Basic Ruby Questions and Answers

1. **What is Ruby?**
    - **Answer:** Ruby is an open-source, dynamic, object-oriented programming language designed for simplicity and productivity.
2. **Explain the Ruby syntax for printing "Hello, World!"**
    - **Answer:** `puts 'Hello, World!'`
3. **What are variables in Ruby?**
    - **Answer:** Variables are used to store data. Ruby has several types of variables: local, global, instance, and class variables.
4. **What is the difference between `puts` and `print` in Ruby?**
    - **Answer:** `puts` prints a string followed by a newline, while `print` prints a string without a newline.
5. **What is an array in Ruby?**
    - **Answer:** An array is an ordered collection of objects, which can hold multiple elements of any type.
6. **How do you create a hash in Ruby?**
    - **Answer:** `hash = { 'key1' => 'value1', 'key2' => 'value2' }`
7. **What are symbols in Ruby?**
    - **Answer:** Symbols are lightweight, immutable identifiers used as keys in hashes or as names for variables or methods.
8. **What is a block in Ruby?**
    - **Answer:** A block is a chunk of code that can accept parameters and is passed to methods for execution.
9. **What is a method in Ruby?**
    - **Answer:** A method is a function defined within a class that can be called to perform a task.
10. **What is the difference between `==` and `equal?` in Ruby?**

- **Answer:** `==` checks for value equality, while `equal?` checks if two objects refer to the same memory location.

11. **How do you define a class in Ruby?**

Answer:

```
class MyClass
  def my_method
    # code
  end
end
```

-

12. **What is an instance variable in Ruby?**

- **Answer:** An instance variable is a variable prefixed with `@` that is accessible within an object instance.

13. **What is a constant in Ruby?**

- **Answer:** A constant is a variable whose value is not supposed to change after initialization, denoted by all uppercase letters.

14. **What is the `self` keyword in Ruby?**

- **Answer:** `self` refers to the current object or the context in which a method is being executed.

15. **What does `nil` represent in Ruby?**

- **Answer:** `nil` represents the absence of any object or value.

16. **How do you handle exceptions in Ruby?**

Answer: Using `begin`, `rescue`, and `ensure` blocks:

```
begin
  # code that may raise an error
rescue StandardError => e
  # code to handle error
ensure
  # code that will always run
end
```

-

17. **What are the different types of loops in Ruby?**

- **Answer:** Ruby supports `while`, `until`, `for`, and `each` loops.

18. **What is a Range in Ruby?**

- **Answer:** A range is an interval that can represent a sequence of numbers, characters, or other objects. Example: `1..5`

19. **What does the `super` keyword do in Ruby?**

- **Answer:** `super` calls a method of the same name in the parent class.

20. **What are regular expressions in Ruby?**

- **Answer:** Regular expressions are used for pattern matching and text manipulation, defined with `/pattern/`.

21. **How do you concatenate strings in Ruby?**

Answer: Using `+` or `<<`:

```
"Hello" + " World"
"Hello" << " World"
```

-

22. **What is the `each` method in Ruby?**

- **Answer:** `each` is used to iterate over an array or a hash.

23. **What is `ARGV` in Ruby?**

- **Answer:** `ARGV` is an array that contains command-line arguments passed to the Ruby script.

24. **How do you check the class of an object in Ruby?**

Answer: Use the `.class` method:

```
"hello".class #=> String
```

-

25. **What is the purpose of `require` and `include` in Ruby?**

- **Answer:** `require` is used to load external files, while `include` is used to mix modules into classes.

Intermediate Ruby Questions and Answers

1. **What is method overloading in Ruby?**
    - **Answer:** Ruby does not support method overloading. However, you can use default parameters or variable-length arguments to simulate it.
2. **What are the differences between a class and a module in Ruby?**
    - **Answer:** A class can be instantiated and inherits from other classes, while a module cannot be instantiated and is used for mixins.
3. **What is the `initialize` method in Ruby?**
    - **Answer:** The `initialize` method is the constructor method that is called when a new object is instantiated.
4. **How do you define a class method in Ruby?**

Answer: Class methods are defined with `self`:

```
class MyClass
  def self.class_method
    # code
  end
end
```

    -
5. **What are Proc and Lambda in Ruby?**
    - **Answer:** Both are types of blocks, but a `Proc` is less strict in argument checking, while a `Lambda` checks the number of arguments passed.
6. **Explain the `yield` keyword in Ruby.**
    - **Answer:** `yield` is used inside a method to pass control from the method to a block.

7. **What is the `attr_accessor` in Ruby?**
    - **Answer:** `attr_accessor` is a shortcut for defining both getter and setter methods for instance variables.
8. **What is the difference between `Array#map` and `Array#each` in Ruby?**
    - **Answer:** `map` returns a new array with the results of the block, while `each` returns the original array and is used for iteration.
9. **How do you handle default values for method parameters in Ruby?**

Answer: You can assign default values to parameters:

```
def greet(name = "Guest")
  puts "Hello, #{name}"
end
```

    -
10. **What is the `to_s` method in Ruby?**

- **Answer:** The `to_s` method is used to convert an object into a string representation.

