**w01-01-ProgrammingAndVariables-2024**

0:02  
I'm gonna hide these tools, hide the panel, chat briefly about the Myplace page.

0:08  
Now the Myplace page is the page where all the resources for this module are hosted and I'm releasing week by week.

0:18  
The reason for that is because sometimes people are super keen and they go faster and further and other people go more slowly.

0:26  
So to try to keep us all in step and releasing a week of time, you'll find when you come to the Myplace page that material is released on a Thursday.

0:35  
So if you want to be ready for the Monday, have a look from Thursday and that will hopefully help.

0:41  
The labs are all on a Tuesday, so if you have a lecture on Monday, you've got the night to think about it before the lab.

0:49  
The purpose of the lab is to practise and what normally happens is the grade that you have for this module is almost the linear function of your lab attendants, right?

1:01  
So if you haven't come to the lab, you haven't tried stuff out when it comes to the assessment, historically students don't do too well.

1:08  
So please come to the lab, try stuff out.

1:11  
It's much better that you ask stupid questions and we can help you rather than you just hide.

1:16  
And then it comes to the assessment.

1:18  
It's not good.

1:19  
OK, what is on this Myplace page?

1:22  
You've got a couple of forums here.

1:25  
Now these two forums are for your questions.

1:28  
So if you want to ask any kind of general question, just post it on the general discussion forum.

1:34  
If I answer it, great, the student might want to answer it.

1:37  
It's fine.

1:38  
We're all here, different backgrounds, so we can help each other.

1:42  
If you've got a question, particularly about an assessment, please put it on the assessment forum.

1:47  
That just separates the discussion.

1:49  
This one is hidden at the moment because nobody's doing a reset at the moment, which is great.

1:54  
In terms of the assessments, the assessments are too.

1:59  
Hopefully, again, we don't need the reset.

2:02  
We have a class test which is already scheduled, that's in November.

2:07  
And the idea there is we are just testing to see.

2:11  
Have you understood the concepts that I mentioned?

2:14  
So there's no software development in that class test, but it's based on you understanding the concepts that are discussed in the course material.

2:21  
The actual programming exam that is in the assessment week, which is either the 1st or the 2nd assessment week, that's just before Christmas time.

2:32  
So I yeah.

2:33  
Question yes, it's a theoretical test.

2:42  
Yeah, I'll come back to it.

2:43  
So it's a theoretical test where you've hopefully thought about what I'm talking about.

2:47  
You know, you've memorised that some of and then I'm going to ask you a question piece of code.

2:52  
What's wrong here?

2:53  
So there's no actual programming to do, but I would perhaps give you a piece of code and say why is this doesn't why doesn't it work anyway?

3:01  
So there's information here about the class test, about the programming exam.

3:05  
In the programming exam it's like 3 hours.

3:08  
So you can sit there and in, you know, try to relax a bit.

3:11  
You've got all the resources so you can download stuff.

3:14  
But in fact all of the stuff you do need is already pre downloaded for you on the PC.

3:18  
But you can download other things if you want.

3:23  
Yeah, right.

3:24  
What else have we got here?

3:25  
We've got a reading list now.

3:27  
On the Myplace page, you'll see a lot of reading references.

3:31  
They are there for support only.

3:33  
You don't need to learn what's in the reading references.

3:37  
Occasionally when somebody's teaching something, they teach it from a particular perspective, and if you've got the same information from a different perspective, it can be very helpful because then it unblocks a kind of learning problem.

3:50  
So the reading here is to essentially give you a different viewpoint on the same thing.

3:56  
Now if we look down here to the first week, so let's expand out this first week, we can see that we have further down reading references.

4:06  
Now as much as possible, I've tried to give you references which are in line with what's in the module, OK, But I'm assessing essentially what is in the lecture and in the lab.

4:17  
And anything you read is just supporting what's in the lecture or the lab.

4:21  
So you don't have to go ahead and read all this stuff if you don't wish to.

4:24  
Then it's online.

4:26  
So you don't even have to buy the book.

4:28  
The book I'm referencing is actually very good, but you don't have to buy it.

4:31  
It's available to you online through the library.

4:34  
There are also some reference pages which you can look up the same sort of information from the reference pages.

4:40  
Now the one thing I do need to say before I go any further is at the top of this myplace page, there is a few other bits of information.

4:50  
Sorry, I'll go up a bit.

4:54  
Well, let me right.

4:56  
So first of all, these documents here are written to support the course.

5:01  
Now, as we go through the lectures will tell you which part of these documents we should be looking at.

5:08  
Now, what's in this Python programming examples actually goes slightly beyond the module content.

5:15  
So just keep in mind what's going on in the lecture in the lab.

5:18  
That should guide me.

5:21  
We will cover virtually all of what's in this Python programme examples by the end of the 11 weeks.

5:27  
Well, by the end of the 10 weeks, actually, the 11th week is just for revision.

5:31  
This one here is an introduction.

5:33  
It's a if you've never done any programming before, you might want to start with this introduction to Python.

5:38  
If you've already done some programming and you're happy, then yeah, jump to this document.

5:44  
We will start to converge as the weeks go on in that you will all be learning something early on.

5:51  
It might be a bit easy now for some people.

5:55  
There's a glossary here.

5:57  
If I use any other word that's a technical word by accident and it's not in the glossary, please tell me and I will happily define it and put it in the glossary.

6:06  
The purpose of the glossary is with software development and computer programming, we do occasionally use technical words, not because we're trying to be obfuscated, but because we're trying to convey an accurate meaning.

6:18  
And I put as many as I can in that glossary.

6:21  
So if I use a word and you think, oh, what's he on about?

