Wow, we have learned so much so far in this module. We've learned that data in a computer program are stored in variables. That every variable has a name or label, a type and a value. We've learned how to associate values to variables using assignments, and we've learned a bit about arithmetic expressions.

Now let's see how this all works together and how we can use it to solve a problem. Here's the problem that we want to solve. It's the middle of the semester and you would like to know what your grade is so far in a course that you're taking.

You know that you have completed all of the participation work, earning 100% so far. You also know your grade for the first exam, your averages for the labs and the homework assignments that you have completed thus far. The final exam hasn't occurred yet. And the syllabus gives you the following table for calculating your grade.

In order to solve this problem we need to make sure that we understand what the problem is asking. In this case, the problem is asking us to calculate a weighted average. In a weighted average, each grade is multiplied by the appropriate percent and they are all added together.

So let's work out an example of this. So if I have 100% for the readings, that means that I've actually earned 5% of my grades so far. Okay, but what happens when I add grades for labs, homework and exams? Let's say that for the labs I have a 97%.

For homework assignments a 99. And for exams, an 89. Then one way that you can solve the problem is with the following calculations. We take each portion and multiply it by the percentage that's on the syllabus. So the 100% is multiplied by five, and we get 500. 97 is multiplied by 15, which would give us 1455.

And you could put these in your calculator and verify. 99 times 50 is equal to 4950. 89 times 30 is equal to 2670. Now if I add all those together, I get 9575. To get an average I need to divide by 100. This gives me 95.75, awesome. That's a great grade in the class.

This is an example of a test case for our program. Let's call it test one. It may seem strange that we're thinking about test cases at this point in our process, but it often helps clarify how to solve the problem. Whenever we do this, however, it's a good idea to think about whether the solution that we've come up with is a reasonable answer.

In this case, 95.75 is totally reasonable. Since we usually want to calculate our grade over and over again during the course of a semester, this is a great thing to write a program for. I can't tell you how many times this semester I run this calculation for students.

So let's design an algorithm and draw a flow chart before we start writing the code. We should start by recognizing that some of the values in our calculation are literals and some are variables. Variables would be different each time we run the program, so that is the averages.

So I would like to define variables for these. So let's start a flow chart. We start a flow chart with the start. And then the next thing I need to do is to have a box that reads in the variables. And the variables I want to read in are for the readings, the labs, the homeworks, and for the exams.

Once I have the input, the next part of my program will be the calculations, similar to what we did on the whiteboard. And I'd like to break these up into individual steps and store them in variables to keep each calculation simple. All right, so that means I'm going to have a regular box, a rectangle, where I'm gonna do the calculations.

So for the reading part, it's going to be readings times 5. This is what I'm getting from my whiteboard example. Lab part is equal to the labs times 15. Homework part is equal to the homework times 50. Exam part is equal to exams times 30. And then, I'm gonna take this and do the calculation for average.

Average is equal to the reading part, plus the lab part, plus the homework part, plus the exam part all divided by 100. Great. That ends my program. Now that I know the flow of the program, I can start to write it in Python. Instead of using the interactive session as we have done before in this module, I'm going to put it into a file so that I can run it over and over again.

So let's start by creating a new file. The Py extension is used for all Python source files. So I'm going to name this weighted average. So, weightedaverage.py. To start this, I need to add the starter code that we introduced in the introduction module. Remember that I'm going to want to put the code for the program in main, such that it's indented one level.

So def main and then I'm gonna have main. Okay, now I'm ready to write code. So all we have to do to write code is to translate our flowchart into Python. The first block of the flowchart shows that a bunch of variables are read in from input. To do this we need to get input from the user and make sure that it's an integer and this is how we do that.

So reading is equal to int input enter grade for readings. This line does a bunch of stuff that we will explain in detail later. But for now, you should know that the thing inside the parentheses happened first. If you look at the innermost parentheses you will see a string.

This is called a prompt because Python will display this message when it prompts the user for input. That's just outside that first set of parentheses, the input. Input is how we get information from the user. By putting the input inside of the parentheses for int, we are then taking whatever the user enters into the keyboard and converting it to an int, and thus assigning that value to the variable called readings.

There is a lot here, but for now, that is all you need to know. You will learn more about both int and input in the module that introduces functions. So let's do the same thing for our other variables. So labs is equal to int input. Enter grade for labs.

Homework is equal to int(input("Enter grade for hw: ")). Exams is equal to int(input("Enter grade for exams:"). Notice that I'm using spaces around the assignment operator. Using spaces around operators helps with program legibility and is considered good style. The next two blocks in the flow chart show calculations for each part of our weighted average.

I'm going to add a blank line to separate this logic from the previous logic. This makes each section of my program easier to find. I'm also going to add a comment to the top of the block so that I know what this block is for. In Python a comment begins with a pound sign or a hash tag.