11. **What is the difference between `Array#select` and `Array#find` in Ruby?**

- **Answer:** `select` returns an array of all matching elements, while `find` returns only the first matching element.

12. **What is the purpose of `freeze` in Ruby?**

- **Answer:** `freeze` makes an object immutable, preventing any further modifications.

13. **What is the difference between `dup` and `clone` in Ruby?**

- **Answer:** `dup` creates a shallow copy of an object, while `clone` creates a deep copy and copies the object's frozen state.

14. **What is a `singleton method` in Ruby?**

- **Answer:** A singleton method is a method that is defined for a specific object, not its class.

15. **What is the difference between `instance_variable_set` and `instance_variable_get` in Ruby?**

- **Answer:** `instance_variable_set` sets the value of an instance variable, while `instance_variable_get` retrieves it.

16. **How do you handle multi-threading in Ruby?**

- **Answer:** You can use the `Thread` class to create and manage threads in Ruby.

17. **Explain how garbage collection works in Ruby.**

- **Answer:** Ruby uses an automatic garbage collector to reclaim memory by removing unused objects.

18. **What is a `Hash#merge` in Ruby?**

- **Answer:** `merge` combines two hashes into one, overriding values with the same keys.

19. **What is the purpose of `Enumerable` module in Ruby?**

- **Answer:** The `Enumerable` module provides methods for collections like arrays and hashes, including `map`, `select`, `reduce`, etc.

20. **What does `respond_to?` do in Ruby?**

- **Answer:** It checks whether an object can respond to a specific method.

21. **What are the differences between `while` and `until` loops in Ruby?**

- **Answer:** `while` loops run as long as the condition is true, whereas `until` loops run until the condition is true.

22. **What is a mixin in Ruby?**

- **Answer:** A mixin is a module that can be included in a class to add functionality.

23. **What is the `self` object in an instance method?**

- **Answer:** In an instance method, `self` refers to the instance of the class the method belongs to.

24. **How do you create a thread in Ruby?**

Answer:

```
thread = Thread.new { puts "Hello from a thread!" }
```

thread.join

- 25. **How does Ruby handle method visibility?**

- **Answer:** Ruby has three levels of method visibility: `public`, `protected`, and `private`.

Advanced Ruby Questions and Answers

1. **What is metaprogramming in Ruby?**
    - **Answer:** Metaprogramming is writing code that writes code, typically using `eval`, `define_method`, and `method_missing`.
2. **What is `method_missing` in Ruby?**
    - **Answer:** `method_missing` is a method called when an object receives a message it cannot respond to.
3. **What are hooks in Ruby?**
    - **Answer:** Hooks are methods that are called automatically by Ruby in certain situations, such as `initialize`, `method_missing`, etc.
4. **What is the use of `define_method` in Ruby?**
    - **Answer:** `define_method` dynamically defines a method at runtime.
5. **What is the difference between `class_variable` and `instance_variable` in Ruby?**
    - **Answer:** Class variables are shared among all instances of a class, while instance variables belong to a specific object instance.
6. **What is `ObjectSpace` in Ruby?**
    - **Answer:** `ObjectSpace` allows you to interact with and inspect all objects currently in memory.
7. **What is the purpose of `super` with arguments in Ruby?**

- **Answer:** It passes arguments to the method in the parent class.

8. **What is a `gem` in Ruby?**
   - **Answer:** A gem is a packaged Ruby library or application.
9. **What is `refinements` in Ruby?**
   - **Answer:** Refinements allow you to modify core classes only within a specific scope.
10. **What is the difference between `include` and `extend` in Ruby?**
    - **Answer:** `include` mixes a module into an instance's method space, while `extend` mixes it into the class's method space.
11. **How do you handle concurrency in Ruby?**
    - **Answer:** Ruby supports concurrency with threads and libraries like `Thread` and `Mutex`.
12. **What are lazy enumerators in Ruby?**
    - **Answer:** Lazy enumerators allow you to evaluate elements in an enumeration only when needed.

13. **What is the `eigenclass` in Ruby?**
    - **Answer:** The eigenclass (or metaclass) is the class of a specific object where singleton methods are stored.
14. **Explain how to use `require_relative` in Ruby.**
    - **Answer:** `require_relative` is used to load files relative to the current file's location.
15. **What is the purpose of `Enumerable#reduce`?**
    - **Answer:** `reduce` combines elements of a collection into a single value using an accumulator.
16. **What are `alias_method` and `alias_method_chain` in Ruby?**
    - **Answer:** `alias_method` creates a copy of a method, while `alias_method_chain` is a Rails-specific method for wrapping methods with

additional functionality.

