Cloud engineer interview questions.

1. Question: Can you explain the role of Infrastructure as Code (IAC) in cloud engineering, and how have you used tools like Terraform and AWS CloudFormation in your previous work?

Answer: Infrastructure as Code (IAC) is a practice of managing and provisioning infrastructure using code. It allows us to define and automate the creation and configuration of cloud resources. In my previous role, I utilized Terraform and AWS CloudFormation extensively to create, modify, and maintain cloud components. This approach ensures consistency, scalability, and automation in managing infrastructure, which is crucial for cloud engineering.

2. Question: Describe your experience with cloud cost optimization. How do you ensure that cloud components are optimized for cost without sacrificing performance and reliability?

Answer: Cloud cost optimization is vital. I've worked on optimizing cloud components by regularly monitoring resource usage, rightsizing instances, using reserved instances, and leveraging auto-scaling where applicable. Additionally, I've used tools like AWS Cost Explorer to identify cost trends and anomalies. My approach is to strike a balance between cost savings and maintaining performance and reliability.

3. Question: Can you explain how you've ensured high availability and reliability in the cloud infrastructure you've managed?

Answer: High availability and reliability are achieved through multiple means. This includes architecting for fault tolerance, using multi-AZ deployments, load balancing, implementing backup and disaster recovery strategies, and setting up comprehensive monitoring and alerting systems. Regularly reviewing and improving infrastructure and conducting incident reviews are also essential.

4. Question: How do you handle monitoring and incident response in a 24x7 operational environment?

Answer: In a 24x7 operational environment, I prioritize proactive monitoring to identify issues before they impact users. I set up robust alerting mechanisms using CloudWatch and custom scripts. When an incident occurs, I follow incident response procedures, diagnose the issue, and work to resolve it. I also keep detailed incident documentation for post-incident analysis.

In a recent experience, our production environment experienced a sudden spike in traffic. The database server became unresponsive, causing downtime.

To address it:

\* First, I identified the issue by examining logs, monitoring metrics, and error reports.

\* I determined that a poorly optimized database query was causing the bottleneck.

\* I temporarily modified the application configuration to mitigate the query issue.

\* Concurrently, I optimized the database queries, added indexing, and performed a rolling update.

\* Within a critical 30-minute window, we had the database and application back to normal, minimizing downtime and maintaining a positive user experience.

5. Question: Explain your experience with integrating cloud services with on-premises systems or third-party APIs.

Answer: I've integrated cloud services with on-premises systems and third-party APIs using various methods, such as VPN, Direct Connect, and API Gateway. It's crucial to ensure secure and seamless communication between cloud and on-premises environments while adhering to best practices and security standards.

6. Question: How have you contributed to the Agile and Scrum methodologies in your previous roles as a Cloud Engineer?

Answer: I've actively participated in Agile and Scrum methodologies by working in cross-functional teams, attending daily standup meetings, and adhering to sprint planning and reviews. I've also used tools like Jira and Confluence for Agile project management. This ensures efficient collaboration and delivery of cloud-related projects.

7. Question: Can you describe your experience with security best practices in the cloud environment?

Answer: Security is a top priority. I've implemented security best practices by securing IAM policies, configuring VPCs with proper security groups and NACLs, enabling encryption at rest and in transit, and setting up WAF for web application security. Regular security audits, compliance checks, and staying updated on security threats are also part of my approach.

8. Question: What automation and productivity tools have you used in your cloud engineering work?

Answer: I've utilized various automation tools like Jenkins for CI/CD, Ansible for configuration management, and custom scripts for automating routine tasks. For productivity, I've used tools like Slack and collaboration platforms to streamline communication and documentation. These tools improve efficiency and collaboration in cloud engineering.

9. Question: How do you stay updated with the latest cloud and DevOps technologies and best practices?

Answer: Staying updated is crucial. I regularly read industry blogs, attend webinars, and participate in AWS and DevOps communities. Additionally, I've pursued relevant certifications, like AWS Certified DevOps Engineer, to ensure that my skills and knowledge are up-to-date.

