# **Lab 4: Decisions and Boolean Logic**

This lab accompanies Chapter 4 of *Starting Out with Programming Logic & Design*.

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Lab 4.1 –Logical Operators and Dual Alternative Decisions**

|  |
| --- |
| Critical Review  The logical AND operator and the logical OR operator allow you to connect multiple Boolean expressions to create a compound expression.  The logical NOT operator reverses the truth of a Boolean expression.  When using the AND operator, both conditions must be true in order for the statements within an if to process.  When using the OR operator, either condition must be true in order for the statements within an if to process.  A dual alternative decision structure will execute one group of statements if its Boolean expression is true, or another group if its Boolean expression is false.  The general structure of an if-then-else statement is  If condition Then  *Statement*  *Statement*  *Etc.*  Else  *Statement*  *Statement*  *Etc.*  End If |

This lab requires you to think about possible true and false conditions using if statements.

**Step 1:** Consider the following values set to variables.

* myAge = 32
* yourAge = 18
* myNumber = 81
* yourNumber = 17
* votingAge = 18

**Step 2:** Based on the values to the variables in Step 1, what is the expected output? Hint: The output will be either what is printed to the screen, or nothing. (Reference: Logical Operators, page 189).

|  |  |
| --- | --- |
| **The condition** | **Expected Output** |
| If myAge == 31 AND yourAge < myAge Then  Display "My age is 31 and your age is less than that"  End If |  |
| If myAge <= 35 AND myAge >= 32 Then  Display "My age is between 32 and 35"  End If | My age is between 32 and 35 |
| If yourAge == votingAge OR yourAge > votingAge Then  Display "You can vote"  End If | You can vote |
| If myNumber == 83 OR yourNumber == 83 Then  Display "One of our numbers is 83"  End If |  |

**Step 3:** Based on the values to the variables in Step 1, what is the expected output? (Reference: Dual Alternative Decision Structures, page 167).

|  |  |
| --- | --- |
| **The condition** | **Expected Output** |
| If myAge == 31 AND yourAge < myAge Then  Display "My age is 31 and your age is less than that"  Else  Display "Our ages do not qualify"  End If | Our ages do not qualify |
| If myAge <= 35 AND myAge >= 32 Then  Display "My age is between 32 and 35"  Else  Display "My age is not within that range"  End If | My age is between 32 and 35 |
| If yourAge == votingAge OR yourAge > votingAge Then  Display "You can vote"  Else  Display "You cannot vote"  End If | You can vote |
| If myNumber == 83 OR yourNumber == 83 Then  Display "One of our numbers is 83"  Else  Display "83 is not our numbers"  End If | 83 is not our numbers |

**Lab 4.2 – Pseudocode: Dual Alternative Decisions**

|  |
| --- |
| Critical Review  A dual alternative decision structure will execute one group of statements if its Boolean expression is true, or another group if its Boolean expression is false.  The general structure of an if-then-else statement is:  If condition Then  Statement  Statement  Etc.  Else  Statement  Statement  Etc.  End If  Module Review  Recall the difference between a reference variable and a value variable. Reference variables are used in the following lab when the value of the variable is modified in the module. You’ll notice some parameter lists include the keyword Ref before the variable that is going to change within the module. |

This lab requires you to think about the steps that take place in a program by writing pseudocode.

Recall the retail company program from Lab 3.2. The company now wants to modify their bonus portion to include different levels and types and eliminate the day off program. The new program is as follows:

A retail company assigns a $5000 store bonus if monthly sales are more than $100,000; otherwise a $500 store bonus is awarded. Additionally, they are doing away with the previous day off program and now using a percent of sales increase to determine if employees get individual bonuses. If sales increased by at least 4% then all employees get a $50 bonus. If they do not, then individual bonuses are 0.

**Step 1:** To accommodate the changes to the program, create the additional variables needed.

* Create a variable named storeAmount to hold the store bonus amount.
* Create a variable named empAmount to hold the individual bonus amount.
* Create a variable named salesIncrease to hold the percent of increase.

