Transcript

>>I want to thank you all for joining us today. My name is Mike Davidson. My pronouns are he/him/his. I'm a librarian at the National Library of Medicine, or NLM. I have a number of different jobs here at NLM, but one of them is what I'm doing today, which is developing and delivering training to help people learn about and learn how to use NLM products and services. And today I get to talk to you about one of my favorite things to teach people about, which is APIs. Also helping out with today's session is Kate Majewski, who will be helping me answer your questions, and we also have Brittney Davis, Michael Tahmasian and Catherine Staley who are handling some of our logistics for us.

Speaking of logistics, a few housekeeping matters before we get into it. We will be recording today's session to make it available to those who are not able to join us today or for you to review and refresh your memory later on. We've also made the slides available for your reference as well as a handout with some helpful links and reference information, and we're going to put links to both the slides and the handout in the chat right now. For those of you who are interested in MLA CE credit, that is available for today's session, just make sure you fill out the evaluation that pops up when you leave, as that will give you access to the CE credit. Though even if you are not interested in CE credit, we really do encourage you to fill out that evaluation anyway. This is our first time offering this particular class very exciting and we'd really love to get your feedback on it. Today's session will be interactive. I'll be walking you through a demo at one point, which you can feel free to follow along with. You'll only need a web browser in order to do that demo, so just any web browser will work. We're going to be using Zoom's polling feature, which we'll try out a little bit later on, and we're also going to use the chat panel, which I've already mentioned. If you have any questions, drop them in chat at any time and Kate will make sure that I address those questions during our Q&A portion at the end of the session. Also, we have some questions that we're going to be asking you and we'd like you to answer them in the chat. When you're using the chat, either to answer our questions or to ask your own, make sure your messages are sent to everyone. So our whole team can see them. As a practice round for this, let's start off with what's your favorite fast food place?

Go ahead and put your favorite fast food place in the chat addressed to everyone. Ohh man, we got a lot of responses here. A lot of Chipotle. I see some Taco Bell. Taco Bell is a perfectly reasonable choice. Culvers. I'm not familiar with Culvers. I'm going to have to look into that one. This is going to be a really good regional exploration of the regional location of our audience. Popeyes. Yes, of course, In-N-Out. OK, so we've got some California folks here. Alright. Fantastic. Alright, I'm going to stop my video now so I don't distract you all.

And we are going to move into our agenda. So here's what we're hoping to accomplish with today's session. By the time we're done, you should be able to explain what an API is, how APIs can help you interact with systems, and why you might choose to use APIs to accomplish certain types of tasks. You'll also be able to describe the basic mechanics of using an API, and you'll be able to identify some of the APIs NLM makes available for our products as well as know which-- when a specific NLM API might be useful. So as I said a few moments ago, my job is to help people learn about NLM products and services, and this is a pretty big job because, well, NLM has a lot of products. There are many, many different types of health and biomedical science information and many types of people who need to access that information. So our portfolio to satisfy all of those needs is pretty broad. And what you see on the screen here is only a small sample of what NLM offers. When a lot of people think about NLM, they first

think about accessing biomedical literature via PubMed, PMC, or Bookshelf. Others think of us as a collector and provider of consumer health information via MedlinePlus, and other products. NLM also does a lot of work in the medical terminology space, provides a number of databases and tools related to molecular biology or bioinformatics, and there's also a great source for drug and chemical information. Additionally, we are the home of ClinicalTrials.gov, which provides access to information about clinical studies conducted around the world, plus many, many more products and services that can be listed on this slide.

And actually I want to take this opportunity to do our first poll. Which of these categories of NLM products do you use and feel free to check off more than one, and if the products you use aren't listed here, let me know in the chat which other MLM products you use. I'll give you a moment or two to respond to that. Couple of DOCLINEs in the chat already. I see that. That is kind of a weird one off, but in terms of doesn't really fit into any of the other categories, you could sort of define that as literature though. "I refer to Gene in the class without PhD students, but use it is pushing it." OK that's fair. That's fair. Alright, looks like responses are already slowing down, so we'll take a quick look at what people have said. Alright, there we go. So it looks like #1 most popular is literature products. Not surprising. With terminology and consumer health close behind those. Though still some drugs and chemicals users out there as well or drugs and chemicals products users out there as well.

While many of you, most of you probably access these resources through our websites every day, some of our users who are pursuing new and innovative ways to use NLM information need a different type of access. So let's talk about some of these special cases. Our first example is one that you may have used yourself. Actually, many clinicians these days, from hospitals to doctor's offices use electronic health record systems, or EHR's that have a patient portal where you can review your own chart, test results and medical history. A lot of the information that's available in an EHR may be confusing to the average patient without further explanation by a clinician. And one solution that many EHRs use is an info button. The patient can click these info buttons next to a lab test, medication, diagnosis, or medical procedure and get more information in an approachable, easy to read format. MedlinePlus, NLM's largest consumer health resource is full of just this sort of authoritative, up-to-date information designed for patients, families, and healthcare providers. Developers and providers of EHR's can configure their systems to call up relevant MedlinePlus information automatically when users click on one of those info buttons. This can help patients stay informed about their health.