10. Question: Describe a challenging incident you've encountered while managing cloud infrastructure. How did you identify and resolve the issue?

Answer: In a past role, we faced a database performance issue that caused slow application response times. I used CloudWatch metrics to identify the problem, pinpointing high CPU usage on the RDS instance. We increased the instance size, optimized queries, and implemented read replicas, resulting in a significant performance improvement.

>11. Question: Could you explain how you handle code deployments and updates in a production environment?

Answer: Code deployments are critical. We follow blue-green deployment practices, ensuring a safe transition. I create deployment pipelines with tools like AWS CodePipeline and monitor the process closely. If issues arise, we can quickly roll back to the previous version to minimize user impact.

12. Question: What's your experience with scaling cloud resources based on demand? How do you ensure scalability and performance in dynamic environments?

Answer: I've worked on auto-scaling configurations, setting up policies based on metrics like CPU and traffic. Additionally, I utilize AWS Elastic Load Balancers for distributing traffic. This ensures that resources automatically adjust to handle varying workloads while maintaining performance.

13. Question: Can you discuss your disaster recovery and backup strategies for cloud infrastructure?

Answer: Disaster recovery is an ongoing process that requires planning, regular testing, and adaptation to changes in your infrastructure. I've implemented regular backups, cross-region replication, and automated snapshots for critical resources. Also we can implement AWS services like AWS Lambda and AWS Step Functions to automate and orchestrate the failover process. We regularly perform disaster recovery testing to ensure data and services can be quickly restored in case of failures.

In my previous role as a DevOps Engineer, one critical task was to ensure the backup and restore of large databases. The manual process was cumbersome, time-consuming, and prone to human errors. I identified the need to automate this process for efficiency and reliability.

Automation Script:

I developed a comprehensive Python script to automate the entire backup and restore workflow for large databases. Here's a breakdown of how the script worked:

\* Backup Scheduling: The script allowed the scheduling of regular database backups, ensuring data was consistently protected. It could be configured to run daily, weekly, or on a custom schedule.

\* Data Compression: Before storing backups, the script compressed the data, reducing storage requirements and improving transfer speed. This was especially crucial for large databases.

\* Backup Execution: It initiated the backup process using the database's native tools, such as mysqldump for MySQL or pg\_dump for PostgreSQL. The script handled the complex command-line arguments and options, ensuring a consistent backup process.

\* Storage Management: Backups were stored in a structured directory hierarchy, making it easy to locate and manage them. Old backups were automatically pruned based on retention policies.

\* Monitoring and Alerting: The script integrated with our monitoring system to provide alerts in case of backup failures or issues. This allowed for timely intervention and resolution.

\* Restoration Process: In the event of a database failure, the script could initiate the restoration process. It selected the appropriate backup based on timestamps and restored the database to the desired state.

14. Question: How do you ensure compliance and governance in the cloud environment, especially when working with sensitive data?

Answer: Compliance and governance are achieved through strict IAM policies, resource tagging, and using AWS Config for rule-based compliance checks. I ensure data encryption, audit trails, and access controls are in place. Regular audits and compliance reporting are part of our process.

15. Question: Can you provide an example of a complex automation task you've performed using scripts or automation tools?

Answer: In a recent project, I automated the process of provisioning and de-provisioning user accounts based on HR data. I developed Python scripts and integrated them with AWS Lambda to create a fully automated onboarding and off boarding process, saving time and reducing manual errors.

16. Question: How do you handle capacity planning in the cloud, especially when dealing with resource-intensive applications?

Answer: Capacity planning involves monitoring resource utilization trends and forecasting future needs. I use AWS CloudWatch insights and AWS Trusted Advisor for resource recommendations. Additionally, we regularly review application performance and scaling policies to ensure we meet demand efficiently.

17. Question: Describe a situation where you had to improve the security of a cloud environment. What steps did you take, and what was the outcome?

