[Captioner standing by]
>> PROFESSOR: Today we will get to this and finish of this lecture. Let me know if you are having trouble with groups. How many of you have Had loops? How many of you have had trouble
with loops? Yes? Loud, loud. A loop is execution of a code, a score that is going to be repeated. It's the main idea. If you want this piece of code to repeat. The next question is how many times? Once the code to repeat. We need some way to control how many times the code
is going to repeat.
>> STUDENT: [Away from mic]
>> PROFESSOR: Loud, loud.
>> STUDENT: Under the conditions true, they will run.
>> PROFESSOR: If none of the conditions are true, repeat the code. That is one way to control
loops. One way to control loops is you have a condition that says if it does not rain I will keep walking. In that case you have a condition. If this condition happens the code will repeat. that is one way to control loops. Specifying a condition that will control the execution of the loop. As long as that condition is true the code will repeat again. Is that completely clear? One way to control groups is by using the condition that has to be tested and as long as that condition is true this code will execute. Is this completely clear? Yes? No? Maybe? Okay. One way to control the loop. There is another way to control the loop. The other way to control the loop is to specify what is the collection of elements that you want to apply that code too. A set of elements we want to apply the code to. Let's say the group of students here, suppose my code is to say hi to everybody. I will say okay. For every
student here, I'm going to say hi. The piece of code is, say hi. Hi. How are you? How are you doing? Hi, how are you?
>> STUDENT: I'm good. How are you?
>> PROFESSOR: I'm good. Thank you. Hi, how are you? It's your name? How do you spell it? Avi, how are you? How are you doing? What is your name? Andrew. What am I doing? I am over
a, every element of the collection, I am doing something. Saying hi. So I will continue. Right? This is quite different to the other way of controlling a loop. Which is having a condition. This picture here designed to capture for you, these two ways. To control a loop. One way is you have some condition. The condition is tested. As long as the condition is true
we will come back to the condition. There is something missing here in this picture. What is missing in the picture? If the condition is true we have to do something. So the condition is true here. I have to do something. Right? That is the piece of code. So here, this would be something like if the condition is true, execute the code and go through the condition

again.
This is the action, this is the code if the condition is true. Is it clear? It's important to understand the diagrams because these diagrams specify the semantics of the language. And the
semantics is independent of the language. It's the same in Python, Java, the same in any language. This is the semantics of a way to control the execution of a piece of code, be a condition. And this is expressed in the language by using this language instruction called WHI
LE raise. Raise the language, formally. To indicate this. Why? This condition is true. You want to have a picture. You can have that if it is true. Execute the code and continue. You will get out when the condition is forced to go and do something else. The only specific instruction in the language for this is this. Yes. What? Loop by loop. It is different, actually. You can simulate one with the other by using semantics but this is the most used way
to control execution of a code. There are other language with other ways to express this. Doing why is one of those. We are not discussing that. We are discussing thesesemantics. Any questions about this? Is this Completely clear? Okay. We are just looking out one way to
control groups. Which is by using the test expression. You want to see an example. This is the same thing we have on the board. The example here you are defining the function. The function has a name, summit, perimeters, CAP,:, we can see the body of the function. Applying
a variable to one. We are going to collect the sum. As this is happening, here the why loop is saying we are going to execute this code here. As long as this condition is true. This is the condition. So how look at this condition, as the interpreter, look at the code. When it comes here it has to have some value. That variable is being compared with something called
cap, CAP. A parameter specified here. As long as this happens, this piece of code is going to execute. Exactly as I'm doing with the mouse here. While this is happening you go here, come
back, go here, come back. Go here, come back. Go here and come back. Question. What has to
happen inside here. Something must happen here. Yeah. I know but I'm telling you to express,
I think you have time to express it a little different. What must happen there in that code? What is the purpose of all of this? The purpose of all of this is while this condition is true, execute this. Yeah. What was happening here? You want this eventually to finish. As long as it is not raining I will keep walking. What must happen for me to stop walking? As long as it's raining I will keep walking. What will have to happen for me to stop walking? Yeah. Yeah. Louder. Okay so if it's not raining I will keep walking. So the condition must be falsified, I, at some point, if it rains, I will stop walking. So for this, to eventually complete, there must be something here inside the code that affects this condition. There

