**w11-01-Consolidation-2024**

0:01  
OK, so this week is basically all about consolidation.

0:06  
There's no additional concepts or ideas this week, we're just practising.

0:10  
Now, if you've looked at the lab, you will see that we have an example question paper, the sort of thing that might set an exam.

0:21  
OK, So what have we got here?

0:22  
We've actually got something that is, yeah, it's about enough with respect to the time.

0:29  
And there is also a solution, so you can have a look at that.

0:32  
Now there's an extended solution where I've gone a bit beyond, and the reason I've gone beyond what you'd need for the actual exam is so that I can demonstrate some of the concepts and ideas that are in the module but aren't in that particular question paper that's given for practise because it would be too long otherwise.

0:52  
So the practise exam should be roughly the right length and right standard of questions.

0:58  
Now, nobody has replied to me with with respect to things that they would like me to look at in particular.

1:05  
So is there any special requests?

1:08  
Otherwise I have a Plan B idea.

1:10  
Any special requests for ideas, things I should cover?

1:15  
Yeah, yeah.

1:23  
So the lab tomorrow is to go through this practise paper.

1:27  
Yeah.

1:28  
And just use it as well for any other questions.

1:31  
Right.

1:31  
Something else that comes to mind, ask in the lab.

1:34  
So it's it's really just to have a go at the practise paper and ask questions as needed.

1:39  
All right, anybody else?

1:44  
All right, OK, I'm going to go with Plan B then.

1:48  
So the name of today is consolidation.

1:52  
No additional material.

1:53  
We're going to go for the previous exam paper just to have a chat through it.

1:59  
So if you haven't done so already and you're sitting right at the very back of the room, you might want to download the question paper from last year since then you better read it.

2:08  
I will try and zoom in at the front of the class here a bit, but it's a bit tricky if you are at the very back.

2:15  
So let's go to the top of this paper.

2:22  
Yeah.

2:24  
OK.

2:24  
So each of these exam papers are similar.

2:29  
There is instructions for you to follow.

2:32  
They're quite precise, so essentially just follow them.

2:36  
Don't create another file called something else because that'll just confuse me.

2:41  
I will test it programmatically, meaning I will try and run your code expecting that what I've asked for will be in the box.

2:48  
And if that doesn't work, I start to sort of take it apart and try and understand why.

2:53  
OK, so we're going to go for it and see if we can go as far as we can with the previous year's paper in the 40 minutes.

3:02  
So we start off with a request to create a class called File in a sorry, the class is called File and it's in a file catalogue File.

3:12  
So let's I'm going to flick between the two things.

3:17  
So I'm going to create something here called File catalogue.

3:25  
Great, let's go back.

3:27  
What does it tell us to do next?

3:30  
OK, we're going to have to have a construction here.

3:32  
It's going to be a class file, and we're going to have to have name, size, executable content modified.

3:38  
Now, bear in mind you will have an easier time than me because in the exam you will have a printed copy of the exam paper and you will not be trying to flick back and forward on your screen.

3:49  
So thankfully somebody asked me that question, right?

3:53  
So\_in it.

3:54  
Now, if you leave the type stuff on like that, I don't mind.

3:58  
It's quite valid to leave it on.

4:00  
I haven't included it in the course because I didn't want to add additional complexity.

4:04  
OK, so I've got my init now.

4:07  
I'm also told further down the exam paper that I need another class called Directories, so I'm just going to go ahead and create that one now.

4:17  
Great.

4:18  
So I'm going to get rid of that extra complexity now I'm told.

4:23  
Let's have a quick look.

4:26  
I'm told I need a yeah, repr function.

4:28  
I also need an EQ function.

4:30  
Great, let's put those in.

4:32  
So we're going to go ahead and put those in.

4:35  
So I need one here, def\_repr.

4:40  
Sorry, that's the wrong one.

4:44  
That one again, I'm just going to get rid of that.

4:47  
And I was asked also to have an EQ function.

4:50  
Now, if you remember EQ, it's a test to see if something is equal to something else.

4:55  
And yeah, I could just leave this window dressing up.

4:58  
I'm going to delete it for now anyway.

5:01  
OK, so we've got an EQ function, we've got an RIPR function, we've got constructor.

5:05  
Let's go back and have a look at the data member information.

5:09  
We need a name, size, executable, content, and modified.

5:12  
Great, let's do that.

5:14  
So I'm going to have a name and you can tell me if I get anything wrong.

5:18  
Name, size, executable and modified.

5:26  
Wonderful.

5:27  
And I'm going to assign these to data members.

5:30  
So we're going to say name equals name.

5:33  
This is easy.