Let's take a look at another example, this one from the world of clinical oncology, and I want to give a huge thanks to Philipp Unberath of Friedrich Alexander University, Erlangen-Nuremberg, and his colleagues for this example. Increasingly, as genomic sequencing becomes cheaper and more feasible, clinicians are able to sequence cancer genomes and take a molecular based approach to cancer treatment, a tool that is used to help facilitate this molecular oncology approach is called cBioPortal, which is an open source platform developed by the Memorial Sloan Kettering Cancer Center that helps clinicians analyze clinical and molecular patient data. However, analyzing this data is only the beginning. What clinicians really need to do is turn that data into a treatment plan and new treatments are constantly being developed and validated through clinical trials. So in order to provide easy access to clinical trials that might enroll patients with a specific tumor profile, Phillip and his team integrated data from NLM's ClinicalTrials.gov directly into their instance of cBioPortal. Users can automatically see clinical trials that are registering new participants and what's more, by using the patient's data from cBioPortal as a filter, the list of trials can then be ranked, helping clinicians find studies that are the best

match for a patient's age, location, sex or cancer gene. All of this is done within the app without the patient or clinician ever needing to visit the ClinicalTrials.gov website.

The third example I want to talk about is a research example using literature citations themselves as data to better understand trends in scholarly publishing. This project was undertaken by a group of NLM associate fellows led by Luke Kudryashov, to explore potential sources of bias among journal publishers. They specifically wanted to look at the nationality of authors and see if specific journals tended to publish authors from specific parts of the world more frequently. Now, as you might know, author nationality is not a metadata field in PubMed, but author affiliation data is often available. This data can often contain evidence of an author's home country. Luke and his team wanted to see if author nationality could be reliably identified using natural language processing and machine learning techniques on PubMed author affiliation data. They could then use those techniques to explore whether journals or publishers exhibited bias toward or against authors of certain backgrounds. First, though, they needed a sizable set of PubMed citations, including author affiliation to train and test their NLP algorithm. What's more, they needed the citation data in a machine readable format to feed into the algorithm. Retrieving the requisite number of citations from PubMed would be rather cumbersome and time-consuming using traditional methods, and doing so in a machine readable format would be even trickier. So the team explored other approaches of accessing PubMed data.

So these three projects are each very different and each accessing information from different NLM resources. We had MedlinePlus, ClinicalTrials.gov, and PubMed. However, the thing these three projects have in common is that they use NLM information, but not NLM websites. In order to work the way they do and to be as successful as they are, they need a different type of access to these products. In the case of the first 2 examples, they need to access NLM information in a different application, not in a normal web browser. They also need to retrieve their data without a human user interacting directly with the resource they need to work automatically to some degree. All three examples need to access NLM information in a specific format, one which may not be available through the products website and in the case of the last two examples, they need to access health information as data, not for a human to read, but for a machine to process and analyze. So how do they accomplish this? Well, as you probably guessed, based on the title of today's talk, they do this using application programming interfaces or API's.

And before we go any further, I want to open up another poll just to get a sense of what previous experience with APIs you might have. Do you use them all the time? Maybe you use them periodically. Perhaps you have used them in the past but not recently. Maybe you know a little bit about APIs but haven't actually used them yourself. Or maybe this is all new to you, which is also totally fine as well. So I'll give you all a moment to answer that one. It looks like responses are starting to slow down. I'll give you another moment or two to get your answers in. Alright, I think that that's about what we're going to get here. Alright, so it looks like that we do have a range of experience here. We have a few people who have used them all-- who use them all the time or use them periodically, but still quite a healthy group of folks who know nothing about APIs or know a little bit about them, but haven't used them yourself. And in some ways, this poll was actually kind of a trick question, because while you may not have worked with an API, you've almost certainly used one. If you've ever checked a website to see if a local branch of a retail chain has a particular item in stock, or if you've ever compared flight or hotel prices on a travel website or use an app to get traffic updates or directions on a road trip or a commute, you've

probably used an API. All of those systems use APIs to gather information from other online sources and present that data seamlessly to you, the user.

