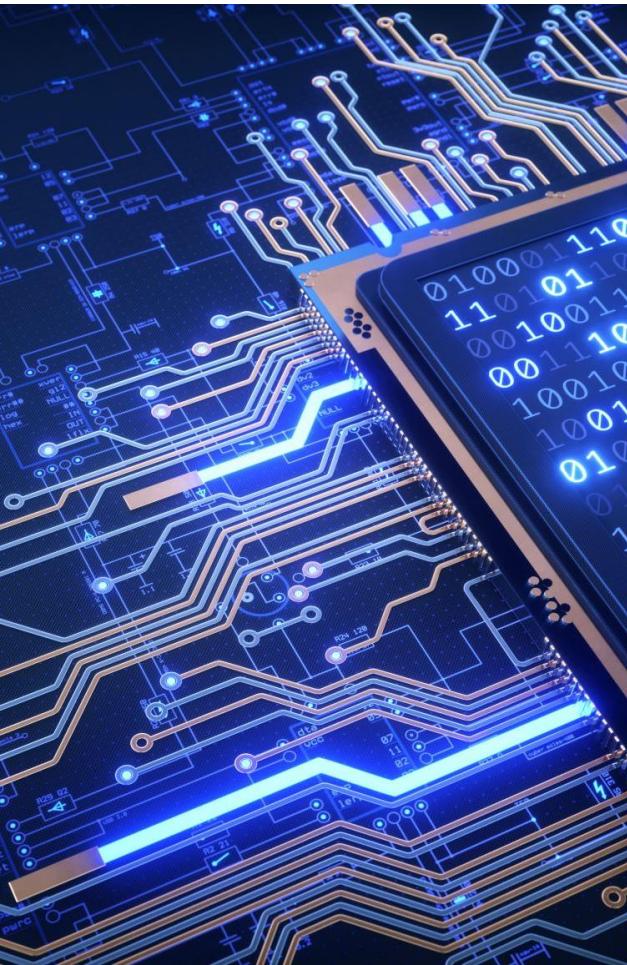


J / T

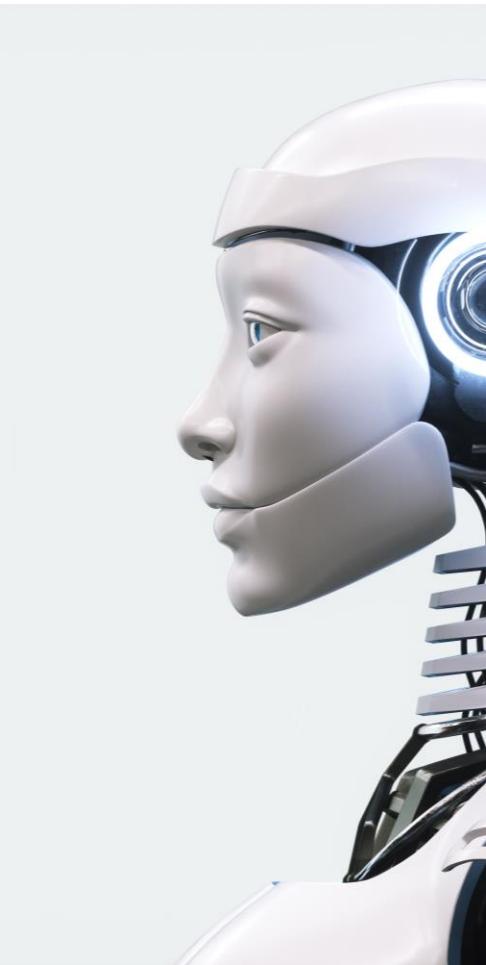
Jit Team™
the human factor of IT



From Zero to (AI) Hero. Can You Become a QA with ChatGPT as a Mentor?

Amelia Walter-Dzikowska

<https://jit.team>



Presentation available here

- <https://github.com/ameliawalter/TestWarez2023>



 Will Robots Take My Job?
https://willrobottakemyjob.com 

Will Robots Take My Job?

Do you feel informed about the data AI systems collect from you? **Yes, fully informed.**
Somewhat informed. Not informed, but would like to know ...
[Job Automation Rankings](#) · [Find a job in the USA](#) · [Fine Artists, Including Painters...](#)

Forbes

FORBES > BUSINESS

BREAKING

Will AI Take Your Job? 27% Of Jobs In Wealthy Countries At High Risk, Report Says

Brian Bushard Forbes Staff
I cover breaking news for Forbes

 Jul 11, 2023, 11:34am EDT

Updated Jul 11, 2023, 02:03pm EDT

 **TOPLINE** More than a quarter of all jobs across the world's wealthiest countries are in industries  where artificial intelligence could replace many  human workers, including in low- and mid-level  skill positions, according to a new report by  McKinsey.

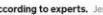
 INSIDER  

HOME > ECONOMY

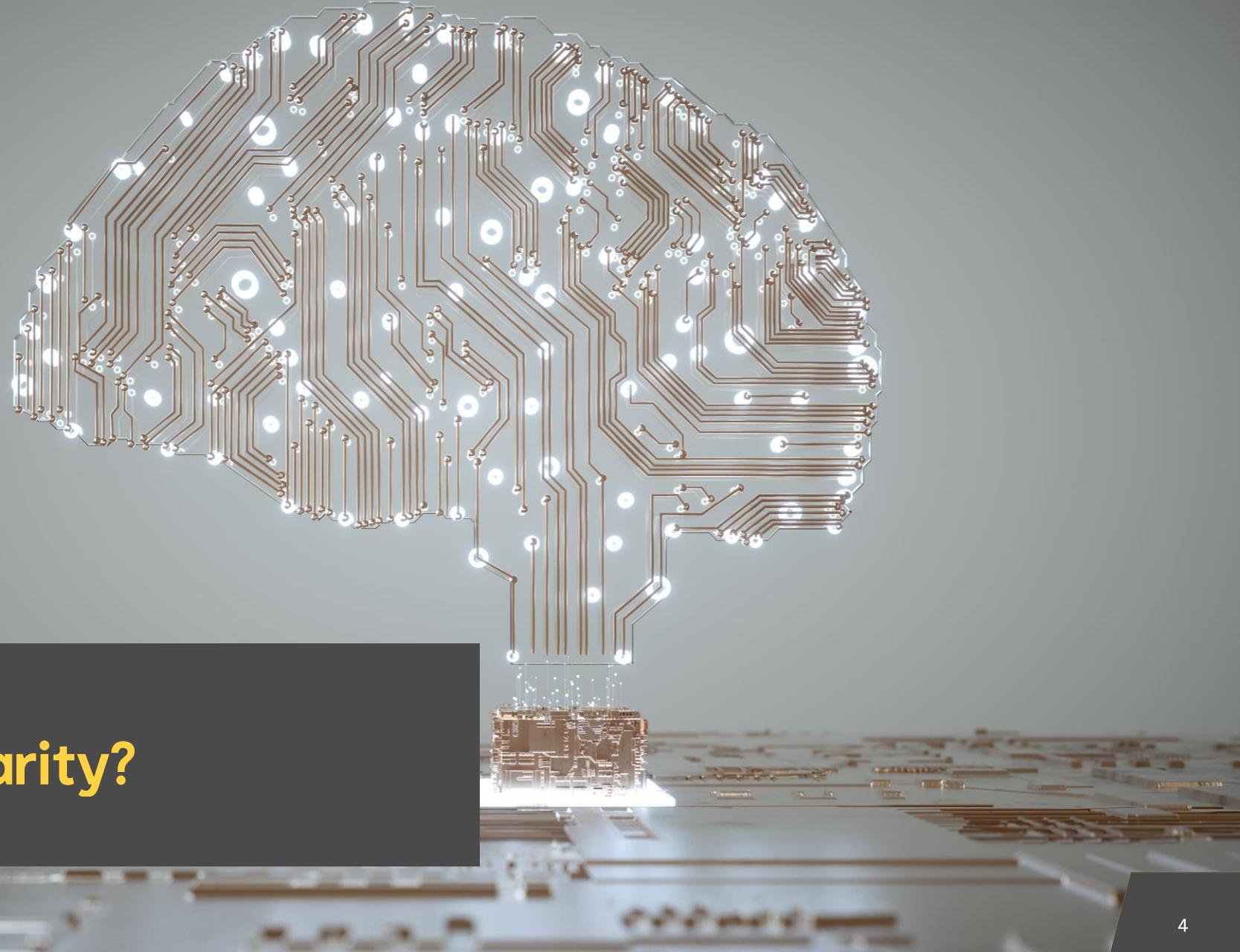
ChatGPT may be coming for our jobs. Here are the 10 roles that AI is most likely to replace.

Aaron Mok and Jacob Zinkula Updated Sep 4, 2023, 4:24 PM CEST



Insider compiled a list of the 10 jobs that could be disrupted by AI tools like ChatGPT, according to experts. 

- Experts say ChatGPT and related AI could threaten some jobs, particularly white-collar ones.
- It could do so by automating mid-career, mid-ability work.
- Insider compiled a list of 10 jobs this technology could replace, according to experts.



Technological Singularity?

istqb.org/certifications/artificial-intelligence-tester

About Us ▾ Certifications Training & Exams ▾

What would you like to search?

