Translating to Code

by Sophia

WHAT'S COVERED

In this lesson, we will learn about the initial steps to translate pseudocode to code. Specifically, this lesson covers:

1. Coding Framework
2. Guided Brainstorming

1. Coding Framework

Looking back at the pseudocode that we created for the demonstration program, we first defined the output of what the program should have. When coding the initial parts of the program, it can be useful to start by adding all of the elements of our plan as comments within our actual project in the IDE. Remember that each commented line should start with a hashtag symbol ("#"). At this stage, this can be a simple task since we can use our existing pseudocode both as the foundation of what to code as well as our starting comments. Using the pseudocode, we can make it all comments (line by line).

EXAMPLE

```
#Function play()
    #Set the rolled value to roll first dice + roll second dice
    #If the rolled value is equal to 2, 3 or 12
      #Set player status to lose
    #Else If the rolled value is equal to 7 or 11
      #Set player status to win
    #Else
      #Set point value to rolled value
      #While the player status is not set, run the following
        #Set the rolled value to roll first dice + roll second dice
        #If the rolled value is equal to 7
          #Set player status to lose
        #Else If rolled value is equal to point value
```

```
            #Set player status to win

#Main Program
    #Set Wins = 0
    #Set Losses = 0
    #Set Total of Win Rolls = 0
    #Set Total of Loss Rolls = 0
    #Set Games Played to 0

    #Input Times to Play from user
    #Loop until Games Played = Times to Play
      #Add 1 to Games Played
      #Call function play()
      #Get if the player won or lost
      #Get the number of rolls
      #If the player won
        #Add 1 to Wins
        #Add the rolls values to the Total of Win Rolls
      #Else If player lost
        #Add 1 to Losses
        #Add the rolls values to the Total of Loss Rolls

    #Average Number of Rolls Per Win = Total of Win Rolls / Wins
    #Average Number of Rolls Per Loss = Total of Loss Rolls / Loss
    #Output results
```

TRY IT

Directions: Try adding these comments of the demonstration program to the IDE and follow along with the coding build.

KEY CONCEPT

Next, it can be helpful to identify the variables that we are planning to use in our program and start to define them in Python. Generally, it's a good idea to initialize numeric values to 0 or its intended value and strings to an empty string or their intended values. This way, we won't be surprised by incorrect values being displayed.

Let's start our coding with the main program first. We will call it the `main()` function.

EXAMPLE

```
#Main Program
def main():

    #Set Wins = 0
```

```
    #Set Losses = 0
    #Set Total of Win Rolls = 0
    #Set Total of Loss Rolls = 0
    #Set Games Played to 0
    wins = 0
    losses = 0
    winRolls = 0
    lossRolls = 0
    gamesPlayed = 0
```

Once we have those variables defined, we can group all of those comments together into a comment called "initializing the variables":

```
#Main Program
def main():
    #initializing the variables
    wins = 0
    losses = 0
    winRolls = 0
    lossRolls = 0
    gamesPlayed = 0
```

Then, we can start working on the other parts of the program:

```
    #Input Times to Play from user
    #Loop until Games Played = Times to Play
      #Add 1 to Games Played
      #Call function play()
      #Get if the player won or lost
      #Get the number of rolls
      #If the player won
        #Add 1 to Wins
        #Add the rolls values to the Total of Win Rolls
      #Else If player lost
        #Add 1 to Losses
        #Add the rolls values to the Total of Loss Rolls
```

BIG IDEA

Notice we have a function called `play()`. This is not something that we have yet, so we'll need to just leave it in a comment for now. As we work through each of the steps, we'll want to fill in the items and segments of code that we're able to do and move to the next. Rarely will we have all of our code at once, but it's a good habit to first define our variables using a consistent naming scheme where the names of the variables are descriptive.

Next, let's code the input that we need the user to enter.

```
#Input Times to Play from user
gamesPlayed = int(input("Enter in the times to play:"))
```

Note that not all of the comments will end up being coded. For example, if the next part to complete is our loop, depending on which type of loop we use, we may not need to add one to the number of the `gamesPlayed` variable. A `for` loop would automatically do that for us:

```
#Loop until Games Played = Times to Play
for count in range(gamesPlayed):
  #Call function play()

  hasWon = True
  #Get if the player won or lost

  rolls = 0
  #Get the number of rolls

#If the player won
  #Add 1 to Wins
  #Add the rolls values to the Total of Win Rolls
#Else If player lost
  #Add 1 to Losses
  #Add the rolls values to the Total of Loss Rolls
if hasWon:
   wins += 1
   winRolls += rolls
else:
   losses += 1
   lossRolls += rolls
```

Take notice that there are parts that are incomplete based on the order in which we are creating the code, but this is quite normal to have for a program. For example, in this program, we still need to define a play function and a way to determine if the player has won or lost. If they won, we need to increment the number of wins. Then, we need to get the number of rolls that the player took. Our if/else statement was able to be implemented but the rolls would need to also be set.

We can also enter code for our statistics based on the results:

```
#Calculating the statistics
print("The total number of wins is", wins)
print("The total number of losses is", losses)
```

```
    print("The average number of rolls per win is %0.2f" % \
          (winRolls / wins))
    print("The average number of rolls per loss is %0.2f" % \
          (lossRolls / losses))
    print("The winning percentage is %0.3f" % (wins / gamesPlayed))
    print("The multi-game has been saved into the log.")
```

Once we have some of these items in place, we'll have to break down the functionality a bit further in terms of the game logic as a whole. At this point, we could create some functions without many details and start some simple coding too. For example, we know we'll have a play function. Instead of implementing it fully right away, we can just create a start to it:

EXAMPLE

```
def play():
  pass
#Function play()
#Set the rolled value to roll first dice + roll second dice
#If the rolled value is equal to 2, 3 or 12
        #Set player status to lose
#Else If the rolled value is equal to 7 or 11
        #Set player status to win
#Else
        #Set point value to rolled value
        #While the player status is not set, run the following
                #Set the rolled value to roll first dice + roll second dice
                #If the rolled value is equal to 7
                        #Set player status to lose
                #Else If rolled value is equal to point value
                        #Set player status to win
```

Remember that the `pass` statement simply means that the body of the function or statement does nothing. By using the `pass` statement, it allows you to call the function and not receive any errors. This is perfectly acceptable at this point and helps to build the structure of the program. We will expand on the core functionality in the upcoming lesson.