So what exactly is an API? It's a little tricky to define since the term has been used to describe a number of different concepts in the world of computing and programming, dating all the way back to the 1960s. However, since a lot of NLM APIs were going to be talking about have some commonalities, we'll come up with a good, workable definition for our purposes today. An API is a set of protocols for contacting a remote system and making requests. In other words, it is a set of protocols for interfacing with a remote system. APIs are designed to be used programmatically as part of a computer program, not directly by humans. In other words, they are an interface to be used when developing and programming applications, or an application programming interface. The APIs we're going to be talking about today include a computer somewhere, which we'll refer to as the server that has information someone might want to access plus a set of rules for making requests or calls to that server.

It can all get pretty technical, so let's take a step back and talk about another way to think about APIs. Thinking back to your favorite fast food restaurant, which I asked about at the beginning of this session, if you want to order something from a fast food place, you can always park your car, go into the dining room, and order at the counter. That's like going to the website of your favorite NLM resource. Or you can go through the drive thru and the drive thru is like an API for your favorite fast food place. It's a quicker, streamlined way of getting the same stuff you get inside. When you pull up to the drive through, you say something into the intercom and based on what you say, you get a response back from the person on the other end. Either they tell you, "Oh I didn't understand what you asked for. Could you please say that again?" Or maybe they tell you "I'm sorry we don't have what it is that you're asking for today" Or, hopefully they tell you to pull through to the next window to pick up what you asked for. You get what you want and you never have to get out of your car.

So APIs work basically the same way. Rather than going to the actual website you, or a computer program that you're using, send a message to the remote system, the server, requesting whatever it is you're looking for from that system and the remote system then responds. If you ask for something the system has available, and you have formatted your request correctly, the system will respond by giving you exactly what you asked for. APIs can save a lot of time and energy in certain situations because they're designed to be integrated into programs. Developers and programmers can incorporate requests or calls to an API directly into their programs or applications, or even into other websites they're designing. Even better, developers can program rules for properly creating API calls right into the application and let the application create and send each request based on those hard coded rules. If you're making dozens or hundreds or even thousands of requests a day, this is the only practical way to do it. In addition to being faster and easier, if a program can access data from an online database directly as part of its normal functions, it can do even more without needing human intervention. Rather than requiring a user to go to the databases website, retrieve the information manually, and input it into the application, all of that process is handled automatically. Finally, some APIs provide access to data and information in different formats than the corresponding websites do. This could mean machine readable formats like XML or JSON that aren't always available to users through websites since they are less useful to the average human user. These machine readable formats may even include data that's not otherwise available through that website.

I don't want to get too technical about how APIs actually do what they do, mostly because I am not an expert in the technical nitty gritty myself. However, in order to see how APIs might help you or your users get things done, it can be useful to know a little bit about how APIs operate. Remember that the APIs we're going to talk about are composed of both a server somewhere, which is the remote computer you're trying to access, plus a set of rules or protocols for interacting with that remote computer. Going back to our drive thru analogy, the server is the specific fast food place that you're ordering from and the rules and protocols are like the menu board outside letting you know what's available to order. The way you or your computer interact with an API is via a URL, and these URLs look just like the kind of URL you'd visit in your web browser. However, the URL you use for an API call includes not only the address of the server for the API you're using, but also the details of what you're requesting from that API. What information you get back depends on how exactly you construct your URL.

So what do these URLs look like? We're going to look at some examples in just a second, but for most of the APIs we'll be talking about, the URLs are made-up of two parts: a base URL which indicates which API you're using, and parameters which indicate what you're asking that API for. So let's start with the base URL. This is the address of an API server. Think of this like the street address of the specific drive thru restaurant you're going to. Every time you go to that restaurant, it's going to be at the same address. Likewise, every time you use a specific API, the base URL for that API is going to be the same and tells your computer where to go to access the remote system for that API. You can see on the screen a few examples of some base URLs for some of NLM's APIs. The rest of the URL is made-up of what we call parameters. These include the details of what you're asking the server for. This can be things like search terms, if your request is about searching a database, or how many results you'd like to see or what format you'd like those results in. This is like your actual order at the drive thru window: what you're actually asking for. And like at the drive thru, these will most likely be different for every request, unless you always want to get the same exact results. On the screen at the bottom here you see an example string of parameters that I've used for retrieving PubMed data from one of our APIs. In this case, I have specified values for four different parameters separated by ampersands. First, we have a DB parameter which equals “pubmed” to indicate which database we want to retrieve data from. There's an ID parameter which specifies a specific PMID for a record that I want to retrieve. A retmode parameter which specifies that I want to retrieve results in XML and the rettype parameter which equals “full” specifying that I want full records returned, not just summaries. Different APIs allow or require different sets of parameters, so you need to look up how to use the parameters for the specific API that you're using.