News SCR Glossary Help

Certified Tester AI Testing (CT-AI)

Find An Exam Provider Find A Training Provider

ISTQB®
Certified Tester
AI Testing

Overview

The ISTQB® AI Testing (CT-AI) certification extends understanding of artificial intelligence and/or deep (machine) learning, most specifically testing AI-based systems and using AI in testing.

Download Materials

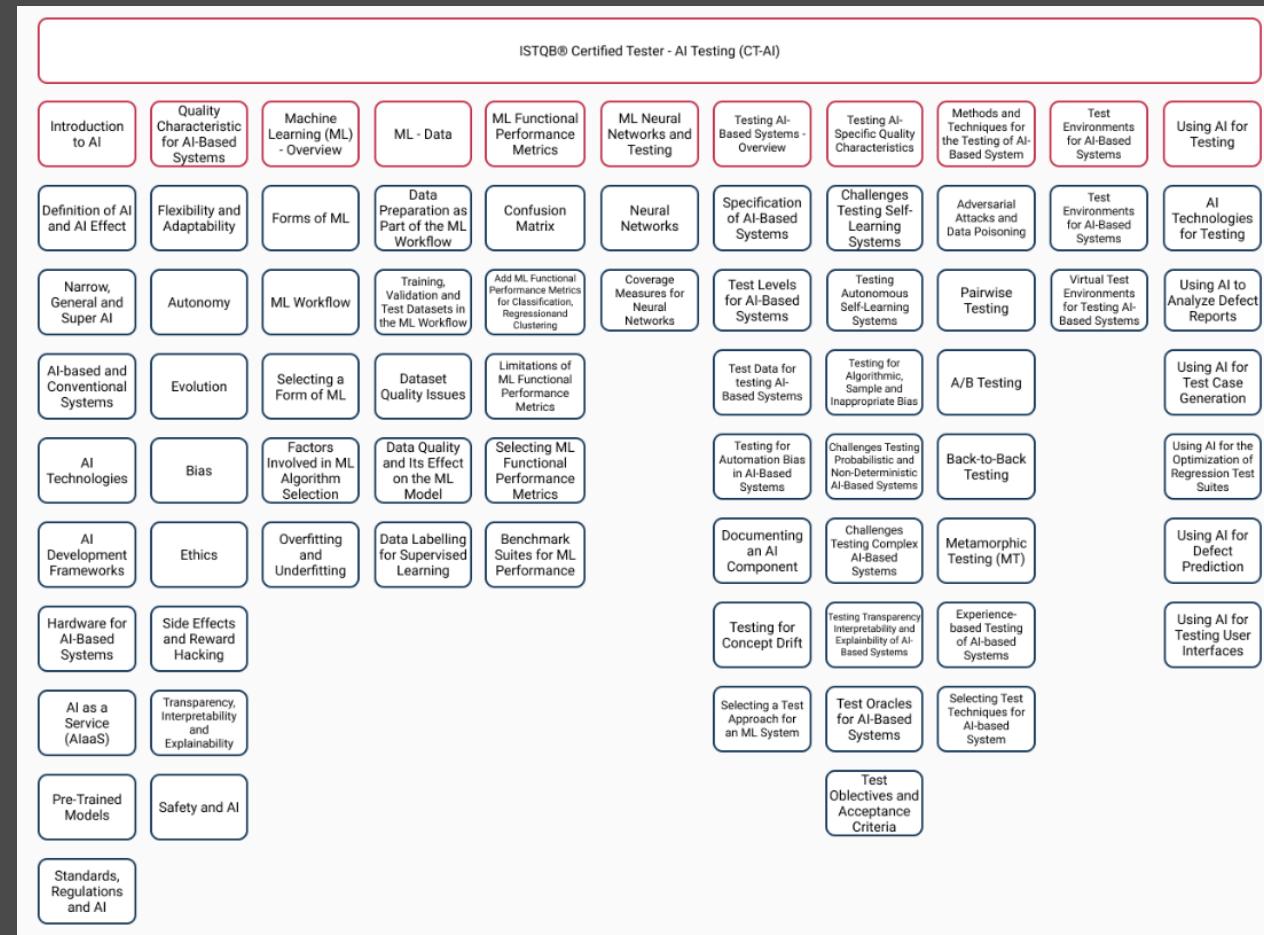
Syllabus

- CT-AI Syllabus v1.0 2MB | Downloads: 18,310 [Download](#)
- CT-AI Syllabus v1.0 2MB | Downloads: 3,267 [Download](#)

Sample Exams

- SampleExam Questions v... 434KB | Downloads: 5,771 [Download](#)
- SampleExam Answers v1.0 501KB | Downloads: 3,128 [Download](#)

ISTQB AI Tester



AI Testing

- Black box issue
- Statistics





OpenAI
ChatGPT 4.0

DALL·E 2

AI tools in everyday life



Bard



Midjourney



DALLE 3





フ /ト

Ethical Issues



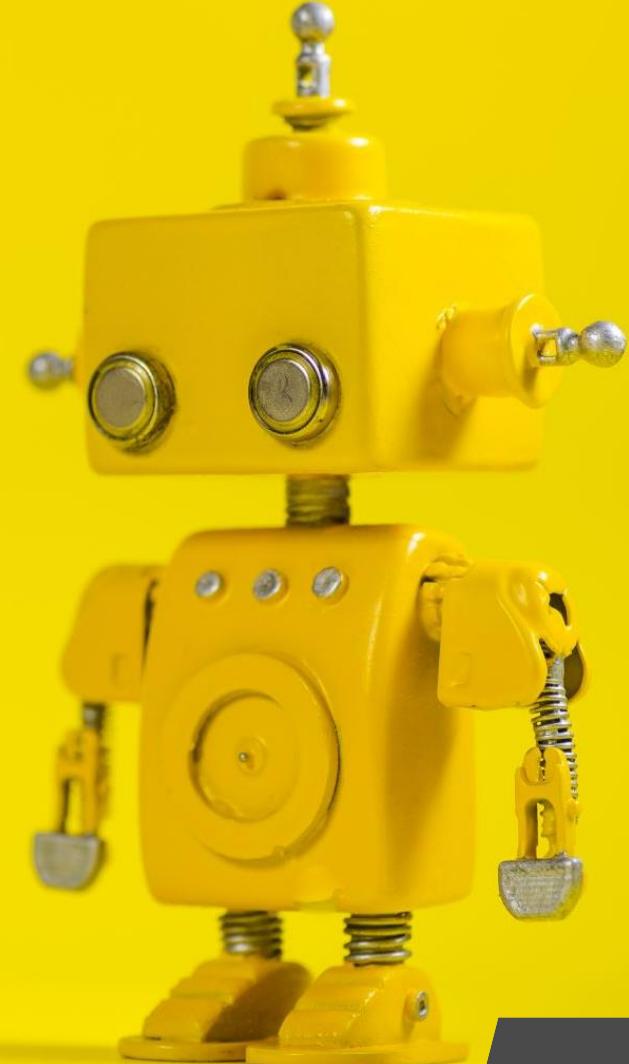
Ethical Issues

7 / 1



ChatGPT

- 3.5 and 4.0
- Plugins
- Browsing the web (new)
- Beware of hallucinations



ChatGPT & Learning

- Whether to use it
- How to use it



Hallucinations & Inadequate Data

leftrightarrow GPT-4



Explain to me boundary value analysis in software testing



Certainly! Boundary Value Analysis (BVA) is a software testing technique that focuses on the boundary values of valid input and output domains. The main idea is that errors are most likely to occur at boundaries.

Key Concepts:

1. **Boundaries:** These are the edges or limits of the allowed input values or range. For example, if an application accepts ages between 18 and 100, then 18 and 100 are the boundary values.
2. **Valid Boundary Value:** These are the values on the edges of the allowed range. Using the age example, valid boundary values would be 18 and 100.
3. **Invalid Boundary Value:** These are the values just outside the allowed range. For the age example, invalid boundary values would be 17 and 101.

⟳ Regenerate

Send a message



Example:

If a function accepts a percentage value between 0 and 100 inclusive:

- Valid boundaries: 0, 100
- Invalid boundaries: -1, 101

You would design test cases for -1, 0, 100, and 101 to test these boundaries.



So if age limit is 18-100 as in your example, give me all possible boundary values



Certainly! Using the age limit of 18-100 as an example, let's identify the boundary values:



1. **Lower Valid Boundary (LVB)**: The smallest value in the valid range.
 - $LVB = 18$
2. **Lower Invalid Boundary (LIB)**: The value just below the lower valid boundary.
 - $LIB = LVB - 1 = 18 - 1 = 17$
3. **Upper Valid Boundary (UVB)**: The largest value in the valid range.
 - $UVB = 100$
4. **Upper Invalid Boundary (UIB)**: The value just above the upper valid boundary.
 - $UIB = UVB + 1 = 100 + 1 = 101$

So, the possible boundary values for the age limit of 18-100 are:

- 17 (LIB)
- 101 (UIB)

Regenerate

Send a message



What about 3-point analysis?