5:34  
Marks right?

5:35  
Size, size, great.

5:41  
No self dot executable equals executable, great.

5:50  
And modified.

5:52  
Now I'm going to leave this wrong and then we're going to go back and see what the exam paper wants us to do.

6:00  
OK, so let's read the exam paper again quickly.

6:04  
It tells us that the executable should have a default of false, OK?

6:10  
And the content should be default of empty string and the modified should be default empty string.

6:15  
Great, let's do that.

6:16  
So it's going to be false, empty string, empty string.

6:19  
So executable, this should be false, and this should be an empty string.

6:29  
And what have I missed?

6:30  
I missed one, haven't I?

6:35  
Yeah.

6:35  
OK, the content.

6:36  
I missed that one.

6:38  
Content empty string.

6:46  
I think that's now the right order.

6:50  
See if I can put it on the same screen, it might help me.

6:54  
Probably not.

6:55  
I can't do that.

6:57  
So we've got contents.

7:00  
Oh yeah.

7:01  
OK, that's fine.

7:02  
Yeah, I think we're on to the right thing so far.

7:05  
OK, so we're then told I'm in this case, I've given you a bit of Python.

7:10  
Why have I given you Python?

7:11  
I have covered this, but not very much.

7:14  
So I just thought, well, here you go have the Python.

7:18  
And the idea is that our input modified should be a text string, but we're going to store it as a date time.

7:26  
And we're going to convert the input text string as given here into a datetime.

7:32  
So let's do that.

7:34  
So let's go back here.

7:37  
So first of all, at the top, I need to import so from from datetime.

7:46  
Import datetime.

7:49  
So you remember, date time is a type.

7:51  
It's both a time and a date.

7:55  
And OK, so modified is an empty string, right?

7:59  
So what we've been asked to do is we've been asked to Oh yeah, I've haven't missed that content.

8:07  
I've known self content equals content and and then we're going to check on modified, right?

8:16  
So if Len modified.

8:23  
Oh, thanks.

8:23  
That's better.

8:24  
Is 0.

8:26  
So if it's zero, what we want to do is we've been told to use the current time dot equals datetime now.

8:37  
Yeah, right.

8:40  
So it tells us here that we should.

8:45  
Yeah.

8:45  
So it says if the modified string is empty value, we should have the current time.

8:50  
So we've just done what it says.

8:52  
We've gone ahead and done that.

8:54  
Let's put in the bit where if it isn't empty, we are going to convert the text string to a date time.

9:02  
So if it's not that, so then we're going to say else and then we're going to have self dot modified equals datetime from from ISO format.

9:19  
That's the one we've been told to use and we're going to pass in modified, OK.

9:25  
And we're going to get rid of that because we don't need that.

9:29  
OK.

9:30  
So that is the first part of the question.

9:32  
Does anybody have any comments or things they want to point out so far?

9:37  
Any mistakes of mine?

9:45  
Well, each OK, this this is the December exam paper.

9:50  
The reset was again create some classes with some constructors and there was some functions inside.

9:58  
So it's, you know, you're building something that's somewhat similar to this.

10:02  
It won't be exactly the same, but it's of the same sort of standard.

10:06  
All right, off the top of my head, I can't remember if I'm using it or not.

10:13  
I if I've, yeah, I put it into the paper as in here it is because I had mentioned it, but it wasn't, you know, front and centre, you hadn't practised using it.

10:23  
So I thought, right, I'll just put it in.

10:26  
All right, anybody else?

10:30  
All right, so let's have a look at the next bit.

10:33  
So we're then told, yeah, we need an RIPR function.

10:36  
And we remember RIPR function should return a text string that is able to be executed by eval.

10:44  
So let's go and do that.

10:47  
Let's see.

10:48  
Now I personally quite like creating a little variable so I don't have very long text strings.

10:53  
So I'm just going to make a little variable here.

10:55  
I'm going to make it into, it's got to be called file.

10:59  
Remember, it has to be as.

11:00  
So we're going to call the constructor when we're done, and we're going to put this little variable here plus equals.

11:07  
We're going to return little variable at the end S And now I'm going to go ahead and use lazy formatting, which is OK for this module.

11:18  
So let's see.

11:19  
We want, we want to have the name, the size, executable and content.

11:24  
Great.

11:25  
So let's put in here name, size, oops, executable, cutable and content, and finally modified.

11:41  
Great.

11:42  
So now I've typed that down quick.

11:44  
So I'm going to put in the quotes around the text.

11:47  
So if you can find a good way of just sort of making mental notes as you go, like putting the code in and then fixing it, that's probably a good idea.