It indicates that Python should disregard the rest of the line from that point. Adding comments to our code can often clarify what we're thinking about when we wrote the code. So I have my comment and then I'm going to do each part. So read_part is equal to the reading times five.

This is just like I did on the whiteboard. Lab part is equal to labs times 15. Homework part is equal to homework times 50, exam part is equal to exams times 30. The last block in our flow chart shows the calculation of the average. So we can write that here in straight line form.

So average is equal to the read_part plus the lab_part plus the homework_part, plus the exam_part all divided by 100. Okay. We are now ready to run our program. Remember from our example that when we run this, the result should be 95.75. This is called the expected value. It's a good idea to think about what you expect the value to be before you run your program.

That way you actually know whether it's right or not. Okay, let's see. Enter the grade for readings which was 100. Enter the grade for labs, which was 97. Enter the grade for homework, which was 99, and the grade for exams was 89. It doesn't look like anything happened.

So what happened? It may look that way. But in reality, something really did happen. The reason that it looks like nothing happened is because unlike in the interactive mode, Python programs need to be told to write things to the screen. To do this, we need to explicitly tell Python to print the average to the screen.

So I'm going to go back to my program and do this. To print to the screen we write the following: print("Average is", average). Like input and int that we used earlier we now are using print. Inside the parentheses are all of the things that I want to print on that line, in a list separated by commas.

Here I'm printing the literal average is, followed by the value of the average variable. Python will be nice and automatically include a space between each element in the list. Notice that elements in the list are really separated by the commas. Now let's see if that makes a difference.

So I save it and then I'm going to run it again. 100, 97 for labs. 99 for homework. 89 for exams.

Average 69.317. Now great the program actually printed a value, but it's not the expected value. This is a logical error. Sometimes the error is obvious but not always.

In this case, we need to find some more test cases that will show which part of the problem is wrong. For each example, we will calculate the expected value. Remember, a test case without an expected value really isn't a test case since we don't know what the answer is supposed to be.

In test one, it's really hard to know what the right answer is. Did we calculate 95.75 correctly? Or did we make a mistake along the way? I don't know about you, but I make multiplication addition mistakes all the time. What about some other examples that give us more obvious results?

Since we already have one test we'll start with test two. How about an example for test two where all of the parts are 92%. Readings is 92, labs is 92, homework is 92, exams is 92. Then the expected value would be 92. What about a test for test three where the reading is 100 but all the rest are zero?

Readings is worth 5% of my grade. So then the expected value would be 5. Test four could be where labs are 100 and all the others are zero. And since labs are worth 15% of my grade, the expected value would be 15. Test five could be where homeworks were 100 and all the others are zeros and since homeworks are worth 50% of my grade, the expected value is 50.

One more where the exams are 100 but everything else is zero. And so because exams are worth 30% of my grade, the expected value is 30. These test cases may look very simple, but they test each part of our calculation one thing at a time. Learning how to find these examples like this is part of what we will do in practicing this course.

So I'm going to run through these examples in reverse order. So here is my program and I'm going to run it with readings is zero, labs is zero, homework is zero, exams is 100. Expected Value 30. And look, the average is 30. This is what we expect. So we're doing something right.

All right, so moving to the next test where readings is zero, labs is zero, homeworks is a 100, exams is zero, expected value 50? Well, we get 5,000. That's not right. So why do you suppose we got 5,000? Let's take a look. Can you see why we got 5,000?

That's right. When we're dividing by 100, precedent says that the division will happen before the addition. So the division is only dividing the exam part, not all of the parts. To fix this we'd better add parentheses. Okay. Now let's do test five again. Readings is zero, labs is zero, homework is a 100, exams is zero.

We expect 50. Yay. Now we get 50. Okay let's try test four. Here reading is zero, labs are 100. Homework is zero, exams are zero. Expected value 15. Yay. That's what we got. You might say well we're done. But we wanna continue to test these because we have these nice test cases.

So we're just going to keep running them. What about test three? Here the readings is 100, the lab is zero, the homework is zero, the exam is zero, the expected value is five and that's what we got. Great. Test two, here they were all 92, so the reading is 92.

The lab is 92. The homework is 92. The exam is 92. Yay, and we get the 92 that we are expecting. What about our original test? The readings were 100, the labs were 97, the homework was 99, and the exams was 89. 95.75, very nice. You've just written and tested your first program.

It might seem like we spent way more time on paper in making sure our program was correct than we did writing code. It's true, we did. In the so called real world, the best estimates are that programmers spend only 10 to 20% of their time writing code. The majority of a programmer's time is spent working with others understanding, designing, testing, debugging and documenting code.

Also, remembering to use good style. Okay, that's all for now. Thanks for watching Align Online.