Certainly! The term "3-point analysis" isn't a standard term in software testing like Boundary Value Analysis. However, in the context of boundary testing and considering the example provided, a 3-point analysis would likely refer to testing three crucial points:

1. **Lower Valid Boundary (LVB):** The smallest value in the valid range.
 - For the age example: **LVB = 18**
2. **Middle Point:** A value from the middle of the range to ensure that values inside the boundaries are also tested.
 - For the age example, a midpoint might be: **Mid = (18 + 100) / 2 = 59** (or any other value within the range)
3. **Upper Valid Boundary (UVB):** The largest value in the valid range.
 - For the age example: **UVB = 100**

Using this 3-point approach, you'd test:

- **18 (LVB)**
- **59 (Mid)**
- **100 (UVB)**

This is a simplified testing approach focusing on key representative values: the start,  Regenerate and end of the valid range. It ensures that the system behaves correctly not only at the

Send a message





There is something called three point BVA where there would be three values at each end: 17, 18, 19, 99, 100, 101



Ah, got it! I understand now. You're referring to an extended version of Boundary Value Analysis where you consider three points around each boundary.



In the "Three-Point BVA" for an age limit of 18-100:

1. At the **lower boundary**:

- One value just below the boundary: **17**
- The boundary value itself: **18**
- One value just above the boundary: **19**

2. At the **upper boundary**:

- One value just below the boundary: **99**
- The boundary value itself: **100**
- One value just above the boundary: **101**

So, for Three-Point BVA, you'd test:

- **17, 18, 19** (around the lower boundary)
- **99, 100, 101** (around the upper boundary)

This approach provides a more comprehensive testing strategy around the boundary values and ensures that any off-by-one errors or similar edge-case issues are captured.

Regenera

Send a message

Patterns

- You are as good as your prompts!



Persona Pattern

- *Act as Persona X*
- *Perform task Y*

Persona Pattern

❖ GPT-4



As an expert in ISTQB test analyst syllabus and a senior test analyst, explain boundary value analysis

« 3 / 3 »



Certainly! Boundary Value Analysis (BVA) is a test design technique used primarily for range-based input domain testing. As defects often occur at the boundaries of input values, this technique focuses on the edges or limits.



Here's a concise explanation:

Persona Pattern

4. **Example:** If an input field accepts values between 10 and 20:

- Lower boundary: 10
 - Test with 9 (just below)
 - Test with 10 (boundary itself)
 - Test with 11 (just above)
- Upper boundary: 20
 - Test with 19 (just below)
 - Test with 20 (boundary itself)
 - Test with 21 (just above)

Persona Pattern with 3.5 model

⚡ Default (GPT-3.5)



As an expert in ISTQB test analyst syllabus and a senior test analyst, explain boundary value analysis



Certainly! Boundary Value Analysis (BVA) is a testing technique used in software testing, including within the ISTQB Test Analyst syllabus. It's particularly valuable for testing input values and their corresponding output boundaries.

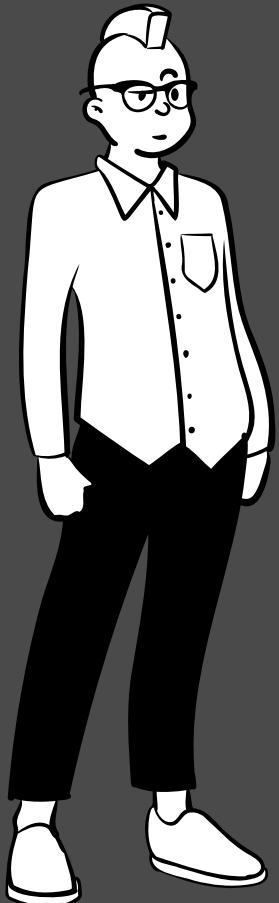


3.5 model gives poorer answers :(

1. **Identify Input Boundaries:** Start by identifying the boundaries of the input domain. For example, if you are testing a system that accepts integers between 1 and 100, the boundaries are 1 and 100.
2. **Select Test Values:** Choose test values that are on or just inside and just outside the boundaries. For our example, you might select values like 1, 2, 100, 101, and 50.



Becoming an Automation Tester with ChatGPT 4.0



Use case: **Automation Tester Roadmap with ChatGPT 4.0**

Question Refinement Pattern

- *From now on, whenever I ask a question, suggest a better version of the question to use instead*

 GPT-4



From now on, whenever I ask a question, suggest a better version of the question to use instead



Of course! I'll help you rephrase your questions for clarity or improvement. Just ask away, and    I'll provide a suggested version before answering.



How to learn to automate software tests?



Suggested question:

"How can one effectively learn software test automation techniques and best practices?"



Answer:

Learning software test automation is a multifaceted endeavor that combines understanding of software development, testing methodologies, and specific tools. Here's a step-by-step guide:

1. Understand the Basics of Software Testing:

- Grasp different types of testing: unit, integration, system, acceptance, etc.
- Familiarize yourself with basic testing principles, such as test-driven development (TDD).

2. Learn About Test Automation:

- Know the advantages and challenges of test automation.
- Understand where automation fits into the software development lifecycle.

3. Choose a Programming Language:

- Many test automation tools require knowledge of a specific programming language, such as Java for Selenium or Python for pytest.
- Familiarize yourself with the basics of the language, including syntax and commonly used libraries.

4. Select a Test Automation Tool:

- There are many tools available, such as Selenium, Appium, JUnit, TestNG, Cucumber, and many others.
- Base your choice on the application under test, platform (web, mobile, desktop), and your comfort with the required programming language.

Regenerate

Send a message





How do I learn to code in Python from scratch?



Suggested question:

"What is a structured approach to learning Python programming for beginners?"





Suggested question:

"What is a structured approach to learning Python programming for beginners?"

Answer:

Starting from scratch means you might not have prior experience in programming. Here's a step-by-step roadmap to learn Python:

1. Understand the Basics of Programming:

- Concepts like variables, data types, and basic operators.

2. Setup Your Environment:

- Install Python from the official website.
- Set up an integrated development environment (IDE) like PyCharm or VSCode.

3. Begin with Python Syntax:

- Learn Python's syntax, indentation, and how it differs from other languages.
- Understand basic operations: arithmetic, relational, and logical operations.

4. Control Structures:

- Conditional statements: `if`, `elif`, and `else`.
- Looping structures: `for` and `while`.

5. Data Structures:

- Lists, tuples, sets, and dictionaries.
- Common operations on these structures.

6. Functions:

- Define and call functions.
- Understand parameters, return statements, and scope.

7. Modules and Libraries:

- Importing modules and using built-in functions.
- Exploring the standard library.

⟳ Regenerate

Send a message ➤

8. Object-Oriented Programming (OOP):

- Classes and objects.
- Inheritance, polymorphism, encapsulation, and abstraction.

9. Exception Handling:

- Understand errors and exceptions.
- Use `try`, `except`, `finally`, and `raise`.

10. File Handling:

- Reading and writing files.
- File modes and handling exceptions related to file operations.

1. Intermediate Topics:

- List comprehensions, lambda functions, generators, and decorators.

1. External Libraries:

- Install packages using pip.
- Explore popular libraries like `requests` for web requests, `pandas` for data manipulation, and `matplotlib` for data visualization.

1. Practice Projects:

- Build small projects like a calculator, to-do list, or a basic web scraper.
- Progress to more complex projects as your understanding deepens.

1. Join the Community:

- Participate in forums like Stack Overflow.
- Engage in open-source Python projects on platforms like GitHub.

1. Join the Community:

- Participate in forums like Stack Overflow.
- Engage in open-source Python projects on platforms like GitHub.

1. Continue Learning:

- Explore more advanced topics like concurrency, design patterns, and frameworks (like Django or Flask for web development).

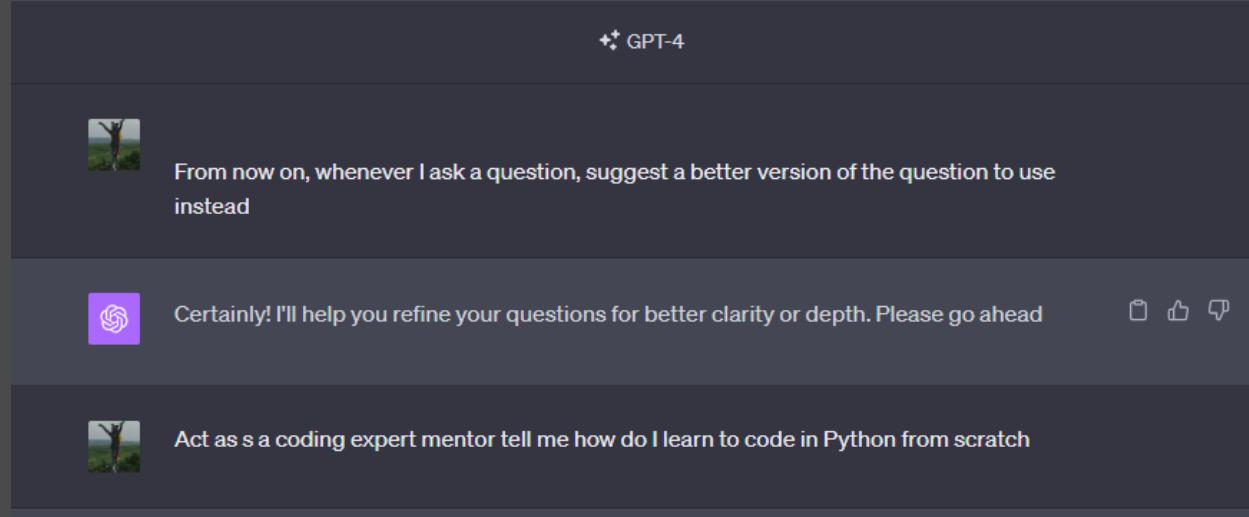
Resources:

- **Books:** "Python Crash Course" by Eric Matthes, "Automate the Boring Stuff with Python" by Al Sweigart.
- **Online tutorials:** Websites like Codecademy, Coursera, Udacity, and edX offer Python courses.
- **Interactive platforms:** LeetCode, Codewars, and HackerRank for coding challenges and practice.