I know we've been talking pretty abstractly with a lot of metaphors about how APIs operate, so now I want to take a moment to actually build and send an API request, and if you like, here's where you can feel free to follow along in your own web browser. You can type this right into the address bar just like you would if you were going to any website. And for this example, we're going to be using the MedlinePlus Web Service API. This API lets you search for and retrieve the information available via MedlinePlus health topics web pages, but in an XML format that can be more easily processed by machine. This allows you to embed the MedlinePlus health topic information on a website that you're building. We're going to start off with the base URL for the MedlinePlus Web Service API. You can see that on the screen here: it's https://wsearch.nlm.nih.gov/WS/query. The base URL for this API is always the same: anytime you want to query the MedlinePlus Web Service, you'll start with the same base URL.

We're going to put this base URL in chat as well, but make sure you copy and paste it into your browser's address bar instead of clicking on it, because, without the second part, the base URL isn't going to do anything useful.

Speaking of the second part, next we need to figure out our parameters. These are really important as they're going to tell the MedlinePlus Web Service server what it is we're looking for. If we don't spell it out in our parameters, it won't be part of our request and won't inform our results. If you're following along with the demo, before you put in any parameters, make sure you put a question mark in immediately following the base URL. This separates the two parts and lets the server know where the address finishes and the parameters begin. There are a few parameters that are required by the MedlinePlus Web Service. The DB parameter here tells the web service which database we want to look in: the English language health topics database or the Spanish language database. In this case, we've chosen to look for health topics in English. If you're following along, make sure you separate the first parameter and the second parameter with an ampersand and then for our second parameter we need to specify what we're actually looking for. In this example I'm going to be searching for health topics pages about acid reflux and if you want to follow along with a different search query, please feel free. Just know that if your search query has spaces in it like mine does, you need to replace each space with a "+" and again put an ampersand to separate this parameter from the next one. There are additional optional parameters that we can use with this API which can help define what format we want our results in or other options. For this example I'm using the parameter called “retmax” to specify the maximum number of results that I want to see. So for my demo, we're going to limit the query to just the first 5 results. If you want to follow along, you can copy the string of parameters we're putting in the chat and paste it into the address bar of your browser right after the base URL. Or just stay tuned until the next slide where we put all of the parts together.

So again, we start with the base URL followed by a "?" to separate it from our parameters. Then we're going to list each of our parameters and corresponding values in order, separating successive parameters with an ampersand. And there we go. We have now built a MedlinePlus web service API call. Now we can just put this URL into the address bar of any web browser and see what we get. And I did this ahead of time and what I got was this big chunk of XML. This XML contains the top five English language health topics in MedlinePlus that match our search for acid reflux. That's what I asked for when I specified those parameters, so that's what I got.

Now at this point you might be asking: Why would I do any of this? The MedlinePlus website has a perfectly nice interface where I don't need to know all of this fancy syntax and it will return actual web pages that I can read without having to be fluent in XML. And the answer to that question is you probably wouldn't do this. Not this way. What we've just done is not an efficient way to use an API. The best uses of API's are by programs or applications or scripts that have the URL creation rules built into them. That way they can take input from a user, build the corresponding URL, request the information from the API, accept the result in a machine readable format like XML, and process that data into whatever format the application needs in order to share with the user. Remember, you can't just walk up to the drive thru window. You need a car, and you need to know how to drive that car. So in this metaphor, the car is a programming or scripting language, which if you know how to “drive it,” can be used to automate the calls to the API. To extend this metaphor a little bit further, you might be able to roll up to the drive through on a bicycle. What we did before, building the URL by hand and putting it in

the browser, that's the equivalent of cycling up to the drive through window. It technically works, but it's not particularly efficient and it's definitely not the way you are supposed to do it.

So what kind of car, or what kind of programming language, do you need in order to make use of these APIs? Well, you have quite a few options, actually. Pretty much any modern programming or scripting language will be able to do it. If you're a developer or you're already working with a developer, whatever programming language you're already using should be fine. If you're new to programming, you may want to look at Python or R. Both of those programming languages are on the easier side to learn, and also both have prebuilt toolkits called packages of commands designed to make using APIs easier, and you can download these and incorporate these packages into your programs. Some programming languages even have downloadable packages designed to work with specific APIs. For example, if you're using the R programming language, the “rentrez” package, which is spelled “RENTREZ,” R Entrez, which Catherine has also just put in chat, that RENTREZ package is designed specifically to help with the NLM E-Utilities API, a specific API that provides access to data from PubMed, PMC, and other NLM products. These packages streamline the process of working with APIs in your program and make it so you don't have to worry as much about the mechanics. It's sort of the equivalent of having an automatic transmission in your car. It doesn't do anything that a manual transmission can't. It just makes the whole process easier. And just to be clear, the amount of programming required to use an API might not actually be that much. We're not necessarily talking about developing a new iPhone app or, as Phillip did in my example, creating a plugin for a clinical oncology platform. Just creating a very rudimentary script that automates data retrieval using an API may be all you need to achieve your specific goals.