Answer: In a previous role, I detected vulnerabilities in an S3 bucket's permissions that could have led to data exposure. I immediately updated the bucket policies, restricted access, and implemented encryption. Regular security audits afterward showed no further issues.

18. Question: How do you keep documentation up-to-date and organized for a complex cloud infrastructure?

Answer: Documentation is crucial for efficient operations. I use Confluence and Wikis to maintain comprehensive documentation. It includes architecture diagrams, configuration details, disaster recovery plans, and troubleshooting guides. Regular reviews and updates ensure accuracy.

19. Question: Can you provide an example of a successful cost-saving initiative you implemented in a cloud environment?

Answer: I initiated an AWS Reserved Instance (RI) purchase strategy that aligned with our usage patterns. This resulted in a 20% reduction in monthly EC2 costs while maintaining performance. Proper tagging and tracking usage patterns are crucial for effective cost-saving initiatives.

20. Question: How do you manage and lead a team in a cloud engineering role, especially in a 24x7 operational environment?

\*Answer: Effective team leadership in a 24x7 environment involves clear communication, delegation, and setting expectations. I ensure that roles and responsibilities are well-defined, have a rotation schedule for on-call duties, and provide training and support to team members to maintain a high level of service reliability.

21. Question: Can you explain how you would design a fault-tolerant architecture for a critical application in the cloud?

Answer: A fault-tolerant architecture involves redundancy and resilience. I would use multiple Availability Zones (AZs) to distribute the workload. Implementing auto-scaling and utilizing features like AWS Elastic Load Balancers ensures that traffic is directed away from failing instances. Frequent testing and failover drills ensure that the system can withstand various failures.

22. Question: Describe your experience with cost optimization in a cloud environment. How do you ensure efficient resource utilization while controlling expenses?

Answer: I actively monitor and analyze AWS Cost Explorer data to identify opportunities for optimization. Leveraging AWS Trusted Advisor recommendations, I review resource utilization and adjust instance types or utilization accordingly. Rightsizing resources, implementing scheduling, and setting up alerts for cost overruns help control expenses.

23. Question: How do you approach ensuring the security of cloud resources when using third-party applications or services in your infrastructure?

Answer: Security in third-party applications is crucial. I follow best practices for secure configurations and access control. Regularly reviewing permissions and performing audits is essential. In addition, I work closely with vendors to understand their security controls and collaborate on maintaining a secure environment.

24. Question: Explain how you handle disaster recovery and business continuity in multi-cloud or hybrid cloud environments.

Answer: Disaster recovery in multi-cloud or hybrid environments requires a well-defined strategy. I set up data replication and backup across cloud providers, ensuring data redundancy. Additionally, I create playbooks for failover procedures and regularly conduct tests to validate recovery processes for seamless business continuity.

25. Question: Can you discuss your experience with orchestrating containerized applications in cloud environments?

Answer: I've worked extensively with container orchestration platforms like Amazon EKS and Kubernetes to manage containerized applications. This includes deploying, scaling, and maintaining containers. I ensure security and networking considerations are addressed to support containerized workloads.

26. Question: In the context of DevOps practices, how do you facilitate collaboration between development and operations teams in a cloud environment?

Answer: Collaboration between teams is vital for successful DevOps. I establish continuous integration and continuous delivery pipelines (CI/CD) to automate code deployment, fostering a DevOps culture. Regular cross-functional meetings and sharing of metrics promote transparency and cooperation.

27. Question: Describe a situation where you had to troubleshoot a complex cloud-related issue. What tools and methods did you employ to diagnose and resolve the problem?

Answer: I encountered a latency issue affecting application performance. Using AWS CloudWatch, I analyzed metrics and logs to pinpoint the root cause. I employed networking diagnostic tools and ran traceroutes to identify bottlenecks. The issue was traced to a misconfigured security group rule, which I promptly corrected.

28. Question: Can you share your experience in managing databases in the cloud, including backup, replication, and optimization?

