Week3 Lecture 1.4

So welcome back. Now we're going to start learning about the Python language and you can think of this as talking to Python itself.

And it turns out that there is a way on most computers - whether it's a Windows computer and command line or a Macintosh or Linux box - to get Python started.

And if you just run Python, it will take, interactively, commands that you can type.

And you have to type "python" at this chevron prompt, is what we call it, but this also sort of fits into what next.

So Python is like "Okay. I'm here. I can handle any Python statement you can send me.

I'm ready to do whatever it is that you want to do. I don't know what to do.

I need you to tell me what to do." And so you can type a series of statements in Python.

And so the first statement that you might type is an assignment statement; x equals 1.

And so what is going on here?

Now this, this assignment statement is something that often confuses people when they move from math to programming.

An equal sign sort of has a direction to it; it's an arrow.

It really is saying "Dear Python, you've got a lot of memory - you've got a lot of memory.

Take a little tiny piece of that memory and remember it and name it x.

I might use that - I'm going to use that later - and stick a 1 in it." So this is sort of like stick 1 in a spare place in memory and name it x. print (x) says, go take whatever that spare bit of memory was and bring it back out and tell me what I put in it.

Now, it's kind of redundant, but usually you're doing something more complex than this.

Put something in memory and then take it back out; that's the first thing.

Now what this is doing is this is an expression and that says take

whatever's in x, which is a 1, and then add 1 to it, which becomes 2, and then stick it back in x.

So that adds 1 to x and then we print that out and it's a 2 and then we quit.

Now if you type wrong things here, you're going to get syntax errors.

And Python is just going to tell you, you know, syntax error, syntax error, syntax error, and away you go.

But this is us talking to Python. So what do we say to Python?

You can almost think of this as like writing an essay where you start - and if you think back, you started learning an alphabet and then you used that alphabet to produce words and then used the words to produce sentences and then you combined sentences to make paragraphs, which then make a story.

And we're going to do the same kind of thing.

This is the program that we're ultimately going to write.

And don't worry about understanding what this story does.

We'll talk about this later, we're going to come back to this many times throughout the course.

This is a Python story about how to count words in a file in Python.

Okay? This solves that problem that I asked you to solve a little while back of what the most common word was and how many there are.

And we'll come back to this several times in this course.

So if we start at vocabulary, the first thing we have in every programming language is what's called reserved words.

Now, what do we mean by reserved words?

Well, these are words that if we use these words, we must use them to mean the thing that Python expects them to mean.

Another way to put that is we can't use them elsewhere.

We can't make up a variable named import.

We can't make a variable named assert; because if Python sees assert, it means something very specific to Python.

If it sees if, it means something.

If it sees for or pass or while - where is while?

There is while. These mean things.

The best way to think about this is Python is kind of like a dog.

And if you're talking to a dog and you say hey, dog, do you like, you know, Beethoven or Bach better?

The dog is hearing - is like blah, blah, blah, blah, blah, blah - it doesn't care anything.

And then you say hey - hey there, Spot.

What do you think of the weather today? Do you really like, you know, sunsets or do you like rain better?

And the dog is like blah, blah, blah, blah, blah, blah.

And then you say something like do you think it's time to go for a walk? And the dog goes, like, walk - got it.

And the dog's been listening to you all along.

Most of it is blah, blah, blah, blah, blah, but walk is a reserved word for a dog.

Okay? Food, treat - those are reserved words for dogs.

Most of stuff you can say anything you want to a dog and they're very good listeners, right?

But when you say a reserved word for a dog and you say walk and you don't deliver and then that dog's gonna bug you for a while.

So you can think of that as Python.

When Python sees global or from or for, it's like whoa - that's a word that means a lot to Python.

So you'd better use it the way Python expects you to use it.

Sentences are lines.

And so when we write a program, we're writing a text file and we put a line and another line and another line.

Each one of these is like a separate line and we've got to get them right.

And then we construct a paragraph out of a series of lines.

And so in this particular example,

we have another assignment statement that's sticking the number 2 into the variable x,

retrieving that 2 back in, adding 2 to it and sticking that sum back into x.

Print is a function and there is a parameter x and that's going to cause 4 to be printed out.

So we're using operators, this plus is what's called an operator, the equal sign is what's called an operator.

This print is actually a function. This is a parameter being passed into the function.