However, I totally understand for some folks any amount of programming may seem like too much programming, and this segues very nicely into my next point. If you don't know how to drive just yet, you have a few options. If programming or scripting is new to you, but you're interested and have a little time and patience, it may be worth your effort to learn. But there is another alternative which is: get a friend to drive you. If you yourself aren't a programmer, but you can find someone you know who is, you can ask them for help in building a project that uses an API. This may mean that you don't have to learn how to program yourself, but it can still be helpful to learn about APIs, how they work, and which APIs are available that might help you do what you want to do with NLM information. If you learn which API options are available, you can better communicate with your programming friend about what you need using language that might be easier for them to understand. Going back to the drive thru analogy, if you can give your driver the address of the fast food place and good directions on how to get there and tell them exactly what you want to order, it makes the whole process go smoother. At the very least, if you can point your programming partner to a specific API that can access the data you need, you can save them some time researching on a subject that they might not be otherwise familiar with.

All right. We're going to sort of change gears in just a second. But before we do that, let's do another quick poll just to see where we're all at in terms of prior programming experience. So please let me know if you are familiar with any of the programming languages listed and please feel free to select more than one option. If you have experience with a language that's not listed, please let us know that in the chat. And if you haven't had any programming experience yet at all, that is fine as well. There is an answer for that too. Remember, you can always find a friend to help you drive where you need to go. We have a SAS in the chat, there we go. I almost put SAS on there but I-- we took that one off at the last second. Basic Assembly. Oh man, some of the old school ones. Very nice. Open Refine. Yeah. Perl. Yep, that's definitely one that some folks have used. I've used APIs with that as well. We'll give folks another

moment or two to get their answers in. VBA. And nice, nice variety here in the chat, people bringing up programming languages that I haven't thought about in a while. Alright, give just another moment or two. Looks like responses are slowing down. Alright, I think we can end that one. Share the result. There we go. OK, so it looks like Python and R are the most popular. Not surprising. And shell scripting and PHP or JavaScript. Again, this is not terribly surprising for me. A small, small number of MATLAB folks in there as well. And you know a healthy percentage of people who aren't experienced with programming yet, which is again totally fair. There are definitely options and we'll talk a little bit about that towards the end. For now, I think we should move on.

And I want to change gears a little bit and I want to spend most of the rest of the session going through some of the NLM's most popular and well-known products and talking about some APIs you can use to access those products' information. Now this won't be an exhaustive list of NLM APIs, however, it should hopefully serve as a sort of sampler platter, giving you an idea of what sort of APIs are available to help you interact with NLM's resources. As you consider the options I'm about to talk about, it's important to remember that APIs are not one-size-fits-all. You can't just use any API you choose to access whatever remote system you want, just like you can't go through the Starbucks drive thru and order a Big Mac or through the Taco Bell drive thru and order a pumpkin spice latte. You have to go to the drive through that offers the food you want, and you have to use the API that offers the data you want. The first step in choosing an API is always going to be looking for one that can access the data that you need. In some cases, an online database might have multiple different APIs that can each retrieve data in different ways or formats, and we'll look at some examples of this in just a few moments. I will also mention that there are links on the handout to more information about each and every one of these APIs. So if one of them catches your eye, you can go check it out in more depth after today's session.

Since we've already talked about the MedlinePlus APIs a few times during this presentation, let's start there. You might remember our example of building the URL - that was using the MedlinePlus Web Service API, which lets you search for and retrieve health topic information in XML. MedlinePlus also makes its data available via a different API designed specifically to be used by electronic health records and patient portals. MedlinePlus Connect is a service that EHR developers can use to provide free, authoritative, up-to-date consumer health information to patients based on specific medications, procedures, lab tests for medical conditions in that patient's health record. This is the API used in the example that I talked about at the very beginning of today's session. Now we can't very well talk about NLM products and services without talking about PubMed. On average over 3.4 million users access PubMed's website each day. However, that number doesn't even include API usage. In the last year over 1.2 billion PubMed searches were conducted via API, which is over 3.3 million API searches a day. The overwhelming majority of those searches were done using E-Utilities, which is a suite of APIs that provide access to over 35 different databases and products developed and maintained by the National Center for Biotechnology Information, or NCBI, which is the division of NLM responsible for PubMed. You can use the E-utilities API to search PubMed and retrieve records in a variety of formats, including the full PubMed XML, which includes some data that isn't otherwise exposed to users on the PubMed website. E-Utilities can also let you grab lots of PubMed records quickly in a machine readable format, so you can treat PubMed citations like data. This is critical for those doing bibliometrics research and portfolio analysis, and that example that we talked about earlier with the Associate Fellows who were trying to identify patterns in authorship nationality? They used the E-Utilities API to retrieve a large

number of PubMed Records in XML format, which they then used as the training data for their machine learning project.

