Lab 2: Modules
This lab accompanies Chapter 3 of *Starting Out with Programming Logic & Design*.

Name: ___________________________

Lab 2.1 – Algorithms

This lab requires you to think about the steps that take place in a program by writing algorithms. Read the following program prior to completing the lab.

A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:

- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

Step 1: Examine the following algorithm.

1. Get the total sales for the month.
2. Multiply the total sales by .04 to calculate the state sales tax.
3. Multiply the total sales by .02 to calculate the county sales tax.
4. Add the state tax and county tax to calculate the total sales tax.
5. Display the calculated county tax, state tax, and total sales tax.

Step 2: Given a total sales of $27,097, answer the following:

What is the calculated state tax? 1083.88
What is the calculated county tax? 541.94
What is the calculated total tax? 1625.82
Lab 2.2 – Pseudocode and Modules

Critical Review

A Module is a group of statements that exists within a program for the purpose of performing a specific task.

Modules are commonly called procedures, subroutines, subprograms, methods, and functions.

The code for a module is known as a module definition. To execute the module, you write a statement that calls it.

The format for a module definition is as follows:

Module name( )
   Statement
   Statement
   Etc.
End Module

Calling a module is normally done from the Main() module such as:

Call name( )

Generally, local variables should be used and arguments should be passed by reference when the value of the variable is changed in the module and needs to be retained. For example:

Module main( )
   Real Integer number
   Call inputData(number)
   Call printData(number)
End Module

//accepts number as a reference so the changed value
//will be retained
Module inputData(Real Ref number)
   Number = 20
End Module

//number does not need to be sent as a reference because
//number is not going to be modified
Module printData(number)
   Display “The number is “, number
End Module

Help Video: Double click the file to view video
This lab requires you to think about the steps that take place in a program by writing pseudocode. Read the following program prior to completing the lab.

A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:

- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

**Step 1:** This program is most easily solved using just four variables. Declare the variables that you will need in the program, using the proper data type and documenting the purpose.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declare Real totalSales</td>
<td>Stores total sales the user inputs</td>
</tr>
<tr>
<td>Declare Real countyTax</td>
<td>Stores the calculated county tax</td>
</tr>
<tr>
<td>Declare Real stateTax</td>
<td>Stores the calculated state tax</td>
</tr>
<tr>
<td>Declare Real totalTax</td>
<td>Stores the calculated total tax</td>
</tr>
</tbody>
</table>

**Step 2:** Given the major task involved in this program, what modules might you consider including? Describe the purpose of each module. (Reference: Defining and Calling a Module, page 78).

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module inputData ()</td>
<td>Allows the user to enter required user input</td>
</tr>
<tr>
<td>Module calcCounty ()</td>
<td>Calculates the county tax at 2% of the monthly sales</td>
</tr>
<tr>
<td>Module calcState ()</td>
<td>Calculates the state tax at 4% of the monthly sales</td>
</tr>
<tr>
<td>Module calcTotal ()</td>
<td>Calculates the total tax as state plus county</td>
</tr>
<tr>
<td>Module printData ()</td>
<td>Prints the necessary information</td>
</tr>
</tbody>
</table>

**Step 3:** Complete the pseudocode by writing the missing lines. (Reference: Defining and Calling a Module, page 78-81). Also, when writing your modules and making calls,
be sure to pass necessary variables as arguments and accept them as reference parameters if they need to be modified in the module. (Reference: Passing Arguments by Value and by Reference, page 97 – 103).

Module main()
    //Declare local variables
    Declare Real totalSales
    Declare Real countyTax
    Declare Real stateTax
    Declare Real totalTax

    //Function calls
    Call inputData(totalSales)
    Call calcCounty(totalSales, countyTax)
    Call calcState(totalSales, stateTax)
    Call calcTotal(countyTax, stateTax, totalTax)
    Call printData(countyTax, stateTax, totalTax)
End Module

//this module takes in the required user input
Module inputData(Real Ref totalSales)
    Display "Enter the total sales for the month."
    Input totalSales
End Module

