3 Pseudocode, Flowcharts and Python

In Chapter 2, we learned how to store information in the computer and the rules governing the manipulation of numbers and logical values. Now we will look at how to organize those rules to create simple programs.

3.1 Sequential Order

Programs are similar to books. In a book, you start reading from the top of the page and continue to the end of the page. In English, each line of text in the book contains information that is read from left to right. Likewise, we write programs for the computer to read in this order from left to right. Remember assignment statements from the previous chapter, *a = 3* means that *a* stores the value of 3, not that 3 is *a*.

Now that we read each line appropriately, we will always start reading from the top line of our program and continuing until the last line in the program. The programs that we will look at in this chapter are all executed in **sequential** order. We will start with the first line and then continue to the next line. This **sequence control structure** can be represented with pseudocode, flowcharts, and Python code.

3.2 Pseudocode

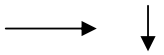
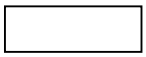
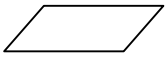
First we will look at outlining a program using **pseudocode**. Pseudocode is a language very close to English that allows us to represent a program concisely. The only thing you need is a statement to show where you are starting and where you are ending a program. We will be using the word **Start** for the start point and the word **End** to show the finish point. Each program will contain statements to accomplish our goal; this will satisfy step 3 from Chapter 1.

3.3 Flowcharts

A more visual way to see the behavior of a program is a **flowchart** which is appealing to the visual learner. A flowchart uses symbols and shapes to represent an algorithm. Pseudocode can be translated into a flowchart and vice versa.

Table 3.1 shows some of the symbols used in a flowchart where text is placed inside of the symbols. The **ovals** are used when you are starting and ending a program. **Rectangles** are used when you are executing assignment statements. **Parallelograms** are used when you print statements to the screen or get information from the keyboard. These **print** and get topics will be discussed in detail in Chapter 4. **Arrows** connect different symbols together to show the direction of flow in the program.

Table 3.1: Flowchart Symbols

Flowchart Symbol	Explanation
Arrow 	Shows the direction of the program; what to execute next
Oval 	Used for the Control and End of a program. Write the word inside the shape.
Rectangle 	Used for assignment statements. Write the statements inside the shape.
Parallelogram 	Used for input and output. Write the I/O statements inside the shape.

3.4 Python

Lastly, we will be coding our solutions in **Python** to execute the program and confirm correctness. Coding a solution is the final stage, bringing together all of the hard work and thoughts written in pseudocode and visually interpreted in the flowchart. In this text we choose Python, for more information on installing and setting up Python, read Appendix A.

Now let us look at some problems and their corresponding pseudocode, flowcharts and Python programs.

Problem 3.1:

Calculate and print the average of three numbers: 5, 10, and 15.

Task 1- Identify your input:
Values of 5, 10, and 15

Task 2- Identify the goal or objective:
Average the input values. The equation for calculating an average is to add all the numbers to create a sum. Then divide the sum by how many numbers you added.

Make sure you use variables to calculate the average instead of the numbers. Getting into the habit of creating variables now will be very helpful when you have longer more complicated programs that use at least one value multiple times. Then updating a value will take one change, where you assign the value to the variable, no matter who many times you use the value throughout your program.

Task 3- Create tasks to meet the objective:

1. Assign values for the input
2. Calculate the sum
3. Calculate the average
4. Print the average

The pseudocode of this program is shown in Pseudocode 3.1.

Pseudocode 3.1: Pseudocode for averaging three numbers.

Start
    num1 = 5
    num2 = 10
    num3 = 15
    sum = num1 + num2 + num3
    average = sum/3.0
    print average
End

Note that indentation is important to clearly show the body of the program. Practice using indentation now, it will become vital as our programs get more complex.

Also note that when calculating the average, we divide by 3.0 instead of 3. This is vital to insure the accuracy of our result. To fully understand this issue, complete the following Self Check.

Self Check 3.1
Use the values for *num1* = 3, *num2* = 5, *num3* = 6. What values do you get in Pseudocode 3.1 when you calculate the average using 3 and again using 3.0?

This program starts by setting the value of three numbers, *num1*, *num2* and *num3,* which are needed to be able to calculate *sum*. Ensuring sequential order is vital to get the result that you expect. Note that I could not have set a variable value after calculating *sum*, as shown below:

```
num2 = 10
num3 = 15
sum = num1 + num2 + num3
num1 = 5
```

We could not calculate *sum* since it would be missing the value of *num1*. This condition occurs because *sum* is **dependent** on the values of *num1, num2*, and *num3.* This is a feature of sequential control structure that specifies that we can only execute code one line at a time from the top to the bottom of the program.