Answer: Managing cloud databases involves thorough backup and recovery strategies. I utilize Amazon RDS for database management, enabling automated backups and multi-AZ replication for high availability. Regular performance tuning and query optimization are essential to maintain efficient database operations.

29. Question: How do you stay updated with the latest cloud technology trends and best practices in cloud engineering?

Answer: I regularly attend AWS webinars, participate in online forums, and follow industry publications. Additionally, I engage in AWS re:Invent events, which provide insights into upcoming features and best practices. I am committed to continuous learning to stay current with evolving cloud technologies.

30. Question: Can you describe your approach to capacity planning when working with cloud resources and infrastructure?

Answer: Capacity planning involves forecasting future resource needs. I analyze historical data, including traffic patterns and performance metrics, to predict demand. Using AWS Trusted Advisor and CloudWatch, I set up alerts to trigger scaling actions when needed. This ensures that we maintain optimal resource allocation.

31. Question: Can you explain the principles of blue-green deployment and how you would implement it in an AWS environment?

Answer: Blue-green deployment involves creating two identical environments (blue and green) and switching between them during deployment. In AWS, this can be achieved by using Elastic Beanstalk or Auto Scaling Groups. Traffic shifts from one environment to the other, allowing for a seamless rollback if issues arise. The process can be automated with infrastructure as code tools like CloudFormation or Terraform.

32. Question: Describe your experience with serverless computing. When would you recommend using AWS Lambda functions, and what are the advantages and limitations of this approach?

Answer: AWS Lambda is ideal for event-driven, short-lived tasks. I've used Lambda to process data, trigger functions based on events, and run serverless applications. The advantages include cost savings, automatic scaling, and simplified maintenance. However, it's not suitable for long-running processes or resource-intensive workloads.

33. Question: Can you elaborate on your experience with AWS Elastic Container Service (ECS) and Kubernetes (EKS)? What are the key differences, and how do you choose between them for container orchestration?

Answer: ECS is an AWS-native container orchestration service, while EKS is a managed Kubernetes service. ECS is simpler to set up and manage, making it a better choice for quick deployments. EKS offers more flexibility and is suitable for complex, multi-cloud, or hybrid cloud scenarios. The choice depends on the project's requirements.

34. Question: How do you handle high availability and disaster recovery strategies for multi-region deployments in AWS?

Answer: Achieving high availability and disaster recovery across multiple AWS regions requires a strategy like active-active or active-passive architecture. Data replication, traffic routing, and automation with services like Route 53, CloudFront, and AWS Global Accelerator play key roles in ensuring failover and minimal downtime. Regular cross-region testing is also crucial.

35. Question: Explain the principles and best practices for securing AWS resources. How would you secure sensitive data stored in AWS, and what are some key security services and tools you'd utilize?

Answer: AWS security involves applying the principle of least privilege, using IAM policies, and enabling MFA for accounts. For securing data, I'd use AWS Key Management Service (KMS) for encryption. Additional tools include AWS WAF for web application security, AWS Config for compliance monitoring, and AWS Inspector for vulnerability assessment. Regular audits and security patches are essential.

36. Question: Can you describe your experience with managing serverless applications and functions at scale? What challenges have you encountered, and how did you address them?

Answer: Managing serverless applications at scale requires optimizing function performance, handling cold starts, and managing dependencies. To address these challenges, I use techniques like provisioned concurrency, warming strategies, and optimizing code. Implementing monitoring and alerting ensures proactive response to issues.

37. Question: How do you automate the provisioning and management of cloud resources using infrastructure as code? Can you provide an example of a complex infrastructure deployed through IaC?

Answer: I leverage Terraform and AWS CloudFormation for IaC. I've automated the deployment of multi-tier applications with IaC, including VPCs, subnets, EC2 instances, RDS databases, and load balancers. This ensures consistent and repeatable resource creation. Infrastructure changes are tracked in version control, improving transparency and collaboration.