6:23  
Go and have a look in the glossary.

6:24  
It should be in there.

6:26  
OK Lastly, before I kick off, if you haven't done so already, please complete the Python experience poll.

6:34  
This is very useful for me because it helps me put you in a group of students who have got a similar sort of experience.

6:41  
If early on when I ran this module, my colleagues said just mix them all up.

6:47  
And so I thought oh OK, so I mixed them all up and then I ended up with a student or two who were extremely good at programming and ones that never done it before.

6:55  
And what happened was the ones with no experience were scared that they should have had experience, which you might think is daft, but it happened.

7:02  
So with the groups for this module, there are closer inexperience.

7:09  
So you probably will end up with either somebody who's experienced as you if you've got more experienced or not as experienced with respect to the rest of the class because you haven't got as much experience.

7:20  
OK, Please complete the attendance for each of the sessions.

7:24  
The attendance is for you to do so you click on the button.

7:27  
It works if you're on the Wi-Fi, if you're in the room, if you're at home, it doesn't work.

7:33  
If you're somewhere else on campus, it may well work.

7:36  
But don't do that.

7:37  
Be honest with the attendance.

7:39  
If you're in the room, say, yeah, I'm in the room.

7:42  
If you forget and you tell me afterwards I forgot to do the attendance, will I can add it, add you after the fact.

7:48  
Well, I'd much rather you just did it yourself.

7:51  
OK, so that is the general introduction.

7:54  
Does anybody have any general questions or comments before we leap into the lecture material?

8:00  
Yes, one from the back, right?

8:07  
Good question.

8:08  
So if we go back to this document and where it says files underneath, so the folder, the zip folder is all the Python files that are discussed in the document.

8:21  
So the idea of what I've given you is here's an explanation of it.

8:25  
There's the file, you play with it, you edit it.

8:28  
It's about giving you things that already work and then you change them, you know, bodge them together.

8:36  
So I'm giving you the files to try to make it faster and easier.

8:39  
Otherwise you'd have to like type in the code yourself.

8:42  
So I'm just taking that step away from you.

8:44  
But all that's in the zip here is discussed in the actual PDF.

8:49  
OK anybody else?

8:54  
All right, fine.

8:55  
We will start with the lecture material.

8:58  
Now before we go any further, this, this is the results of the and it's anonymous.

9:06  
Don't worry, I did say it would be anonymized.

9:09  
These are the results of the Python skills, the the pole.

9:14  
So what can we see?

9:15  
This cohort changes year to year.

9:20  
Sometimes we have more people with no experience.

9:22  
Sometimes it's, you know, a lumpy shape.

9:25  
This year you can see that most people have either got no experience or basic experience.

9:30  
There's a few people are happy writing functions and and one person you can find them who's happy writing classes in Python.

9:39  
So that means as we approach the exercises, the people down here might see something completely new.

9:44  
All right, So we have to sort of encourage each other.

9:47  
Great.

9:48  
That's that.

9:50  
Let's go back to my We'll come back.

9:53  
Good.

9:53  
It will.

9:58  
This is the lecture for this today.

10:06  
All right, so today's lecture is a sort of introduction to computer programming and it's a little bit of an introduction to the Python programming language and also a little bit of introduction to software engineering.

10:17  
Now we'll actually carry on with software engineering next semester, but I'm going to sow the seeds of it in this lecture.

10:24  
So welcome to the module.

10:26  
We assume no prior programming experience.

10:29  
Obviously some people do have experience, some more than others.

10:32  
We're going to start from the basics.

10:34  
So if it's known to you, you, yeah, you're going to end up maybe reading around, spending that time early on, looking ahead a bit.

10:43  
We introduce ideas quickly.

10:45  
So although I'm trying to give you an introduction from nothing, we have to keep trying to try it out, learn it as we go.

10:54  
Otherwise we become further behind quickly.

10:58  
Yes, we've got a range of experience and abilities and hopefully that's useful.

11:03  
You can go beyond the class material.

11:05  
Remember, I'm assessing what's in the lectures and what's in the lab.

11:09  
So if you're feeling bored, just keep going, you know, learn something new.

11:14  
Don't bind yourself in.

11:16  
Now when you're learning computer programming, it's a different style of learning with respect to other learning.

11:22  
Now I know we had one person who said history was their their degree in history.

11:28  
You might read a lot of books, you might think deeply about them.

11:32  
You might then write a paper and reference all of these books, right?

11:37  
So you spend an awful lot of time reading things and thinking deeply about what you're reading.

11:43  
Now, some people, because that's their learning style, will do that.

11:47  
They will find a book on Python programming and they will take it home and they will read it from page to page and try to understand everything in that book before they start programming.

11:57  
Don't do that.

11:58  
All right, Why?

12:00  
Because the computer is your friend.

12:02  
If you try something and it's wrong, the computer will give you an error.

12:07  
So you can use the computer as a learning tool.

12:10  
You can, yes.

12:11  
Find out what you should be doing roughly, then try it out.

12:15  
Just put it into the computer terminal.

12:18  
Try the Python programme.

12:19  
It breaks.

12:20  
OK, where's it broken?

12:21  
And that that sort of quick test out style of learning is much faster for computer programming.

12:29  
You won't break the computer.

12:31  
It might slow down.

12:33  
OK, you're going to have to stop the programme.

12:35  
You won't break the computer.

12:36  
So it's different with respect to say, chemistry.

12:39  
We could blow up the building or something, right.

12:41  
But with with a computer programme, if you're playing with small programmes, you're not going to do any damage.

12:47  
So we need to learn a little bit and try, try, try, don't learn nothing, right?

