**IT Business Analyst**

**Screening Questions**

**Question: Can you explain the difference between procedural and object-oriented programming?  
Answer:**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Procedural Programming** | **Object-Oriented Programming** |
| Program Structure | Divided into functions (procedures) | Divided into objects and classes |
| Approach | Top-down approach | Bottom-up approach |
| Access Specifiers | No access specifiers | Has access specifiers (private, public, etc.) |
| Adding New Elements | Adding new data and functions is not easy | Adding new data and functions is easy |
| Data Hiding | No proper way of hiding data (less secure) | Provides data hiding (more secure) |
| Overloading | Not possible | Possible |
| Inheritance | No concept of inheritance | Utilizes inheritance |
| Program Size | Used for medium-sized programs | Used for large and complex programs |
| Abstraction | Procedure abstraction | Data abstraction |

**Question: How would you handle an exception in your code? Can you provide an example?**  
**Answer:**

1. Try Block: We wrap the code that might cause an error in a "try" block. This is where we put the code we suspect could throw an exception.

2. Catch Block: We use a "catch" block to handle the exception if it occurs. The "catch" block specifies the type of exception it can handle and provides instructions on what to do in that situation.

Example in Python:

try:

  numerator = int(input("Enter the numerator: "))

  denominator = int(input("Enter the denominator: "))

  result = numerator / denominator

  print("The result is:", result)

except ZeroDivisionError:

  print("Error: Cannot divide by zero”)

**Question: Explain the concept of polymorphism in object-oriented programming.  
Answer:**

Definition:

* + Polymorphism refers to the ability of different objects to respond to the same method call in their own unique way.
  + It allows you to treat objects of different classes uniformly through a common interface.
  + Polymorphism simplifies code by promoting reusability and flexibility.

Types of Polymorphism:

* + Static (Compile-Time) Polymorphism:
    - Achieved through method overloading (multiple methods with the same name but different parameters) within the same class.
    - The compiler determines which method to call based on the arguments passed during method invocation.
  + Dynamic (Run-Time) Polymorphism:
    - Occurs when a subclass provides a specific implementation for a method already defined in its superclass.
    - The actual method called is determined at runtime based on the object type.

Benefits:

* + Code reusability
  + Flexibility
  + Abstraction

**Question: How do you optimize a SQL query for better performance?  
Answer:**

1. Use Indexes: Create indexes on frequently queried columns to speed up data retrieval.
2. Limit Results: Use “LIMIT” to restrict the number of rows returned.
3. Avoid Subqueries: Prefer “JOIN” over subqueries for better performance.
4. Minimize Wildcards: Avoid excessive use of wildcard characters (% and \_ ).

Ex: SELECT \* FROM customers WHERE last\_name\_city LIKE 'P%';

1. Understand Query Purpose: Optimize based on the query’s intended use.

Ex: SELECT \* FROM customers WHERE last\_name\_city >= 'P' AND last\_name < 'Q';

**Question: What is the difference between GET and POST in HTTP, and when would you use each?  
Answer:**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **GET Request** | **POST Request** |
| Purpose | Retrieve data from a resource | Send data to a server to create/update a resource |
| Safety and Idempotence\* | Safe and idempotent (same result on repeated calls) | Not safe or idempotent |
| Data Transmission | Parameters appended to the URL (query parameters) | Parameters sent in the body of the request |
| Caching | Requests can be cached by browsers and intermediaries | Requests are not cached by default |
| Bookmarking | Requests can be bookmarked and shared | Requests are not typically bookmarked or shared |
| Data Length Restrictions | Limited by URL length (maximum 2048 characters) | No specific length limitations in POST body |
| Data Visibility | Visible in the URL and may be cached | Hidden in the request body |
| Security | Less secure (data sent in URL) | Safer (parameters not stored in browser history) |

**Question: Explain the concept of MVC (Model-View-Controller) architecture.  
Answer:**

1. Purpose:
   * MVC is a software architectural pattern that separates an application into three main components: Model, View, and Controller.
   * It aims to organize code, improve maintainability, and enhance scalability.
2. Components:
   * Model: Represents data and business logic.
   * View: Handles UI presentation.
   * Controller: Manages user input, business logic, and communication between Model and View.
3. Responsibilities:
   * Model: Deals with data manipulation, storage, and retrieval.
   * View: Displays data to the user.
   * Controller: Orchestrates interactions between Model and View.
4. Benefits:
   * Separation of Concerns: Isolates different aspects of an application.
   * Reusability: Allows components to be reused independently.
   * Scalability: Facilitates adding new features without major changes.
5. Usage:
   * MVC is widely used in web development, mobile apps, and desktop GUIs

**Question: Explain the concept of garbage collection in programming languages.  
Answer:**

Purpose:

* + Garbage collection (GC) is the process by which programs reclaim memory space that is no longer used by objects.
  + It automatically identifies and frees up memory occupied by objects that are no longer reachable or needed in the program.

Implementation:

* + High-level programming languages (such as Java, C#, Python) typically include built-in garbage collection mechanisms.
  + Low-level languages (like C or C++) may add garbage collection through libraries or manual memory management.

**Question:** What are the programming languages you have experience in and how many years?

I am having 2.7 years of experience.

1 year – Python

1 Year – Angular

7 months – Figma, Share point development  
  
**Question:** What are the business solutions proposed post analysing the data?

1. **Descriptive Analytics**: This type of analytics focuses on interpreting historical data to identify trends and patterns. By understanding past performance, businesses can make informed decisions about their future strategies.
2. **Predictive Analytics**: Building upon descriptive analytics, predictive analytics uses historical data to forecast future outcomes. It helps organizations anticipate trends, customer behavior, and market changes.
3. **Diagnostic Analytics**: When a problem arises, diagnostic analytics helps identify its root cause. By analyzing data, businesses can understand why certain events occurred and take corrective actions.
4. **Prescriptive Analytics**: This type of analytics recommends specific actions to optimize outcomes. It involves testing various scenarios to determine the best course of action in a given situation.