LAB 2 ASSIGNMENT

Ch. 2 Input, Processing, and Output

Start: 01/17/2024 06:16 PM
Name: Imani Hollie

LAB 2.1 – ALGORITHMS

Critical Review

An algorithm is a set of well-designed logical steps that must take place to solve a problem.

The flow the algorithm takes is sequential. Ex. Before you process calculations, all data should be retrieved.

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

```
Write a program that will take in basic information from a student, including
student name, degree name, number of credits taken so far, and the total
number of credits required in the degree program. The program will then
calculate how many credits are needed to graduate. The display should include
the student's name, the degree name, and the credits left to graduate.
```

Step 1: Examine the following algorithm:

```
1. Get the student’s name.
2. Get the degree program name.
3. Subtract the number of credits taken so far from the required credits
   for the degree.
4. Get the number of credits required for the degree program.
5. Get the number of credits the student has taken so far.
6. Display the input information in Steps 1 and 2.
7. Display the calculated information.
```

Step 2: What logic error do you spot and how would you fix it?

Line 3 needs to go after Line 5, because we would not be able to calculate for credits taken and required because the input has not been given yet.

Step 3: What steps require user interaction (Ex. User must type in some input)?

Lines 1, 2, 4, and 4 require user interaction.

LAB 2.2 – PSUEDOCODE

Critical Review

Pseudocode is an informal language that has no syntax rules and is not meant to be compiled or executed.

The flow the program takes is sequential. Ex. Before you ask for input, you should display what information you want from the user.

```
//Comments are done by putting two forward slashes
//before lines you want to document. Comments
//are used to explain code.
```

Variables are named storage locations.

`Declare` is the keyword used before naming a variable.

Data types are: (1) `Real` for decimal numbers; (2) `Integer` for whole numbers; (3) And `String` for a series of characters.

Follow the rules for naming variables: (1) Must be one word, with no spaces; (2) Usually no punctuation characters, only letters and numbers; (3) And cannot start with a number.

`Display` is the keyword used to print something to the screen. Any information needed to be displayed to the user should be put inside quo tation marks such as:

Display "This is how you print something to the screen"

When using display to print both a string and the value of a variable, a comma is used, such as:

Display "Here is the average: ", average.

`Input` is the keyword used to get the user to enter data. The data value entered by the user will be placed in the variable that follows the keyword input such as `Input variableName`.

Set is the keyword used before a calculation. Standard math operators are used, such as `+ - * / MOD ^`. Operators can be combined in one calculation, but it is wise to group expressions together using parentheses. Remember the order of operations. Some examples are:

```
Set sale = price – discount
Set average = (test1 + test2 + test3) / 3
```

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

```
Write a program that will take in basic information from a student, including
student name, degree name, number of credits taken so far, and the total
number of credits required in the degree program. The program will then
calculate how many credits are needed to graduate. The display should include
the student's name, the degree name, and the credits left to graduate.
```

Step 1: This program is most easily solved using just five variables. Identify potential problems with the following variables declared in the pseudocode. Assume that the college has the ability to offer half credits: (Reference: Variable Names, pg. 39 – 40)

Variable Name	Problem (Yes/No)	If Yes; What is wrong?
`Declare Real creditsTaken`	No	
`Declare Real creditsDegree`	No	
`Declare Int creditsLeft`	Yes	`creditsLeft` needs to be a `Real` data type because it is dealing with decimals, not whole numbers.
`Declare Real studentName`	Yes	`studentName` needs to be a `String` data type because it is dealing with characters, not numbers.
`Declare String degreeName`	No	

Step 2: Complete the pseudocode by writing the three missing lines: (Reference: Prompting the User, pg. 42)

```
▪ Display "Enter student name."
1. Input studentName
▪ Display "Enter degree program."
▪ Input degreeName
2. Display "Enter the number of credits needed to earn degree."
▪ Input creditsDegree
▪ Display "Enter the number of credits taken so far."
3. Input creditsLeft
```

Step 3: What are two things wrong with the following calculation: (Reference: Variable Assignment and Calculations, pg. 43)

```
creditsLeft = creditsTaken - creditsDegree
```