// Declare local variables

Declare Real monthlySales

Declare Real storeAmount

Declare Real empAmount

Declare Real salesIncrease

**Step 2:** The first module in the program is getSales(). Since this is still required, leave this module as is. This module should be written as follows:

// MODULE 1

// this module takes in the required user input

Module getSales(Real Ref monthlySales)

Display "Enter the total sales for the month."

Input monthlySales

End Module

**Step 3:** The second module in the program was isBonus(). Since there are two types of bonuses now, rename this module and the module call to storeBonus(). Write an if-then-else statement within this module that will set the bonus amount to either 5000 or 500. Also, pass the variable storeAmount to the module as a reference. Complete the missing lines. (Reference: Dual Alternative Decision Structures, page 167).

// MODULE 2

// this module will determine what the bonus levels are

Module storeBonus(Real monthlySales, Real Ref storeAmount)

If monthlySales >=100000 Then

Set storeAmount = 5000

Else

Set storeAmount = 500

End If­­­­­­­­

End Module

**Step 4:** Write a module that will ask the user to enter the percent of sales increase in decimal format. This module will have to accept salesIncrease as a reference. Complete the missing lines.

// MODULE 3

// this module takes in percent of increase in decimal

// format such as .02 for 2 percent.

Module getIncrease(Real Ref salesIncrease)

Display "Enter the percent of sales increase."

Input salesIncrease

End Module

**Step 5:** Write a module that will determine individual bonuses. If the sales increase percent was 4% or more, then all employees get a $50 bonus. If the sales increase was not reached, then the bonus amount should be set to zero. This module should be called empBonus and accept salesIncrease as a normal variable and empAmount as a reference.

// MODULE 4

// this module will determine what the bonus levels are

Module empBonus(Real salesIncrease, Real Ref empAmount)

If salesIncrease >= .04 Then

Set empAmount = 50

Else

Set empAmount = 0

End If­­­­­­­­

End Module

**Step 6:** Write a module that will print the store bonus and the employee bonus amount. Name this module printBonus() and pass the two necessary variables.

// MODULE 5

// this module will display store and employee bonus info.

Module printBonus(Real storeAmount, Real empAmount)

Display "The store bonus is $", storeAmount

Display "The employee bonus is $", empAmount

End Module

**Step 7:** The final step in completing the pseudocode is to call all the modules with the proper arguments. Complete the missing lines.

Module main ()

// Declare local variables

Declare Real monthlySales

Declare Real storeAmount

Declare Real empAmount

Declare Real salesIncrease

// Function calls

Call getSales(monthlySales)

Call getIncrease(salesIncrease)

Call storeBonus(monthlySales, storeAmount)

Call empBonus(salesIncrease, empAmount)

Call printBonus(storeAmount empAmount)

End Module

**Lab 4.3 – Pseudocode: Nested Decision Structures**

|  |
| --- |
| Critical Review  To test more than one condition, a decision structure can be nested inside another decision structure. This structure can become very complex, and often an if-then-else-if statement is used instead.  The general structure of the if-then-else-if statement is:  If condition\_1 Then  *Statement*  *Statement*  *Etc.*  Else If condition\_2 Then  *Statement*  *Statement*  *Etc*.  *Insert as many Else If clauses as necessary*  Else  *Statement*  *Statement*  *Etc.*  End If  A case structure lets the value of a variable or an expression determine which path of execution the program will take. This is often used as an alternative to a nested if else decision. |

The company now wants to add additional levels to their store and employee bonuses. The new levels are as follows:

Store bonuses:

If store sales are $80,000 or more, store bonus is $3000

If store sales are $90,000 or more, store bonus is $4000

If store sales are $100,000 or more, store bonus is $5000

If store sales are $110,000 or more, store bonus is $6000

Employee bonuses:

If percent of increase is 3% or more, employee bonus is $40

If percent of increase is 4% or more, employee bonus is $50

If percent of increase is 5% or more, employee bonus is $75

**Step 1:** Modify the storeBonus module to write a nested if else statement to set the new bonus levels. Complete the missing lines. (Reference: Dual Alternative Decision Structures, page 167).