In addition to E-Utilities, there are two other more specialized APIs that you can use to access PubMed data in specific ways. The Literature Citation Exporter API can be used to convert PMIDs or PMCIDs into formatted citation strings in a variety of citation formats, if you need a way to quickly and automatically generate that citation information. The inverse of this is the Citation Matcher API, which takes the citation matcher technology embedded in the search box on the PubMed website and makes it available for developers to use programmatically. Using this API to search for a citation string will return the set of PMIDs most likely to match that string. Sticking with the literature theme, let's talk about two other NCBI literature products, PubMed Central, or PMC, and Bookshelf. PMC is NLM's free full text archive of biomedical journal literature, while Bookshelf provides free online access to books and other documents. You can also use E-Utilities with both PMC and Bookshelf. These are two of the other databases that E-Utilities API can retrieve data from. However, there are other API options for these databases as well, which are particularly suited for retrieving full text from articles in the PMC Open Access subset or from books in the NLM Literature Archive Open Access subset. These APIs are built using a standard called Open Archives Initiative Protocol for Metadata Harvesting, or OAI-PMH. This is a standard used by a lot of different online digital depositories for their APIs, and the standardization means that if you can find a program that's already written to use one OAI-PMH API or a programmer that's familiar with that standard, it's much easier to adapt that work and knowledge to retrieve full text from the PMC and Bookshelf Open Access collections.

Now let's talk about some of the more terminology focused products starting with Medical Subject Headings, or MeSH. Many of you probably already know that many PubMed records are indexed with MeSH descriptors to help improve discoverability. Some libraries, including NLM, also use MeSH for descriptive cataloging. If you want to explore or download the MeSH vocabulary programmatically, you have a few different API options. If you recall the E-Utilities API which we just talked about on the last two slides, that one works for MeSH as well. However, because of the MeSH vocabulary's hierarchical nature, the E-Utilities API is a little bit limited in how it can retrieve MeSH data. This is one of the reasons why we can't have a one-size-fits-all approach for APIs: the nature of the data we want to retrieve varies from resource to resource, so the tools and protocols we need to retrieve it have to vary as well. Fortunately, NLM provides another API called MeSH RDF which helps you query and retrieve MeSH vocabulary information in a different way. As some of you might be familiar with, RDF stands for Resource Description Framework, which is a standard that represents metadata as linked data. The MeSH RDF API can be used to query a linked data representation of the MeSH vocabulary. Using MeSH RDF is a little bit more complex than the other APIs I've discussed, as you or your programming partner need to be familiar with querying linked data, which is a slightly different skill set than, say, searching PubMed or MedlinePlus. However, if you're looking for a way to dive deep into MeSH, MeSH RDF may be the API for you.

While we're talking about terminology APIs, I also want to give a brief mention to RxNorm and the RxNorm API. I know it may not be quite as well-known as some of the other products, but it is a great resource with a particularly useful API. As you may know, there are some unique challenges in designing computer systems to keep track of prescription drugs. The same medication can be called by different brand or generic names and may be identified by different codes in different drug terminologies. RxNorm provides a terminology of clinical drug names and codes that also serves as a crosswalk

between many other drug vocabularies, including First Databank, Micromedex, and Gold Standard. Developers of pharmacy management software, hospital information systems, and even clinical and public health research tools can integrate the RxNorm API into their projects to ensure that their applications can keep clear and consistent track of each individual's medications, even if the drugs are entered using different naming schemes. And the reason I wanted to mention the RxNorm API specifically is that there's a particularly useful tool called RxMix that can make it easier to get started using the RxNorm API. RxMix, which you can see pictured on the slide, is a website where you can experiment with the different features of the RxNorm API. You can try out API calls and workflows and figure out how to make the API work well for your needs. Once you've found the right combination of API calls for your project, you can take what you've learned and program it into your application. This makes it easier when developing and testing a project to tell whether the problems are in how you form the API calls or somewhere else in your code.

We talked about ClinicalTrials.gov earlier on, specifically when we looked at the example about clinical oncologists using cBioPortal to explore treatment options based on genetic tumor profiles. That project uses the ClinicalTrials.gov API (which doesn't have a special fancy name; it's just the ClinicalTrials.gov API) to help clinicians find possible clinical studies which might be a good fit for their patients. The modified version of cBioPortal uses the relevant patient information to formulate a search request to the ClinicalTrials.gov API. The results which the API sends in machine readable format are then ranked by cBioPortal and converted into a human readable interface for the clinicians to review. Like with RxNorm, the ClinicalTrials.gov API also has a web-based graphical user interface shown here on the slide that you can use to develop and test out your API calls before you get started.