//this module calculates county tax
//totalSales can be a value parameter because it is not changed in the module.
//countyTax must be a reference parameter because it is changed in the module
Module calcCounty(Real totalSales, Real Ref countyTax)
    countyTax = totalSales * .02
End Module

//this module calculates state tax
Module calcState(Real totalSales, Real Ref stateTax)
    stateTax = totalSales * .04
End Module

//this module calculates total tax
Module calcTotal(Real countyTax, Real stateTax, Real Ref totalTax)
    totalTax = countyTax + stateTax
End Module

//this module prints the total, county, and state tax
Module printData(Real countyTax, Real stateTax, Real totalTax)
    Display "The county tax is ", countyTax
    Display "The state tax is ", stateTax
    Display "The total tax is ", totalTax
End Module
Lab 2.3 – Flowcharts

Critical Review

The flowchart symbol used for a function call is a rectangle with vertical bars on each side:

```
Main ( )

//Used in main to represent a function call
Method ( )

End
```

Help Video: Double click the file to view video

This lab requires you to think about the steps that take place in a program by designing a flowchart. Use an application such as Raptor or Visio. Read the following program prior to completing the lab.

A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:

- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

**Step 1:** Start Raptor and save your document as *Lab 2-3*. The *.rap* file extension will be added automatically. Start by adding a Comment box that declares your variables. Here is a start to how your Comment box should look.
Step 2: The next step in your flowchart should be to call your methods. Click the Call Symbol on the Left and Drag and Drop to the flow lines between Start and Stop. Double click on the Call Symbol and type the name of your first method. For example, type inputData in the Enter Call box. Do not put the () when using Raptor. Click the Done button. A new box will pop up that will ask you to create a new tab. Click Yes. A new tab will be created for your new method. Notice the new Tab called inputData.

Step 3: Continue this process to add your additional methods, which are calcCounty(), calcState(), calcTotal() and printData().

Step 4: Click on the inputData tab and add the necessary code as identified in your pseudocode in Lab 2.2. In Raptor, there is no need to pass variables as References as in pseudocode. Your inputData method might look like the following:

Step 5: Continue to code the remaining methods, which are calcCounty() calcState(), calcTotal() and printData(). If you happened to execute your program without completing your program, an error will occur such as:
Your calculations methods input box might look like the following:

![Image of input box with set statement and calculation]

Your output data methods box might look like the following:

![Image of output box with calculated output]

**Step 6:** After your program is complete, click Run, then Execute to Finish. For your input, enter 67854 as your total monthly sales. If your program is coded correctly, the output should be as follows:
The county tax is $1357.0800
The state sales tax is $2714.1600
The total tax is $4071.2400
----Run finished----

Step 7: The final step is to insert your finished flowchart in the space below. Inside Raptor, select File and the Print to Clipboard from the menu. Inside Word in the space below, select Edit and Paste. You will have to do this for each module you created.

PASTE FLOWCHART HERE
inputData Module

```
Start

"Please enter total sales for the month: "
GET totalSales

End
```

calcCounty Module

```
Start

countyTax ← totalSales * .02

End
```

calcState Module

```
Start

stateTax ← totalSales * .04

End
```

calcTotal Module

```
Start

totalTax ← countyTax + stateTax

End
```

printData Module

```
Start

PUT "The county tax is $" + countyTax

PUT "The state sales tax is $" + stateTax

PUT "The total tax is $" + totalTax

End
```
Lab 2.4 – Python Code and Functions

Critical Review

The code for a function is known as a function definition. To execute the function, write a statement that calls it.

To create a function, write its definition. The keyword `def` is used before a function name, followed by parentheses and a colon. Here is the general format of a function definition in Python:

```
def function_name():
    statement
    statement
    etc.
```

Calling a function is done in order to make the module execute. The general format is:

```
function_name()
```

Function names must be flushed to the left.

Statements within a module must be aligned evenly in order to avoid syntax errors.