// MODULE 2

// this module will determine what the bonus levels are

Module storeBonus (Real monthlySales, Real Ref storeAmount)

If monthlySales >= 110000 Then

Set storeAmount = 6000

Else If monthlySales >= 100000 Then

Set storeAmount = 5000

Else if monthlySales >= 90000 Then

Set storeAmount = 4000

Else if monthlySales >= 80000 Then

Set storeAmount = 3000

Else

Set storeAmount = 0

End If­­­­­­­­

End Module

**Step 2:** Modify the empBonus module to write a nested if else statement to set the new bonus levels. Complete the missing lines. (Reference: Dual Alternative Decision Structures, page 167).

// MODULE 4

// this module will determine what the bonus levels are

Module empBonus (Real salesIncrease, Real Ref empAmount)

If salesIncrease >= .05 Then

Set empAmount = 75

Else If salesIncrease >= .04 Then

Set empAmount = 50

Else if salesIncrease >= .03 Then

Set empAmount = 40

Else

Set empAmount = 0

End If­­­­­­­­

End Module

**Step 3:** Modify Module 5 by adding an if statement that will print a message if both the store bonus and the employee bonus are the highest amounts possible. (Reference: Logical Operators, page 189).

// MODULE 5

// this module will display store and employee bonus info.

Module printBonus(Real storeAmount, Real empAmount)

Display "The store bonus is $", storeAmount

Display "The employee bonus is $", empAmount

If storeAmount == 6000 AND empAmount == 75 Then

Display "Congrats! You have reached the highest bonus amounts possible!"

End If

End Module

**Lab 4.4 – Flowcharts**

Critical Review

A dual alternative decision structure has two possible paths of execution – one path is taken if a condition is true, and the other path is taken if the condition is false.

A diamond with a true and false value is used in flowcharting a dual alternative decision structure.



Nested if-then-else flowcharts look as follows:



This lab requires you to convert your pseudocode in Lab 4.2 to a flowchart. Please use Visio or Draw.io to create your diagram.

**Insert** your finished flowchart in the space below for later reference. You will also upload this flowchart in Blackboard. If you are using Draw.io, please convert your flowchart to a pdf file before uploading it.

**PASTE FLOWCHART HERE**



**Lab 4.5 – Python Code**

Critical Review

In code we write a dual alternative decision structure as an if-else statement. Here is the general format of the if-else statement:

if *condition*:

*statement*

*statement*

*etc.*

else:

*statement*

*statement*

*etc.*

Here is the general format of the if-elif-else statement:

if *condition\_1*:

*statement*

*statement*

*etc.*

elif *condition\_2*:

*statement*

*statement*

*etc.*

*Insert as many elif clauses as necessary…*

else:

*statement*

*statement*

*etc.*

The logical operators and, or, and not are used in Python to connect Boolean expressions.

**Step 1:** Start the IDLE Environment for Python. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as *Lab4-5.py*. Be sure to include the .py extension.

**Step 2:** Document the first few lines of your program to include your name, the date, and a brief description of what the program does.

**Step** **3:** Start your program with the following code:

# Lab 4-5

# The main function

def main():

monthlySales = getSales() #call to get sales

# This function gets the monthly sales

def getSales():

monthlySales = float(input('Enter the monthly sales $'))

return monthlySales

# This function gets the percent of increase in sales

def getIncrease():

# This function determines the storeAmount bonus

def storeBonus():

# This function determines the empAmount bonus

def empBonus():

# This function prints the bonus information

def printBonus():

# calls main

main()

**Step 4:** Under the getIncrease function, add the necessary code to allow the user to input sales increase. Your code might look as follows:

# This function gets the percent of increase in sales

def getIncrease():

salesIncrease = float(input('Enter percent of sales increase: '))

salesIncrease = float(salesIncrease)

salesIncrease = salesIncrease / 100

return salesIncrease

**Step 5:** Under the call to getSales(), add a function call to getIncrease().

salesIncrease = getIncrease() #call to get sales increase

**Step 6:** Under the storeBonus function, add the necessary code so that the program will determine what the proper storeAmount variable should have. This function might look as follows:

#This function determines the storeAmount bonus

def storeBonus(monthlySales):

if monthlySales >=110000:

storeAmount = 6000

elif monthlySales >=100000:

storeAmount = 5000

elif monthlySales >=90000:

storeAmount = 4000

elif monthlySales >=80000:

storeAmount = 3000

else:

storeAmount = 0

return storeAmount

**Step 7:** Under the call to getIncrease(), add a function call to storeBonus().

#call to get the store bonus

storeAmount = storeBonus(monthlySales)

**Step 8:** Repeat the similar process in step 6 and 7 for writing the empBonus() function and making a call to it. Recall that this function uses salesIncrease to determine empAmount.