is
nothing here to affect the condition, what will happen? If it doesn't rain, you were in the desert. What happens? You will never stop walking. Never. Never. So if, there is no mechanism inside this code that affects the condition that creates what is called an infinite loop. The condition is not affected in the code, this will continue forever. And that is called an infinite loop. That condition is one of the most pervasive. Mistakes in developing code in any problem. With that in mind, tell me what is it here that is affecting this condition in this piece of code? What is happening here that is definitely affecting this condition? Do you know? Do you know the answer? Yeah?
>> STUDENT: [Away from mic]
>> PROFESSOR: Okay. What he is saying is if I understood correctly what you are saying, this
statement here. The way to understand this statement is this variable is being implemented by
one. This is the way to understand this. This variable a UM is being implemented by what? This is a way to understand this. Any time you come here that variable is implemented by one
so next time variable is implemented by one, next time the variable is unlimited by one. Eventually, we expect because the variable is being increased we expect this condition will fail because this is the upper bound for this condition. As long as this is less than or equal to that, whatever it is, this will continue. Without this statement, without this statement this is going to be always true. Raise your hand if you got that. This kind of mechanism to implement something inside a loop that will affect the condition is called accounted. In the language of computer science you say that here we are using this variable as a counter. Ms. variable is counting how many times this is happening. What happens if you do this backwards?
What does it mean backwards? You look at this piece of code here. What is different? Again, we have a condition. The condition is using this parameter here. This is saying this parameter is greater than zero. We have this, means implement this variable by this value. What's happening here? What is this doing? Ya? We are reusing the value by one. This is the meaning of that statement. This is counting backwards. This counter is a different type of counter but in the same idea. In the previous case we were implementing the counter. In this
case we are - - the counter. This is the one affecting this condition here. This is the one affecting this condition here. This is another way to write this process here. It happens to be that infinite loops are something that you really have to be aware of. It should be a scale of infinite loops. There is a way to protect yourself from infinite loops. By using the break condition. Here we have something similar to before. What we are doing is adding a check on
this. If it happens to be this is zero for whatever reason. It could be something silly here. For whatever reason, if at some point biscuits to zero you say break. What that does is push

the code out of the loop. And when I say out of the loop, notice that this statement here has
an interesting condition which is true. This is not a normal condition, it is saying while this is true, do this stuff. How is this going to be affected? There is nothing here affecting this directly. That is why you break. .2, Notice there is a matching indentation between this while loop, the why loop, and the value of this. Okay. There is another piece of
jargon used in computer science. It is called a guide or a - - can anyone tell me what the heck is that?
>> STUDENT: The final condition of a repeating to make sure the rules of this value - -[away from mic]
>> PROFESSOR: Very good, good. In computer science jargon, when we say okay, I'm going to run
some kind of loop over this loop. In this case, you guys. And I'm going to say hi, as long as you are registered in the class. So you get to - - and say are you in the class? Hi, how are you? Are you? Yes, etc. At some point you get a value that is not in the class. Then you are using the condition as a Sentinel. Hey, this is not part of the loop. This is jargon to be aware of because when you design code with other people people say oh, let's put a guard there.
But a sentinel value there. Here is an example. Here is a very simple loop. You input something, the value is zero, you break. You were implementing the value of that as before,
whatever is being entered, etc. In this case a, the value is negative. Excuse me, the value is not an integer. It's just jargon. Okay. Let's look at something a little bit more contrived. Not difficult but contrived. What is a prime number? A prime number is an integer that what?
Only one person? Only two people? It is only this group by? One and itself. So if I give you a number, how do you test if that number is prime? Check this condition. You take that number, raise the number if it is divisible by two. If not then you can go to three. If it is divisible by three, if not, continue. Right? Well this is a very intelligent way to do this.
If the number is I don't know, 10 million. Whereas they are going to be doing this test for two, four, five, etc. Too many. Too many checks. There's a couple of things you can do. To test if the number is prime, you don't have to go through all of the elements that are smaller
than the number. You only have to go until the square root of the number. To check a number,
to see if it is prime or not, only check until the square root of one. Can anybody give me a reason why this is the case? You have to check two, three, four, five, six, up to the square root number. You don't need to go past that because you already checked the previous numbers.
And if it is something bigger than the square root that is because it was divisible by something smaller than the square root. This is a limitation. Square root of N is a number