11:59  
Speed you up a bit, right?

12:02  
So we're then going to put this on a separate line.

12:07  
So S plus equals, probably faster than me with no mouse as well, plus equals executable.

12:19  
Oops, that's me pressing the wrong button, plus equals content.

12:31  
And finally, the last one.

12:34  
Great.

12:36  
So we're going to put the values in now equals and then lazy formatting equals.

12:48  
Now here we just want to give it obviously the data member values.

12:52  
So I'm gonna put in self.

12:54  
Just try and speed myself up a bit.

13:06  
OK, so unless you size in there and these should all be lazy formatted text strings, Format, format, format and my end bracket.

13:32  
Now we're told that we have a name text string, therefore this has to be quoted.

13:37  
Now, you could use single quotes if you want to avoid the back slash double quote.

13:43  
So you could actually do that.

13:44  
That would work.

13:46  
Or you could use double quotes.

13:49  
The size we're told is an integer, so no quotes needed there.

13:53  
This executable thing, that is actually a boolean, so we're not going to quote it.

14:01  
All right?

14:01  
So when it becomes a text string, it will become a valid boolean, so we're not going to quote that.

14:07  
The content, we're told is a text string, so we're going to quote that.

14:10  
And we're also told that the input modified date time should be a text string, so we're going to quote that.

14:19  
Great, and we're done.

14:21  
Anybody got any questions about that one?

14:25  
All right, no shaking your heads.

14:27  
Great.

14:28  
So let's go to the next bit.

14:30  
So an EQ function is going to return true in this case if all the data members are equal of the two objects.

14:39  
Great.

14:39  
So let's go ahead and do that.

14:42  
What do I say?

14:45  
Return.

14:46  
Now there are many ways of doing this.

14:48  
You could maybe have a variable and then you could just check if the each part is true or not.

14:53  
I'm just going to do this in one line.

14:55  
If now here's a bracket.

14:57  
So let's see if self dot name equals equals.

15:05  
I prefer to call this other rather than value because it's the other object.

15:09  
I know the editor thinks that I should have value and let's put this round here will then allow me to continue the line.

15:20  
Let's \*\*\*\* all these together.

15:25  
Bit of copy pasting.

15:27  
If you're copy pasting, be careful in case you copy paste the wrong thing or something or forget to edit it.

15:36  
So modified that should be there content and this should be executable.

15:55  
All right now we've got size and name, size and name.

16:03  
So I've only got 5.

16:04  
I think I've forgotten any 12345.

16:08  
No, got them all.

16:09  
Great.

16:11  
So and then we're done.

16:16  
OK, so I'm being careful.

16:18  
I don't want a very long line.

16:19  
So you remember with PEP 8 standard you're like, I think it's 78 characters is the maximum length.

16:26  
Hence I'm keeping my line short.

16:29  
Anybody got any questions about that?

16:34  
No, OK, I'll keep going.

16:36  
So right.

16:37  
What are we told to do next?

16:40  
The next thing we're told to do.

16:41  
So we've done that, RIPR, we've done that.

16:44  
We've done the EQ function.

16:46  
So we're told we want another class, and it's called Directory and it has two input parameters, name, string and files, which should default to an empty list.

16:58  
So let's do that.

17:00  
So we've got name, so name and the files, and this should default to an empty list.

17:14  
Great.

17:15  
What else?

17:18  
OK, so assign them to and it tells us where to create a shallow copy.

17:23  
So you remember a shallow copy.

17:24  
You've got two ways of doing this.

17:26  
You can either have the copy module and say dot copy, or because it's a list, you can just say the list dot copy.

17:33  
That's a shallow copy.

17:34  
I'm going to use the second one of the two in an exam.

17:37  
If you're doing this, I don't mind.

17:38  
You could use the first one or the second, doesn't actually matter.

17:41  
So let's say nameself dot name, name, self dot files, files dot copy.

17:54  
OK, that's a shallow copy.

17:57  
What are we told to do next?

17:58  
Let's have a quick look.

18:02  
Next off, we need another repr function.

18:05  
OK, this one's easy.

18:07  
Notice we've got a list.

18:08  
We don't need to quote the list, but we have a text string, so we're going to quote that one.

18:12  
So let's do that quickly.

18:14  
So def\_\_repr.

18:18  
Thank you very much.

18:19  
I'm going to get rid of that for now.

18:24  
And I'm going to say S my S in here directory and then have S plus that edited trying to be helpful.

18:46  
Oops, wrong button on my keyboard, right?

18:49  
So let's put in the others.

18:51  
So we're going to say name equals name, self dot name, obviously self dot name, and put the other one in F files, self dot files.