17. **What is `Object#clone` used for?**
    - **Answer:** `clone` creates a shallow copy of an object.
18. **How do you define a singleton class in Ruby?**

Answer: A singleton class is defined for a specific object by opening the class:

```
class << obj
  # singleton methods
end
```

    -
19. **What are `unshift` and `shift` in Ruby?**
    - **Answer:** `unshift` adds elements to the beginning of an array, while `shift` removes the first element.
20. **Explain `uninitialized constant` error in Ruby.**
    - **Answer:** It occurs when Ruby cannot find a constant or class that is referenced.

21. **What are the `assert` and `refute` methods used for in Ruby testing?**
    - **Answer:** `assert` checks if a condition is true, while `refute` checks if a condition is false.
22. **What is a callback in Ruby?**
    - **Answer:** Callbacks are methods that are automatically invoked at specific points in a program, commonly used in Rails.
23. **What is `Time.now` in Ruby?**
    - **Answer:** `Time.now` returns the current system time.
24. **What is `ruby-debug`?**
    - **Answer:** `ruby-debug` is a debugger for Ruby that allows step-through debugging.

25. **What is the `ActiveRecord` pattern in Ruby on Rails?**

    - **Answer:** `ActiveRecord` is a pattern used in Rails for database access, representing records as objects and handling CRUD operations automatically.

Technical Ruby Questions and Answers

1. **How do you implement logging in Ruby?**

Answer: Use the `Logger` class to create and write logs:

```
 require 'logger'
logger = Logger.new(STDOUT)
logger.info('Info message')
```

    -

2. **Explain Ruby's memory management model.**

    - **Answer:** Ruby uses a garbage collector for automatic memory management.

3. **What are `Marshal` and `YAML` used for in Ruby?**

    - **Answer:** `Marshal` serializes Ruby objects, while `YAML` is used for storing and reading data in human-readable formats.

4. **How can you optimize Ruby code for performance?**

    - **Answer:** Use memoization, avoid unnecessary object creation, reduce method calls, and leverage native extensions or C extensions for computational-heavy tasks.

5. **What is the difference between `hash` and `set` in Ruby?**

    - **Answer:** A `hash` stores key-value pairs, while a `set` stores unique elements without keys.

6. **What is the use of `db:migrate` in Rails?**

    - **Answer:** `db:migrate` is used to apply changes to the database schema.

7. **What is a `singleton` in Ruby?**

- **Answer:** A singleton is a design pattern that ensures a class has only one instance and provides a global access point to that instance.

8. **How do you handle database transactions in Ruby?**
   - **Answer:** Use `ActiveRecord::Base.transaction` to handle database transactions.

9. **What is the `Rails console`?**
   - **Answer:** The Rails console is an interactive environment for running Ruby code in the context of a Rails application.

10. **How do you prevent SQL injection in Ruby on Rails?**
    - **Answer:** Use ActiveRecord queries and parameterized queries to prevent SQL injection.

11. **What are background jobs in Ruby on Rails?**
    - **Answer:** Background jobs handle tasks that need to be performed asynchronously, often using tools like `Sidekiq` or `Resque`.

12. **What is `Bundler` in Ruby?**
    - **Answer:** `Bundler` is a tool that manages Ruby gem dependencies for projects.

13. **Explain the purpose of `RSpec` in Ruby.**
    - **Answer:** `RSpec` is a testing framework used for behavior-driven development (BDD) in Ruby.

14. **What is `Capybara` used for in Ruby?**
    - **Answer:** `Capybara` is a tool for acceptance testing in Ruby, especially useful for simulating user interaction with web pages.

15. **What is `ActionController::Base` in Rails?**
    - **Answer:** `ActionController::Base` is the parent class for all controllers in Rails, providing methods for handling web requests.

16. **What is the `render` method in Ruby on Rails?**

- **Answer:** The `render` method is used to generate views in Rails controllers.

17. **What is the difference between `locals` and `instance variables` in Rails views?**

    - **Answer:** `locals` are passed from the controller to the view, while instance variables are set in the controller and accessible in the view.

18. **How can you secure sensitive data in Ruby on Rails?**

    - **Answer:** Use `dotenv` or Rails credentials to manage environment variables, and ensure database connections are encrypted.

19. **What is the `scope` method in Rails?**

    - **Answer:** The `scope` method defines a custom query that can be reused within models.

20. **What are concerns in Ruby on Rails?**

    - **Answer:** Concerns are modules used to extract reusable code, typically shared across models or controllers.

21. **What is the purpose of `rails new`?**

    - **Answer:** `rails new` generates a new Rails application, setting up the directory structure, configuration, and initial files.

22. **What is ActiveJob in Rails?**

    - **Answer:** ActiveJob is an abstraction layer for background job processing in Rails, supporting various job frameworks like `Sidekiq` and `Resque`.

23. **How does Rails handle sessions and cookies?**

    - **Answer:** Rails stores session data either in cookies (client-side) or in the database (server-side) depending on the configuration.

24. **How do you configure routes in Rails?**

    - **Answer:** Routes are configured in `config/routes.rb` where you define how requests map to controller actions.

25. **What is the difference between `get` and `post` in Rails routing?**

- **Answer:** `get` handles requests that retrieve data, while `post` handles requests that create or modify data.