that is a lot smaller than and in this case. So and is really here. God is.2. If the number is not divisible, Can it be divisible by four? If it is not divisible by some K, then you can use that as information not to check multiples of that. So the point is, this is not a classic number theory. But the point is that this has some limitations that can be applied. This particular piece of code here is telling you how to do that. To check if the number is prime, less than one. We have positive numbers. If the number is one then it is not considered a prime number, if the number is two. If the input number is two, a prime number. Two is a prime number. After that if the number divided by two is equal to zero that means this number
is a multiple of two. It is not prime. So this is easy. Based cases. After that you start with equal three and you are going out to take the Y loop while this number is less than a square root. Using the library math and the SQ RTA. Looking at the square root of that number. As long as it leads to that you were going to take your number, divide by D and of remainder is zero you say the number is not prime. After doing that you implement D by two
every time. You go to the Y loop again and repeat this. Raise your hand if you got this? Raise your hand if you got this. How to check the number is prime in Python. So what happened
with you? You don't get that?
>> STUDENT: I got it.
>> PROFESSOR: You guys? Yeah? How to check if the number is prime in Python? Yes? Yes? Okay. Now we are going to use that. This is a whole function. A base case. Has a Y loop, returning to the low distance. So now we are going to use the function inside something. That
is going to wrap up that function definition before. And this is another function. To say try prime. Now look here, what we want to simplify, this is a Y loop which is why true, meaning this whole thing is going to be executed unless this becomes Forest. So somewhere on the line
this condition has to be fortified. Somewhere along the line here. Is that clear? Somewhere along the way it has to be fortified and it will be an infinite loop. So here we have try and accept which is the topic we called last lecture in which we are going to try this piece of code and there is an exception that is going to be raised. So we go inside the piece of code. And, input something, some integer if the value is zero. Great. Get out of here. If the value is less than one then the input must be an integer. If it is less than one it is because it is something funny. Now here, can anybody tell me, especially those people that claim they
know Python. What is this doing here? This is a Python instruction. Yes, back there? Louder, please. Okay. Thank you. Good. That continues saying you know what, I want my code
to complete gracefully. Regardless of this funny input. To compute you type continue so you
don't have your coat crushing. Is that clear? It's a safety mechanism incodevelopment. That

is another form of this language. At some point we will use the function that you claim you understood to check if the number is prime or not. And print it out etc., etc. Okay. You can check the results of that. We mention continue now. There is a modification on this fundamental Y statement. That will allow you sometimes to do very creative stuff. Here we have why - - which is what we are discussing. Until the expression is forced. Then at some point if this goes correctly, that means that this code here executed beautifully. You get here and this code is executed beautifully without problems. So sometimes you want to add this
close here after the Y loop because that is going to give you confidence that you are code reaches this statement. These are things to help you make your code more reliable. So that is
this first construction in Python. Close the Y loop. My condition is true, execute the code and if inside the code there is something after the condition to make it force at some point then you get out because look at the other. The other control of a loop. It is quite different. It has the flavor that is the same thing, is not the same thing. You look at the diagram and they look very much the same. They are not the same. And it is important to understand the difference between what is called a full loop and a Y loop. The main difference
is here. Over here there is a condition that will be written down while this number is less than this for example. Or some expression that is going to be - -. Over here we don't have such a condition. Over here we are going to specify that we are dealing with a collection. Give me an example of a collection we have discussed in class. Only one person? You guys don't have collections? Come on. Yes? A list is an example of a collection. Give me another example. Sam, same. Same. What? I'm sorry, I cannot hear you. A dictionary. Excellent. That is another example of a collection. Give me another example of a collection? Back there.
What? Do you like that one? Yes? Okay. This is a lot more powerful. Because what this is saying is my collection is the set of students in CS 210 and we can specify that. I can do plus, that can be a prime to the set of students here. As long as I specify that is my collection and this is the list of students. Then I say for every element in this collection, say hi. Every element in this collection, send a coupon. For every element in this collection, but bonus points. Just by specifying the collection you can apply very complex processes to all elements in the collection. Is this completely clear? The elements in this collection are what I call intelligent meaning you can integrate over. The Python language part consists of these two. We elements that is just a variable. Where in that collection, these are the two pieces of Python. It is going to be executed for each elements in the collection. Somehow when one needs to specify, this is the sequence, the set, the map. And now for every element in the collection, if the element is in the collection, if that is true we execute code and come back to another elements in the collection whenever we get to the
point in which an element is not in a collection we at question. If I am integrating over a collection how can you make sure at some point you get out of that? What can you make