So now that we've run down the menu of NLM APIs, what looks appetizing? We're going to put up another poll, and I want you to let me know which of the APIs we've talked about are you most interested in learning more about or working with? Just pick your one top choice, and if you're still not sure why or how you would use APIs, there is an option for that as well. Alright, give you a few more moments to make your pick. Got a comment for Literature Citation Exporter in the chat. Yeah, that's one that I feel like not as many people know about and is super useful in the right circumstances. Alright. Last five seconds, get your answers in now. Alright, I think we can wrap that up. All right. So in a runaway, not surprising: E-Utilities, considering first of all that it accesses 35 different databases, but also that it accesses PubMed which is the one that I think a lot of people are very interested in, though you know we got some votes for MeSH RDF, too; that's a fun one to explore that I think folks are not as familiar with. I see another one for Citation Matcher in the chat; that's a newer one so I enjoy-- I'm glad that people are starting to discover that. And we still got a decent percentage here that are not quite sure why they would use an API which is totally fair. And we'll actually hopefully help you out with that in just a moment. We can stop sharing that.

And what I want to do next, as we sort of head into the end of today's session, is go back over a few of the key concepts that we've covered. This is some of what I hope you take away with you today, aside from a deep craving for fast food. And this might also help out some of those, those of you who aren't quite sure yet when or why you would need to use an API. So APIs, or application programming interfaces, are great ways to retrieve data from NLM resources. In order to make the best use of an API, you probably want to be using the API in the context of a program, script, or application that you're developing, as APIs are designed specifically to be used programmatically. APIs often provide access to data in machine readable formats like XML or JSON, and since APIs query the same databases available

on NLM websites, they are great ways to get the most current up-to-date data quickly and on demand. Also, APIs are useful when you are searching for or requesting specific things from a database. In other words, when you know more or less what it is you're looking for. If you put these last two elements together, APIs are great at getting you exactly the data you need exactly when you need it.

Now, despite how cool and useful I think APIs are, there are still some times when using an API is not probably going to be the right call. First of all, this may seem obvious based on the last slide, but I do want to underline it: if you aren't programming, or aren't working with somebody who's programming, APIs are probably not going to be the right choice, since they work to their best advantage when integrated into programs or scripts. Also, as I've said many times before, APIs are not one-size-fits-all. You have to use an API designed specifically for the resource you're trying to retrieve data from. If the resource you're trying to access doesn't have an API, you can't use an API to access that resource. Fortunately, as we've talked about already, many NLM resources are accessible via their own specific APIs. If you're not exactly sure what you're looking for in a resource, and are still exploring or browsing or getting to know that resource, you may want to stick with the web version until you're a little bit more familiar with it. Because APIs require precise syntax to make specific requests, and because the output is often in machine readable, not human readable formats, you're often better off starting your exploration in the web interface and waiting to start building API requests until you have a better idea of what you're looking for. Finally, while APIs are really good at retrieving data in response to specific requests, if you're working on a project that needs access to the entirety of a databases data, like for example, if you want to do an analysis of all of PubMed's 34 million plus citations, retrieving that data via API may be a long and cumbersome process. Many APIs have throttles or gates to prevent people from abusing or overusing the system. These restrictions can make it difficult to do really massive bulk downloads. However, there may be another option for projects that require you to download all of the data in an NLM database. This isn't really the point of this presentation, so I'm just going to touch on it briefly.

The data from many NLM products are available in bulk downloads via FTP. This includes PubMed data, the Open Access subset of PMC, and NLM catalog records, MeSH, ClinicalTrials.gov records, and much more. For the right project, these bulk downloads can be invaluable, but they can take quite a while to download and require a lot of space to store locally. Also, in some cases you'll need to set up your own local database system to be able to search and query these downloaded datasets. Depending on your project, you'll also have to consider how often you need to stay up-to-date. The bulk downloads are a snapshot of the database at the time you download them. If your project requires the newest data, you'll have to continually download the latest updates. Going back to our drive thru metaphor, bulk downloads are kind of like the equivalent of going to the grocery store, buying all the ingredients, then bringing them back home and cooking the meal yourself. You might well end up with something way tastier and better than what you can get at the fast food place, but you have to do a lot more work to get your meal, and you'll probably have to go back to the store again next week to get fresh ingredients.

