Week6 lecture4.2

So now we're going to build some of our own functions.

We sort of saw a little bit of that syntax at the very beginning.

We're going to explore using the def keyword and

some of the choices that we get to make.

And so the essence of this, I'll say this probably 20 times.

The def statement only defines the function.

There's a side effect to it, but it doesn't actually run the code.

It just remembers the code, and then later we will call and invoke the function.

从 ::36 开始播放视频并学习脚本0:36

So here's an example of some dysfunctional code that's not going to work very well.

And so we come in here, we set x = 5.

And you don't have to have the def at the beginning of the program.

We do a print statement, so out comes Hello.

And then we do a def statement and that cruises through here.

And the result is, is you have extended Python to have this thing called

the print\_lyrics function, and it's got two bits of code in here.

Now, this print does a de-indent, so that sort of blocks this off.

That little bit inside the block is the function,

that stuff's been copied up to there, and now it continues on and it prints Yo.

从 :1:11 开始播放视频并学习脚本1:11

And it adds 2 to x and then prints x, and out comes a 7 and now the program is done.

What happened to these lines of code?

And the answer is they never executed, they never executed.

And that's because the def statement doesn't automatically run the code.

You must invoke.

从 :1:32 开始播放视频并学习脚本1:32

We did not invoke the function.

从 :1:36 开始播放视频并学习脚本1:36

Not sure if that's a happy face or a sad face, it's a wry face,

it's a wondering what's going on face.

从 :1:45 开始播放视频并学习脚本1:45

So we didn't invoke it, right?

We didn't invoke it and so it did not work and we didn't do any output.

Now why we did this?

I don't know why we did this.

Question mark, question mark.

Who knows why we did this.

But what we really meant to do probably is something that more looks like

the next bit.

Once you define it we got to call it, and that's the reuse part.

We had the store part working but not the reuse part.

So here we go, this is a little more of a functional bit of code.

Set x to 5, print out hello, out that comes.

Define it, this creates no output but the de-indent causes this block to be

从 :2:22 开始播放视频并学习脚本2:22

part of that, and that's been stored.

Store.

Now we print out Yo, and so out comes Yo.

And then we call print\_lyrics.

And again, print is a function, has parentheses with parameters.

This has parentheses with no parameters.

We didn't put any parameters in, we'll get to that in a second.

Print\_lyrics now says, oh, run this code.

So out comes these two print statements,

then we add 2 to x, print it, and then out comes 7.

So this is the store, this is the reuse, this is the call, this is invoke.

从 :2:57 开始播放视频并学习脚本2:57

Okay?

从 :2:59 开始播放视频并学习脚本2:59

Okay.

从 :3:1 开始播放视频并学习脚本3:01

So everything we've shown so far has no arguments,

except this max did have an argument.

That max is input and

then we will get something back, the w, and then the w will get stuck into big.

So arguments are input, so if we want to define arguments in our

own functions that we're making, we just put them in the parentheses.

Now it's like a variable, we're choosing a variable name.

But this variable sort of doesn't exist.

This lang variable is best thought of as an alias.

从 :3:33 开始播放视频并学习脚本3:33

It's an alias for whatever the first parameter is going to be when that

particular function invocation happens.

So we're going to invoke this function three times,

we're going to put in parentheses three different things.

This is a way to have code that's almost the same but a little bit different.

By pushing in different parameters, then the code does something different.

从 :3:53 开始播放视频并学习脚本3:53

And so this really is just a placeholder to say whatever the first parameter was.

If that first parameter is es, then print this.

If that first parameter is fr, print that.

Otherwise, print that.

So.

从 :4:7 开始播放视频并学习脚本4:07

As our program runs, this accomplishes nothing, doesn't run any code, but

it creates this thing called greet out there.

And we've indicated to Python we expect a parameter.

We expect that when this is invoked or called, we want one parameter.

So then we call greet, and again, Python remembers to come back to here.

And then it fires up to here and starts this line of code.

And en, lang is an alias for en.

So it's like lang is not es,

lang is not fr, so this code runs, out comes Hello, and then we're done.

And it says go back to where you remembered you were supposed to pick up

next, which is here.

从 :4:48 开始播放视频并学习脚本4:48

Okay? So it says pick up next.

Then it goes on to the next statement here, greet.

So it's going to run this function again.

从 :4:57 开始播放视频并学习脚本4:57

Remember where to come back to, which is right here.

从 :5:1 开始播放视频并学习脚本5:01

Jump up to here, and then make es this time be what lang means.

So now, lang is es, so this is true.

We run that code, we come out and

we print out Hola and then we go back to where we remembered before.

Then we continue on, come to this code, okay?

So now we're in this code,

is time to run greet again, so we jump up to here.

And then this time fr is lang, and so

it runs here, that's false, this is true so it prints that out, prints out Bonjour.

Then it comes and it remembers.

Oh, forgot to remember where to come back to, and continues on, okay?

So you see the whole pattern there, a lot of different pretty colors there,

but you get the idea.

从 :5:54 开始播放视频并学习脚本5:54

That it gets run a couple of times and lang is the alias for

whatever the first parameter happens to be.

And that's the best way to think about it.