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 `Lab2-4.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. Description of the program should be:

```
#This program will demonstrate various ways to use functions in Python.
```

Step 3: After your documentation, add the following function definition and function call.

```
#This function is to welcome people to my program
def welcome_message():
    print 'Welcome to my program using functions'
    print 'My name is Joe Student'

#This is a function call
welcome_message()
```
**Step 4:** Click Run, then Run Module to see your output. It should look like the following:

```
IDLE 1.2.1
>>>  ==================  RESTART  ==================
>>>  Welcome to my program using functions
>>>  My name is Joe Student
```  

**Step 5:** Change your program so that the function call is tabbed over, such as:

```python
#This function is to welcome people to my program
def welcome_message():
    print 'Welcome to my program using functions'
    print 'My name is Joe Student'

#This is a function call
welcome_message()  #tab this line over
```

**Step 6:** Click Run and Run Module again. You’ll notice that nothing is printed. This is because in Python, each line in a block must be indented and aligned. Function calls must be flushed to the left, and each line within a module must be aligned evenly. The following will cause a syntax error.

```python
def my_function():
    print 'And now for'
    print 'something completely'
    print 'different.'
```

**Step 7:** Since programs normally center around a main function, modify your program so that it looks as follows:

```python
#The main function
def main():
    welcome_message()  #causes welcome_message to run

#This function is to welcome people to my program
def welcome_message():
    print 'Welcome to my program using functions'
    print 'My name is Joe Student'

#This is the main function that starts the program in #motion
main()  #calls main
Step 8: Add an additional function to your program that is called goodbye_message(). The contents of this function should print a goodbye message. Execute your program so that it works and paste the final code below.

```python
# Danica Myers
# Date
# Description: This program tests out the different ways to use functions

# The main function
def main():
    welcome_message() # causes welcome_message to execute
    goodbye_message() # causes goodbye_message to execute

# This function is to welcome people to my program
def welcome_message():
    print ‘Welcome to my program using functions’
    print ‘My name is Danica Myers’

    def goodbye_message():
        print ‘Goodbye…thank you for using my program’

    # This is the main function that starts the program in motion
    main() # calls main
```
Lab 2.5 – Python Code and Variables

Critical Review

Variables can either be local or global in scope.

A local variable is created inside a function and cannot be accessed by statements that are outside a function, unless they are passed.

A local variable that needs to be used in multiple functions should be passed to the necessary functions.

An argument is any piece of data that is passed into a function when the function is called. A parameter is a variable that receives an argument that is passed into a function.

A global variable can be accessed by any function within the program, but should be avoided if at all possible.

---

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 Lab2-5.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. Description of the program should be:

```python
#This program demonstrates how to use variables and functions.
```

Step 3: Add a function called main() and a function call to main. Your code might look like this:

```python
#This program uses functions and variables

#the main function
def main():
    print 'Welcome to the tip calculator program'
    print   #prints a blank line

#calls main
main()
```

Step 4: Add a function called inputName() under the def main(): function. Your code might look as follows:

```python
#this function inputs name
def inputName()
```
Step 5: Under your function definition, write a statement that allows the user to enter their name. Inside of the main function, call inputName() and write a print statement that displays the name. Your code might look as follows:

```python
#This program uses functions and variables

def main():
    print 'Welcome to the variable program'
    print   #prints a blank line
    inputName()
    print 'Hello', name

#this function inputs name
def inputName():
    name = raw_input('Enter your name: ')

#calls main
main()
```

Step 6: Compile and run your program. Notice that when the program attempts to display the name, a syntax error occurs. This is because name is declared as a local variable within the inputName() function and main cannot access it.

Step 7: There are a couple of ways to fix this error. Examine the following code:

```python
#This program uses functions and variables

def main():
    print 'Welcome to the variable program'
    print   #prints a blank line

    name = inputName()
    print 'Hello', name

#this function inputs data
def inputName():
    name = raw_input('Enter your name: ') return name

#calls main
main()
```

The local variable name is declared in main and set equal to whatever the inputName() function returns. Notice the return name that is at the end of the inputName() function. This passes the value that was taken in back to main.
**Step 8:** Add an additional function to your program that is called `inputAge()`. The contents of this function should be structured similar to the `inputName()` function excepts that it asks the user to enter their age. Additionally, make a call to this new function such as `age = inputAge()`. You should also display the value of age after the name is displayed. Execute your program so that it works and paste the final code below.

PASTE CODE HERE

---

# Danica Myers
# Date
# This program demonstrates how to use variables and functions

# This program uses functions and variables

# the main function
```python
def main():
    print ‘Welcome to the tip calculator program’
    print # prints a blank line

    name = inputName()
    age = inputAge()
    print ‘Hello’, name
    print ‘Your age is’, age