Hopefully this presentation whetted your appetite to learn more about and maybe even start working with NLM APIs. If you're trying to figure out what your next step is, here are a few suggestions. If you aren't familiar with programming or scripting, that might be a good place to start. There are plenty of online courses that teach the fundamentals of R or Python, including the basics of how to use those language to engage with APIs. Another good starting point for this is Library Carpentry. They host intensive multi day workshops that cover a range of data and technical skills aimed at librarians and

information professionals. Some Library Carpentry workshops include an intro to R or Python so take a look at the curriculum and see if you can find a workshop that works for you. Also, ask around at your institution to see if basic programming classes are available or if perhaps one of your coworkers already has the programming skills you need. Something else that can help if you have a project in mind is thinking carefully about the project before you get started. Identify what information you already have going in, be it a search term, an author name, a list of PMIDs, whatever. And also what information you're trying to retrieve. Because API requests require specific syntax to retrieve specific information, a little planning ahead of time can save you a lot of trial and error work in formulating the right request. And finally, make sure you've identified the API that is actually relevant to your project. We've talked about a bunch of APIs today that provide access to NLM data, but there are many more as well. To help you find the right API for you, you may want to check out NLM Data Discovery. The Data Discovery Portal serves as a directory of NLMs datasets and access points, including APIs and bulk downloads. Additionally, some NLM datasets that don't have dedicated APIs can be retrieved directly from Data Discovery using Data Discovery's own API. Once you've found the API you think will work for you, settle in to spend some time reading the APIs documentation. These docs can be dense and occasionally confusing, but often contain a wealth of genuinely useful information. APIs-- API documentation usually tells you what an API can and can't do, so you'll know if you're spending your time on the right API. Docs will also outline the syntax for creating an API request, including the base URL and all of the possible and required parameters. Most APIs have guidelines or restrictions for usage which are also documented. These guidelines prevent the system from being overloaded with too many calls at once and help protect our resources against malicious code. Finally, most API documentation, at least most of the good ones, include example API calls that you can review, modify, or pop into a web browser. The handout has links to help you find the documentation for all of the API's that we've discussed today. So that's most of what I wanted to cover today. But we also want to hear from you about what else you feel like you need in order to get started with APIs. So we've thrown open this poll here, but if there's other things not listed here that would help you get going with APIs, go ahead and put them in the chat. Also, while that poll is running, if you have any questions, you can put those in chat as well, and I will take a sip of water and turn my video back on and go to Kate to see what questions we have.

>>All right. Thanks Mike. So Sarah just popped a question in chat about viewing data in a bulk download. She's asking: what is a common way to view data in a bulk download? How sophisticated a program is necessary?

>>So that's a good question and it really depends on the bulk download in question and on how big of a download and how big of a data set you're talking about. For PubMed, it's the file-- the PubMed data set is broken into many files and is compressed. So you need something that can sort of uncompress it and combine those files together and then help you search through it. And there are some solutions out there for that. Other datasets that are large but not as large, any tool that can work through a large XML file. I think Oxygen is one of them. There are others as well-- can work through those. It really depends on which data set that you're looking at.

>>Thanks, Mike. And Jacqueline asks if there's a base code or example code for E-Utilities to get started on extracting data.

>>For E-Utilities, that's a good question. There is something, there is a tool out there called E-Direct, which is sort of basically a set of commands that were developed by one of our researchers here at NCBI

that works in a Unix or Linux environment and sort of implements the E-Utilities API stuff into some easy commands that also let you process the XML. That's called E-Direct, and there's documentation on that on the-- It's available via Bookshelf. Maybe one of our support folks can find a link to that. I don't have it at hand, but that's that is one way to go about it. Another way is again like looking at packages for programming languages that are designed to work with specific APIs. So RENTREZ is another example that I could think of off the top of my head.

>>Great. And I'm pulling up the E-Direct documentation. I'll get that in the chat in just a moment while Mike addresses the next question or request, could you talk a little bit about API keys?

>> Sure. I could talk a little bit about that I'm not an expert. Many APIs either allow you or require you to have what's called an API key, which is sort of an individualized string of numbers and letters that identifies you as the person who's making an API request. And usually you will incorporate that API key as a parameter. So remember, we talked about the base URLs and the parameters. One of your parameters would be your API key, which again will be specific to that API and to you and that will let the sort of managers of the API know who's making the requests, and so if you're misusing the API or abusing the API, they know who they can cut off. Sometimes APIs, E-Utilities is an example of this, you don't have to have an API key, but if you get one you are able to make more requests more frequently so that sort of-- the throttle on how much you can use the API is limited because the API knows who you are, so they can know who to who to come for. Well, I know we have a couple more questions I will-- I do want to address the poll here. Let's end this poll here and we'll share the results there. So people are looking for more examples of APIs in action. 70% there. Help with programming, and then also more info on NLM-- available NLM APIs. So I can already help you a little bit with that last one. Data Discovery and then the API documentation, which again you can find links to in the handout. Those are great places to start with that, but for the other things, more examples of APIs in action and help with programming that might be something that we need to sort of spend some time on as a training team and figure out how best to deliver that.

>>And you just addressed the final request. So thank you very much.