12:54  
So you just Will's talking to me, fingers and ears.

12:57  
Just fire up Python And bash in some keys, right?

13:00  
And hopefully something will happen.

13:02  
That's also going to be a waste of time.

13:04  
So you need to learn some of the content and then try out, raise questions quickly if you get stuck.

13:13  
Oh, what's that?

13:15  
That's much easier in terms of learning.

13:19  
If you ask quickly, if you've got a question, somebody else might have the same question.

13:24  
Yeah, I'm dropped matter most actually.

13:26  
I should take this out, use the forum effectively.

13:30  
So when we are writing a computer programme, it's a logical process.

13:36  
It doesn't actually matter which language we are using.

13:39  
Different programming languages suggest a different style of writing.

13:46  
But that's that process of taking a problem and breaking it into steps and then expressing those steps in a computer programme is a common skill.

13:57  
So we are going to look at that process and we are also going to look at debugging programme and we will see two programming styles.

14:05  
So procedural is where we have a series of functions that call other functions.

14:10  
And the other style we'll look at is object oriented where we put functions together into an object with some data and we'll come back to that later, don't worry.

14:20  
So already, yes, the resources you've got lectures, they are there to give you the beginnings of the ideas.

14:28  
You really have to go and try out the lab exercises to get, you know, to gather some more information.

14:33  
There are worked examples that are there for you to test that and modify, as we said earlier, to try to understand what's going on.

14:40  
Lab exercises in the lab, you're going to try and solve a problem.

14:44  
So by all means work in a group of friends.

14:47  
I've allocated you groups because I'm hoping you'll talk to each other.

14:50  
So until you come to the assessment, it's much better to talk to each other about everything in the module because you'll learn faster.

14:59  
If you get stuck as a group, just ask.

15:01  
There's going to be one or two demonstrators and me in the lab and then supported reading.

15:06  
That's there to broaden your understanding.

15:08  
I'm not going to assess you on the reading.

15:12  
As we've already said, they've got two assessments.

15:14  
The class test, which is a closed book theoretical test on have you understood this or not?

15:20  
The programming exam, this is 3 hour long.

15:24  
You get to sit in the lab, you're given a problem which is very carefully described.

15:29  
You just build the code to follow the instructions and hopefully you'll be fine.

15:35  
OK, so why are we doing software development?

15:39  
Well, software allows you to do things with your hardware.

15:44  
Essentially.

15:45  
You might have a good idea, but you are not really able to present this to anybody without a piece of software.

15:52  
For example, you might have a mobile phone, and you may think to yourself, if only this app existed, if only there was something that could do this, because I really need it.

16:01  
If you are able to develop in software, you could build that app, and there may well be a bunch of other people who need it too.

16:09  
You'll find software in all sorts of other places though.

16:12  
It's used in the energy sector.

16:14  
So when you're trying to simulate or understand nuclear reactors, yeah, it's used there.

16:20  
It's used in cars, a lot of cars these days.

16:23  
In fact, most of them now have computer programmes running in them.

16:28  
They actually have a little computer which controls how the engine works.

16:33  
So you have the optimization of the fuel burn.

16:37  
You may have other computers in there as well like GPS link UPS that it's detecting where you are or how you're driving.

16:44  
But anyway, aviation, they have a lot of computers in the in aeroplanes because they're monitoring the systems.

16:51  
So for example, what is the engine doing it?

16:54  
Does the engine need to be serviced?

16:56  
You have computer programmes that are calculating how steeply you can pull the airframe before it stalls.

17:04  
So there's stall warning systems.

17:07  
You generally speaking have little bits of computers inside, lots of different technology.

17:12  
So for example, if you go home and you have a clever television, most televisions these days are clever or smart as people like to call it.

17:21  
They normally have a microprocessor or microcontroller in there which is going to actually run some computer programme code.

17:29  
Now that code has to be downloaded.

17:32  
You can't download it into the Tele yourself.

17:35  
Sometimes maybe you can depending on how the telly is made, but it's what we'd call an embedded solution.

17:41  
So the little computer is inside the TV.

17:45  
If you go to the airport, the check in places there are computers in there.

17:51  
And again, there's some kind of embedded technology.

17:55  
You have loads of data these days, and if you want to analyse data you could try and use Excel or spreadsheet or something.

18:01  
But when you've got a lot of data that doesn't work anymore, you need a computer programme.

18:06  
And often it's Python with some libraries to perform data analysis.

18:11  
And then if you've got any crazy ideas, yeah, you may well end up needing a computer programme.

18:17  
OK, that's the motivation.

18:19  
Now I'll move this thing out of the way.

18:22  
Come on.

18:24  
No, All right, I'll just ignore it.

18:26  
So in terms of a computer inside, very generically, what do we have?

18:32  
We have a box or a board or whatever it is.

18:36  
And in there we have a processor and its job is to run calculations essentially.

18:42  
So it receives instructions like add these two numbers together, put this number into this memory space.

18:49  
So it does that and it picks up the data from a memory space which is on the device.

18:56  
So that memory space could be random access memory.

18:59  
Normally you've got memory in your phone.

19:02  
In your normal computer, there's some storage, right?

19:05  
So you could have a magnetic disc or a, it could be an SSD disc.

19:09  
Or in your phone you've got storage and you've often got some way of connecting it to the world.

19:16  
So there might be buttons on your computer if you're in an airport and you're scanning your boarding pass, the input could be the barcode rather than keyboard presses anyway.

19:26  
So that's the sort of box you have.

19:28  
Now these days the processor is often packaged as well with a graphics processor which does the graphics as a so-called system on a chip, which is what's inside a phone.

