# Transcript: IT Podcast - Ep 84 - C949 Computer Science with DeNece Meyer and Jim Ashe

*The following transcript is a verbatim account of the video or audio file accompanying this transcript.*

Speaker #1 (Narrator):

WG use IT audio series, flexible, portable, profound.

Speaker #2 (DeNece Meyer):

Welcome to today's podcast. I'm here today with Jim Ashe. My name is DeNece Meyer. I'm a Program Manager in the computer science program andtoday we're going to talk to you about C949, data structures and algorithms 1, because some are the common questions that students havebefore they address this course or while they're in this course. Jim are you out there with me today?

Speaker #3 (Jim Ashe):

I am.

Speaker #2 (DeNece Meyer):

Good. Well, I have a few questions that students are commonlyasking me about this course as they start on this journey to this course. I'm going to throw a few questions at you and we'll get some best answers for students. What are the benefits of the data structures and algorithms in the BSES degree?

Speaker #3 (Jim Ashe):

Well, you mentioned that being Part 1 though, there's a Part 2. The same Part 1 implies, and so it does lay the groundwork for that Part 2 C950 data structure and algorithms. Part 2, both conceptually and technically, understanding data structures from a general, not a language specific point. I think a lot of our incoming students, they know about using Java or Python or wherever, the preferred language they use, understand how the data structures work. But this is more general, abstract look into those data structures. That presentation that's commonly used for data structure that I think is very useful, helpful for computer science students. They should be more than Java, Python or C, fancy programmers. They should be thinking about big picture things, they should be looking at being problem solvers, which is what algorithms is about. These go together very well hand in hand and lay a foundation that's very important, I think for computer science students. We do need to point out though that this course C949 is really designed for computer science students and then everybody else. We have a lot of non-computer science students take it, and the assessment is really suited better for everybody else. That's something to consider as you prepare for it. For the Part 2, 950, which only computer science students take, that's a Python programming project. The technical stuff that's presented for Python, even though it's not really a Python course per say, it's still very helpful because it lays that groundwork to introduce Python, particularly if it's not your first language. So you get ready for the technical side for Part 2. But for the ejective assessment for the OA, you only need a surface understanding of the Python technical syntax and stuff. You need a surface understanding ofthe more complex things like time complexity make a big O. But computer science should know, although they should know, of course, a lot more about the surface. I think those are all very important things forcomputer science major studies to be familiar with.

Speaker #2 (DeNece Meyer):

Great point. Well, that leads up to the very next question students are always asking me; is C949 a Python course, or where's Python used in this particular course, so it leads up to this?

Speaker #3 (Jim Ashe):

Yeah, I get asked that a lot too, and they're like, look at all these Python exercises. No, it's not a Python course. Python is a tool that we're using here to illustrate and learn data structures and algorithm concepts. That should be pointed out on the objective assessment. They will actually use many different types of languages. Maybe presented a pseudo code to test your understanding of howthat pseudo code with Java or Python or C might be interpreted. But you don't have to write any code on the assessment, just read it. Some of that specific language things probably shouldn't be there, but it's more of a surface technicality and it hasn't been a real hurdle for students. They're either infrequent enough or they'reeasier enough that it hasn't slowed students down, the inclusion of language specific parts. For you, 950 students in that unit, computer science students, you're going to need it. So it does have value learning the technical stuff with Python. If you're familiar with Java or some other object oriented language, transition into Python is usually not that hard. I mean, as far as one of the languages to learn isusually considered one of the easier ones to learn. It's usually just transitioning into the specifics of the Python. I do recommend that students work through and look atthe technical Python stuff because it helps make 950 a little easier. But if you do that stuff, be aware that you should use your own IDE, PyCharm is what we recommend and what 950 recommends. Use PyCharm, download it, write the code on that, so you're using an actual IDE, and if you can write the code to your own satisfaction, you're good. Design books learning resource has its own built-in editor and greater, but it's clunky and it's really too picky with the questions, and that sometimes slows students down justbecause they can't get it to accept their answers, but their coding is fine. Just make sure the code does what you want and that's good enough.

Speaker #2 (DeNece Meyer):

That it has great information on that. As we've talked about, we know that C949 is an objective assessment. What's the best approach for students to do in preparing for that course?

Speaker #3 (Jim Ashe):