19:15  
I already told you that the name is a text string, so we're going to quote that.

19:19  
The list obviously is a list, so we're not going to quote the list itself.

19:24  
And we're done.

19:25  
We're just going to return it.

19:30  
All right, easy.

19:32  
Let's go to the next part.

19:34  
Yeah, yes, Thank you very much.

19:39  
Good spotting.

19:39  
I'm glad you're awake.

19:42  
And this one should be as well.

19:45  
Yeah.

19:45  
So instead of having a long line, I'm just using plus equals.

19:50  
OK, let's go to the next part.

19:55  
Next, we're told we need a member function called total size for the Directory class, and the function should return the total size of all files in the directory object.

20:06  
Now, very simple, what are we going to do?

20:08  
We're going to loop over all the files that are in that list, and we're going to add up the size and return it as a total value.

20:16  
Let's do it.

20:18  
This is similar to when we're dealing with the solar panels or when we're dealing with purchases for a customer, same idea.

20:27  
So let's put in here def total size and this is also a member function.

20:38  
And then we are going to loop over for file in self dot files, right?

20:48  
And what is the function called?

20:49  
If we've forgotten, we can always go up here.

20:51  
Oh, look, it's the data members called size.

20:54  
Great.

20:55  
So let's have a variable up here calling.

20:58  
Let's call it total.

21:00  
Set it equal to 0.

21:02  
And then we'll just say file dot size.

21:06  
And we'll say total plus equals that.

21:11  
And then we are going to return total.

21:16  
Great.

21:18  
OK, easy.

21:21  
Let's have a go at the next bit.

21:24  
Now what I haven't done as as I've been doing this, I've been quite quick, that total size should really have a multi line comment, right.

21:33  
So we'll come back to this where we just say what it does.

21:38  
Actually, maybe I should just do it now.

21:40  
You don't really need to do any commenting for the init or the repr because that's the standard.

21:46  
We might want to put a comment in here for the class to say what the class represents.

21:52  
I mean, when you're coding in the exam, you can just plough through it all and then remember, comment the code when you're done, right?

22:02  
Let's go and have a look.

22:04  
So the next thing we need is we need a 2 CSV.

22:08  
And the idea here is we're going to take the files.

22:12  
So it says the function should save each file object as a separate row in output CSV file.

22:19  
It should contain these things and OK, great.

22:25  
And that's about it really.

22:27  
So let's do it.

22:31  
We want 2 CSV, and we're going to have those things as outputs, 12345 of them.

22:38  
Now, if you've been paying attention here, I need to give a text file name for this output file.

22:45  
I haven't actually said that, but I do need to do this, so let's just do that.

22:50  
So we're going to have another function here, and this one's going to be called def.

22:57  
Let's just triple check I'm doing the right thing.

22:59  
Member function.

23:00  
Yeah, great.

23:02  
Dev 2 CSV self.

23:08  
All right, we're going to pass in a file name, filename.

23:14  
Now we could use open and close, or we could use with if you prefer to use with, I don't really mind.

23:21  
Let's use open and close for the sake of argument.

23:23  
So output file and this is going to be given Yeah, that's fine.

23:38  
This is going to be given the file name file name, which of course needs to be a write written file.

23:50  
Now you really should give an encoding.

23:53  
So let's do that.

23:55  
I'm going to give this UTF 8 which is the standard one.

24:00  
And the other thing is if I'm writing CSV, I want to disable the
So
And give it that.

24:08  
And because I'm not using with, I'm going to remember to close my file at the end.

24:14  
Great.

24:14  
So you could use a normal dict writer or you could use a CSV writer, right?

24:22  
So I'm going to use the dict writer from the CSV module.

24:25  
So let's go ahead and do that.

24:26  
I'm going to add CSV at the top up here.

24:29  
Scroll to the top of the file.

24:32  
Here we go.

24:33  
So import CSV.

24:37  
PEP standard is I have to have two spaces between these imports and this, if you run PY code style, it'll tell you that, but I'm just telling you anyway, remind you about the style.

24:49  
OK, so we're now going to create something called Dictwriter.

24:53  
I'm going to call it Dictwriter and it's going to be CSV Dictwriter.

25:01  
Now to instantiate Dictwriter, we need to give it an output file name or file connection rather, and we need to give it some field names.

25:13  
So let's go ahead and give it the I put here.

25:20  
Oops.

25:22  
So I'm going to call this thing field names, which is what it calls it, just so I don't confuse myself.

25:29  
Now, what are we told?

25:31  
We're told that we need a field name that corresponds to each of these.

25:35  
All right, So if I'm paranoid, I would probably copy and paste it.