38. Question: Describe a situation where you had to optimize cloud resources for both performance and cost efficiency. What strategies and AWS services did you employ to achieve this balance?

Answer: I encountered an environment with underutilized EC2 instances. By right-sizing instances based on utilization metrics from CloudWatch, I achieved cost savings. I also implemented Auto Scaling and spot instances to maintain performance during traffic spikes while reducing costs. Continuous monitoring and optimization practices helped strike the balance.

39. Question: As a Lead Cloud Engineer, how do you stay updated with AWS service releases and updates? Can you share an example of a recent AWS feature that you found valuable for your projects?

Answer: I stay updated through AWS blogs, webinars, and announcements. For example, I found AWS Lambda Extensions to be valuable. They enable easier integration of third-party monitoring and security tools with Lambda functions, enhancing observability and security without extra effort in code.

40. Question: Can you provide an overview of your disaster recovery architecture and incident response plan? How do you ensure rapid recovery and minimal data loss in case of an outage?

Answer: My disaster recovery architecture involves multi-region data replication, automated failover, and backup strategies. Regularly tested runbooks outline incident response procedures, including the role of each team member. Data loss is minimized by ensuring frequent backups and replication consistency checks.

Situational questions based on past work experience:

1.Can you describe a challenging situation that you faced in you career.

In a recent experience, our production environment experienced a sudden spike in traffic. The database server became unresponsive, causing downtime.

To address it:

\* First, I identified the issue by examining logs, monitoring metrics, and error reports.

\* I determined that a poorly optimized database query was causing the bottleneck.

\* I temporarily modified the application configuration to mitigate the query issue.

\* Concurrently, I optimized the database queries, added indexing, and performed a rolling update.

\* Within a critical 30-minute window, we had the database and application back to normal, minimizing downtime and maintaining a positive user experience.

2.Can you describe a complex automation project you've undertaken in your role as a DevOps Engineer? What challenges did you face, and how did you overcome them?

Complex Manual Process: One of the complex manual processes I automated involved the release and deployment pipeline for a large-scale e-commerce platform. Before automation, this process was cumbersome, error-prone, and time-consuming. It encompassed activities like code integration, testing, packaging, and deployment to various environments.

Challenges:

\* Time-Consuming

\* Human Errors

\* Lack of Consistency

\* Automation:

To tackle these challenges, I implemented an end-to-end CI/CD (Continuous Integration and Continuous Deployment) pipeline:

\* Version Control Integration: The process started with code integration into the version control system (e.g., Git). Automated triggers were set up to initiate the pipeline when new code was pushed.

\* Automated Testing: Automated testing suites were integrated to ensure code quality and reliability. This included unit tests, integration tests, and end-to-end tests.

\* Artifact Creation: After successful testing, the pipeline automatically created deployment artifacts, such as Docker containers or application packages.

\* Orchestration and Deployment: Using tools like Kubernetes, the pipeline orchestrated the deployment process. Also ansible was used that ensured consistent configurations across various environments.

\* Monitoring and Rollback: Automated monitoring solutions were implemented to detect issues in real-time. If a problem occurred, the pipeline could initiate an automated rollback to a stable version.

Results:

The automation of the release and deployment process had several positive outcomes:

\* Efficiency: New code changes were automatically and rapidly integrated and deployed, reducing lead times.

\* Quality: Automated testing minimized human errors and improved the overall quality of the software.

\* Consistency: All environments (development, staging, and production) followed a consistent release process.

\* Reliability: Monitoring and automated rollback mechanisms ensured high system reliability.

3.Share an experience where you had to troubleshoot a complex technical issue that impacted a critical system. How did you approach the problem, and what was the resolution?

In a recent experience, our production environment experienced a sudden spike in traffic. The database server became unresponsive, causing downtime.

To address it:

\* First, I identified the issue by examining logs, monitoring metrics, and error reports.

\* I determined that a poorly optimized database query was causing the bottleneck.

\* I temporarily modified the application configuration to mitigate the query issue.