(1) The assignment statement in pseudocode needs to have `Set` before the named variable on the left side of the equal sign. (2) The `creditsDegree` and `creditsTaken` needs to be switched since it reads from left to right.

```
1. Set creditsDegree = 65
2. Set creditsTaken = 35
3. Set creditsLeft = creditsDegree – creditsTaken
```

Step 4: Write the exact output you would expect from the following line of code if the user of the program enters "Bill Jones". (Reference: Displaying Multiple Items, pg. 40 – 41)

```
Display "The student's name is ", studentName
```

I would expect the output to be as follows:

```
1. The student's name is Bill Jones
```

Step 5: Write the exact output you would expect from the following line of code if the user of the program enters a degree that is 63 credits total and they have taken 40 credits. (Reference: Variable Assignment and Calculations, pg. 43)

```
Display "This program requires ", creditsDegree, " credits and they have taken ", creditsTaken, " so far."
```

I would expect the output to be as follows:

```
1. This program requires 63 credits and they have taken 40 credits so far.
```

Step 6: Complete the following pseudocode to solve the programming problem.

```
//This program takes in student information and calculates
//how many credits the student has left before graduation.
//Information is then printed to the screen.

//Declare variables
Declare Real creditsTaken
Declare Real creditsDegree
Declare Real creditsLeft
Declare String studentName
Declare String degreeName

//Ask for user input
Display "Enter student name."
Input studentName
Display "Enter degree name."
Input degreeName
Display "Enter total credit amount for ", degreeName
Input creditsDegree
Display "Enter total credits taken so far."
Input creditsTaken

//Calculate remaining credits
Set creditsLeft = creditsDegree - creditsTaken

//Display student name, degree program, and credits left.
Display "The student's name is ", studentName
Display studentName, " program is ", degreeName
Display studentName, " has taken ", creditsLeft, "credits so far."
```

LAB 2.3 – FLOWCHARTS

Critical Review

A flowchart is a diagram that graphically depicts the steps that take place in a program. Symbols are used to depict the various steps that need to happen within a program.

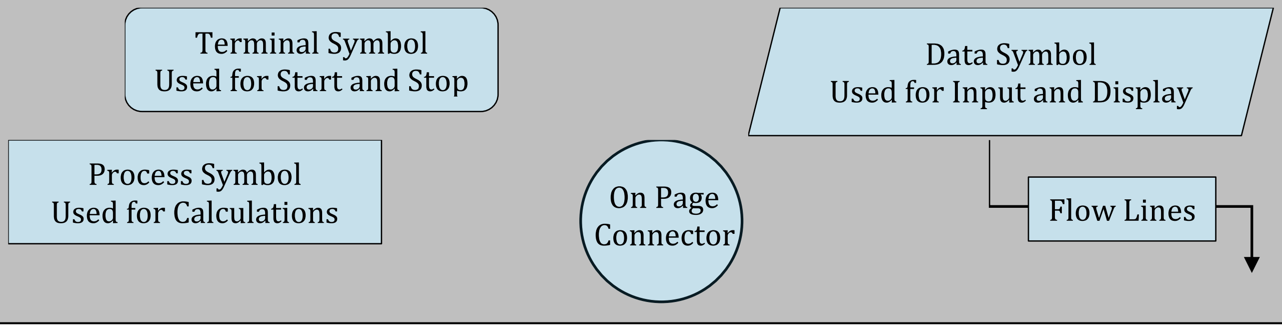
Ovals are used as terminal symbols, which indicate a start and stop to a program.

Parallelograms, the data symbol, are used for input and display statements.

Rectangles, the process symbol, are used for calculations and variable declarations.

On page connectors are used to link a flowchart that continues on the same page. The connecting system starts with the letter A, whereas A would appear in the two connectors that show the flow.

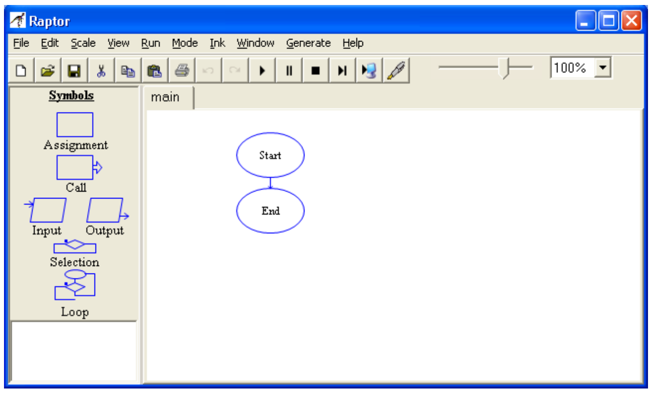
The statements inside the data and process symbols can be written similarly to the statements used in pseudocode.