So there's a whole series of different kind of syntaxes that we use to produce the lines that we have to produce.

Now we start combining these - the lines into paragraphs and make some sense.

And so that interactive Python, where you're talking with the three chevron prompt - that's fun and it's fine to get started and type x equals and x equals x plus 1 and whatever.

But don't do too much.

And I see students all the time, they try to write whole programs in that and it gets a little

frustrating because you have to type it perfectly from beginning to end.

It's much more common to type something in a file.

Once we get maybe beyond three lines of Python, we tend to just use a programming text editor like Atom or something and we put them all in a file and then we tell Python -

start at the beginning of this file and then read through the file.

We call this a script or a Python program.

And in Python, we tend to call that file .py. And some text editors,

like the Atom text editor which is one of those that I recommend, when you have a file that ends in .py, it syntax highlights it and gives you colors, and makes things pretty and helps you understand and sometimes even leads you towards syntax errors and mistakes that you have.

And so scripts are stored sets of instruction in text files that you can then hand to Python to run them.

And I have a whole bunch of videos that sort of get you to learning Python - getting Python installed, doing something interactively, and then writing a script and then running from the script.

So like I said, you can do interactive in Python or you can run a script.

And a script is much more common.

Once you - after about the first 15 minutes, you pretty much do everything with a script.

So what do you put in that script?

Well, it's a series of steps.

Now there are a couple of basic patterns that we use and we compose them.

The most basic pattern is what's called sequential.

We do one thing, then we do the next thing, then we do the next thing.

Conditional is sort of intelligent where you're doing something and then may or may not be doing something.

Do, do, do, something,

something, something, maybe not, maybe - and you have this "if".

If this is true, do this statement; if it's false, do some other statement.

Those are called conditional steps and we'll show you how - in a sec how those look.

And then the real power of computers come when we tell it to do something over and over and over again a million times, if necessary.

And that's when we have a series of steps that need to be repeated.

And then the fourth pattern is the store and retrieve pattern, which we will visit in Chapter 4.

So this is an example of a four-line Python program that has four basic steps.

Python, we - as if you typed these four lines into a file and then told Python to execute them.

Python will stick 2 in the variable x, then it will print that out; our output will be this.

It'll add 2 to it and then print it out again and it'll be 4.

You can - this over here is called a flowchart.

I don't really make you write flowcharts.

In the old days they made us write flowcharts.

I'm not sure that's all that valuable.

All I'm trying to show you is the sequence of things.

It runs this, does something, then it moves onto the next one, does this, then it moves onto the next one, does this, then it moves onto the next one.

Meaning that when you're done with the statement, where are you going to go?

Well, in this case, it's the simplest of all possible things.

Where we're just going from one thing to the next thing.

Now, it gets a little more interesting when we do conditional steps.

And the thing that makes conditional happen is this if, right?

The if statement is a reserved word, okay?

And so you'll see, it starts at the beginning and the if has embedded in it

a question that leads to a true or a false, ends in a colon, and then there's an indented block.

And what happens is this is the conditional statement.

Print smaller is the conditional statement, meaning that if x is less than 10 evaluates to true, then it's going to execute;

otherwise, it's going to be skipped.

So it can either execute it and continue or it can skip right over it.

And so it's probably easiest to show this on this diagram, where you start with x equals 5, then you hit the if statement and the if statement is asking a question.

If the question is true, you go down one road and if it's false, you go down the other road.

In this case, x is 5; so this is true and so we go down this path and print out smaller and then we rejoin.

And the next thing we encounter is another if statement.

Is x greater than 20?

Well, in that case, it's no, because it's not. Because x is 5.

So we're asking the question.

We're not changing x, we're just asking a question about x.

So we skip right over this.

So the way this code ultimately runs is it comes in, runs this,

sequentially comes to here, conditionally does run this line of code and then skips completely over it.

So it's this code never ran.

And that's a super simple example.

So it said the output is smaller and then finis and it's a super simple one.

And what you'll find is we compose these things to make more complex programs.

So conditional is the second of the three patterns we're going to see.

Now the most exciting pattern is the repeat.

And it looks a little more busy because we can do more stuff with it.

So the basic idea is there are a couple of different looping keywords, reserved words - the while and the for.

The while loop basically says - it functions kind of like an if statement in that there's a question in it.

It doesn't hurt n; n doesn't get changed by this, it just looks at n and asks the question is n greater than zero?