TRY IT

Directions: Go ahead and try to add each of the code snippets since the last Try It and see if you follow the build. If you need help, the code so far is located below.

```
def play():
  pass
#Function play()
```

```
    #Set the rolled value to roll first dice + roll second dice
    #If the rolled value is equal to 2, 3 or 12
      #Set player status to lose
    #Else If the rolled value is equal to 7 or 11
      #Set player status to win
    #Else
      #Set point value to rolled value
      #While the player status is not set, run the following
        #Set the rolled value to roll first dice + roll second dice
        #If the rolled value is equal to 7
          #Set player status to lose
        #Else If rolled value is equal to point value
          #Set player status to win

#Main Program
def main():
    #initializing the variables
    wins = 0
    losses = 0
    winRolls = 0
    lossRolls = 0
    gamesPlayed = 0

    #Input Times to Play from user
    gamesPlayed = int(input("Enter in the times to play:"))

    #Loop until Games Played = Times to Play
    for count in range(gamesPlayed):
      #Call function play()

      hasWon = True
      #Get if the player won or lost

      rolls = 0
      #Get the number of rolls

    #If the player won
      #Add 1 to Wins
      #Add the rolls values to the Total of Win Rolls
    #Else If player lost
      #Add 1 to Losses
      #Add the rolls values to the Total of Loss Rolls
```

```
    if hasWon:
        wins += 1
        winRolls += rolls
    else:
        losses += 1
        lossRolls += rolls

#Average Number of Rolls Per Win = Total of Win Rolls / Wins
#Average Number of Rolls Per Loss = Total of Loss Rolls / Loss
#Output results
```

---

2. Guided Brainstorming

The goal of these initial steps is for us to understand how to break down our pseudocode into smaller pieces so that we can implement them part by part. It's a good idea to identify all of the variables needed first. As we do that, we can then move on to breaking down any functions or methods that we have. In this step, we can use the `pass` statement to create a function or method without fully implementing the functions.

As we start to write our code, we can start to remove comments for logic that we've fully implemented. We can also think about the bigger picture. Beyond the pure logic of the program, think about the program that we plan to build.

Remember that with our Drink Order program, we had defined the following pseudocode.

↷ EXAMPLE

If drink is equal to water
    Ask user to enter in hot or cold
    Store input in heat
    If heat equal to hot
        Add hot to outputString
    Else If heat equal to cold
        Add cold to outputString
        Ask user to enter in ice or not
        If ice is yes
            Add ice to outputString
        Else
            Add no ice to output String
Else If drink is equal to coffee
    Ask user to enter in decaf or not
    Store input in decaf
    If decaf equal to Yes

```
        Add decaf to outputString
    Ask user to enter in Milk, cream or none
    Store input in milkCream
    If milkCream equal to milk
        Add milk to outputString
    Else If milkCream equal to cream
        Add cream to outputString
    Ask user to enter in sugar or not
    Store input in sugar
    If sugar equal to Yes
        Add sugar to outputString
Else If drink is equal to tea
    Ask user to enter in teaType
    Store input in teaType
    If teaType equal to green
        Add green to outputString
    Else If teaType equal to black
        Add black to outputString
print outputString
```

The converted code ends up being fairly simple as we'll have the hashtag (#) being added to each line as a comment.

↷ EXAMPLE

```
#If drink is equal to water
    #Ask user to enter in hot or cold
    #Store input in heat
    #If heat equal to hot
      #Add hot to outputString
    #Else If heat equal to cold
      #Add cold to outputString
      #Ask user to enter in ice or not
      #If ice is yes
        #Add ice to outputString
      #Else
        #Add no ice to output String
#Else If drink is equal to coffee
    #Ask user to enter in decaf or not
    #Store input in decaf
    #If decaf equal to Yes
      #Add decaf to outputString
    #Ask user to enter in Milk, cream or nonet
```

```
    #Store input in milkCream
    #If milkCream equal to milk
      #Add milk to outputString
    #Else If milkCream equal to cream
      #Add cream to outputString
    #Ask user to enter in sugar or not
    #Store input in sugar
    #If sugar equal to Yes
      #Add sugar to outputString
#Else If drink is equal to tea
    #Ask user to enter in teaType
    #Store input in teaType
    #If teaType equal to green
```

For our Drink Order program, we'll implement the functionality that's in our pseudocode in the next lesson.

TRY IT

Directions: This lesson is meant to just prep your program for conversion to code. It always helps to just comment out your pseudocode from the beginning and start to fill in sections of the code that you're comfortable with in the same steps that we took. Now is a good time to convert your pseudocode into comments for your project.

SUMMARY

In this lesson, we started to translate the pseudocode into code. To add the **coding framework**, we took the demonstration program's pseudocode from our past lesson and added all of its lines as comments in our coded program. This is a great starting point since the lines of pseudocode can act as both our foundation of what to code as well as our starting comments. We started to code our demonstration program by identifying the variables that we are planning to use in the program. We also coded some initial functions and expected output. Then, in the **Guided Brainstorming** section, we converted the pseudocode of the Drink Order program as comments to set that example up as well.

Best of luck in your learning!