Regular practice and application of what you learn is the key. Happy coding!



Question Refinement Pattern + Persona Pattern



Question Refinement Pattern + Persona Pattern

Your original question:

Act as s a coding expert mentor tell me how do I learn to code in Python from scratch

Suggested version:

As a coding expert mentor, how would you recommend I learn Python from scratch?

To learn Python from scratch, follow these steps:

1. **Set Clear Goals:** Before you start, understand why you want to learn Python. Whether it's web development, data science, artificial intelligence, or automation will influence your learning path.
2. **Begin with Basics:**
 - **Install Python:** Download from the official Python website (python.org) and install it.
 - **Interactive Shell:** Familiarize yourself with the Python interactive shell, which allows you to run Python commands in real-time.
 - **Syntax & Structure:** Learn basic Python syntax - indentation, variables, data types, and basic operators.
3. **Grasp Fundamental Concepts:**
 - Control structures: loops (for, while), conditionals (if, elif, else)
 - Functions and modules
 - Data structures: lists, tuples, dictionaries, and sets
 - Exception handling
4. **Hands-on Practice:**
 - **Simple Projects:** Start with basic projects like a calculator, to-do list, or a basic text-based game.
 - **Coding Challenges:** Platforms like LeetCode, HackerRank, and Codewars offer a plethora of challenges suitable for beginners.
5. **Explore Advanced Topics:**
 - Object-Oriented Programming (OOP)
 - File handling and I/O operations
 - Regular Expressions

6. **Specialize:**
 - If you're into **web development**, learn frameworks like Django or Flask.
 - For **data science**, focus on libraries like NumPy, Pandas, Matplotlib, and Scikit-learn.
 - For **automation**, explore libraries like Selenium and PyAutoGUI.
7. **Continuous Learning:**
 - Read Python's official documentation.
 - Join online communities like Stack Overflow, Reddit's r/Python, and other forums.
 - Follow Python-related blogs, podcasts, and YouTube channels.
8. **Feedback Loop:**
 - Share your code with mentors or peers for review.
 - Collaborate on projects to learn best practices.
9. **Advanced Resources:**
 - Read books like "Fluent Python" by Luciano Ramalho or "Python Crash Course" by Eric Matthes.
 - Take advanced online courses from platforms like Udacity, Coursera, or Udemy.
10. **Work on Real-world Projects:** Building real-world applications gives you practical exposure and makes the learning more engaging.

Remember, coding is as much about problem-solving and critical thinking as it is about knowing the syntax of a language. Keep practicing, remain persistent, and always stay curious.

< 2 / 2 >



Act as s a coding expert mentor tell me how to learn to code in Python from scratch for test automation



Original Question:

"Act as s a coding expert mentor tell me how to learn to code in Python from scratch for test automation."

Suggested Question:

"As a coding expert mentor, can you provide a structured roadmap for learning Python from scratch, specifically for test automation?"

Answer:

Certainly! Here's a step-by-step roadmap to learn Python for test automation:

1. Basics of Python

- Syntax, Variables, and Data Types
- Basic Operators
- Control Structures (if, for, while)
- Lists, Tuples, Dictionaries, and Sets
- Functions and Modules

Recommended Resource: Automate the Boring Stuff with Python by Al Sweigart

2. Intermediate Python

- Object-Oriented Programming (Classes, Inheritance, Polymorphism)
- Error and Exceptions Handling
- File Handling (Reading/Writing)

Recommended Resource: Python Crash Course by Eric Matthes

3. Introduction to Testing in Python

- Understanding the importance of testing
- Writing your first unit tests using the `'unittest'` module

Recommended Resource: The Python Standard Library Documentation on `'unittest'`

4. Test Automation Basics

- Introduction to Selenium WebDriver for web automation
- Setting up Selenium with Python
- Writing basic tests to automate browser tasks

Recommended Resource: Selenium with Python Documentation

5. Advanced Test Automation

- Page Object Model (POM) for structuring your test automation suite
- Cross-browser testing
- Handling AJAX and dynamic content
- Implicit and Explicit waits in Selenium

Recommended Resource: Test Automation using Selenium WebDriver with Python by Navneesh Garg

6. Frameworks and Tools

- Introduction to popular Python testing frameworks like PyTest and Behave
- Data-driven testing
- Continuous Integration (CI) with tools like Jenkins

Recommended Resource: "Python Testing with pytest" by Brian Okken

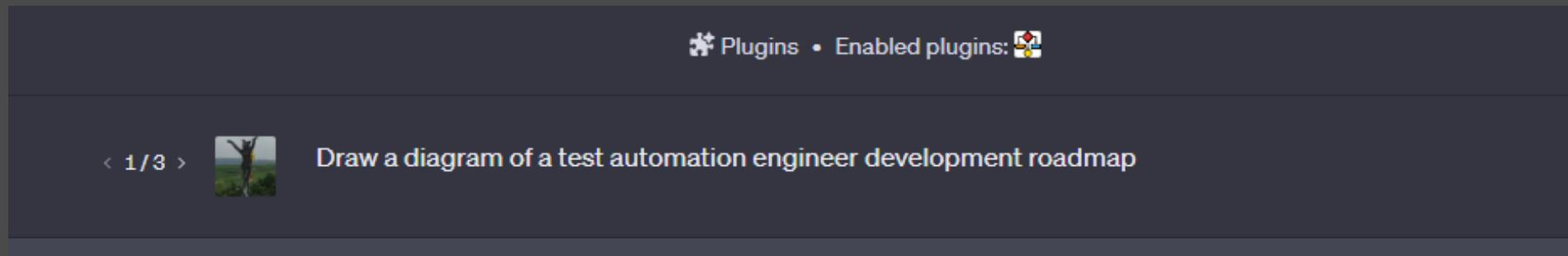
7. Best Practices & Advanced Topics

- Code maintainability and readability in automation
- Mocking in Python for isolating parts of your system to ensure accurate tests
- Performance testing with tools like JMeter or Locust (though not Python-specific, it's beneficial to understand performance testing)

Remember, hands-on practice is key. Work on mini-projects or replicate existing web application tests to solidify your understanding. Good luck!

One More Persona Pattern Example

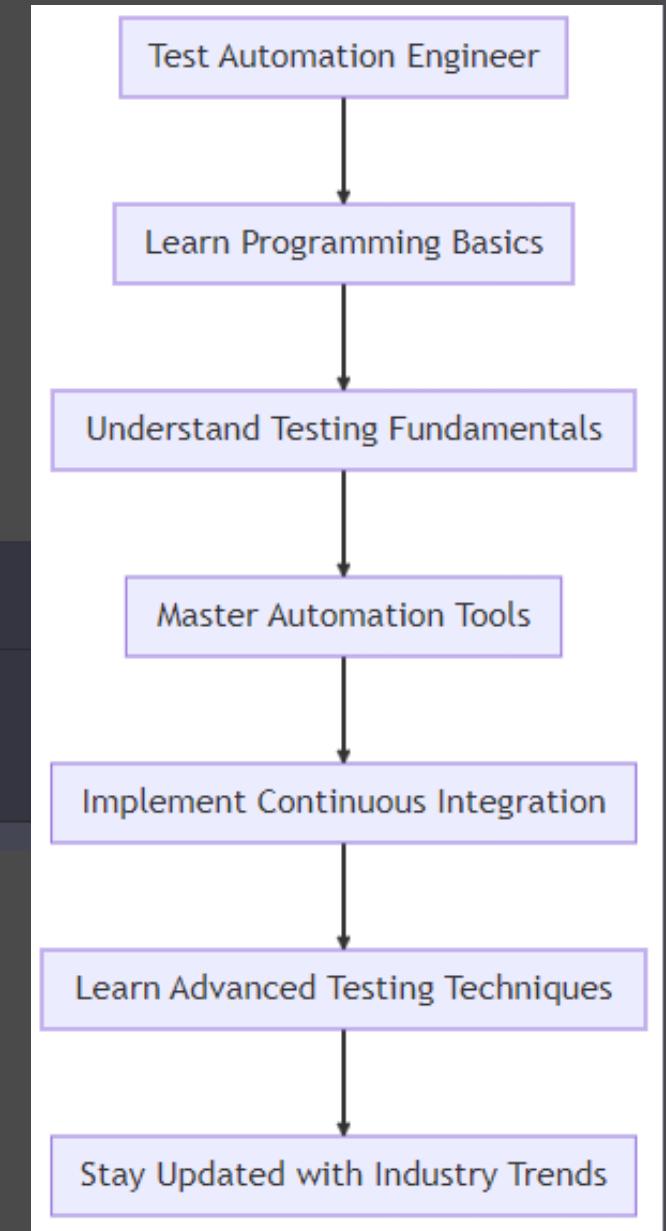
- First without any pattern...