**Step 9:** Code the printBonus() function to print the storeAmount and empAmount. This function might look as follows:

#This function prints the bonus information

def printBonus(storeAmount, empAmount):

print 'The store bonus amount is $', storeAmount

print 'The employee bonus amount is $', empAmount

if storeAmount == 6000 and empAmount == 75:

print('Congrats! You have reached the highest bonus amounts possible!')

**Step 10:** Under the call to empBonus(), add a function call to printBonus. This call might look as follows:

printBonus(storeAmount, empAmount) #call to print amounts

**Step 11:** Click Run and Run Module to see how your program processes. Test the following values to verify the expected output.

|  |  |
| --- | --- |
| **Input Values** | **Expected Output** |
| monthlySales = 120500  salesIncrease = 5 | The store bonus amount is $ 6000  The employee bonus amount is $ 75  Congrats! You have reached the highest bonus amounts possible! |
| monthlySales = 93400  salesIncrease = 5 | The store bonus amount is $4000  The employee bonus amount is $75 |
| monthlySales = 75000  salesIncrease = 1.5 | The store bonus amount is $0  The employee bonus amount is $0 |
| monthlySales = 82000  salesIncrease = 3.6 | The store bonus amount is $3000  The employee bonus amount is $40 |
| monthlySales = 125000  salesIncrease = 4.5 | The store bonus amount is $6000  The employee bonus amount is $50 |

**Step** **12:** Execute your program so that it works and paste the final code below

#######################################################  
# Name: David White  
# Class: CIS-1400  
# Assignment: Lab 4-5  
# File: lab4-5.py  
# Purpose:   
#######################################################  
  
print('\n\*\*\*David White\*\*\*\n') # Display author's name  
  
  
# main function  
def main():  
 print('Program Start')  
 monthlySales = getSales()  
 salesIncrease = getIncrease()  
 storeAmount = storeBonus(monthlySales)  
 emplAmount = emplBonus(salesIncrease)  
 printBonus(storeAmount, emplAmount)  
 return  
  
  
# get monthly sales  
def getSales():  
 monthlySales = float(input("Enter the monthly sales $"))  
 return monthlySales  
  
  
# get the percent increase in sales  
def getIncrease():  
 salesIncrease = float(input('Enter the percent of sales increase: '))  
 salesIncrease = float(salesIncrease)  
 salesIncrease = salesIncrease / 100  
 return salesIncrease  
  
  
# determine store bonus  
def storeBonus(monthlySales):  
 if monthlySales >= 110000:  
 storeAmount = 6000  
 return storeAmount  
 elif monthlySales >= 100000:  
 storeAmount = 5000  
 return storeAmount  
 elif monthlySales >= 90000:  
 storeAmount = 4000  
 return storeAmount  
 elif monthlySales >= 80000:  
 storeAmount = 3000  
 return storeAmount  
 else:  
 storeAmount = 0  
 return storeAmount  
  
  
# determine employee bonus  
def emplBonus(salesIncrease):  
 if salesIncrease >= .05:  
 emplAmount = 75  
 return emplAmount  
 elif salesIncrease >= .04:  
 emplAmount = 50  
 return emplAmount  
 elif salesIncrease >= .03:  
 emplAmount = 40  
 return emplAmount  
 else:  
 emplAmount = 0  
 return emplAmount  
  
  
# display bonus info  
def printBonus(storeAmount, emplAmount):  
 print('The store bonus amount is $', storeAmount)  
 print('The employee bonus amount is $', emplAmount)  
 if storeAmount == 6000 and emplAmount == 75:  
 print('Congrats! You have reached the highest bonus amounts possible!')  
 return  
 return  
  
  
# call main  
main()

**Lab 4.6 – Programming Challenge 1 – Tip, Tax, and Total**

Write the Pseudocode, Flowchart, and Python code for the following programming problem.