Particularly for computer science students who are typically alittle more comfortable with the programming stuff, particularly if you've taken software 1 and 2 or feel comfortable with the types of courses, take the pre-assessment now. It lets you see where the bar is, what types of questions you need to prepare for. It lets you see how to more efficiently navigate through the textbooks. I think when you take it two things are going to stand out. First, there's just a lot of words to know, and you'll be adjusting what you've been used to, to those particular words. A lot of different data structures and how they behave. For example, what happens when there's a stackand you put something in and then what happens when you take something out? All these different data structures are defined by their behavior. What they're called in your particular language may be a little different, but they fall under the umbrella category of one of these data structures. You just need to know their words and there's a lot of it, its memory stuff. Data types is a third of the test. By character strings, information about that, a lot of memory stuff, data structures is a third of the test. These two sections alone are almost enough to pass, and programming, people withprogramming experience will be familiar with all that stuff already. They'll just feel they're adjusting what they know from their languageto the more general format or Python. The algorithm section, which is a third, is conceptually, and I think much more difficult. But the second thing we'll see when we take that pre-assessmentis how accessible I think it is on the assessment. For example, they'll ask you what's the worst-case time complexity? Big O. That's what that is. Like say bubble sort. Which of the following is a characteristic of quick sort, sorting algorithm? Well, that's stuff you just need to memorize. If you know you need to memorize it, those questions will be pretty straightforward. You either know them or you don't. If you're not prepared, you're going to be in trouble. There is a big chunk of things onthe algorithm section on the data structure sections like data type, that's just memory, flashcards stuff is what I call it. There is harder questions like determiningthe big-O time complexity of code you're presented. But most of this is just recognizing constant time loops or loops within loops, 01, 0n, 0n squared respectively. When students get on board withour understanding of that's how we're going to be a test, and you can see that when you take the pre-assessment when you start out, it makes that stuff seem a lot more accessible. Number 1, take the pre-assessment, get a feel for things. We also have a study guide that's been really helpful. It says, here are sort of things that you can focus on. It's not complete, but it seems to be a pretty good guide. Second, work through a textbook. You've seen the assessment, you got a learning guide, so you more efficiently go through the textbook. You may focus a little more time on Python, learning Python if you have the time. That's something to consider, something you should talk to your programmer about. Like DeNece, because she'll know that stuff. I won't. After you go through learning resource witha focus on the stuff you see on the pre-assessment and the study guide, retake the pre-assessment, this time for real, sit down like a test, treat it like a test, and also the course planning tool. It's basically a mini pre-assessment. I'm saying here, take it afterwards. It's just more assessment material, more practice to let you know where you are, let you know stuff that you need to polish up. From here you'll get a list of things that you need to know. Of course, you need to know why all the correct answers are correct, but also know what the incorrect choices are. Those are all things in their problem thing that direct you the things that you need to know in areas that you need to review. You identify those things. Some things would be like, oops, that was silly of me. I need to be more careful. Some things will be like okay, that's something that I didn't know but I know it now, other things you'll maybe you should talk to us about, and that's what we're here for. Please do know why the correct answers are right and know why the others are wrong. All these multiple choice questions are about comparing close situations. We will narrow it down to two or three, maybe two. But you're going to have to have some small thing or thatone or two incidences you're given to pick one option over another. That's the mentality we should take.

Speaker #2 (DeNece Meyer):

Well, that's great answer. I learned a lot from just listening to you on that. You led into the very next question that I have. What is the best time to reach out to course instructor? I know when I'm talking to students, I'm always listening for a point where they're struggling or what's going on so we can really utilize that subject matter expert, and I know all you all are amazing. So definitely for this particular course, what's a great time to really reach out to that course instructor?

Speaker #3 (Jim Ashe):

First always give the same answer. Anytime you want, it is what we're here for. But particularly after that last step, you've gone through the pre-assessment of course planning tool, learning resource. Now some things need to polish or you needdirection about getting ready for the assessment. Often students just need a little polish, particularly on time complexity. Students are often surprised that sometimes it'sa 10 or 15 minute conversation getting things a little clarified, and then they go off and do well. As you get started, great time to talk or maybe you took the pre-assessment, you got to preview the material, but you're not sure how to best proceed, given your timeline or your goals, maybe your pre-assessment, your program manager has let you know, hey, we are a little behind, we want to catch up and that would bea great time to talk to us about how to best do that. Yeah, and so of course, if you have limited time or worried about learning Python, you have a special situation, that would be a great time to talk to us.

Speaker #2 (DeNece Meyer):

Again, Jimmy, after we have talked about this, a lot of students come back to me and they ask me, are there any additional resources that a student needs touse other than Zybooks or do Zybooks cover the course well.

Speaker #3 (Jim Ashe):

I think it does cover a little well. I think it has a little more than you need, which is probably preferable than having too little. The study guide has been very helpful. Getting direction about how to best approach the course, I think is important, and another additional resource we have, and it's in the study guide, it will be linked there. Is a time complexity cheat sheet, where we have some practice problems. Discussing those practice problems with studentsalong with the staff in the pre-assessment has beena real help in understanding how to best approach time complexity on the assessments, which is always one of the hardest things. In general though, I think that computer science courses, this is one that students get through the fastest, and so we've had probably the least resources for this, but there are some webinars and development and some other stuff we'll see in the future.

Speaker #2 (DeNece Meyer):

Great. Well, thank you so much for your time today, Jim. I really appreciate all this information and I know our students will as well. Thank you for your time and like I say, I'll see you in the next podcast.

Speaker #3 (Jim Ashe):

Thank you. I look forward to it.

Speaker #1 (Narrator):

Schedule time with your course instructor to explore more deeply. WGU, a new kind of you.