One More Persona Pattern Example

- First without any pattern...

A screenshot of a digital workspace interface. At the top, there's a header bar with a profile icon, a search bar containing "Test Automation Engineer Development Roadmap", and a "Plugins" section showing "Enabled plugins: 🎨". Below the header is a dark grey sidebar with navigation icons. The main area features a card with the title "Draw a diagram of a test automation engineer development roadmap". On the left side of the card is a small thumbnail image of a person standing in a field. The card has a "Due" date of "1 week ago" and a "Last updated" time of "1 day ago". There are also "Assignee" and "Comments" sections.



One More Persona Pattern Example

- ...with persona pattern...

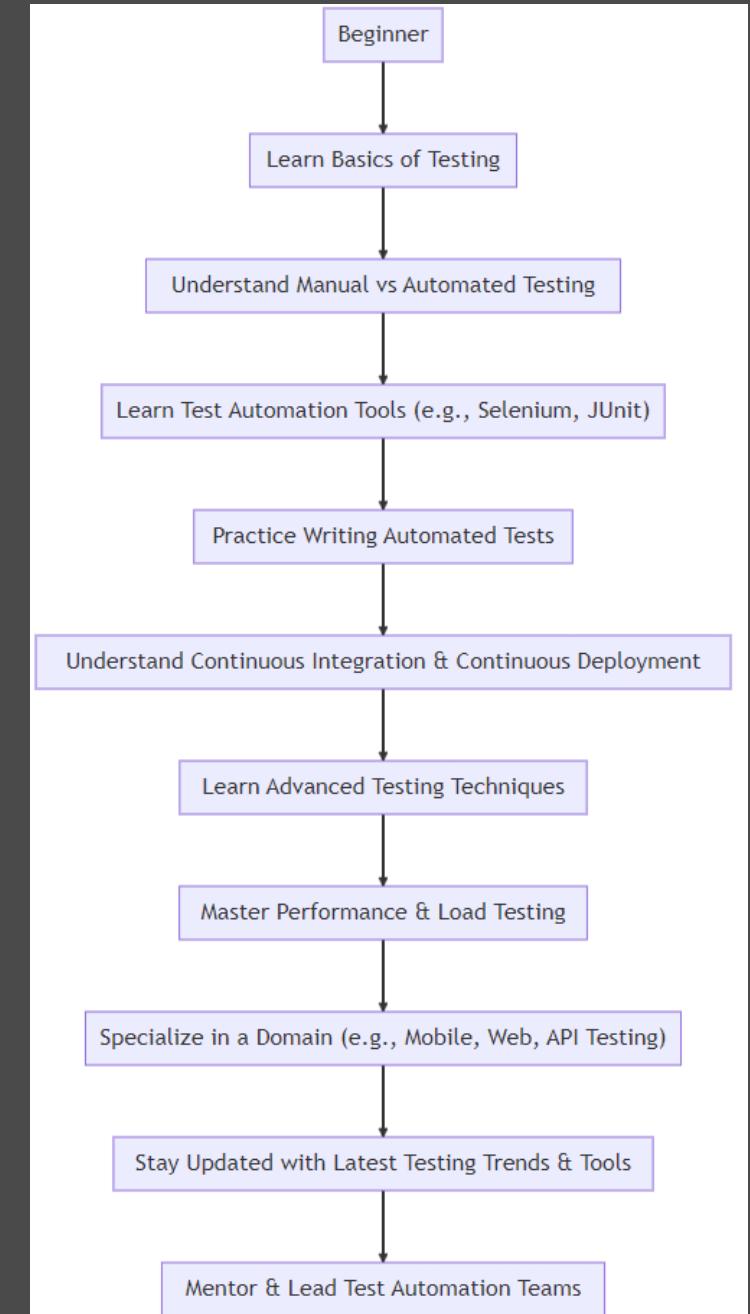
A screenshot of a digital interface, likely a software application or a web-based tool. At the top, there is a dark header bar with the text "Plugins • Enabled plugins: 🎨". Below the header, the main content area has a dark background. On the left side of this area, there is a small icon of a person with arms raised, followed by the text "< 2 / 3 >". To the right of this, there is another small icon of a person with arms raised. The main text in the center reads: "As an experienced software development mentor, draw a diagram of a test automation engineer development roadmap".

One More Persona Pattern Example

- ...with persona pattern...

A screenshot of a software application interface. At the top, there's a header with a gear icon and the text "Plugins • Enabled plugins: 5". Below the header, there's a navigation bar with arrows for navigating through tasks. The main content area contains a task card with the following details:

- Icon: A person icon.
- Text: "As an experienced software development mentor, draw a diagram of a test automation engineer development roadmap".



One More Persona Pattern Example

- ...finally asking for a more detailed response

A screenshot of a digital interface for a persona pattern example. At the top, there's a header with a gear icon, the word "Plugins", a dot separator, and "Enabled plugins" followed by a small icon. Below the header, the page number "3 / 3" is shown next to a small profile picture of a person standing outdoors. The main content area contains a task description: "As an experienced software development mentor, draw a detailed diagram of a test automation engineer development roadmap, including examples of programming languages and frameworks worth learning and links to webpages with courses and learning materials".

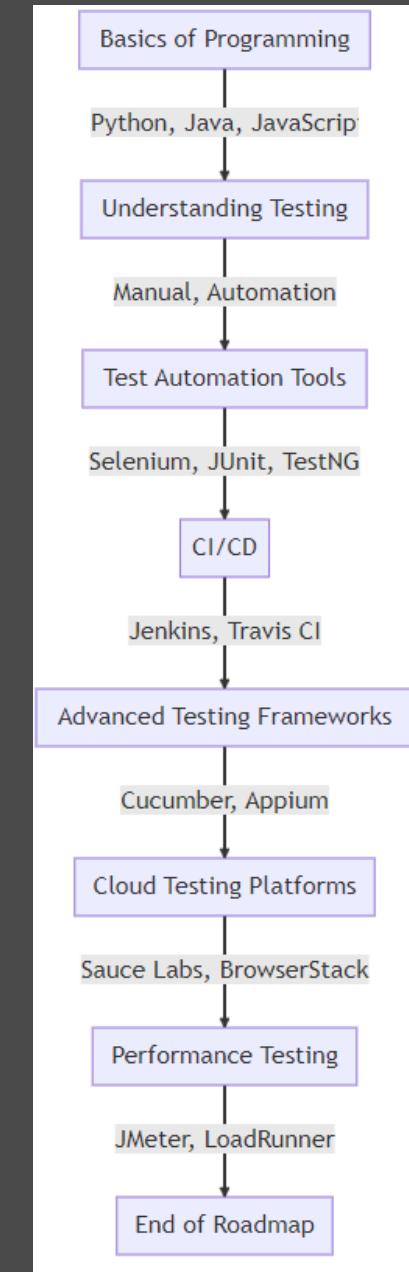
One More Persona Pattern Example

- ...finally asking for a more detailed response

Plugins • Enabled plugins: 🎨

< 3 / 3 >

As an experienced software development mentor, draw a detailed diagram of a test automation engineer development roadmap, including examples of programming languages and frameworks worth learning and links to webpages with courses and learning materials



One More Persona Pattern Example

- ...finally asking for a more detailed response

Plugins • Enabled plugins: 🎨

As an experienced software development mentor, draw a detailed diagram of a test automation engineer development roadmap, including examples of programming languages and frameworks worth learning and links to webpages with courses and learning materials

7 / 10

Sure! I'll create a detailed diagram for a test automation engineer development roadmap. Here's the breakdown:

- Basics of Programming:**
 - Languages: Python, Java, JavaScript
 - Python Course
 - Java Course
 - JavaScript Course
- Understanding Testing:**
 - Manual Testing
 - Automation Testing
 - Software Testing Course
- Test Automation Tools:**
 - Selenium, JUnit, TestNG
 - Selenium Course
 - JUnit and TestNG Course
- Continuous Integration/Continuous Deployment (CI/CD):**
 - Jenkins, Travis CI
 - Jenkins Course
 - Travis CI Course
- Advanced Testing Frameworks:**
 - Cucumber, Appium
 - Cucumber Course
 - Appium Course
- Cloud Testing Platforms:**
 - Sauce Labs, BrowserStack
 - Sauce Labs Tutorial
 - BrowserStack Tutorial
- Performance Testing:**
 - JMeter, LoadRunner
 - JMeter Course
 - LoadRunner Course

Let's visualize this roadmap in a diagram.

Send a message

46

Cognitive Verifier Pattern

- *When you are asked a question, follow these rules*
- *Generate a number of additional questions that would help more accurately answer the question*
- *Combine the answers to the individual questions to produce the final answer to the overall question*

Cognitive Verifier Pattern

* GPT-4

 When you are asked a question, follow these rules. Generate a number of additional questions that would help you more accurately answer the question. Combine the answers to the individual questions to produce the final answer to the overall question.

 Okay, I understand your instructions. Please go ahead and ask your question, and I will generate additional questions to help provide a comprehensive answer. ✖️ ⌂ ⌂

 I want to learn to code, how to do it?



Alright, in order to provide a comprehensive answer to the question "I want to learn to code, how to do it?", I would need to ask a series of more specific questions:

1. Why do you want to learn coding?

- Career change?
- Hobby or personal projects?
- Enhance current job skills?
- School or academic reasons?