sure
that you have completed the process for all of the elements in the collection? We just said that minutes ago. You use what? A guard or a sentinel. So if it happens to be that somehow in the input at some point at the end there is something that does not belong to the collection
you can specify that as the guard. A sense and it took control. Let's look at an example of this? One way to specify collection is by using range. Can anyone tell me what ranges? What does it specify?
>> STUDENT: [Away from mic]
>> PROFESSOR: What kind of numbers? Range is a way to specify a collection of integers. The
range has two parameters. How you begin and end of the range. When that is not specified there is assume. Assume. For example we say range six, that is taking numbers from 0 to 5. Remember in Python everything begins with zero. So zero, one, two, three, four, five. For R in this range what that means is for R, here in this collection, what is it you are going to do? You were going to implement the sum that you have already initialized here by R. You are
implementing that and printing it. Why the answer here is - -? Why is that the answer? Are you following this? Are you lost and confused? Can you tell us? Why is it 15? The answer to that piece of code.
>> STUDENT: It's because it is adding up to five.
>> PROFESSOR: We are adding the numbers. Zero, one, two, three, four, five. We are adding
those numbers and getting a perfect 10. Yes? Suppose that now I want you to compute the sum
of numbers. For the first hundred integers. How would you do it?
>> STUDENT: R range, 100.
>> PROFESSOR: For R, 104 101? You would take one more number, right? So you want one of the
first elements to be up to? 99? You have to be careful with this. The point is just by making the change, okay. Then you can use the same piece of code to add all of the numbers.
Yes? This is a very simple way to specify the collection. If it is integers, use range. So what happens with the next one? Which is a little bit more fun. You are specifying a different range. Starting from? One. Up to? 10. In that range, what we are going to do is implement these elements by two. This specification is telling you this is the collection. One, three, five, seven, nine. You start with one and you go up to, blah blah blah. Think about how powerful this is. This is one of the nice things about Python. You cannot specify complexes by specifying in the case of integers the right parameters to specify range. So now
you are getting some of these, the same as the operation we did before but now you are doing

something a little bit more interesting. And actually, you can also specify when this is one minus, all right. Start with a, you want to Pastaria, you can play all of these things. Range is a great tool. Okay, one important thing about loops. That is sometimes neglected is that yes, these are two basic constructions. But you can have loops, within the why loop you can have another wide loop. Within a why loop you can have a for loop. Within four loop you can
have a why loop. Loops can be nested and now this gives you a lot of power. An example of how
you can start with this and check the details. But this is good to learn these things. How would you bring this? Simple but relevant. We said here as long as we have something that can
be integrated on we can use a for loop. While a string of characters is something you can integrate. So in many processing activities you have a whole bunch of text. Social media costs, this is a test. This is a test. And there are many processes, nouns that you can apply to test. One of the simplest things to do is okay, you want to count the number of hours. In the number of consonants, for example. It's important to learn these things so that you understand that things like chat GPT, things that have become so is because since there is a
lot of text processing that is happening. When you go to chat GPT antitype one of these prompts, then chat GPT has the process for those roles, the collection of characters. The characters are consonants, some are vowels, it has to pass that to make sense of what is the
question you are asking chat GPT to do. This is a very little example so that you get the sense of how in Python you can do things like this. What about counting the number of vowels
and consonants in a string him a so, the function. I get count. Give me a name. Input, the number of vowels, zero. Consonants, zero. To start. Look at how nice this is because all you do is say I'm going to have a full. I have a valuable, input of string, whatever it is. So you're going to take that, apply a method to that. The method is saying if it happens to be the character is in this string of characters, this is AEI OU. This is a way to specify in a string, so what Liz's checking is that character is one of those vowels. This is what it is doing. If the characters is one of the is done the count is implemented, as is not a vowel. Character count is implemented. This is a use to count the number of vowels and consonants in
a string. Now you may decide to take care of this because you don't want to count consonants
here. So check for that. You can go over these examples. The same thing we did for why loops
you can use breaks to get out of a full loop. You have a full loop, this is controlling statement for the full loop. This is the code for the full loop and inside you have a break and get out of it. You can put continuous statements. For example in this case if you want to