19:39  
OK, Now when we're programming these devices, we've got a choice and this is a choice.

19:45  
It's not a right or wrong answer, it depends on what you're doing down here.

19:49  
At the far end we have the computer which is controlled with a low level set of instructions which are called machine code.

19:59  
All right, so it actually works using binary and we express binary normally with hexadecimal.

20:06  
So you can, yes, you can write in machine code.

20:10  
It's not very easy to read, but it will run fast.

20:13  
Now, a little step up from that is so-called assembly code.

20:17  
Again, it's not very easy for humans to read, but it's very close to what the computer actually runs.

20:22  
So when you compile it down, it runs fast.

20:25  
C is used a lot, so Linux, the Linux kernel and the Windows kernel are mostly C still because when it compiles down, it's fast.

20:36  
However, it's not as quick to write in.

20:39  
So you've got a balance.

20:41  
It's a language that runs quickly, or it's a language that's easier to write in, which doesn't run as quickly, and it depends what you're doing as to which one you take.

20:51  
Python is so-called high level programming language, so it's much easier to write your programme in it, but it doesn't run as fast as C It's a lot slower than C The good thing about Python is you can actually put the hard bits in C and then you can connect them together.

21:08  
So you can actually put the bits that matter in a different programming language.

21:13  
So it's a choice.

21:15  
Now I'm going to cover a brief elements of software engineering before we actually look at the programming language.

21:20  
And we'll pick up software engineering next semester in 993.

21:25  
Now software engineering has been around as a discipline for quite a long time.

21:31  
This is the Apollo lunar Module from Apollo 11.

21:36  
And it was, yes, controlled by software.

21:41  
This is the Deskey user interface.

21:45  
So this is the user interface that the astronauts had to push buttons on.

21:50  
And in here there was a very limited computer programme.

21:54  
Now, the problem was that at the time, before this, everybody wrote code, right?

22:00  
So I've got an idea.

22:01  
I'll write it.

22:03  
Just write it.

22:04  
And the danger with that, it's a bit like the Wild West.

22:07  
You come in and you shoot your mouth off.

22:08  
You're like, right, code.

22:10  
Yeah, there it is.

22:11  
And nobody really knows.

22:12  
What did you do?

22:13  
Why did you do that?

22:15  
So during the Apollo guidance computer development process, they realised if they're going to put astronauts on the moon, they had to be careful about what they were writing.

22:25  
And so the discipline of software engineering really became something they specified exactly what was going to go in because that had limited memory.

22:33  
And if they got it wrong, the astronauts died.

22:36  
And so this is really the beginning of where it comes from.

22:38  
It is an engineering discipline that's continued to be developed since that point in time.

22:43  
So the thought process is this.

22:46  
You first of all need to ask yourself what the problem is.

22:48  
Now, if you're lazy, you're just like, I want something to fix my problem.

22:52  
Well, what is that problem?

22:54  
And before you can write the software, you need to expand out and think about exactly what do you want.

23:00  
And this is where we have the so-called user requirements, where we write down what does the person want?

23:05  
The little pieces, what do they want it to look like?

23:08  
The user interface, the buttons, how is the screen going to look?

23:12  
How is it all going to fit together?

23:13  
That's the high level design.

23:15  
The pieces of software, how do they plug together?

23:17  
And then we start thinking inside those big blocks, what are the components, the pieces of functions, the pieces of computer code they're going to plug together to make this thing happen.

23:28  
And then we implement the programme so we don't just pick up our keyboard and go, yeah, let's code.

23:35  
We actually think a bit before we write some computer code.

23:38  
Otherwise, you can actually generate gobbledygook code, right?

23:44  
When you run it, it's like, oh, it's broken.

23:45  
Why is that?

23:46  
I don't know.

23:47  
Why did we do it that way?

23:48  
I have no idea.

23:51  
You think it's silly, but it does actually happen without some thinking.

23:55  
And then lastly, we want to test the programme.

23:56  
Like does it do what we think it should do in different circumstances, like when we give it bad inputs or whatever?

24:03  
OK, here we go.

24:05  
You're lazy and you're just thinking, right?

24:07  
I want something to solve my problem.

24:10  
So you need to sit down and then ask yourself or ask somebody else to often, well, what is it you actually want?

24:17  
And then the person tells you, Oh yeah, I actually want a device.

24:21  
It's going to be an embedded device, and it's going to have a couple of buttons on the outside.

24:25  
That's the user interface, just two buttons.

24:28  
And there's going to be some data flowing in here from somewhere or another.

24:31  
And this thing is going to have to display on a very small LCD screen here the time or some other messages.

24:39  
And yet it's then going to process some data.

24:41  
And there's going to be a data flow out.

24:42  
And you say, OK, fine, but what about inside?

24:45  
What will happen inside?

24:47  
And then the person says, well, this device is going to read some data.

24:50  
It's going to have to handle some user inputs to do with these buttons.

24:54  
It's going to have to process the values from the data or the buttons, and then it's going to have to put out some outputs and then write some data probably to the screen or something like that.

25:04  
And so you see you've got this general idea and you're breaking it down into pieces.

25:08  
And then you think, all right, OK, what's inside one of those pieces?

25:12  
Well, I'm not going to build this whole thing in one go.

25:16  
So this is the like the final version where I've got 2 buttons and I'm doing all the bits and pieces.

25:21  
You often want to say I'm going to build a so-called minimum viable product.

25:25  
So I'm going to build something that's going to be useful but isn't the final version.

25:29  
So I'm going to get rid of one of those features.

25:32  
Right now I've decided roughly what I'm going to build in terms of the feature list, the user requirements.

25:38  
I'm then going to think about how is this going to fit together inside.