2. Do you have any prior experience or background in coding?

- If yes, what languages or platforms have you used?
- How comfortable are you with computer technology in general?

3. How do you best learn new skills?

- Through structured courses or curriculums?
- By working on real-world projects?
- Through books or written material?
- By watching video tutorials?

4. What's your preferred mode of learning?

- Online or in-person?
- Self-paced or scheduled?

5. What resources are you willing to invest?

- Are you willing to spend money on courses, books, or tools?
- How much time per week can you dedicate to learning coding?

6. What specific areas of coding are you interested in?

- Web development?
- Mobile app development?
- Game development?
- Data science and machine learning?
- Embedded systems or Internet of Things?

7. What's your long-term goal with coding?

- Becoming a professional software developer?
- Building a particular application or project?
- Just gaining a general understanding?



1. To be able to start automating test as a software tester
2. None
3. I like working with a mentor or watching online tutorials
4. Online, self-paced
5. I can spend some money, but not much and I have 5 hours per week
6. Test automation
7. Becoming a professional automation tester



Thank you for providing detailed responses. Based on your answers, here's a comprehensive answer tailored to your needs:

Given that you want to learn coding to automate tests as a software tester, your journey will be focused more on understanding test automation frameworks and tools. As you have no prior coding experience, starting with a language that's commonly used in automation will be beneficial.

1. **Start with a language:** Given the popularity and ease of use, Python is an excellent choice for beginners and is widely used in the test automation field.

- Additional Questions:**

- Which Python version should I start with?** It's best to start with the latest stable version.
- Where can I find Python tutorials?** Websites like Coursera, Udemy, and YouTube have beginner-friendly Python tutorials. Ensure you choose courses tailored for complete beginners.

2. **Focus on Test Automation Libraries:** Once you have a basic understanding of Python, focus on libraries and frameworks that help with test automation.

- Additional Questions:**

- Which Python libraries are essential for test automation?** Selenium for web application testing, pytest for writing tests, and libraries like Requests for API testing.
- How do I set them up?** Installation guides and tutorials on their official websites or GitHub pages are your best starting point.

3. **Practice:** Set up a simple web application or use publicly available ones and practice writing automation scripts for them.

- Additional Questions:**

- Which websites can I use for practice?** Websites like "the-internet" by Dave Haeffner are great for practicing your automation skills.
- How do I start writing automation scripts?** Follow online tutorials specific to Selenium with Python to begin your journey.

4. **Invest in a Course or Mentorship:** Given that you prefer learning with a mentor or through online tutorials, consider investing in a course specifically on test automation with Python.

- Additional Questions:**

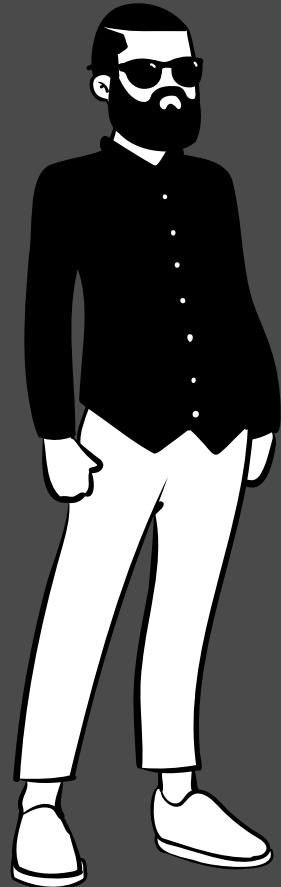
- Which platforms offer courses on test automation?** Udemy, Pluralsight, Test Automation University, and Coursera have courses focused on test automation.
- How do I find a mentor?** Platforms like Codementor or even LinkedIn can help you connect with experienced automation testers willing to mentor.

5. **Stay Updated and Network:** Join communities like Ministry of Testing, attend webinars, and engage in forums. This helps you stay updated and provides networking opportunities.

- Additional Questions:**

- Which forums or communities should I join?** Apart from Ministry of Testing, you can join Stack Overflow, Reddit's r/QualityAssurance, and TesterHome.

By combining all the answers and following this roadmap, you should be well on your way to becoming a proficient automation tester. Remember, consistent practice and real-world application of your knowledge will be key. Good luck!



Use case: Diving into Coding Details with ChatGPT 4.0

Loops





What is the difference between for and while loops?



Both `for` and `while` loops are used to repeat a block of code multiple times, but they serve different purposes and have different structures. Here's a breakdown of their differences:

1. Purpose and Usage:

- **For Loop:** Used when the number of iterations is known ahead of time. It's common to use `for` loops when iterating over a sequence (like a list or range).
- **While Loop:** Used when the number of iterations is unknown and depends on a specific condition being met.

2. Syntax:

- **For Loop** (in many languages like Python, C, Java, etc.):

```
python
for variable in sequence:
    # code to execute
```

Copy code

- **While Loop:**

```
python
while condition:
    # code to execute
```

Copy code

3. Control Mechanism:

- **For Loop:** Uses a counter or an element from a sequence to control the number of iterations.
- **While Loop:** Uses a condition to control the number of iterations. The loop will continue executing as long as the condition evaluates to `True`.

4. Example:

- **For Loop** (Iterating over a range of numbers from 1 to 5 in Python):

```
python
for i in range(1, 6):
    print(i)
```

Copy code

- **While Loop** (Printing numbers from 1 to 5 in Python):

```
python
i = 1
while i <= 5:
    print(i)
    i += 1
```

Copy code

5. Potential Pitfalls:

- **For Loop:** It's generally safer because the number of iterations is defined upfront (by the length of the sequence). There's usually less risk of accidentally creating an infinite loop.
- **While Loop:** There's a greater risk of creating an infinite loop if the condition never evaluates to `False`. It's essential to ensure that the condition can change and eventually become `False` to exit the loop.

6. Flexibility:

- **For Loop:** Its structure is more rigid, designed mainly for iterating over sequences.
- **While Loop:** It's more flexible and can handle a wider variety of scenarios since it operates based on a condition. You can even mimic a `for` loop using a `while` loop, but the opposite isn't always straightforward.

In summary, while both `for` and `while` loops are fundamental constructs for controlling the flow of programs, their use-cases, structure, and inherent risks can differ. The choice between them largely depends on the specific requirements of the task at hand.



What is the difference between for loop in Python and in C#?

Both Python and C# offer `'for'` loops, but the way they're typically used and their syntax can differ considerably. Here are some of the key differences:

1. Basic Syntax:

- **Python:**

```
python
for variable in iterable:
    # code block
```

- **C#:**

```
csharp
for (initialization; condition; iterator)
{
    // code block
}
```

3. Iterating over Collections:

- **Python** (iterating over a list, for example):

```
python
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

- **C#** (using `'foreach'` for collection iteration):

```
csharp
string[] fruits = {"apple", "banana", "cherry"};
foreach (string fruit in fruits)
{
    Console.WriteLine(fruit);
}
```

6. Flexibility:

- In Python, the `'for'` loop is primarily used to iterate over sequences like lists, tuples, dictionaries, strings, etc.
- In C#, while the `'foreach'` loop is used to iterate over collections, the traditional `'for'` loop provides a bit more flexibility in terms of controlling the initialization, condition, and iteration expressions.

7. Type Declaration:

- **Python:** Python is dynamically typed, so you don't need to declare the type of the loop variable explicitly.
- **C#:** C# is statically typed, so in traditional `'for'` loops and `'foreach'` loops, you typically need to declare the type of the loop variable (unless you're using the `'var'` keyword for type inference).

4. Control Variables:

- In Python, the control variable in a `'for'` loop is not required to be initialized before the loop; it's created when the loop starts.
- In C#, if you're using the traditional `'for'` loop (not `'foreach'`), you usually initialize the control variable as part of the loop declaration.

5. Scope of Loop Variable:

- **Python:** The loop variable retains its value even after the loop ends and it's accessible outside of the loop.

```
python
for i in range(5):
    pass
print(i) # Outputs: 4
```

- **C#:** The loop variable's scope is limited to the loop. If you try to access it outside the loop, you'll get an error.

