hand)):
        print(cardString(hand[i]))

# return a string that represents the given card.
def cardString(card):
    return '    ' + card[0] + " of " + card[1]
```

playerPlays and PlayerHit

```
# Player takes hits until Busted or stops requesting
hits.
def playerPlays(player, dealer, deck):
    while playerHit(handScore(player)):
        dealTo(player, deck)
        displayGameState(player, dealer, False)


# Ask player whether she wants another card.
def playerHit(playerScore):
    if playerScore > winningScore:
        return False
    answer = input("Hit? (Y/N) ")
    return answer[0].lower() == 'y'
```

reportWinner function

```
# Figure out who won.
def reportWinner(player, dealer):
    playerScore = handScore(player)
    dealerScore = handScore(dealer)
    if dealerScore > winningScore:
        print("DEALER IS BUSTED, YOU WIN")
    elif dealerScore > playerScore:
        print("DEALER WINS")
    else:
        print("YOU WIN!")
```