25:41  
So this is my high level design.

25:43  
I might have a bit of the programme that just listens to the buttons.

25:46  
Ah, the Button's been pressed.

25:48  
Great.

25:48  
That's what its job is.

25:50  
I might have another bit of the programme that receives some data and its job is to put it on the screen.

25:55  
There might be another bit of programme which actually just takes the data and stores it.

25:59  
And there might be another bit where it's just calculations, right?

26:02  
So it takes in some numbers, calculates something and then chucks it out.

26:06  
So you can break this bigger problem down into pieces.

26:11  
And then within that piece you can decide I know what needs to go in there.

26:15  
So it's much easier to sort of focus your mind on a small piece.

26:21  
Now when you're building a programme, you need to describe what you're going to build before you actually build it.

26:28  
Now, if you're more experienced, the computer programming language starts to become something that you can just write in.

26:36  
So it is like I speak Python or whatever.

26:39  
Often though, you need to think in terms of the process.

26:43  
So you for some complicated processes, you might want to draw a flow chart just to explain to others.

26:49  
Well, what happens.

26:50  
Oh yeah, this has got to be in input here.

26:52  
Then that.

26:53  
And then if that isn't, then we have to go here.

26:55  
So a flow chart pseudo code is very helpful.

26:59  
Pseudo code is a bit like English or whatever your native language is.

27:04  
It's similar to a computer programme in that it's got some rules.

27:09  
The idea here is that we are trying to carefully express what we want out of the computer programme.

27:16  
We'll do a little bit of this in Week 2 just to sort of understand the idea.

27:21  
It can be really helpful for writing comments.

27:23  
Sometimes I think to myself, what do I need to do with this computer programme?

27:27  
And I write down the pseudo code in comments.

27:30  
Oh yeah, I need to load this file, this, that, the other, and I've got all the comments and then I start working through them and implementing the code.

27:38  
It's a good way of recording your thoughts and breaking stuff down into small steps.

27:44  
OK, so that's the sort of software engineering introduction.

27:48  
Does anybody have any comments or questions about that before we jump into Python that seem mostly sensible?

27:56  
All right, fine.

27:59  
Now Python, rather like a lot of programming languages, is actually just a text file.

28:06  
And the text file is then read by something and that something runs the programme.

28:13  
So when we write a Python programme, what we have is we actually have some text.

28:18  
So you type it in now instead of the file being called dot TXT, which would be a text file on on Windows or whatever, we call it dot PY, but it's still just inside a text file.

28:31  
And then what happens is the Python interpreter is given this text file, it runs it and that's how the programme works.

28:38  
So in this module we are using Visual Studio Code, which is what you can see on the screen.

28:44  
We do use that in other parts of the of the MSC, which is why I'm using this rather than P rather than PyCharm.

28:51  
If you really want to use PyCharm yourself, go for it for the assessment in the lab, we're on Visual Studio Code, so you might want to try it out so you're ready for the assessment.

29:01  
The other reason I use Visual Studio Code is it's it's very simple to debug stuff and control what's going on.

29:09  
Nothing is hidden in Visual Studio Code.

29:12  
Well, mostly nothing compared to some other editors.

29:15  
What do we see in this editor?

29:16  
We can see the files that are in the current directory over here.

29:20  
This is the content, the file I've currently got open.

29:23  
I can run the file by clicking on the run button.

29:26  
I can edit the text here and then I've got the output down at the bottom.

29:30  
So for example, this code, when it runs, prints the text hello there onto the screen.

29:37  
With Visual Studio Code, you've got extensions.

29:39  
So when you use Python, it will want to load the Python extension.

29:44  
They won't do that in the lab because the Python extension is already installed.

29:47  
But if you install Visual Studio Code on your laptop, when you first open a Python file, it will say, do you want to install the Python extension?

29:54  
And you say, yes, thank you, because then that connects it to the Python interpreter so that when you say run, it will run it for you.

30:03  
What else can you find here?

30:04  
The debug button is here.

30:05  
We will get into debugging in the lab.

30:07  
We'll try it out.

30:08  
Debugging is very useful because you can step through a programme.

30:12  
This is common to most programme languages.

30:15  
You can step through a programme and you can see what's in the computer's memory at different points and you can see what happens next.

30:23  
So if you can't spot the problem, debugging will help you.

30:27  
It's a sort of oh, oh, I didn't realise that value is in there and then you realise what you've done wrong.

30:35  
So to start with I'm going to discuss variables and then containers where we put variables.

30:43  
Now these are the sorts of things we are going to play with in the lab just to get used to the idea.

30:48  
Now computer programming sometimes looks like maths.

30:52  
However, it is not actually like a maths equation.

30:56  
What happens is that this equals is actually an instruction to the computer to say put what is on the right into what is on the left.

31:06  
So this is an equation makes no sense.

31:08  
Right box equals box plus one.

31:11  
Now that as a maths equation doesn't work because you'd have box minus box, so that would be 0 and you'd end up with 0 = 1, which is obviously not true.

31:21  
But what is happening in the computer, remember, is it calculates what's on the right and then it puts it into what's on the left.

31:28  
So here the number one is put into the bit of memory that's labelled box.

31:34  
A variable is just a way of saying this piece of memory over here is now called box.

31:40  
And so when you say box equals one, we're putting the number one into that piece of memory.

31:47  
And you can think of memory as a bit like a fixed.

31:50  
Well, it is a fixed space that contains a number up to a fixed size.

31:56  
So what happens here is we we've made an integer, a whole number, a variable.

32:01  
We put the number one in and then here we've said, OK, 1 + 1 is 2.