If it is true, it runs this code.

So just like an if.

So up to here, it looks like an if - if n is true, n greater than zero is true, it runs this code.

Now this is a little sequential bit.

When you're done with this, what do you do? Oh, we just fall onto the next one.

And you'll notice that this is indented at the same level as the print statement.

And that's how we have repeating or even ifs with more than one statement in it.

And then when it de-indents, that's the end of the loop.

And what happens is it runs this and it runs that,

but then what does it do?

Well, it's at the end of the loop, so it actually goes back up to the top of the loop.

So it goes back up to the while statement.

And we actually printed n out, which was 5.

And then n became 4 and then it reevaluates this question - is 4 greater than zero?

Yes, it is. So it prints 4 out and then it subtracts 1 and it becomes 3.

Now, is 3 greater than zero? Yes, it is.

Yes, it is.

And so it goes 4, 3, 2, 1.

Then it comes through and it prints 1 and then

it subtracts 1 from it, so it becomes zero.

Now it comes up and asks this question - is zero greater than zero?

Because n has become zero by the successive iterations through this loop.

So n becomes - is not greater than zero.

So this switches from a true to a false.

And false, it leaves - goes out and exits the loop.

So over here, it goes 5, 4, 3, 2, 1 - and then out she comes.

Okay? And as soon as this becomes false, then this loop exits.

And so that's the way we tell a program that - you know, we tell the computer that we want it to keep going until something has happened.

Some - we have achieved something.

Who knows what it is we're doing?

None of these - none of these programs make any sense particularly, we're just sort of learning baby steps.

Okay?

So loops have this notion of iteration variables to make sure that they are not infinite loops.

And this variable n is carefully constructed to start and be checked and then be changed each time through the loop.

And that's how we make sure this loop only runs five times and not forever.

Because you can construct what we call infinite loops that run forever, but it's not too practical; you just run out of battery or whatever after awhile.

So you generally construct loops to finish so that they have to check to make sure that they're finished. Okay?

So if we take a look back at that story, we see that in that story, that is, how you count the most common word in a file and print what the word is and how many there are,

you find that there is some sequential code.

And in Python, the way you can tell that it's sequential code is when it's not being indented - it just goes - it goes down, down, down, down, down.

Now the for is another of the reserved words that a loop - so this has a colon and it says that's an indented block and there turns out to be a for within a for.

This is called nesting.

We'll get to that.

And then this loop ends and it runs for awhile.

And then there's more sequential stuff.

There is a for, which is repeated code.

And then there's an if nested within that for; that's conditional code.

This'll run for awhile and then it comes out and does some sequential code.

So there is, in this, sequential code, repeated code, and conditional code.

So the three of the four basic patterns of programming

we see in this file.

And so if you look at this from an outline perspective,

this is sort of our story.

These are the paragraphs of our story.

This top bit here says read the name

of a file from the user and open that file so we can read it.

This second paragraph says oh, as we read through the file, create a histogram that maps the number of words to the frequency of the words and each time you see a word,

update the histogram so that we have a running total of all of the words, using a thing that we'll learn later, like in Chapter 10 or 9, called dictionaries.

And that's cool, that's cool. But don't worry about it,

don't worry about it, don't worry about it now.

So this is making a histogram.

Then once we have the histogram and we've read all the words, then we read through the histogram to find the largest.

And then we print out the largest word and the largest count.

And so this is a little short story with some sentences and paragraphs and an alphabet and reserved words and vocabulary and all that stuff.

And we're kind of learning how to do it.

Don't try to get this all right now.

We're going to touch on this, we've got a whole chapter on variables and a whole chapter on conditional, a whole chapter on loops.

Right? I just - I'm trying to give you this big picture so that when we get there,

you can start putting those things together.

And like I said, it's a little confusing.

Generally it really starts to clear up in Chapter 6 and Chapter 7.

So just understand that we learn little bits and little pieces and then it gets better toward the end.

So thanks for listening.

Chapter 1 was really trying to get you an overview of the kind of stuff that we're going to learn throughout the course and lay down a little bit of vocabulary so I can talk about things like microprocessors and RAM and memory and stuff like that so that you - so I can communicate, say hey, we're going to do this thing that's going to do something

with this CPU and so that - so you can use that.

So I hope you find this value and see in the next chapter.