\* Concurrently, I optimized the database queries, added indexing, and performed a rolling update.

\* Within a critical 30-minute window, we had the database and application back to normal, minimizing downtime and maintaining a positive user experience.

4.Could you provide an example of a complex script you've developed to automate a specific process? What was the process, and how did the script improve efficiency or reliability?

In my previous role as a DevOps Engineer, one critical task was to ensure the backup and restore of large databases. The manual process was cumbersome, time-consuming, and prone to human errors. I identified the need to automate this process for efficiency and reliability.

Automation Script:

I developed a comprehensive Python script to automate the entire backup and restore workflow for large databases. Here's a breakdown of how the script worked:

\* Backup Scheduling: The script allowed the scheduling of regular database backups, ensuring data was consistently protected. It could be configured to run daily, weekly, or on a custom schedule. scheduler library in Python (e.g., schedule or APScheduler). Accepted user-defined parameters for scheduling intervals (daily, weekly, custom).

\* Data Compression: Before storing backups, the script compressed the data, reducing storage requirements and improving transfer speed. This was especially crucial for large databases.Incorporated data compression libraries like gzip or bzip2 in Python.

\* Backup Execution: It initiated the backup process using the database's native tools, such as mysqldump for MySQL or pg\_dump for PostgreSQL. The script handled the complex command-line arguments and options, ensuring a consistent backup process.

\* Storage Management: Backups were stored in a structured directory hierarchy, making it easy to locate and manage them. Old backups were automatically pruned based on retention policies.

\* Monitoring and Alerting: The script integrated with our monitoring system to provide alerts in case of backup failures or issues. This allowed for timely intervention and resolution.

\* Restoration Process: In the event of a database failure, the script could initiate the restoration process. It selected the appropriate backup based on timestamps and restored the database to the desired state.

\* Backup Scheduling:

\* Technical Implementation: Utilized a scheduling mechanism, possibly a cron job or a scheduler library in Python (e.g., schedule or APScheduler).

\* Configuration: Accepted user-defined parameters for scheduling intervals (daily, weekly, custom).

\* Data Compression:

\* Technical Implementation: Incorporated data compression libraries like gzip or bzip2 in Python.

\* Compression Process: Compressed the database dump files before storage to save disk space and accelerate data transfer.

\* Backup Execution:

\* Technical Implementation: Utilized subprocess management in Python (subprocess module) to execute native database backup commands.

\* Command Handling: Dynamically generated and executed commands (e.g., mysqldump or pg\_dump) with specified arguments and options.

\* Storage Management:

\* Directory Hierarchy: Designed a structured directory hierarchy to organize backups, possibly based on date or database name.

\* Retention Policies: Implemented logic to automatically prune old backups based on defined retention policies, preventing unnecessary storage consumption.

\* Monitoring and Alerting:

\* Integration: Integrated with a monitoring system (e.g., Nagios, Prometheus) using APIs or custom scripts.

\* Alerts: Triggered alerts in case of backup failures or issues, sending notifications to administrators or logging events for further analysis.

\* Restoration Process:

\* Technical Implementation: Wrote scripts to initiate the database restoration process.

\* Selection of Backup: Identified the appropriate backup based on timestamps or other criteria.

\* Database Restoration: Executed the restoration commands (e.g., mysql or psql) to restore the database to the desired state.

\* Error Handling:

\* Technical Implementation: Implemented robust error-handling mechanisms using Python's try-except blocks.

\* Logging: Recorded detailed logs of each step, including errors, for debugging and auditing purposes.

\* Configurability:

\* Technical Flexibility: Made the script configurable by using configuration files or command-line arguments.

\* User Inputs: Accepted user-defined parameters for backup locations, compression methods, and other settings.

\* Scalability:

\* Parallelization: Considered implementing parallelization for backup and compression tasks to enhance performance, especially for large databases.

\* Optimizations: Employed optimizations to handle increasing data sizes efficiently.

\* Documentation:

\* Technical Documentation: Ensured comprehensive technical documentation for the script, including setup instructions, configuration details, and troubleshooting guides.