25:41  
Yeah, let's do that in case I'm going to make a mess.

25:44  
Copy paste is your friend sometimes.

25:49  
So we're going to put these in here.

25:52  
I'm going to put here field names and paste that in there.

26:01  
And then I'm just going to make these text strings because that's what they're going to be and get rid of that.

26:12  
So the field names are going to become the names at the top of the CSV output file.

26:21  
So do that.

26:23  
That's a list.

26:27  
Watch my time here, right?

26:44  
And the last one.

26:47  
So there we go, we've got the field names.

26:50  
So the next thing we're going to do is we're going to tell the CSP writer to write the where is it going?

27:01  
Right header.

27:03  
So right header implies that it writes those field names as text at the top of the CSV file.

27:11  
Now you could specify, I haven't done it yet.

27:15  
You could specify you're quoting or well, potentially a quote character, although that's pretty standard.

27:23  
So I'm going to specify quoting and I'm going to say CSV, where is it?

27:31  
Quote?

27:31  
I'm going to go quote non numeric.

27:37  
That's a fairly OK one.

27:38  
Now I'm going to use the same quoting when I write the file as when I read the file.

27:45  
Otherwise I might get myself in a muddle, right?

27:48  
Quote non numeric.

27:50  
So write the header.

27:52  
OK, so now we're going to have to loop over each of those files and write them out.

27:57  
OK, let's do that.

27:58  
So for file in self dot files and what are we going to do next?

28:06  
Well, we're going to say our row.

28:10  
So let's say row dictionary is going to be equal to.

28:18  
Now I'm going to cheat, well, cheat a little bit.

28:20  
So I'm just going to take that copy that paste.

28:26  
Now, if you're clever and you went beyond this module, you could think to yourself, ah, but will we can use attributes?

28:32  
Yes, but that's not in the module content.

28:34  
So which is why I'm doing it this way.

28:37  
OK, so then we're going to have here set file dot name, and this is going to be file dot size.

28:49  
And yes, there's a comma there.

28:51  
And this is going to be executable executable and the file in front and this one is going to be file dot content.

29:07  
Yeah, I know I've missed one more of these and here I really should have file dot modified.

29:17  
Yes.

29:19  
And I'm going to call where is it called?

29:23  
ISO.

29:26  
Forget my function here, just got to look up where the function is.

29:37  
Date time, it's called to no ISO, Awesome format.

29:47  
Is that right?

29:50  
Yeah, that's right.

29:52  
Fine, so let's do that.

29:57  
So I'm gonna come back here.

29:58  
So I could put in here that that would be absolutely safe.

30:06  
So remember, date time is a variable.

30:09  
If you actually try and cast it to a string, it will generate something that is almost the ISO formatted.

30:15  
And yeah, it will work.

30:17  
Here I'm specifying ISO format.

30:20  
What happens is it will put in the date, the time, and it actually puts a little something in to say which time zone are we using?

30:29  
Fine.

30:30  
So I'm going to do the same down here.

30:34  
I missed one there and that is like that.

30:38  
Great.

30:39  
So I've got my row dictionary.

30:41  
So then I'm going to say dict writer dot right, right, right.

30:49  
Row.

30:49  
Thanks very much.

30:50  
Row dict.

30:53  
OK, and that should be fine.

30:56  
Anybody got any comments or questions about that or is it reasonable?

31:02  
No.

31:02  
All right.

31:03  
So we won't have a, a matching function where this one's called from CSV.

31:11  
And yes, you can guess what it's got to do.

31:13  
It's got to read the CSV data back in and create new files.

31:19  
Now, if we're paranoid, which is as a programmer, always be very careful, we want to potentially clear.

31:28  
It's a choice.

31:29  
We could clear the files list or we could just append to it.

31:34  
So I'm just going to clear it.

31:38  
Why not?

31:39  
Now, if you chose one or the other, I don't really mind that.

31:44  
You know, it's probably a good idea to clear it because then what we've read is what we have rather than we're appending to it.

31:51  
But if you didn't clear it, then yeah, that would just be appending.

31:56  
OK, let's go ahead and do that.

31:57  
So we're going to have another one, another member function here, and it's going to be called ROM CSV down here somewhere from CSV.

32:12  
And yes, it's going to have to have a file name.

32:15  
File name.

32:16  
And yes, is meant to be a function, so we'll put that in there.

32:21  
And now we're just going to use the Dictreader.

32:24  
So what do we need to do?

32:26  
Let's copy what we did before our friend copy paste.

32:30  
And this time we're going to call this an input file for clarity's sake, input file.

32:38  
And then we're going to have input file dot close.