skip spaces in the character is a space, I'm sorry. Then you use continue and don't count. This is not difficult to understand but it is important to identify what is the loop? What is the control in the loop? How to get out of the loop and then this is saying at the end I want to put another statement meaning if everything is good and I go here it is because everything
is good. At that point you print what you have. So getting out of the loop by this statement allows you to get confirmation of what you were doing is good, is correct. This is usually a new type of mistake. Do not touch the variable. That is controlling the loop. If you have something like what I have in this range, specification and you're doing something. Don't change that for something else. That is going to create a very undesired effects. So don't touch inside the loop the variable you are using to control the loop. Okay. Now, loops can also be integrated over sets. Here is an example related to your many projects. You set up recommended books, using this, this initialization from the books dictionary. You can set up a
set that you would like to check. I like this particular one. He would like to put that in the set and now you would like to check what's in this particular set to do something with that. So for those autos in this specialized set that you have here, you apply some computation and this is part of what you are doing in your many projects. As long as that is the case you go to the next order, repeat, etc. You get something on here then you get out of
it. Why am I pausing here? There's a lot of mistakes here. That is important to be aware of. Here we are using maybe 70 percent. Of what we have covered in the class so far. This is initializing what kind of collection is this? What? What kind of collection of Bruce? If you have a symbol, what kind of collection is that? What? A list. Here you are initializing the list. What about this list? What kind of collection is this? What? Dictionary or? Yeah? What is happening here when you say set something like that equals or not? What is it doing?
It is defining a set to be assigned to the order. This is initialization but it is important to understand because these things are different. Nothing is happening here, really. From a
computational point of view. FO R, part of Python, - - whatever this is in this set. Python set is this and this. This is the way you integrate over this set. Here you have an example of this dictionary. This is part of what you are doing in your many projects. This dictionary starts here, ends here. We have inside the stuff. This piece is a list of books associated with this key. So this is a key value. This is the key. This is the order, in this case and this is the association of a list with that order that is established by this column. So this is only one keyvalue in the dictionary. This is another keyvalue and another keyvalue. So orders, books. So this is how you are defining this particular mapping from order to books. Now, another one here, we are defining a set, a set that is different to this one. In that set I'm specifying the name of an order, the name of another. Separated by, something missing here
that you have fantasies. This is a set of orders. So these two things have the input to the

stuff above. And you are going to do something with that. So now, whatever the order, in your
favorite orders, you are doing a full loop in this set of orders. You are going to do something. That something is you are going to the dictionary, to start the particular order. For this order, this dictionary, this here. You extend it. Recommended books, you put the recommended books. Notice that this process that appears sophisticated is very simple. If you
understand the instructions you are using. There is something called order books dictionary,
there is something called you prefer order set and now what you are doing is for every order in
this set, you are looking at the other dictionary and this is the other dictionary. You are starting the list of books for that order. And extending the recommended books. Remember that
we covered this. Here you are creating something out of these two pieces of input. So this very sophisticated order here, I hope you can generalize this to many different attempts. Which would be recommended. Soccer players, you prefer soccer players and now you see there is
not too much of a difference. There is a difference but the process is the same. So that is a great thing. You also integrate where dictionaries. Here, also related to your project you can, we want to integrate a dictionary. You set up some - - because you were looking up the guys that are rated high and this is just, we have a dictionary that is initializing a variable. To unnumbered. This is initializing a list. This is initializing the dictionary.
And now you are going to use a full loop. That is going to iterate, this is going to integrate books over this dictionary here. And is going to do something with that. Is going to extract the rating for that book. If the rating is bigger than the score than that is going to go into the high rated books. I want you to look at this logic here. Because there are patterns. Through the years people have realized, there are many patterns. That can be applied to different corrections. You understand this and how to translate it to Python. There are many
other processes that can be mapped to this. The dictionaries may be different. What you are
trying to do might be a little bit different but the template of the code is basically the same. That is why it's important to understand these very well to every level of detail. For the purpose of this. So you look to God. Try to make the obstruction. Once more, once more.
You have a dictionary from books to dictionary. Now you take for every book in that collection
or rate of books, you are going to take the rating of that particular book. You are extracting from not dictionary a particular value which is the rating which is there in the data. That is a number. That number you're going to check that with your threshold because you want everything to be 12.9. Excuse me, nine point. He wants to change this. If that is the case