# this function inputs name
def inputName():
    name = raw_input(‘Enter your name:’)  
    return name

# this function inputs age
def inputAge():
    age = raw_input(‘Enter your age:’)  
    return age

# calls main
main()```
Lab 2.6 – Writing a Complete Program

Help Video: Double click the file to view video

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 Lab2-6.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. Description of the program should be:

Write a program that will calculate a 20% 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.

Step 3: Add a function called main() and a function call to main.

Step 4: Add the function definition for input_meal(), calc_tip(), calc_tax(), calc_total(), and print_info(). Your code might look like the following:

```python
#This program uses functions and variables

#the main function
def main():
    print 'Welcome to the meal calculator program'
    print  #prints a blank line

    #this function will input meal price
    def input_meal():

    #this function will calculate tip at 20%
    def calc_tip():

    #this function will calculate tax at 6%
    def calc_tax():

    #this function will calculate tip, tax, and the total cost
    def calc_total():

    #this function will print tip, tax, the meal price, and the total
    def print_info():
```
#calls main
main()

**Step 5:** Inside of main() under the print #prints a blank line statement, create a local variable named mealprice that is set to the input_meal() function. This should look like the following:

```python
mealprice = input_meal()
```

**Step 6:** Add the following lines of code inside of input_meal() function. This should look like the following:

```python
mealprice = input('Enter the meal price $')
mealprice = float(mealprice)
return mealprice
```

The first line asks the user to enter their meal price. The second line converts the value to a float, since it will likely be a decimal value. This must be done with all potential decimal values that the user enters. The third line returns the input value of mealprice to the place where it was called (in Step 5).

**Step 7:** Inside of main() under the meal = input_meal() statement, create a local variable named tip that is set to the calc_tip() function. In this case, you must pass mealprice to the function, so it must be placed between the parentheses. This should look like the following:

```python
tip = calc_tip(mealprice)
```

**Step 8:** Add the following lines of code inside of calc_tip(mealprice) function. The entire function should look like the following:

```python
def calc_tip(mealprice):
    tip = mealprice * .20
    return tip
```

The first line is the function definition. It accepts mealprice as a parameter. The second line is to calculate tip as 20% of the mealprice. The third line returns the calculated tip to the place where it is called.

**Step 9:** Inside of main() under the tip = calc_tip(mealprice) statement, create a local variable named tax that is set to the calc_tax() function. In this case, you must pass mealprice to the function, so it must be placed between the parentheses. This should look like the following:

```python
tax = calc_tax(mealprice)
```
Step 10: Add the following lines of code inside of calc_tax(mealprice) function. The entire function should look like the following:

```python
def calc_tax(mealprice):
    tax = mealprice * .06
    return tax
```

The first line is the function definition. It accepts mealprice as a parameter. The second line is to calculate tax as 6% of the mealprice. The third line returns the calculated tax to the place where it is called.

Step 11: Inside of main() under the tax = calc_tax(mealprice) statement, create a local variable named total that is set to the calc_total() function. In this case, you must pass mealprice, tip, and tax to the function, so they must be placed between the parentheses. This should look like the following:

```python
total = calc_total(mealprice, tip, tax)
```

Step 12: Add the following lines of code inside of calc_total(mealprice, tip, tax) function. The entire function should look like the following:

```python
def calc_total(mealprice, tip, tax):
    total = mealprice + tip + tax
    return total
```

The first line is the function definition. It accepts mealprice, tip, and tax as parameters. The second line is to calculate the total of all three values added together. The third line returns the calculated total to the place where it is called.