32:45  
And yes, it's going to be a read file rather than a write file.

32:51  
And then we're going to create a dict reader equals CSV dict reader.

33:00  
Wonderful.

33:01  
And what are we going to give this thing?

33:03  
We have to give it an input file.

33:05  
Wonderful.

33:07  
OK, so I'm going to clear the files list at the top.

33:12  
So self dot files dot clear.

33:17  
So remove the files.

33:19  
And then in here we're going to say for dict row in Dictreader.

33:29  
And then we're going to go ahead and say file, file.

33:36  
And in here we can put everything we need.

33:41  
So let's do that.

33:42  
So I'm going to say it's the opposite of what we're doing here.

33:46  
Now, the one tricky thing here is that we are going to want to be careful with the text that is the executable.

34:00  
We can't just cast that.

34:02  
We're going to have to check if that word that's been saved, it's text and the output file is true or false.

34:09  
So that we have to be careful with the other parts.

34:12  
We can convert them more easily.

34:15  
So let's do this.

34:17  
So I'm going to say executable.

34:21  
I'm going to deal with this first executable.

34:25  
And what have I got?

34:27  
So I want to say and in here is executable.

34:37  
So let's set this to false.

34:40  
Why not?

34:41  
And then we're going to say if dict row lower equals equals true.

34:57  
Now, you could do this in different ways.

34:58  
It's up to you.

35:01  
But I'm checking to see if that text string is actually true or false in this case.

35:08  
Yeah, it's going to be true if it's true.

35:10  
Otherwise it's false.

35:11  
Now, if you're writing this up, you could put it in your little user guide.

35:14  
If you give any other thing other than true, it's going to be false.

35:19  
All right, so that's what I'm decided to do.

35:23  
Great.

35:23  
Now let's put everything else in here.

35:26  
So what have we got to remember?

35:28  
We need a name, so let's put that in here.

35:32  
So I'm going to put here, sorry, that should be executable, not Dicturo, right?

35:44  
This should be the name.

35:49  
And let's see what else we've got here.

35:51  
We've got size.

35:53  
Fine, so that actually should be an integer size.

35:55  
So I'm just going to cast that one size and then we've got executable which we've already got, We've already dealt with this executable and what else have we got here Modified right now modified is a date to time.

36:20  
So what we need to do here is go on my way is we're going to put this back.

36:28  
So we've got a text string.

36:29  
We're going to read it back into a date time.

36:32  
How do we do that?

36:33  
Well, we can go back and copy our friend up here that we previously used if we couldn't remember.

36:40  
So up here, this is what we want to use like that.

36:43  
So it's taking a text string that's isoformatted and converting it into a date time.

36:49  
So let's go ahead and do that down here.

36:51  
And that's the other one.

36:56  
Let's put this in here.

37:00  
And we are going to put this inside rather than the other way around.

37:17  
All right, so this is now looking OK ish.

37:20  
Oh yeah, I've missed my content act.

37:23  
I need the content.

37:26  
And that is going to be this thing up here content here.

37:38  
OK, now I have a file object, so I'm going to append the file object to the list.

37:43  
So I'm going to say self dot files append file.

37:49  
All right.

37:51  
Anybody got any comments or questions or does that seem reasonable?

37:56  
Yeah.

38:02  
So that is being dealt with up here so that the the point is that when we're writing the what is a boolean will become text in the output file.

38:15  
So in Python it will be true with a capital T or false with a capital F.

38:21  
Now when we read it back in, we need to check if that text is actually true or false, and then that will give us a boolean.

38:34  
So here I've got a boolean.

38:37  
I've set it to true if the text that I've read is true.

38:44  
Yeah, yeah.

38:50  
So row 90, I'm just using that variable that you can see at row 86.

38:57  
You could put it on one long line if you really want to.

39:01  
You could have all of this.

39:03  
I mean, another way of doing it would be just take that.

39:07  
I mean, that itself will return a boolean, right?

39:09  
True or false.

39:10  
So what we could do is we could put that here.

39:15  
That would also work.

39:17  
Yeah, it's, it's up to you.

39:19  
I just decided I'll put it at the top.

39:22  
I mean, with programming you can stuff as much stuff as you as you can into like a constructor call like this.

39:28  
But at certain point it becomes messy and you're like, well, should have put it above, should have put it inside, doesn't really matter.

39:34  
I mean, both will work.

39:36  
Yeah, try not to be too messy.

39:37  
If it starts to become busy in the constructor call like that or in function call, you put it outside.

39:43  
Anybody else?

39:48  
Yeah, yeah.

39:52  
If if we did this, if we did that, we don't need lines 8485 and 86 that are redundant.