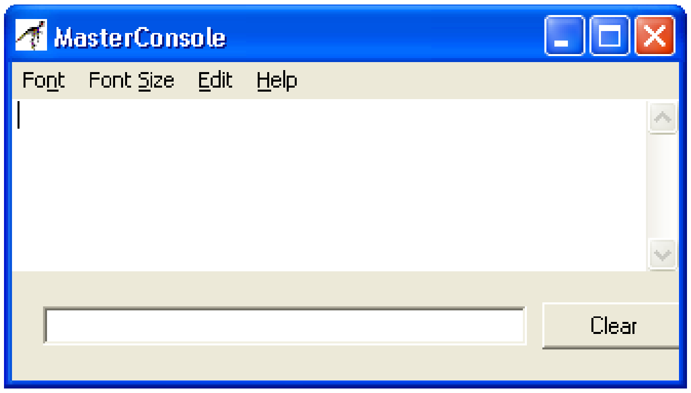
This lab requires you to think about the steps that take place in a program by designing a flowchart. While designing flowcharts can be done with paper and pencil, one mistake often requires a lot of erasing. Therefore, a flowcharting application such as Raptor or Visio should be used. This lab will give you a brief overview of Raptor. Read the following program before completing the lab.

```
Write a program that will take in basic information from a student, including
student name, degree name, number of credits taken so far, and the total
number of credits required in the degree program. The program will then
calculate how many credits are needed to graduate. The display should include
the student's name, the degree name, and the credits left to graduate.
```

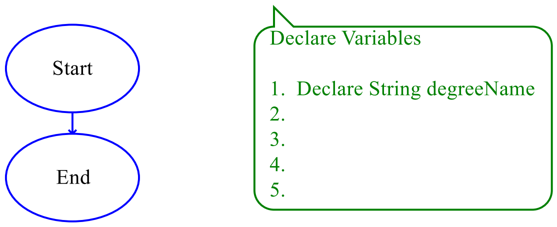
Step 1: Start Raptor; Notice the **Raptor screen.** This window is your primary tool for creating a flow chart. Before adding symbols, save your document by clicking **File** and then **Save.** Select your location and save the file as **Lab 2.3.** The **.rap** file extension will be added automatically.



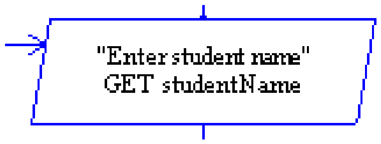
Step 2: Notice the **MasterConsole screen.** This window is used to show your program output once your flowchart is completed. The **Clear button** will clear the console to view a fresh run of your program.



Step 3: Return to the **Raptor screen** to begin adding symbols to your flowchart. Your flowchart should follow the pseudocode in **Lab 2.2 – Step 6.** While a rectangle is normally used for declaring variables, there is no easy way to do this in Raptor. Since this is an important part of flowcharting, we will do this using a **comment box.** To do this, right-click on the **Start symbol** and select **Comment.** In the **Enter Comment box**, type the variables your program will need. Below is a start to how it should look:

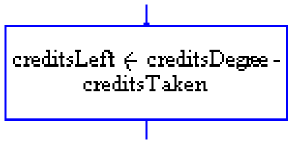


Step 4: The next step in your flowchart should be to ask for user input. Click the **Input Symbol** on the Left and Drag and Drop to the flow line between Start and Stop. **Double-click** on the **Input Symbol** to begin entering information. Type "`Enter student name`" in the top box. Enter `studentName` in the variable box. Below is how it should look:

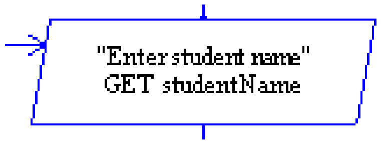


Step 5: Repeat Step 4 directions for all your input statements, changing each **Input symbol** to reflect the appropriate user information.