Step 13: Inside of main() under the total = calc_total(mealprice, tip, tax) statement, call the print_info() function. In this case, you must pass mealprice, tip, tax, and total to the function, so they must be placed between the parentheses. This should look like the following:

```python
print_info(mealprice, tip, tax, total)
```

Step 14: Add the following lines of code inside of print_info(mealprice, tip, tax, total) function. The entire function should look like the following:

```python
def print_info(mealprice, tip, tax, total):
    print 'The meal price is $', mealprice
    print 'The tip is $', tip
    print 'The tax is $', tax
    print 'The total is $', total
```
The first line is the function definition. It accepts mealprice, tip, tax, and total as parameters. The following lines print the mealprice, the calculated tip, the calculated tax, and the calculated total.

**Step 15:** Run your module and fix any errors you may have. The most common errors may be that you have misspelled something when typing, or that your indentations are not aligned properly. When running your program, enter 24.50 as the meal price. Your output should look as follows:

```
Welcome to the tip and tax calculator program

Enter the meal price $24.50
The meal price is $ 24.5
The tip is $ 4.9
The tax is $ 1.47
The total is $ 30.87
```

**Step 16:** When your program is completed and you have tested your output in Step 15, paste your completed program below.

```
PASTE CODE HERE

#This program uses functions and variables

#the main function
def main():
    print 'Welcome to the tip and tax calculator program'
    print  #prints a blank line
    mealprice = input_meal()
    tip = calc_tip(mealprice)
    tax = calc_tax(mealprice)
    total = calc_total(mealprice, tip, tax)
    print_info(mealprice, tip, tax, total)

#this function will input meal price
def input_meal():
    mealprice = input('Enter the meal price $')
    mealprice = float(mealprice)
    return mealprice

#this function will calculate tip at 20%
def calc_tip(mealprice):
    tip = mealprice * .20
    return tip
```
#this function will calculate tax at 6%
def calc_tax(mealprice):
    tax = mealprice * .06
    return tax

#this function will calculate tip, tax, and the total cost
def calc_total(mealprice, tip, tax):
    total = mealprice + tip + tax
    return total

#this function will print tip, tax, the mealprice, and the total
def print_info(mealprice, tip, tax, total):
    print 'The meal price is $', mealprice
    print 'The tip is $', tip
    print 'The tax is $', tax
    print 'The total is $', total

#calls main
main()
Lab 2.7 – Programming Challenge 1 – Retail Tax

Help Video: Double click the file to view video

lab2-7.wmv

This lab requires you to translate your work in the pseudocode and flowchart from Lab 2.2 and Lab 2.3 to actual code using Python. Read the following program prior to completing the lab.

A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:

- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

Consider the following functions for your program:

- main that calls your other functions
- inputData that will ask for the monthly sales
- calcCounty that will calculate the county tax
- calcState that will calculate the state tax
- calcTotal that will calculate the total tax
- printData that will display the county tax, the state tax, and the total tax

If your program is correct, sample output might look as follows:

Welcome to the total tax calculator program

Enter the total sales for the month $12567
The county tax is $ 251.34
The state tax is $ 502.68
The total tax is $ 754.02

The Python Code

PASTE COMPLETED CODE HERE
#This program uses functions and variables
#the main function
def main():
    print 'Welcome to the total tax calculator program'
    print  #prints a blank line
    totalsales = inputData()
    countytax = calcCounty(totalsales)
    statetax = calcState(totalsales)
    totaltax = calcTotal(countytax, statetax)
    printData(countytax, statetax, totaltax)

#this function will input meal price
def inputData():
    totalsales = input('Enter the total sales for the month $')
    totalsales = float(totalsales)
    return totalsales

#this function will calculate tip at 20%
def calcCounty(totalsales):
    countytax = totalsales * .02
    return countytax

#this function will calculate tax at 6%
def calcState(totalsales):
    statetax = totalsales * .04
    return statetax

#this function will calculate tip, tax, and the total cost
def calcTotal(countytax, statetax):
    totaltax = countytax + statetax
    return totaltax

#this function will print tip, tax, the meal price, and the total
def printData(countytax, statetax, totaltax):
    print 'The county tax is $', countytax
    print 'The state tax is $', statetax
    print 'The total tax is $', totaltax

#calls main
main()