then the particular rated books collection. Every time you identify one of those you append it
to the collection and go back. You have a dictionary for every element in the dictionary. You are extracting some data about that element in the dictionary. You are using that data to do
something with it. After that you are creating by appending another collection which is what
you are looking for. Okay. I would like to close with a couple of reservations about functions. Sometimes when you were trying to develop code there are two approaches. The dome
approach is to start typing the code. That is a dumb approach. What is a clever approach? The clever approach is to design what is called pseudocode. Pseudocode is not covered in a particular language. It is a description for what it is you want to do that you understand. In order to get the description you need to first identify what is my data? What is my data? What is my data? What is my data? The most important thing is, what is my data? If you don't
push yourself to do that you will see you start typing stuff and say I don't have that, I have to go back and eventually what you do is scratch the whole code and start again. So, what is
my data? What is two? What is it? But I want to do. Don't take this lightly. In a particular example we just gave, the data has a whole bunch of dictionaries. There were two
orders, boot ratings, and something we asked you to do. So you have to be able to write and
are very specific manner what it is you want to do. Not saying how, I am saying what. And that, in my experience is one of the pitfalls. Of programming for many people. This data. Second, what? The goal. What is the goal, what is it you want to do? What is number three? Only one person? Yeah? Before that. You understand your data, you can describe your data.
It is a dictionary, a set. My data is a list. Okay. A list of what? Fine, that's the description. What is the goal? I want a mini projects, to, what was it? Better rated books. For your preferred orders. But you write that down. What issix? Yeah. Something more generic then that. This is a human question. What? Oh, okay. Okay, okay. That is a good answer. How? How are you going to achieve this? He said something very important. One of
the most effective tools to specify the how, is by drawing your flowchart. It doesn't have to be pretty. Why? Flowcharts are nothing different than a specification of code. High level. Specs of code. In a flowchart you are going to use it like this. Oh, I need to test this condition. Draw that. What is the condition? Put it here. If that is true then mistakes might be like 20 seconds. No, no, for this I need a lie loop and I prefer for loop. Well, what about this? If you're able to do that, that's about 60 to 70 percent of code development.

Then I can start typing my coat? No. There is another little trick. That you can do which we refer to in computer science as function stubs. Stubs. Which is a layout of what you want
to do. Which is a layout of what you wrote in your flowcharts. In English, as you want. To find the name of a function that you want to implement. That you want to implement. You don't
need to write down the code. You put there after that the term pass. Pass. This is getting closer to code. This can be executed in the interpreter. Even though you have not written particularities over this function. You are telling the interpreter you know what? I'm coming back later to fill in the blanks. Okay? Until you complete this, what happens in real life? You complete this, high level description of the functions. And the logic. In real life you going and tell them so, you are very good. You are very good at Python, right? Yeah? Yeah? Yeah? Okay. Here it is. The specification I wrote in this project. Now you go and translated to Python code. This is real life. High-level code assignments and there are the others that make the translation. Why is that essential? I don't like Python. Oh, I like Java. Are you good at Drago? This specification of this process I want to translate to Java. Code, this is a specification. You translate that to - -. Translation to code, a good, high-level specification is a very simple task. The better you do the spec, the simpler you have the translation. And that is why some people are saying that AI is going to substitute many programmers. Well, that's not quite true. Last level translation, yes, AI programs can do that well. But the high-level specification of the task that we humans, we shine. It is good for you to learn to specify this programming task at very high level so you can say, you
know what? My friend is very good at that and can do that. Is this completely clear? Raise your hand if you got this. To me, this is the main model of coding I would like to inject into you for this class. Yeah? What's the bad news? Bad news is, we are humans. Bad news, in a good sense. So, we are lazy. We obey what is called the law, - - that is the bad news. My advice to you is learn to design sub processes that can be translated in this class into Python
very quickly. Because I can always give you a piece of very complicated Python code that will
make your head spin. So it is better to understand the process behind. Good, written code is
code that has comments that say this piece of code is this and that. And that is the specification you are starting with. I guess it is time to go home. No? Yes? It's time, no? Two more minutes. Okay, what about if we just go home? Yeah. Past and what? Because this
code, when you put pass you don't have the code to put the function. Right? I'm going to put
the code.