Step 6: The next step in your flowchart is to process any calculations that exist. Click on the **Assignment symbol** and drag it to the flow line between the last input statement and the end symbol. **Double-click** on the **Assignment symbol** to enter your code. In the **Set box**, put the name of your storage variable. In the **To box**, put the expression part of your formula. Below is how it should look:



Step 7: The next step in your flowchart is to display the requested output on the screen. Click the **Output symbol** and drag it to the flow line between the assignment statement and the end symbol. **Double-click** on the **Output symbol** to enter your code. Under **Output Type**, select **Output Expression** since we want to display both a sentence and the contents of a variable. In the box, type `"Student name is " + studentName`. Below is how it should look once you click **Done**:



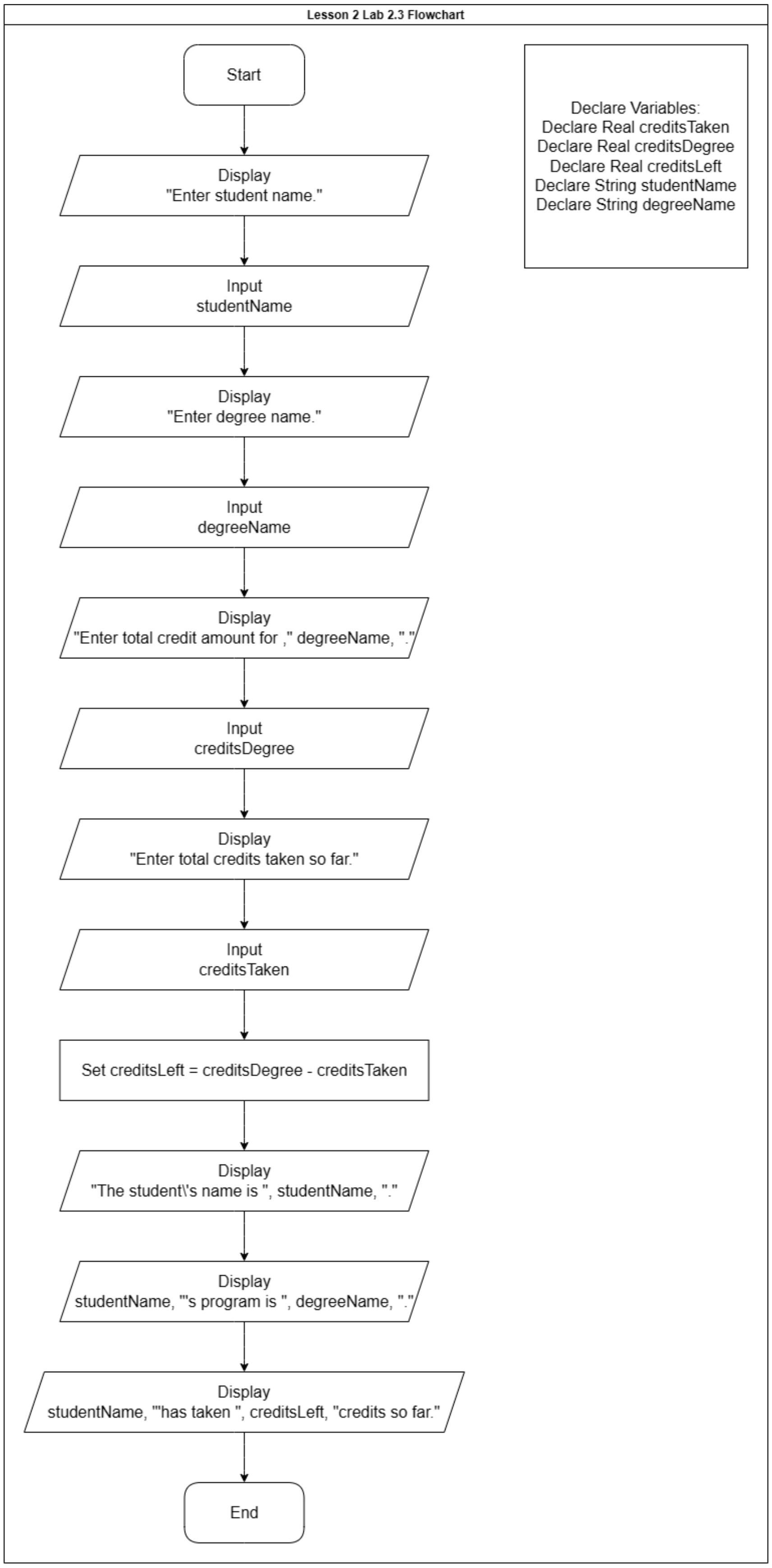
Step 8: Repeat Step 7 directions for all your output statements, changing each **Output symbol** to reflect the appropriate requested output information.