40:02  
So we could just get rid of these if we wanted to.

40:07  
Yeah, because the other that will return true, false that condition.

40:12  
OK.

40:14  
And we can to keep that if you like.

40:18  
Great.

40:21  
OK, so what's next in our shopping list?

40:24  
Let's see if we can go any further before we run out of time.

40:28  
So we've got a fine name.

40:29  
Now what I'm going to do is I'm going to pause going through these other questions.

40:33  
I'm going to go ahead and build the unit tests because that would probably be a good idea.

40:39  
Any any strong votes?

40:41  
Should I go for the unit tests or carry on with the other questions?

40:43  
What would you like?

40:47  
Yeah.

40:48  
All right, let's do the unit test.

40:50  
Fine.

40:51  
So we'll have a go at the unit test.

40:53  
Bear in mind you would have 3 hours to do this paper and I've taken, I don't know, 40 minutes.

40:58  
So test file catalogue.

41:08  
So we're gonna now start off with the way I'm writing unit tests.

41:12  
I know the book uses itest if you wanted to do that and it works fine.

41:17  
I'm just going to use the unit test framework because it's a bit like the Java 1.

41:22  
So import unit test.

41:25  
Great and now I am going to follow my instructions.

41:29  
What am I told to do right?

41:32  
I'm going to Scroll down to the unit test bit.

41:34  
Here we go.

41:35  
I'm told that I need to have a test for the our PR functions and I need to test the two and CSV and from CSV and I need to also test these other two that I haven't implemented yet.

41:47  
Fine, so let's go ahead and write perhaps 3 tests.

41:53  
Let's see what we can do.

41:54  
So now there are 15 marks here and normally when I'm doing this, I'm thinking that 15 marks is divided by 5, right?

42:02  
So if I managed to do each unit test is worth 3 marks, right?

42:09  
Let's go for it.

42:10  
So we're going to have an repr function.

42:12  
So I'm going to import.

42:14  
So I'm going to say from file, catalogue, import file and dictionary.

42:23  
Sorry, directory rather not dictionary.

42:25  
So I've got my file.

42:28  
Catalogue helps if you spell right.

42:31  
What have I done?

42:32  
Catalogue.

42:33  
Yeah, it is.

42:34  
I made a mess.

42:38  
Doesn't like my spelling.

42:39  
Of course, it's probably set to American or something horrible.

42:42  
OK, so we're going to go ahead and create a test for the file first.

42:47  
So test file.

42:49  
And this thing is going to inherit from unit test test case oops, not that one test case.

42:57  
And inside here we are going to have a test and we're going to call this test repr.

43:03  
And then in here, it's going to be self, oops, self.

43:09  
And what are we going to do?

43:09  
We're going to say file equals file.

43:12  
And what does it want?

43:13  
It wants a name, OK, my dot TXT and a size.

43:20  
So let's say this is worth 10 bytes or something.

43:24  
Executable is going to be true and content is going to be high there and modified is going to be leaving that blank.

43:38  
Let's leave it blank.

43:38  
Why not?

43:39  
No, because then we can't unit test it.

43:41  
The problem with leaving it blank is we're going to have to check the value.

43:45  
So we want to actually set that modified string ourselves.

43:49  
So let's say a time now time now equals.

43:55  
I'm going to put in date time, import date time.

44:04  
Great.

44:05  
And then we're going to get actually from date time, from date time, import date time.

44:12  
And then here we're going to say date time.

44:15  
Bless you.

44:16  
Now, so I've I've got a string.

44:21  
Now you could set the date time with a text string.

44:25  
That's another way of doing it.

44:26  
But you must have a value that you can verify is the value.

44:31  
You can't just let it set the the the now inside because it might not be the one you thought it was.

44:38  
So we're going to set the modified to time now.

44:41  
Great.

44:43  
OK, so we are going to test out the RIPR function.

44:47  
Now what I suggested to you before is use eval to do this.

44:49  
So that's what we're going to do.

44:51  
So I'm going to say new file is equal to eval STR file.

45:01  
Now if you've been paying attention you will remember that I have an EQ function inside file, therefore I could just use on the file itself.

45:12  
So self assert like that file new file.

45:26  
Great.

45:27  
Now if you're writing code, it's good to test as you go.

45:31  
So I'm going to go ahead and test it.

45:32  
So I'm going to say if name equals equals main.

45:41  
And here I'm going to put main, sorry, unit test main.

45:48  
So now we're going to see does this actually work or not?

45:51  
Why have we made an error?

45:54  
Great.

45:55  
So yes, that doesn't work.