32:06  
And then the two goes into box.

32:08  
And then here we're printing to the word print in a computer programme means put it onto the screen.

32:15  
It does not mean send it to the printer and print it on a piece of paper.

32:18  
That's something different.

32:21  
Now that's numeric edition.

32:23  
So you can see in numeric edition, Python supports the concept of using addition for other things that aren't numbers.

32:32  
So when you write a piece of text like this, you can have it in double quotes or single quotes.

32:38  
All of my stuff is in double quotes.

32:40  
Just to keep it simple.

32:43  
We've got a piece of text here.

32:44  
So these are characters, text characters.

32:48  
When we say text plus text, what happens in the computer programme is it doesn't add the numbers.

32:55  
It says, oh, the box I'm going to append to the end contains.

33:00  
So now in here we've got the box contains.

33:04  
Now you could also take what's a number?

33:06  
That's an integer number and you could say to Python, I want to make that into a text string.

33:11  
So then this is going to be a text string which is going to contain 2:00.

33:15  
So at the end of this, it's going to be in text.

33:17  
It will be the box contains 2.

33:21  
OK, you can have floating point numbers or decimal numbers as well.

33:26  
You just described those with a dot.

33:29  
SO apologies if you're French and French, you use a virgin.

33:32  
But in computer language, we use a decimal point for a decimal.

33:38  
So here I've just made a decimal variable π and I've calculated something.

33:43  
So this is π R-squared, right?

33:45  
So it's r \* r \* π There's your area and print it on the screen.

33:50  
So that's basically how you use variables.

33:52  
Does anybody want to ask me a quick question about that?

34:00  
No, it's good to play with these things.

34:03  
If you, if you're not sure, go ahead and play with it.

34:05  
I give you the programmes, run it, change something, see if you can break it, understand what's going on.

34:12  
OK, There are two common ways of putting several variables into what I'm describing as a container.

34:22  
It is a way of justice putting several into one thing.

34:25  
And you are saying there are several numbers in here.

34:28  
The most common 1 is a list.

34:30  
Now lists are a little bit like arrays that you may have seen in other programming languages in that they are sequential one after another after another.

34:39  
It's a bit more like AC plus plus vector.

34:41  
If you like C++ in that it's dynamic, you can keep adding to the end.

34:46  
So here you can see you've got numbers and numbers is actually a list and it contains 123 numbers.

34:54  
And you can see there's a comma here.

34:56  
And that is saying to the computer, I want 3 three things in my list because there are three, sorry, there are two commas separating these three things.

35:07  
So that's going to be of length three.

35:09  
Now, once we've got a list, we can just print the list on the screen and it will print the numbers as is written here.

35:17  
The square bracket, 453 square bracket, it'll print that out.

35:22  
We can say in the computer programme, I just want one element.

35:27  
Now most computer programmes, BASIC is different, all right.

35:32  
So not listening with BASIC Fortran is I think different as well from deep memory.

35:38  
But most computer programme languages count from zero.

35:41  
You can think of the number here as being an offset.

35:43  
So you are starting from Indiana, my case, the number four.

35:47  
And when you say, OK, my offset is 0, I'm pointing at the number 4.

35:53  
So this will actually print the element that's in the 0 offset position here as 4.

36:01  
Python hopefully gives you -1 so -1 is actually saying one element from the other end.

36:09  
So what's what's in here?

36:11  
So not the maximum, what's this element?

36:13  
So -1 will actually give you this.

36:16  
Now you can loop over a list in Python.

36:18  
In other languages, it's sometimes called for each and you can basically say 4 number in numbers.

36:25  
Now what's happening here, we will come back to for loop later, don't worry.

36:28  
I am just putting this in now.

36:30  
So you can see is we are getting a value and that value is dropping into this variable number and you go around that loop until you run out.

36:40  
You can think of it like playing cards, right?

36:43  
So you have got three playing cards 4-5 and three and the dealer is the for loop and you say 4-5, 30, no more cards stop.

36:57  
So you can cycle through them like that.

37:00  
You can also join lists together.

37:02  
So what this means here is you can say I would like 0.

37:07  
In fact it's up to 1.

37:09  
So that means I would like this and then I would like in this case the 6th element, which isn't going to work here, sorry.

37:18  
It's going to add the element 6, isn't it?

37:21  
So you're sticking this together with the list 6 and then you're stuck in a bit of the list at the end.

37:26  
Run the programme, try it out.

37:28  
So you lists behave like strings.

37:31  
So you see here we the plus means append on the end.

37:35  
And here the plus means again append on the end.

37:39  
So you're sticking together pieces of list.

37:43  
Anybody got any comments or questions about lists?

37:46  
Yes.

37:48  
So generally good practise to put each line on a
Or is it just for this example to make it clear or?

37:53  
Yeah, so here I've just done it to make it clearer.

37:57  
You can put them all on one line.

37:59  
The problem you've got is when you write computer programme code, somebody else has got to read it.

38:05  
So sometimes programming languages allow you to sort of jam as much as you can in one line, and that makes it harder to read.

38:13  
So when you're writing it, we'll do this as you go along.

38:15  
Have a look at my programmes and think to yourself, oh, he's done that because it's easier to read.

38:19  
So here, yeah, I did do that for that reason.

38:22  
You could put it all on one line, it will work.

38:25  
But if you've got numbers jammed together, then they become harder to read.

38:29  
With Python, as with other things, there is a standard way of writing things.

38:33  
So with Python, the standard is called PEP 8.

38:35  
We'll come to it next semester, but it's a way of just verifying is your pricing programming code written in a OK manner for other people to read.

38:44  
So it will check, for example, is your line too long?