Step 9: Once your flowchart is complete, click on **Run** and then **Execute to Completion** on the **Raptor menu**. Follow the flow of your program to see if it processes properly. Your **Master Console window** should show output similar to:

```
The student's name is Bill Jones
The degree program is in Computer Programming
Credits left to graduation is 39
----Run finished----
```

Step 10: The final step is to insert your finished flowchart in the space below. Inside Raptor, select **File** and **Print to Clipboard** from the menu. Inside **Word** in the space below, select **Edit** and **Paste**:

I used the following website to complete this assignment: [draw.io]()


Lesson 2 Lab 2.3 Flowchart
Start
Display "Enter student name."
Input studentName
Display "Enter degree name."
Input degreeName
Display "Enter total credit amount for ," degreeName, "."
Input creditsDegree
Display "Enter total credits taken so far."
Input creditsTaken
Set creditsLeft = creditsDegree - creditsTaken
Display "The student\'s name is ", studentName, "."
Display studentName, "'s program is ", degreeName, "."
Display studentName, "'has taken ", creditsLeft, "credits so far."
End
Declare Variables:
Declare Real creditsTaken
Declare Real creditsDegree
Declare Real creditsLeft
Declare String studentName
Declare String degreeName

LAB 2.4 – PYTHON CODE

Critical Review

Comments in Python are preceded by the `#` sign.

Input of strings into a variable is down by using the `input` function. This function converts the input to a series of characters so they can be used later in the program. This is written as a statement such as:

```
stringVariable = input('Enter a word. ')
```

The *input* function always returns the user's input as a string, even if the user enters numeric data. If you want your program to read a numeric value as input, you must use a conversion function along with the `input` function: (1) To convert input to an integer, you must use the `int()` conversion function; (2) To convert input to a real number, you must use the `float()` conversion function.

- Ex. The following statement reads input from the user, converts that input to an integer, and assigns the integer value to a variable named `number`:

```
number = int(input('Enter a number.'))
```

- Ex. The following statement reads input from the user, converts that input to a real number (AKA: Floating-Point Number), and assigns the integer value to a variable named `number`:

```
number = float(input('Enter a number.'))
```

Equations are written similarly to the method used in pseudocode, but without the *Set* keyword. Ex.

```
total = apples + oranges
```

Complex formulas should use parentheses to group processes. In addition, if input values are taken in as integer, but will be used to calculate a decimal value, they must be converted to real values. Ex.

```
average = (test1 + test2) / 2
```

To display information to the screen, the `print` function is used with the string, which is written within single quotation marks. If the value of a variable needs to display after the string, a comma separates the two. Ex.

```
print('The average is', average)
```

This lab requires you to translate your work in the pseudocode and flowchart to actual code using Python. Read the following program before completing the lab.

```
Write a program that will take in basic information from a student, including
student name, degree name, number of credits taken so far, and the total
number of credits required in the degree program. The program will then
calculate how many credits are needed to graduate. The display should include
the student's name, the degree name, and the credits left to graduate.
```

Step 1: Examine the following line of code. What do you expect as output to the screen?

```
studentName = input('Enter student name. ')
```

```
Enter student name.
```

Step 2: Examine the following line of code. What type of value do you expect the user of the program to enter?

```
creditsDegree = int(input('Enter credits required for degree.'))
```

I expect the user to enter a whole number, as partial credits are not likely. Ex. 63

Step 3: If the user types "Bill Jones" to the question in Step 1, what do you expect the output to the screen to be when the following line of code processes?

```
print('The student\'s name is', studentName)
```

```
The student's name is Bill Jones
```

Step 4: Examine the following code. If the program requires 63 credits, and the student has 20 left, what do you expect the output to the screen to be?

```
print('The program requires', creditsDegree,
'credits and they have taken', creditsTaken,
'credits so far.')
```

```
The program requires 63 credits and they have taken 43 credits so far.
```

Step 5: Start the **IDLE Environment for Python**. If the **Edit window** for entering code does not come up, go to **Options**, **Configure IDLE**, click on the **General tab**, and under **Startup Preferences** select **Open Edit Window**. Close and reopen the Environment. 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.