```
csharp
for (int i = 0; i < 5; i++)
{
    // ...
}
// Console.WriteLine(i); // This would be an error.
```

8. Performance:

- **C#:** The traditional `'for'` loop in C# can be faster than Python's `'for'` loop due to the compiled nature of C# versus the interpreted nature of Python, especially when iterating over a range of numbers or arrays. However, the actual performance can vary based on many factors.
- **Python:** While Python's `'for'` loop might not match C#'s performance, the simplicity and ease of use make it a popular choice among Python developers.

In conclusion, while both Python and C# offer `'for'` loops, they have differences in syntax, usage, and features. It's essential to understand these differences when transitioning between these languages or comparing their capabilities.

Page Object Model





Explain Page Object Pattern like I'm new to automated testing



Alright! Imagine you're putting together a play (a theatrical performance). Each scene has props, characters, and specific actions that characters do. If you had to describe each scene, you'd mention where each prop is located, which characters are present, and what they're doing.

In the world of automated testing, particularly for web applications, each "scene" can be thought of as a different page or view in your application. Each page has its own elements (like buttons, text boxes, and labels) and actions (like clicking a button or entering text).

The Page Object Pattern is a way to model these pages, just like you'd describe scenes in your play. Instead of scattering the specifics about how to interact with a page throughout your tests, you create a separate object, a "Page Object", for each page or view. This object knows about the elements on that page and how to interact with them.

Here's a breakdown:

1. What is a Page Object?

A Page Object represents a page (or part of it) in your application. It contains attributes to find and interact with elements on that page, and methods that define the actions you can take on the page.

2. Why use the Page Object Pattern?

- **Maintainability:** If something changes in the structure of your web page, you only have to update the corresponding Page Object, not every single test that interacts with that page.
- **Readability:** By using descriptive method names in your Page Object, your tests read more like human-friendly stories, rather than a bunch of technical code.
- **Reusability:** Once you've created a Page Object, you can use it in many tests, reducing the amount of repetitive code.

3. How does it work?

Let's imagine a login page of a web application. Instead of writing a test script that directly interacts with the username field, password field, and login button, you'd create a `LoginPage` object.

This `LoginPage` object might have methods like `enterUsername()`, `enterPassword()`, and `clickLoginButton()`. Your test script will then use this object and its methods to perform the test, like so:

python

```
login_page = LoginPage(driver)
login_page.enterUsername("myUsername")
login_page.enterPassword("myPassword")
login_page.clickLoginButton()
```

Copy code

Notice how this makes the test script more readable and focused on the high-level actions rather than the specifics of element selection and interaction.

4. Adaptable to Changes:

Now, let's say the developers change the ID of the login button. Without the Page Object Pattern, you'd have to scour your tests and update each instance where you referenced that button. With the pattern, you simply update the button reference in the `LoginPage` object, and all the tests using it automatically adapt.

In summary, think of the Page Object Pattern as a way to organize your play (tests) by describing each scene (web page) in a dedicated script (Page Object). This makes the play easier to read, update, and maintain.



Explain Page Object Pattern like I'm an experienced automation tester



Alright! Given that you're an experienced automation tester, you likely understand the challenges that come with maintaining and scaling test automation suites, especially when they deal with complex user interfaces.

The **Page Object Pattern** is a design pattern that seeks to address these challenges, mainly by promoting code reusability and maintainability.

1. Abstraction of UI Interaction:

The primary idea behind the Page Object Pattern is to abstract the UI interactions and representations of a web page (or part of it) into a separate class, known as a "Page Object".

This class will contain all the locators (like IDs, XPaths, CSS Selectors) and methods for interactions (like click, input text, retrieve text) specific to that page.

2. Encapsulation:

Every detail about how the page is interacted with (locators, UI details, etc.) is encapsulated within the Page Object. This ensures that if there's a change in the UI, you only need to update the Page Object and not all the test cases that interact with it.

3. Code Reusability:

When you want to interact with a particular page in multiple test cases, you don't need to rewrite the interaction logic. Instead, you instantiate the Page Object and call its methods.

4. Cleaner Test Scripts:

Your actual test scripts become more human-readable because they'll look more like business flows and less like low-level UI interactions. For instance, instead of:

python

```
driver.find_element(By.ID, "username").send_keys("myUsername")
driver.find_element(By.ID, "password").send_keys("myPassword")
driver.find_element(By.ID, "login").click()
```

Copy code

You'll have something like:

python

```
loginPage.enterUsername("myUsername")
loginPage.enterPassword("myPassword")
loginPage.clickLoginButton()
```

Copy code

5. Enhanced Collaboration:

Even testers or developers who aren't familiar with the specifics of the UI can still work with the test cases, because the logic is abstracted behind clear method names in the Page Object.

6. Separation of Concerns:

The Page Object Pattern aligns well with the principle of separation of concerns. Your Page Objects deal with "how" an operation is performed, while the test scripts deal with "what" operations to perform.

7. Consistent Locator Strategy:

Because locators are centralized within Page Objects, you can ensure consistency in how elements are located, making the process more efficient and less error-prone.

Note: While the Page Object Pattern provides numerous advantages, it's not a silver bullet.

For very large applications, a single Page Object per page might become too large. In such cases, you might consider breaking it down further into smaller, more manageable components, or consider other design patterns like Page Component Pattern.

Given your experience in automation testing, you'll appreciate that the Page Object Pattern isn't just about organizing code – it's about fostering collaboration, promoting good coding practices, and ensuring that as your application evolves, your test suite can evolve with it in a maintainable way.



Use case: **Improving Coding Skills with ChatGPT 4.0**

Codewars

The screenshot shows the Codewars dashboard at codewars.com/dashboard. The user is in the 'Psychologia' category. The dashboard features a sidebar with sections for Home, TRAINING (Practice, Freestyle Sparring), CAREER (Opportunities), and COMMUNITY (Leaderboards, Chat, Discussions). A central area displays a 'Suggested Challenge' for '7kyu' level in Python Fundamentals. The challenge is titled 'Binary Addition' and describes implementing a function that adds two numbers together and returns their sum in binary. It includes examples: (Input1, Input2 → Output (explanation)). Below the challenge are promotional cards for upgrading to Codewars Red, earning honor, and GitLab ads.

Home
Report home for your next assignment

IT Psychologia

TRAINING

Practice Complete challenging Kata to earn honor and ranks. Re-train to hone technique

Freestyle Sparring Take turns remixing and refactoring others code through Kumite

CAREER

Opportunities Find your next career challenge - powered by Andela

COMMUNITY

Leaderboards Achieve honor and move up the global leaderboards

Chat Join our Discord server and chat with your fellow code warriors

Discussions View our GitHub Discussions board to discuss general Codewars topics

ABOUT

Docs Learn about all of the different aspects of Codewars

Suggested Challenge

7kyu Binary Addition

Implement a function that adds two numbers together and returns their sum in binary. The conversion can be done before, or after the addition.

The binary number returned should be a string.

Examples:(Input1, Input2 → Output (explanation))

TRAIN SKIP

BINARY FUNDAMENTALS

Upgrade To Codewars Red!

Support Codewars and get some fancy upgrades like Pro Stats, Head-to-head comparisons, No ads and more.

LEARN MORE

8 Reasons Why Codewarriors Practice Coding with Codewars

Not everyone trains the same. Discover new ways to leverage Codewars in your education and career.

Earn extra honor and gain new allies!

Honor is earned for each new codewarrior who joins.

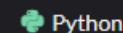
LEARN MORE

GitLab is the only place where enterprises build mission-critical software.

Ads via Carbon

8kyu A wolf in sheep's clothing

☆ 446 🏷 131 📈 91% of 3,199 📈 9,702 of 29,327 🎙 jkempster34



Python

Details

Solutions

Forks (5)

Discourse (91)

Translations

DESCRIPTION:

Wolves have been reintroduced to Great Britain. You are a sheep farmer, and are now plagued by wolves which pretend to be sheep. Fortunately, you are good at spotting them.

Warn the sheep in front of the wolf that it is about to be eaten. Remember that you are standing **at the front of the queue** which is at the end of the array:

[sheep, sheep, sheep, sheep, sheep, wolf, sheep, sheep] (YOU ARE HERE AT THE FRONT OF THE QUEUE)
7 6 5 4 3 2 1

If the wolf is the closest animal to you, return "Pls go away and stop eating my sheep". Otherwise, return "Oi! Sheep number N! You are about to be eaten by a wolf!" where N is the sheep's position in the queue.

Note: there will always be exactly one wolf in the array.

Examples

Input: ["sheep", "sheep", "sheep", "wolf", "sheep"]

Output: "Oi! Sheep number 1! You are about to be eaten by a wolf!"

Input: ["sheep", "sheep", "wolf"]

Output: "Pls go away and stop eating my sheep"

VIEW

- All
- Following
- Mine
- Invalidated ?

SORT BY

- Best Practices
- Clever
- Newest
- Oldest

by CopperWye, Shanmuga-raj, Diwya Ganesan, matekaszas, phil@gridvisi.com, RolandMarton, SakoSucks, Stevee1715, PaddyKaspar, maamerio24 (+ 23)

```
def warn_the_sheep(queue):
    n = len(queue) - queue.index('wolf') - 1
    return f'Oi! Sheep number {n}! You are about to be eaten by a wolf!' if n else 'Pls go away and stop eating my sheep'
```

11 similar code variations are grouped with this one ⓘ

[^ Best Practices 146](#) | [^ Clever 239](#) | [💬 13](#) | [🔗 1](#) | [Fork](#) | [Compare with your solution](#) | [Link](#)

by falsetru, AlexTelon, toma95, Toxic_Stark, Viktor Volkov, Truther24, cristimihescu, onetraul, RustambekSafarov, Sarayjm (+ 1)

```
def warn_the_sheep(queue):
    i = queue[::-1].index('wolf')
    if i == 0:
        return 'Pls go away and stop eating my sheep'
    return f'Oi! Sheep number {i}! You are about to be eaten by a wolf!'
```

6 similar code variations are grouped with this one ⓘ

[^ Best Practices 68](#) | [^ Clever 42](#) | [💬 4](#) | [Fork](#) | [Compare with your solution](#) | [Link](#)

by Oleg_, sobakagavgav, narancs, metin-akyol, Yeongho, AAlp6, soma03, bruhaps, OleksiiFedinchyk

```
def warn_the_sheep(queue):
    queue.reverse()
    return 'Pls go away and stop eating my sheep' if queue[0] == 'wolf' else 'Oi! Sheep number {}! You are about to be eaten by a wolf!'.format(queue.index('wolf'))
```

4 similar code variations are grouped with this one ⓘ

[^ Best Practices 24](#) | [^ Clever 33](#) | [💬 2](#) | [Fork](#) | [Compare with your solution](#) | [Link](#)



Solve this coding task using Python and the most efficient solution:



Here's a Python function that accomplishes this:

0 ⌟ ⌚

python

Copy code

```
def warn_the_sheep(queue):
    # The reversed() function will give us the elements in the list from right
    reversed_queue = reversed(queue)
    position = next(i for i, animal in enumerate(reversed_queue) if animal == "wolf")

    # If the wolf is the closest animal to you.
    if position == 0:
        return "Pls go away and stop eating my sheep"
    else:
        # Subtracting 1 from the position since position starts at 0.
        return f"Oi! Sheep number {position}! You are about to be eaten by a wolf!"