Note that the order in which *num1, num2*, and *num3* are defined does not matter. Therefore, our pseudocode could look like:

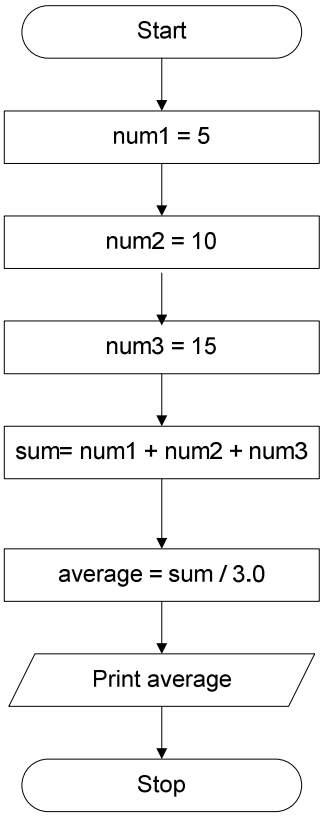
```
Start
        num3 = 15
        num2 = 10
        num1 = 5
        sum = num1 + num2 + num3
        average = sum/3.0
        print average
End
```

and it would still execute correctly since the variables *num1, num2*, and *num3* are defined before *sum* is calculated.

Self Check 3.2
One dependency was identified in problem 3.1, can you find another dependency?

Now let's examine a more visual solution to the pseudocode problem from Pseudocode 3.1 by creating a flowchart. The flowchart in Flowchart 3.1 begins with Start and ends with Stop, as all programs will. Following the Start, all assignment statements are in rectangles and the *print* statements are in parallelograms. Note that all statements must be placed in their appropriate flowchart symbol with arrows showing the direction of the execution.

Flowchart 3.1: A Flowchart for averaging 3 numbers.



Run through the problem by hand executing the flowchart to confirm that everything works as expected.

Since num1 = 5, num2 = 10, num3 = 15
sum = *num1 + num2 + num3*
sum = 5 + 10 + 15
= 30
average = 30/3.0
= 10

You can check your results with a calculator to confirm your solution.

Now let's finish the exercise with the corresponding Python program and the output. They are shown in Pseudocode 3.1 and Output 3.1.

To start making your program, you will need to open the IDLE (Python GUI) that you installed in Appendix A. Then click on File, New Window to open a screen that will allow you to type your program in. Note that the benefit of using Python is that the syntax is very similar to the pseudocode that we are using. See Python 3.1 for this program.

When creating a program, it is important to document information with comments. Comments allow any user to understand the purpose of the function and how to use it appropriately. In Python, the pound sign, '#' will start a comment from the '#' to the end of the line.

Python 3.1: Python program to average three numbers

```
# This program prints the average of three numbers
num1 = 5
num2 = 10
num3 = 15
sum = num1 + num2 + num3
average = sum/3.0
print average
```

Once you have typed in your Python program, press F5 (or Run, Run Module) to execute your program. Note that the computer will prompt you to save your code before it will run your program. See Output 3.1 for the results displayed in the original IDLE screen.

Output 3.1: Corresponding output for Python 3.1

We will see how to create output in detail in the next chapter.

Problem 3.2:

Calculate and print the square and cube of a number.

The square of a number is calculated by multiplying the number by itself. The cube is calculated by multiplying the number by itself twice.

Assuming that the number we want to square and cube is 4, let's first look at the pseudocode to outline the steps in Pseudocode 3.2.

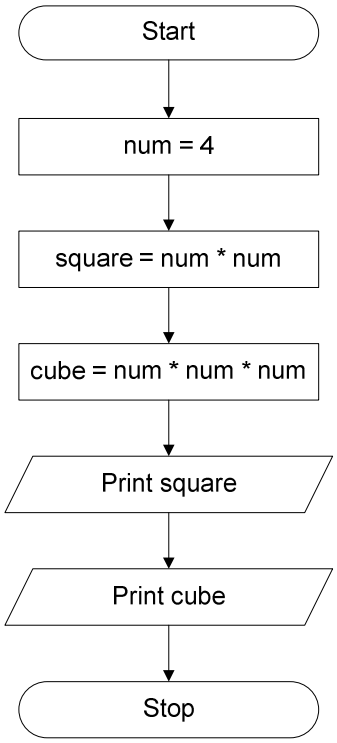
Pseudocode 3.2: Pseudocode for Problem 3.2.

```
Start
        num = 4
        square = num*num
        cube = num*num*num
        print square
        print cube
End
```

Looking at the pseudocode, you can find two dependencies. Both *square* and *cube* require that *num* be defined before we can calculate their values.

