



IIT Madras

ONLINE DEGREE

Computational Thinking

Indian Institute of Technology, Madras

Tutorial 3.3

(Refer Slide Time: 00:12)



```
SumBB = 0
CountBB = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumBB, CountBB = AddIfBigBazaar(X, SumBB, CountBB)
    Move X to Pile 2
}
AvgBB = SumBB/CountBB
```

```
Procedure AddIfBigBazaar(X, SumBB, CountBB) {
    if (X.ShopName == "Big Bazaar") {
        SumBB = SumBB + X.TotalBillAmount
        CountBB = CountBB + 1
    }
    return([SumBB, CountBB])
End AddIfBigBazaar
```

Hello CT students. In this tutorial, we are going to look at 3 common errors with respect to using procedures in your code. So here, we have a piece of code where, let us just go through it step by step, we have 2 variables, SumBB and CountBB, both being initialized to 0. And this, there is a BigBazaar mentioned here. So, BB is basically BigBazaar. And now, we are going through the cards in the pile. And we are reading the card at top to the pile.

And now we have this statement where we are assigning Sum BB and CountBB to the result of this procedure called AddIfBigBazaar. And to this procedure, we are sending these parameters which are, X is the card, SumBB is 1 variable, which is likely the sum variable and CountBB is likely the count variable, both for BigBazaar. These are the 3 parameters, we are sending to this procedure. And whatever output comes out, that we are taking to be SumBB and CountBB's new values. And then we move on to the next card.

And the end of going through all the pile, when the while loop is done, we are basically calculating Average BB, AvgBB, which is the Sum by the Count, Sum divided by the Count. So, we are essentially calculating the average for BigBazaar cards. Now, let's look at what is the code for AddIfBigBazaar, that procedure. So, here we have this procedure AddIfBigBazaar, which takes an X, a SumBB and a CountBB as parameters. And in this procedure, we are checking if the ShopName of the card is Big Bazaar.

So, this is the first check, we are doing. We are interested only in BigBazaar cards. So, we are checking if the ShopName is BigBazaar. If it is BigBazaar, then to the sum variable we are adding the TotalBillAmount of that card X. So, if it is BigBazaar, we are adding that bill amount to our sum variable, which means our sum variable represents the TotalBillAmount of BigBazaar overall. At the end of the whole task, this is what we are going to get to.

And CountBB is being incremented by 1, if the card is of BigBazaar, which means we are counting the number of BigBazaar cards. And then the procedure returns, SumBB and CountBB, after having done these modifications. So, this is what is being returned by this procedure here. And therefore, we are assigning SumBB and CountBB to their new values in this way. That end of the procedure, we are saying End the procedure. So, there are a few ways this could go wrong. This is basically the use of the procedure.

(Refer Slide Time: 03:47)



#1 Wrong Procedure Name

And let us look at the first way. In the first way, you could simply be writing the wrong procedure name.

(Refer Slide Time: 03:54)



```
SumBB = 0
CountBB = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumBB, CountBB = AddingIfBigBazaar(X, SumBB, CountBB)
    Move X to Pile 2
}
AvgBB = SumBB/CountBB
```

```
Procedure AddingIfBigBazaar(X, SumBB, CountBB) {
    if (X.ShopName == "Big Bazaar") {
        SumBB = SumBB + X.TotalBillAmount
        CountBB = CountBB + 1
    }
    return([SumBB, CountBB])
End AddingIfBigBazaar
```

Here, I have put AddingIfBigBazaar, which could be the procedure's name, but it is not. The actual procedure's name is AddIfBigBazaar. So, this is only going to work if this is changed, the procedure name is changed to AddIfBigBazaar or we should change the usage of the name here, the procedure name here. So basically, when you are calling a procedure, when you are using a procedure, you have to use the actual name given in the definition. So, this is the first common error.

(Refer Slide Time: 04:37)



#2 Wrong number of parameters

In the next kind.

(Refer Slide Time: 04:41)



```
SumBB = 0
CountBB = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumBB, CountBB = AddIfBigBazaar(X, SumBB, CountBB, AvgBB)
    Move X to Pile 2
}
AvgBB = SumBB/CountBB
```

```
Procedure AddIfBigBazaar(X, SumBB, CountBB) {
    if (X.ShopName == "Big Bazaar") {
        SumBB = SumBB + X.TotalBillAmount
        CountBB = CountBB + 1
    }
    return([SumBB, CountBB])
End AddIfBigBazaar
```

Here, the problem is, we have these parameters that the procedure definition is expecting. These are the 3 parameters, which is card X, SumBB and CountBB variables. And when we are calling the procedure however, we are taking Sum, the card X, the SumBB variable, the Count BB variable, but we are also trying to pass the Average BB variable, which means it is more parameters than actually given in the definition.

(Refer Slide Time: 5:15)



```
SumBB = 0
CountBB = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumBB, CountBB = AddIfBigBazaar(X, SumBB)
    Move X to Pile 2
}
AvgBB = SumBB/CountBB
```

I

```
Procedure AddIfBigBazaar(X, SumBB, CountBB) {
    if (X.ShopName == "Big Bazaar") {
        SumBB = SumBB + X.TotalBillAmount
        CountBB = CountBB + 1
    }
    return([SumBB, CountBB])
End AddIfBigBazaar
```

A similar error would be if you forgot to give all the parameters required. So here, I have given only 2 parameters. This is also not okay. Even in this case, your code is going to fail.


(Refer Slide Time: 05:28)



#3 Incorrect order of parameters

Now, in the third error.

(Refer Slide Time: 05:33)



```
SumBB = 0
CountBB = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumBB, CountBB = AddIfBigBazaar(X, CountBB, SumBB)
    Move X to Pile 2
}
AvgBB = SumBB/CountBB

Procedure AddIfBigBazaar(X, SumBB, CountBB) {
    if (X.ShopName == "Big Bazaar") {
        SumBB = SumBB + X.TotalBillAmount
        CountBB = CountBB + 1
    }
    return([SumBB, CountBB])
End AddIfBigBazaar
```

```

SumBB = 0
CountBB = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumBB, CountBB = AddIfBigBazaar(X, SumBB, CountBB)
    Move X to Pile 2
}
AvgBB = SumBB/CountBB

```

```

Procedure AddIfBigBazaar(X, SumBB, CountBB) {
    if (X.ShopName == "Big Bazaar") {
        SumBB = SumBB + X.TotalBillAmount
        CountBB = CountBB + 1
    }
    return([SumBB, CountBB])
End AddIfBigBazaar

```

We are calling the AddIfBigBazaar correctly. We have the exact number of parameters expected, which is 3. And this is card X, but this is the count variable and this is the sum variable, whereas here, the sum variable is second and the count variable is third. What is going to happen here is, the procedure is going to assume that this second value that you are sending to it is the sum variable. As well as assume that this third value that you are sending to it is the count variable.

And within it, you will see this exchange of assignments, the assignments are going to be wrong. So, your code is going to fail. So, in order to avoid this, you should always make sure that your parameters are given in the exact same order as the definition of the procedure expects. Only then when the procedure be executed as you require. Thank you