45:58  
Let's go and have a look.

46:00  
See if we get to the unit test.

46:01  
We suddenly start realising, oh we made an error.

46:05  
So it says object of date time has no length, right?

46:09  
Let's go and have a look at it.

46:21  
Yes, you've probably noticed the error is not in the code, the error is in the unit testing and the person doing it.

46:29  
So we need to convert this to an ISO formatted text string.

46:37  
All right, because we're meant to be putting it in a text string, not in a date time.

46:41  
So now will it work?

46:47  
Yes, it does.

46:48  
Tremendous.

46:49  
One down.

46:51  
OK, so my my unit test now works.

46:54  
Job done.

46:55  
Let's have a go at the next one.

46:57  
So def test.

47:02  
Well, yeah, we've got 3 minutes left, so bless you.

47:09  
So let's do the next one and I'll stop.

47:13  
So in this case, what I'm going to do is I am going to have a directory now test dealer, and this is going to be equal to directory.

47:30  
All right, So it's got a name.

47:32  
So let's just say user and it's going to have some files.

47:37  
Now, when you're writing unit tests, put and it's got a list.

47:40  
You might want to put in more than one file, otherwise you're not quite sure is it going to loop over or not.

47:46  
So let's do that.

47:48  
So here I'm going to say files, and it is a list inside.

47:55  
And then what are we going to say?

47:57  
Let's take our friendly file up here.

48:02  
Why don't we do that?

48:03  
We use the time now again.

48:06  
Oops.

48:08  
And we'll \*\*\*\* this in here again.

48:12  
Copy.

48:14  
No problem.

48:14  
Use the same one.

48:16  
I'm gonna have a different file, and this one's gonna call my two.

48:22  
I'll put an exclamation mark or something.

48:24  
I'm gonna make this false, and let's make this 12.

48:31  
Great.

48:34  
So then we can do the same as we did before.

48:38  
Now in this case, will it work?

48:41  
No, it won't.

48:42  
We don't have an EQ function, so we can't play this game.

48:46  
We're going to have to check each data member in turn, so let's do that.

48:51  
So I'm going to say new dir.

49:00  
Oops.

49:02  
And this is going to be test DIR.

49:07  
Let's see if this works to start with for a laugh.

49:10  
Yeah yeah, it doesn't actually got an unexpected argument file.

49:15  
Yes.

49:15  
OK, so I think that's me making a typo.

49:18  
This should be files.

49:22  
Let's run it now.

49:28  
Can I do 2 functions that have the same name?

49:32  
You mean test functions like this?

49:35  
Well, I can in this case because.

49:38  
Well, when I Yeah, not like that I can't.

49:41  
But what I was intending to do, and I'm in a hurry, which is probably why I'm making a a dog's dinner of it, is to have a separate class for it.

49:52  
Now it's now it's absolutely fine.

49:55  
Yeah.

49:57  
Because they're in two different classes in, you can't have the same function name within one class, right?

50:03  
You know the languages you can.

50:05  
And as long as I have different lists of input parameters, you will know that they're separate.

50:12  
So here, So you can see I've made another error.

50:16  
Great.

50:17  
Yes, yes, I can see what I've done wrong there.

50:20  
So let's see.

50:21  
It shouldn't be a tupper.

50:22  
Will you get?

50:24  
It's fine.

50:27  
Debugging is really good.

50:28  
Because then you're sure it works.

50:30  
It says perhaps, ah, perhaps I've forgotten a comma.

50:33  
I probably have forgotten a comma.

50:34  
Yes.

50:37  
Let's go back and have a look at that code.

50:43  
Yes, I have forgotten a comma.

50:44  
That's what was wrong.

50:45  
I put it in, was the wrong place.

50:51  
Yeah, good.

50:52  
Now it works fine now.

50:54  
So you see just by running a bit of unit testing a couple of minutes function's fine and then we're going to go ahead and write some more unit tests and that's basically it.

51:04  
Now we're we're out of time for this quick run through.

51:09  
Has anybody got any comments or questions about what I've done so far?

51:12  
Yes, yes, yes.

51:25  
So in the question, yeah, good, good question.

51:29  
So for each of these functions, or each of these constructors even, I've told you it should have, in this case, 2 input parameters.

51:43  
And I told you this one should have a default.

51:45  
I haven't said anything here.

51:47  
There's no default, so you've got to give it all right?

51:50  
So if I'm telling you it's got to be a default of this, put it as that, as the default.

51:54  
If I don't give you default, there isn't one.

51:56  
OK, all right, so hopefully that's useful.

52:02  
I'll save the code and put it somewhere safe and hopefully see you in the lab.