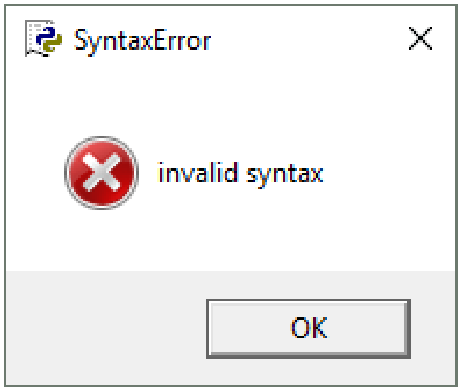
Step 6: Code should start with documentation. Document the first few lines of your program to include your name, the date, and a brief description of what the program does. Each line that you want to comment out must begin with a # sign. Ex.

```
#Sally Smith
#January 15
#Thsi program…
```

Step 7: After documentation, enter the following line of code into your program:

```
studentName = input('Enter student name.')
```

Step 8: On the menu, select **Run** and then **Run Module**. Observe your program in action. If you get a syntax error, you must fix it before you are able to run your program. Click **OK** and review the highlighted syntax error to fix it.


SyntaxError

invalid syntax

Step 9: Repeat Step 7, but change the statement so that it asks the user to enter their degree name. It is up to you whether you want to repeat Step 8 each time you code a line. It is recommended for beginning programmers so they can immediately identify syntax errors; One syntax error at a time is better than many all at once.

Step 10: Next, you should write the code that will ask the user how many credits are required in the degree. This can be done using the `input` function since it is a numeric value. Enter the following line of code into your program:

```
creditsDegree = int(input('Enter the number of credits required for the
degree.'))
```

Step 11: Repeat Step 10, but change the statement so that it asks the user to enter the number of credits they have taken so far.

Step 12: Next, add your calculation. Enter the following line of code into your program:

```
creditsLeft = creditsDegree – creditsTaken
```

Step 13: Add the following line of code into your program:

```
print('The student's name is', studentName)
```

Step 14: If you have not evaluated your program yet, do so now. Go to **Run** and **Run Module** and observe what happens… SYNTAX ERROR!

Step 15: While nothing stands out as being wrong in Step 14, notice that the word '`student's`' is actually causing the problem. To the python language, the apostrophe looks as if it is the end of the statement. Since it is not, it must be quoted out by putting a \ in front of it. change the line to the following:

```
print('The student\'s name is', studentName)
```

Step 16: Finish your code by printing the remaining of the requested statements. Your final output might look like the following:

```
Enter the student name. Bill Jones
Enter degree name. Computer Programming
Enter the number of credits required for the degree. 63
Enter the number of credits taken so far. 24
The student's name is Bill Jones
The degree name is Computer Programming
There are 39.0 credits left until graduation.
```

Step 17: When your code is complete and runs properly, on the **Menu**, go to **Edit** and then **Select All**, then **Edit** and **Copy**. **Paste** the code below.

```
#Imani Hollie 01.22.2024
#This program will collect basic student information (name, degree, credits needed, and
credits taken)

#Use the pound '#' sign to comment, Python will ignore everything after it

#Input-------------------------------------------------------------------------------------
------
#Use the 'input' function for strings: variable = input(prompt)
studentName = input('Enter student name: ')
degreeName = input('Enter your degree: ')
```

```
#You can convert strings to a numeric type using 'int()'
#Use the 'float' function since credits can be partial: variable = float(input(prompt))
creditsDegree = int(input('Enter the total credits needed for your degree: '))

#Use a forward slash '\' for apostrophe's since Python cannot read them
creditsTaken = int(input('Enter the total credits you\'ve taken so far: '))

#Calculations--------------------------------------------------------------------------------
------
#Python reads calculations from left to right.
creditsLeft = creditsDegree - creditsTaken

#Output--------------------------------------------------------------------------------------
------
#To display output in Python use the 'print' function: print(result)
print('Your name is:', studentName)
print('Your program is:', degreeName)
print('You need', creditsLeft, 'credits to graduate.')
```