Recall the Tip, Tax, and Total program from Lab 2.6. Modify your program to include new requirements.

Write a program that will calculate a XXX% tip and a 6% tax on a meal price. The user will enter the meal price and the program will calculate tip, tax, and the total. The total is the meal price plus the tip plus the tax. Your program will then display the values of tip, tax, and total.

The restaurant now wants to change the program so that the tip percent is based on the meal price. The new amounts are as follows:

|  |  |
| --- | --- |
| **Meal Price Range** | **Tip Percent** |
| .01 to 5.99 | 10% |
| 6 to 12.00 | 13% |
| 12.01 to 17.00 | 16% |
| 17.01 to 25.00 | 19% |
| 25.01 and more | 22% |

**The Pseudocode**

Module main()

//declare variables

Declare Real mealPrice

Declare Real tip

Declare Real tax

Declare Real total

//call functions

inputMeal(mealPrice)

calcTip(mealPrice)

calcTax(mealPrice)

calcTotal(mealPrice, tip, tax)

printInfo(mealPrice, tip, tax, total)

End Module

//get mealprice

Module inputMeal(Real Ref mealPrice)

Display “Enter the meal price”

Input mealPrice

End Module

//calculate tip

Module calcTip(Real mealPrice, Real Ref tip)

If mealPrice >= 25.01 Then

Set tip = mealPrice \* .22

Else If mealPrice >= 17.01 Then

Set tip = mealPrice \* .19

Else If mealPrice >= 12.01 Then

Set tip = mealPrice \* .16

Else If mealPrice >= 6 Then

Set tip = mealPrice \* .13

Else

Set tip = mealPrice \* .10\

End Module

//calculate tax

Module calcTax(Real Ref tax, Real mealPrice)

Set tax = mealPrice \* .06

End Module

//calculate total

Module calcTotal(Real mealPrice, Real tip, Real tax, Real Ref total)

Set total = mealPrice + tip + tax

End Module

//Display information

Module printInfo(Real mealPrice, Real tax, Real tip, Real total)

Display ‘The meal price is $ ‘, mealPrice

Display ‘The tip is $ ‘, tip

Display ‘The tax is $ ‘, tax

Display ‘The total is $ ‘, total

End Module

**The Flowchart**



**The Python Code**

#######################################################

# Name: David White

# Class: CIS-1400

# Assignment: Lab 4-7

# File: lab4-7.py

# Purpose: calculate a tip and a 6% tax on a meal price.

#######################################################

print('\n\*\*\*David White\*\*\*\n') # Display author's name

# This program uses functions and variables

# the main function

def main():

print('Welcome to the meal calculator program')

print() # prints a blank line

mealPrice = inputMeal()

tip = calcTip(mealPrice)

tax = calcTax(mealPrice)

total = calcTotal(mealPrice, tip, tax)

printInfo(mealPrice, tip, tax, total)

return

# user input function

def inputMeal():

mealPrice = input('Enter the meal price $')

mealPrice = float(mealPrice)

return mealPrice

# calculates 20% tip

def calcTip(mealPrice):

if mealPrice >= 25.01:

tip = mealPrice \* .22

return tip

elif mealPrice >= 17.01:

tip = mealPrice \* .19

return tip

elif mealPrice >= 6:

tip = mealPrice \* .13

return tip

else:

tip = mealPrice \* .10

return tip

return tip

# calculates 6% tax

def calcTax(mealPrice):

tax = float(mealPrice \* .06)

return tax

# calculates the total

def calcTotal(mealPrice, tip, tax):

total = mealPrice + tip + tax

return total

# displays tip, tax and the total

def printInfo(mealPrice, tip, tax, total):

print('The meal price is $', mealPrice)

print('The tip is $' + format(tip, '>.2f'))

print('The tax is $' + format(tax, '>.2f'))

print('The total is $' + format(total, '>.2f'))

return

# calls main

main()