38:48  
Because if you are, if your eyesight's not very good and you've got super long line, then it's like, oh, Oh yeah, it works on my screen because I've got this resolution, but doesn't work on your screen.

38:57  
So you're thinking about other people as you write the the code.

39:01  
Anybody else?

39:05  
All right, so that's lists.

39:08  
Lists are a sequential container one after another.

39:11  
You can append to the end.

39:13  
You can actually pre append as well.

39:16  
Then we've got dictionaries.

39:17  
Now dictionaries are a container where instead of a position.

39:22  
So here we had a position where we're saying zero is the beginning of the list and then one is the next element.

39:28  
So this would be 01 and two position offset.

39:32  
Here we have a key and the key is used to get the element.

39:37  
So we've got in this case a dictionary and you can see the first key is A and then the value is five.

39:45  
And then we've got the key B and the value is 6.

39:49  
So if we find this dictionary values we could say right?

39:52  
I want to get hold of the the value 5.

39:55  
So what I'm going to do, I'm going to use the key A to get to that value 5.

40:01  
So rather like we did here, you can see here we we use the position offset here we go ahead and we use the key to get at the value.

40:09  
Now we also use the key to put in a new value.

40:12  
So here we're actually saying create me a new key C and associate it with a value 15.

40:19  
Here we're saying take the the key value B, but now overwrite the value 6 with the value 10.

40:29  
Dictionaries are sometimes useful because you may want to find something not based on a increasing number 012345, but it might be based on something else like, oh, this has happened #55 has appeared, right?

40:45  
I'm going to go just to that one.

40:47  
I'm going to skip the others.

40:50  
And we can also update a dictionary.

40:52  
So here we've added one new key and value pair.

40:55  
Here we've added two new key value pairs.

40:58  
And then yes, we can print out the whole dictionary or we can print out the value using the key.

41:05  
OK, so that's a dictionary.

41:07  
Does anyone want to ask me any quick comments or questions about dictionaries?

41:12  
Yes, yes, good question.

41:21  
There's a lot of what we call in computer language syntax, meaning that the computer understands some things in one way.

41:29  
So here, as soon as we use square brackets, the computer, when we're declaring it, the computer says, aha, the person is making a list, right?

41:38  
Because it's to the right of the sequel.

41:41  
So we're declaring that it's a list here.

41:44  
The computer to the right of the equals sees a curly bracket, so it says aha, the person is declaring a dictionary.

41:51  
OK, so the type of the variable is set here where it's first defined.

41:57  
So that's a dict or dictionary.

41:59  
And this is a list.

42:01  
Now when we're actually accessing it, the computer takes the square brackets here as being the position.

42:09  
So we've got the, this is the dictionary variable name values.

42:14  
We could call this thing instead of values.

42:16  
It could be called cars or soap boxes or something, right?

42:21  
And that's a way of saying I want the value that's associated with this key.

42:26  
So this square bracket is getting at the value or setting it, whereas this is declaring the thing as being a dictionary.

42:34  
All right, we'll play with the code in a minute.

42:37  
Anybody else got any questions?

42:42  
OK, fine.

42:46  
So I'm going to mention comments briefly and then I'm going to play around with code to see if we understand what we're doing so far.

42:52  
So comments are very useful.

42:54  
Occasionally you bump into so-called modern computer programmers and they think, Oh no, no, we'd never write comments because they might become out of date.

43:03  
You want to write some comments in your code because even you after six months will forget perhaps what you were doing six months ago.

43:11  
So you just little things in your code where you're saying what is the intention?

43:16  
You're not replacing what the code is actually doing.

43:19  
So you don't say looping over 5 variables.

43:22  
Yeah, OK, well I can see that it's a for loop.

43:25  
It's going over 5 variables.

43:26  
You haven't told me anything.

43:28  
You want to say something like processing user input to find total or you know you're trying to say what are you trying to do?

43:38  
That's why a comment is useful.

43:40  
Comments have to be maintained as well.

43:42  
It's no good putting a comment in and then updating the code and then the comments wrong because that will confuse people.

43:49  
In Python we have two comment styles.

43:51  
It's similar.

43:53  
Other languages have two comment styles.

43:55  
We have a multi line comment, which is where we've got these 3 double quotes, some text and then three double quotes and we have a single line comment which is where we have a hash character.

44:04  
Now the editor helpfully shows these in different colours.

44:09  
The text itself is just text.

44:11  
The editor shows it in a different colour to remind you it's a comment.

44:15  
This is a multi line comment.

44:17  
This is a single line comment.

44:18  
Now, a comment isn't processed by the computer as something that it needs to run.

44:23  
It's just there for you to understand what it is doing as a piece of code.

44:31  
Now we will mention it again next semester, but you can write comments that are associated with functions and classes, which when you want to understand what's in a file, you can get at that using pydoc in this case, and it will show you a summary of all the comments so you don't have to read the code.

44:48  
So commenting is really helpful because one, it tells the other person what's going on, and two, it will actually become a kind of documentation of the code.

44:58  
OK, yeah, that's all my slides.

45:00  
Anybody got any comments or questions about commenting?

45:02  
Yes, yes, yes, you are.

45:10  
So the assessment actually has a part on it which is use appropriate.

45:15  
I think I'd say something like use appropriate commenting.

45:18  
I'm not looking for anything very complicated.

45:22  
It's basically I've written a function and I've written a multi line comment to say what the functions for.

45:30  
I've written a class, I've got a multi line comment to say what the class is for.

45:33  
Maybe I've got a complicated piece of code and I've left a single line commenting to say what the intention is.

45:39  
So it doesn't need to be stupidly commented in the sense of comment every line.