Whatever the first parameter is,

I will use lang within here, I'll use lang to refer to the first parameter.

You might name this thing, don't do this, but you could name it p1, p1, p1.

So parameter one, no, Python wouldn't care but it's really just the first parameter.

Lang itself is not really

a variable in that there is no piece of memory that lang belongs in.

从 :6:29 开始播放视频并学习脚本6:29

It really sort of is an alias of something else.

Okay?

从 :6:34 开始播放视频并学习脚本6:34

Okay.

从 :6:35 开始播放视频并学习脚本6:35

Now, when we're looking at like the float statement or the input statement,

we see things like input parenthesis blah,

blah, blah, parenthesis equals x, right?

And so we're going to assign something.

The question is. what is this residual value that comes back?

And within the function, we, as the writer of the function,

get to determine what that residual value is by the return statement.

So, basically even though this is really a trivial function, has no parameters,

the return statement does two things.

One, it stops the function, and it jumps to that next line, right?

But two, it also determines the residual value.

从 :7:22 开始播放视频并学习脚本7:22

So by saying return, if we come in here, and so

we do this and then we're in the middle of this print statement.

So it sort of pauses here and then it runs this code and then the return says,

从 :7:36 开始播放视频并学习脚本7:36

oh, this little bit that was the function, send that back right there.

So that means that this is really print Hello Glenn, which prints out Hello Glenn.

从 :7:45 开始播放视频并学习脚本7:45

And then it goes on to the next line, it runs greet again.

从 :7:52 开始播放视频并学习脚本7:52

Runs greet again, comes in here and so now Hello becomes this residual value.

Now it's not very clever but you get the idea.

This residual value, it's like Hello Sally, so it prints out Hello Sally.

从 :8:8 开始播放视频并学习脚本8:08

So here's a little better example of this where we're using

both arguments and a return value.

So again, this just remembers it.

We're going to say greet and en, so en is lang.

greet is going to run [NOISE] and it runs this.

Here now we've returned Hello, so this Hello becomes this.

So Hello is the residual value of the function evaluation, and so

out comes Hello Glenn.

从 :8:38 开始播放视频并学习脚本8:38

So now, we do this, greet es.

So we're going to call this code, and es is going to come in as lang.

And in this case, it's going run this and return Hola.

So Hola is going to then become this replaces, and it says Hola Sally.

从 :8:58 开始播放视频并学习脚本8:58

And so now we're going to do it again, greet fr, greet comes up here,

we're passing in fr as the language, false, true, and then return.

从 :9:9 开始播放视频并学习脚本9:09

Return doesn't come out here, return is done.

That's it, you're done at that point, and you also specified the residual value of

this little bit, and that's Bonjour Michael.

By the way Glenn, Sally, and Michael are real people.

I always just use names of real people that are actual people I know.

从 :9:30 开始播放视频并学习脚本9:30

Okay, so now what we're going to do is we're going to take a look again at that

max function, but we're going to understand how the return works.

This is really more about the return.

So if we take a look at max, so we're going to

evaluate the right-hand side of this so we have something to send into big.

So max, Hello world, says take this parameter and send it in to max.

Here's the max function, right?

And so Hello world comes in to max as some parameter.

We don't know what it's named inside max, we really don't care.

But, it's going to do something,

you know, read some stuff, check to see if it's a string or something.

And then it's going to loop through a bunch of stuff and

figure out what the largest thing is.

And figure this out, read through, pick the w as the largest one.

And then it's going to execute a return statement because it has to pass back

the w to us. Right?

So, to communicate out of the function back to the real world,

here's our real world here, it says return 'w'.

Now it's probably a variable, but you get the idea.

It has to run a return statement, and that's the end of execution.

It doesn't continue, it doesn't even run a single statement after return.

But then it sends back to us to replace in

that original statement the w then becomes the residual value.

And then the w is what goes into big.

Now after a while you won't need to look at all of this syntax in such detail.

I just want you to be able to know every single character and what they mean.

And slow this down into slow motion because after a while you'll just type

this stuff and you won't even think about it.

But you need to be able to slow it down to sort of slow motion if you need to.

从 :11:9 开始播放视频并学习脚本11:09

Okay, so you can have more than one parameter as you might well expect.

This says I want two parameters.

Again, the name of these things inside here hardly matters.

Just that's the first parameter, that's the second parameter.

In a call, we pass in 3 and 5.

So that is easy, the number and order of

the parameters is the same in the function invocation as it is in the function definition.

And we just add these two numbers together, make it a local variable, and

then return it.

So 8 becomes what comes back and gets stuck into x.

So more than one parameter, more than one argument, no problem.

The names of the parameters inside the functions, no big deal.

从 :11:51 开始播放视频并学习脚本11:51

So some functions do not return values.

We call them non-fruitful functions,

and if they return values then we call them fruitful functions.

从 :11:59 开始播放视频并学习脚本11:59

So in summary, this was a pretty quick introduction to functions.

Like I said, I don't really expect you to build a bunch of functions,

but I want you to understand them.

And so when the time comes, and

you're thinking to yourself, oh wow, it's time to build a function.

You'll know how to build them and you'll know how to use them.