# Test
print(warn_the_sheep(["sheep", "sheep", "sheep", "wolf", "sheep"]))
print(warn_the_sheep(["sheep", "sheep", "wolf"]))
```

When you run the above code, it will give you the desired outputs for the provided examples.



Give me alternative solutions to this task



Certainly! Let's look at a couple of alternative approaches to the problem:



Approach 1: Using `index()`

Using the `index()` method of lists, we can directly find the position of the "wolf". Then, using the total length of the list, we can calculate the sheep's position in front of the wolf.

```
python Copy code
def warn_the_sheep(queue):
    wolf_position = queue.index("wolf")

    distance_from_front = len(queue) - 1 - wolf_position

    if distance_from_front == 0:
        return "Pls go away and stop eating my sheep"
    else:
        return f"Oi! Sheep number {distance_from_front}! You are about to be
```

Approach 2: Using `for` loop

By iterating through the list from the end, we can find the position of the wolf and then calculate the sheep's position accordingly.

```
python Copy code
def warn_the_sheep(queue):
    for i in range(len(queue)-1, -1, -1):
        if queue[i] == "wolf":
            distance_from_front = len(queue) - 1 - i
            if distance_from_front == 0:
                return "Pls go away and stop eating my sheep"
            else:
                return f"Oi! Sheep number {distance_from_front}! You are abo
```

Each of these approaches provides a valid solution to the problem and you can choose based on your preference. The first approach (using `index()`) is probably the most concise and clear.



Solve this task using only three lines of code



Certainly! Using Python's list methods and conditional expression, here's the task solved in just three lines:

python

Copy code

```
def warn_the_sheep(queue):
    n = len(queue) - 1 - queue.index("wolf")
    return "Pls go away and stop eating my sheep" if n == 0 else f"Oi! Sheep
```

This solution utilizes the `index()` method to find the wolf and calculates the position of the sheep in front of the wolf with a simple subtraction. It then uses a conditional expression to determine the correct message to return.

Better Way to Ask?

GPT-4



As a seasoned coding expert and clean code proficient programmer, solve this coding task using Python and the most efficient solution:



Alright! Let's solve this problem.



To find the position of the wolf, you can use the `'index'` method of a list. Once you find the wolf's position, you can determine the position of the sheep that is in front of it. Here's how you can do it:

1. Reverse the list to simulate the view from the front of the queue.
2. Find the position of the wolf.
3. Check if the wolf is at the beginning of the reversed list (i.e., the closest to you in the original setup). If so, return the specific message.
4. Otherwise, return the message indicating which sheep is about to be eaten.

Here's the Python code to achieve this:

Better Way to Ask?

Here's the Python code to achieve this:

```
python                                Copy code

def warn_the_sheep(queue):
    reversed_queue = queue[::-1]
    wolf_position = reversed_queue.index('wolf')

    if wolf_position == 0:
        return "Pls go away and stop eating my sheep"
    else:
        return f"Oi! Sheep number {wolf_position}! You are about to be eaten"

# Test cases
print(warn_the_sheep(["sheep", "sheep", "sheep", "wolf", "sheep"])) # "Oi!
print(warn_the_sheep(["sheep", "sheep", "wolf"])) # "Pls go away and stop e
```

This solution should work efficiently for the given problem.

Better Way to Ask?



A seasoned developer would do that challenge in 2-3 lines of code.



You're right. We can make the solution more concise by using Python's list comprehension and string formatting capabilities. Here's a more succinct solution:

python

Copy code

```
def warn_the_sheep(queue):
    n = queue[::-1].index('wolf')
    return "Pls go away and stop eating my sheep" if n == 0 else f"Oi! Sheep
```

This concise version essentially performs the same operations but is condensed into fewer lines for brevity.



Use case: Technical Interview with ChatGPT 4.0

Audience Persona Pattern

- Assume that I am X

GPT-4



Assume that I am a mid software automation tester. Lead a technical interview with me



Certainly! I will assume the role of the interviewer and lead a technical interview tailored towards a mid-level software automation tester. Here we go:



Interviewer: Good morning/afternoon! Thank you for taking the time to interview with us today. I hope you're doing well. Before we dive into the technical questions, can you briefly walk me through your experience



I have four years of overall QA experience and two years of experience as a test automation engineer. I've coded with C# and JavaScript, using tools such as NUnit, Playwright, Selenium



Interviewer: Great! Thanks for sharing that. Let's delve deeper into some specifics:



1. **Tools and Frameworks:** You mentioned you've used NUnit, Playwright, and Selenium.

- Can you explain the main differences between Playwright and Selenium in terms of their capabilities and scenarios where you'd choose one over the other?
- In your experience, what are some of the challenges you've faced with Selenium and how did you overcome them?

2. **Programming and Scripting:**

- Since you've worked with both C# and JavaScript, can you explain a situation where you chose to use one over the other for a specific automation task? What were your considerations?
- Could you walk me through a situation where you had to debug a particularly tricky automation script? What was the problem and how did you go about troubleshooting it?

3. **Test Strategy and Approach:**

- How do you prioritize which test cases to automate and which ones to leave for manual testing?
- Describe a situation where an automation script you wrote failed, but not because of an issue in the application under test. How did you handle that?

4. **Continuous Integration & Continuous Deployment (CI/CD):**

- Do you have any experience integrating your automation scripts into a CI/CD pipeline? If yes, can you describe the tools and processes you've worked with?
- How do you ensure that your automation scripts remain robust and reliable as the application changes over time?

5. **Collaboration and Communication:**

- Automation doesn't exist in a vacuum. Can you share an experience where you had to collaborate closely with developers, business analysts, or other stakeholders? How did you handle differences in opinion or any challenges?
- When a test fails, how do you communicate the issues to developers or stakeholders? Do you follow any specific format or method?

6. **Looking to the Future:**

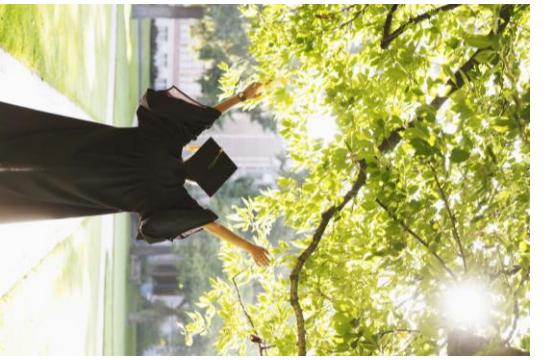
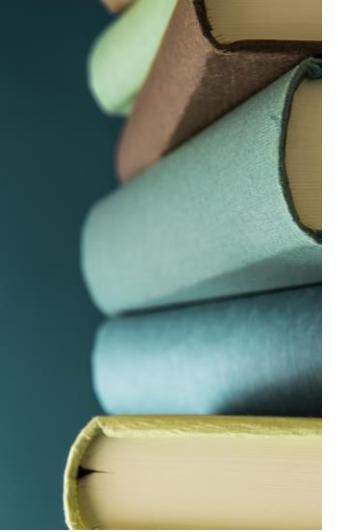
- In the constantly evolving field of automation testing, how do you keep your skills current? Are there any new tools or technologies you're excited about or keen to learn?
- How do you envision the future of automation testing? What changes or advancements do you foresee?

Interviewer: Before we conclude, do you have any questions for me about the role, our company, or the team you'd potentially be working with?

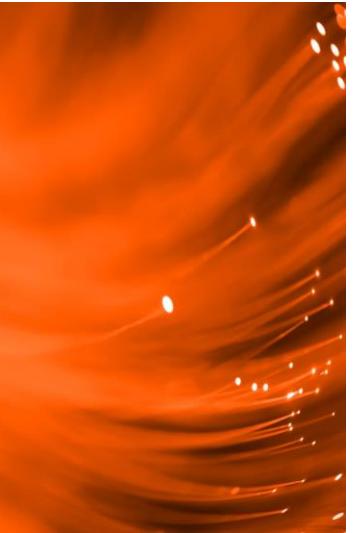
(Note: This is just a broad template for the interview. Depending on the responses given, an interviewer would likely follow-up with deeper, more probing questions on specifics.)

Patterns Summary

- Persona
- Audience Persona
- Question Refinement
- Cognitive Verifier
- ...and many more!



フ / ツ



JIT

https://amelia.qa

<https://jit.team>

JIT