Now let us practice again with the flowchart for this problem shown in Flowchart 3.2.

Flowchart 3.2: Flowchart for Problem 3.2.



Again, run through the problem to confirm that your code works.

square = 4 * 4
= 16

cube = 4 * 4 * 4
= 64

You can now check your results with a calculator to confirm your solution. The corresponding Python program and the output are shown in Python 3.2 and Output 3.2.

Python 3.2: Python program for Problem 3.2.

```
#This program prints the square and cube of a number
num = 4
square = num * num
cube = num * num * num
print square
print cube
```

Output 3.2: Python program for Problem 3.2.

By now you should be getting comfortable using flowchart and pseudocode symbols. In the next chapter we are going to add to your pseudocode knowledge and flowchart symbols as we solve more complex problems..

Key Terms

Pseudocode	Flowchart
Start	arrows
End	Oval
Print	Rectangle
	Parallelogram

Exercises:

Draw flowcharts and create Python code to do the following:

1. *num1 = 16*
   num2 = -12
   sum = num1 + num2
   print sum

2. *length = 5*
   width = 11
   *area = length * width*
   print area

3. Create a complete program that will calculate the diameter, area, and circumference of a circle with the radius of 4.25. Use the following equations (assuming pi = 3.14159):

*diameter = 2 * radius*
*area = pi * area * area*
*circumference = 2 * pi * radius*

Answer the following questions:

4. Susan wants to put wallpaper on four walls of her room. What are the three things you would like to know before you can calculate the cost?

5. Assuming 1% of your income is spent on school supplies. Create a program that will create two variables to store the amount you are paid and another to calculate the amount which is spent on school supplies.

Projects:

1. You have decided to enter a model boat race. You put your boat at the start line next to your best friend Jill's boat. Create a program to read in the speed your boat travels in (feet per minute), the speed that Jill's boat travels and the number of minutes in the race. Print out the distance that both of your boats traveled.

*Distance = speed * time*

Sample Data:

speedMe = 6.2
speedJill = 5.9
time = 2
*distanceMe = speedMe * time*
*= 6.2 * 2*
= 12.4
*distanceJill = speedJill * time*
*= 5.9 * 2*
= 11.8

2. A recipe you are reading states how many grams you need for the ingredient. Unfortunately, your store only sells items in ounces. Create a program to convert grams to ounces.

   *ounces = 28.3495231 * grams*

   Sample Data:

   grams = 45
   *ounces = 28.3495231 * grams*
   *= 28.3495231 * 45*
   = 1275.72854

3. Read in the rate of pay (in dollars per hour) and the number of hours an employee has worked for a week. Calculate the amount the employee should be paid.

   Sample Data:

   rate = 6
   hours = 30
   *pay = 6 * 30 = 180*

4. Read in a Fahrenheit temperature. Calculate and display the equivalent centigrade temperature. The following formula is used for the conversion:

   $$C = 5 / 9 * (F - 32)$$

   where F and C are the Fahrenheit and centigrade temperatures.

   Sample Data:

   For F = 72
   *C = 5 / 9 * (72 – 32) = 5 / 9 * 40 = 200/9 = 22.22*

   For F = 32
   *C = 5 / 9 * (32 – 32) = 5 / 9 * 0 = 0/9 = 0*

5. Calculate the amount obtained by investing the principal P for N years at the rate of R. The flowchart in Figure 1.5 shows the sequence of steps necessary to accomplish this task. The following formula is used for the conversion:

$$A = P * (1 + R) \wedge N$$

   Sample Data:

   For P = 1000
   N = 5
   R = .05
   A = 1276.28

   For P = 1000
   N = 10
   R = .05
   A = 1628.89

6. Read the electricity meter at the beginning of the month and at the end of the month. Also read in the price/unit of the electricity consumed. The following formula is used for the calculation:

   Number of units consumed: *ending meter reading – beginning meter reading*

   Cost: *Number of units consumed * price/unit*

Answers to self-check problems:

3.1:
When dividing with 3, you get 4.
When dividing with 3.0, you get 4.666.

3.2
The variable *average* is dependent on *sum* being defined.