5.Could you provide an example of a script you've developed to automate daily routine tasks? What was the process, and how did the script improve efficiency or reliability?

Process:

In my previous role as a DevOps Engineer, I had to perform daily log file analysis to monitor system health and identify potential issues. The process involved manually accessing log files, parsing them for specific events, and then generating reports based on this data. It was a time-consuming and error-prone task.

Automation Script:

To streamline this process, I developed a Python script that automated log file analysis and reporting. Here's how the script worked:

\* Log File Access: The script used SSH to remotely access log files from various servers and retrieve them to a centralized location.

\* Log Parsing: It parsed the log files to extract relevant data, such as error messages, warning signs, and system performance metrics.

\* Data Analysis: The script performed real-time analysis on the log data, identifying critical events, and calculating key performance indicators (KPIs).

\* Report Generation: It generated daily reports summarizing the key findings. These reports included a breakdown of critical issues, performance trends, and recommendations for improvements.

\* Alerting: The script also integrated with our alerting system to send immediate notifications if it detected severe issues.

6.Can you describe a scenario where you had to troubleshoot an API configuration issue? How did you identify and resolve the problem?

Answer: In a previous role, we faced an issue where API requests were failing intermittently due to misconfigured load balancing. I identified the problem by analyzing logs and metrics, then adjusted load balancer settings to distribute traffic evenly. Regular monitoring and alerting were crucial in detecting and resolving the issue promptly.

Python Coding Questions:

Python:

The split() function separates the given string by the defined delimiter.

Mutable objects: They can be updated once defined. e.g., list.

Immutable objects: They cannot be updated. e.g., tuples.

We can delete the file in Python using the os module. The remove() function

delete elements from the list By using the remove() function, By using the pop() function

We can delete a list in Python using the clear() function.

reversing an array are as follows:

1. Using the flip() function

arr1[::-1] (manual)

 Pickling is converting a Python object (list, dict, function, etc.) to a byte stream(0s and 1s), and unpickling is converting the byte stream back to a python object.

\_init\_ method is used in Python to initialize the attributes of the object when the object is created

The join() method is a built-in function of Python’s str class that concatenates a list of strings into a single string.

\* strip(): This function removes leading and trailing whitespace from a string.

\* replace(): This function replaces all occurrences of a specified string with another string.

Subprocess Module

Probably the second most important module in the DevOps chain. If you want to run the external program and read their output in Python code then subprocess is the module for you.

Sys Module

The third most important module for me is sys. If you want to manipulate different parts of the Python runtime environment or want to work with a command-line argument passed to your python script, sys is the module for you.

Platform module

Platform module is used to access underlying platform data such as operating system, interpreter version information, and hardware. It has also been used to write cross-platform script(the script which works on multiple platforms for e.g. same script works on Linux as well as on Windows)

Paramiko module

If you want to execute a command to the remote system, download or upload the file without login, then you can use the paramiko module.

Boto3 Module

If you are managing AWS infrastructure, then you must need to know boto3. Boto3 is the Amazon Web Services (AWS) SDK for Python. It enables Python developers to create, configure, and manage AWS services, such as EC2 and S3. Boto3 provides an easy-to-use, object-oriented API, as well as low-level access to AWS services.

GitHub API or PyGithub Module

If you are dealing with frequent tasks like creating, deleting, or listing GitHub repo on a frequent basis then you should check out this module.

\* Requests - To send HTTP requests (GET, POST, PUT, DELETE) and interact with RESTful APIs.

\* JSON - To parse JSON file.

\* YAML - To parse YAML file.

\* virtualenv - To create an isolated environment.

\* Pytest - To write test cases for your code.

\* Boto3 - Allows managing AWS infrastructure using Python script.

https://www.analyticsvidhya.com/blog/2022/07/python-coding-interview-questions-for-freshers/

(The above link has few python question that would be enough but we need to prepare for any follow up question that in real time)