45:43  
But I'm looking for the basics that you find in industry, which is that you've got a comment per large block or functional class.

45:51  
You'll see in my examples I'm consistent with respect to what I'm assessing.

45:55  
So as you go through, OK, that's what he's after.

45:58  
So I just need you to replicate back to me what I'm showing you is all right, OK, anybody else?

46:05  
So I'm not going to some more advanced stuff like the there's a kind of comment style we'll touch on next semester, which is where you are going to put in some parameter input types and other things, you know, you can be more descriptive.

46:18  
That would be next semester.

46:20  
Anybody else?

46:22  
OK, great.

46:23  
Nothing else.

46:24  
OK, so I'm gonna fire up our Visual Studio Code Now then let's have a new file.

46:33  
Let's first of all make ourselves a window or something else to play in.

46:39  
Now, when you first use Visual Studio Code, it doesn't actually have anything open.

46:44  
So I'm gonna go to my desktop and on here I'm gonna create a folder and I'm just going to call this temp for now.

46:57  
And now I'm going to say select folder.

46:59  
It's a better idea to open a folder because when we open that folder, we can then add several files to it.

47:06  
We'll do this over and over and over again in examples.

47:09  
So what can you see at the moment?

47:10  
Yeah, I trust myself.

47:12  
Thanks very much.

47:14  
On the left, we've currently got no files, and on the right we've currently got no code.

47:20  
So what we can do next is we can say right, I want a file, new file please.

47:24  
So I'm going to call this.

47:26  
I'm just going to call this test 1 dot PY.

47:31  
I've now got a Python file.

47:32  
It's got no code in at the moment.

47:35  
I'll make it bigger so you can read it.

47:38  
There's nothing here at the moment, but I can go ahead and start writing a programme.

47:43  
So maybe I'm going to say name.

47:49  
All right, so I've now got a variable.

47:51  
Now you can see this little dot here.

47:53  
This is telling you this white circle that I've not saved it.

47:57  
So you can then save it by either clicking on save or in my case, I'm just going to do control S It's now saved it.

48:03  
So if I run this, it doesn't do anything interesting.

48:06  
Yes, it's going to add.

48:08  
So I put the text computer into name, but it doesn't print it on the screen.

48:14  
So let's just put print in here.

48:17  
Print name.

48:19  
Yeah, it sometimes suggests would you like to do this?

48:22  
And it becomes more useful the further on you go.

48:25  
But for now, I'm just going to ignore the suggestion.

48:28  
So now we can run the programme again.

48:30  
And you can see in here, we've got the output.

48:33  
It's run the programme.

48:34  
You can see there's the Python interpreter, and that's my file.

48:37  
And the output is computer.

48:40  
Now I'm going to put in a bit more, so let's make this smaller.

48:46  
So let's put in here something like what we were doing earlier.

48:50  
I'm going to add my to the beginning of this, and then I'm going to append.

48:58  
So what's happening here is that I've got the text my.

49:02  
I'm going to stick on the end what's already in name.

49:04  
I'm going to put it back in name.

49:07  
So now of course, what's going to happen if I print that out?

49:09  
So this is experiential learning that I was talking about earlier.

49:13  
I'm now going to run this and you can now see it's printing out at the bottom.

49:19  
Make it bigger again, two things.

49:22  
It's printing out computer and my computer.

49:25  
So you're stuck bits on the end.

49:27  
Now this is where debugging comes in handy.

49:29  
If you're stuck, you're like, what on earth is going on?

49:31  
We can put here.

49:32  
You see when you hover with the mouse, there's a little red dot.

49:37  
Now that is going to become a breakpoint.

49:40  
The word breakpoint is used in all sorts of debuggers, not just Python.

49:44  
It's a way of saying I want the programme to stop there.

49:48  
That's so-called breakpoint.

49:50  
So let's break here.

49:52  
Breakpoint.

49:53  
Now you can see it's an active breakpoint.

49:55  
So we're going to go over to our debugger because I've zoomed in a long way.

49:59  
It's it's not the debug symbol.

50:02  
Yeah, I'm going to run a debug this Python file, please.

50:07  
Yeah, just this one.

50:09  
And what happens with the debugger is it will run Python, but it will allow us to see what's going on as it's running.

50:19  
It's not as fast as running without the debugger, but it allows us to see what's going on.

50:24  
OK, Once the debugger started, finally it will stop at that red dot and there it is.

50:32  
You can see it's stopped.

50:33  
And what we've got is now the ability to see what's in the variables.

50:37  
We can hover with the mouse and we can see, Oh yeah, in name, we've got computer.

50:42  
We can see on the left hand side the variables that have been defined at the moment.

50:46  
Well, there's one, it's name and it's got the computer in.

50:49  
We can hop forward one step.

50:52  
So like this.

50:53  
OK, now, yeah, it's printed on the screen.

50:55  
Now we're going to hop forward another step, and you can see that now in the variable name.

51:00  
We've now got my computer.

51:02  
So with the debugger you can see exactly what's going on and, you know, start to understand the language.

51:10  
Does anybody have any comments or questions about the debugger?

51:15  
No, OK, well, hopefully that's a bit of a taste, bit of an introduction.

51:20  
Please come to the lab, try stuff out, run the programmes.

51:24  
As I said earlier, I've given you the Python from the documents to the top, so you can play with it.

51:30  
It's a bit like Lego.

51:31  
If you get used to building with it, it becomes more familiar and you become more relaxed and confident that you are able to build things.

51:39  
So that's it, Thanks very much.

51:41  
I'm going to stop my video recording.

51:43  
I'll hang around if anybody is shy who wants to ask me a question.