Ruby cours numéro 2

Définition de la classe `Book` :

```
ruby

class Book
  attr_accessor :title, :author, :genre

  def initialize(title, author, genre)
    @title = title
    @author = author
    @genre = genre
  end

  def to_s
    "#{title} by #{author} (Genre: #{genre})"
  end
end
```

Définition de la classe `Library` utilisant un module pour le suivi des emprunts :

```
ruby

module Borrowable
  def borrow
    @borrowed = true
  end

  def return
    @borrowed = false
  end

  def borrowed?
    @borrowed || false
  end
end

class Library
  attr_reader :books

  def initialize
    @books = []
  end

  def add_book(book)
    @books << book
  end

  def list_books
    puts "Library Catalog:"
    @books.each do |book|
      status = book.borrowed? ? "Borrowed" : "Available"
      puts "#{book} (Status: #{status})"
    end
  end
end
```

Utilisation des classes et modules :

```
ruby

# Création de quelques livres
book1 = Book.new("The Great Gatsby", "F. Scott Fitzgerald", "Fiction")
book2 = Book.new("To Kill a Mockingbird", "Harper Lee", "Classics")
book3 = Book.new("1984", "George Orwell", "Dystopian")

# Création d'une bibliothèque
library = Library.new

# Ajout de livres à la bibliothèque
library.add_book(book1)
library.add_book(book2)
library.add_book(book3)

# Affichage de la liste des livres
library.list_books

# Emprunt d'un livre
book1.extend(Borrowable)
book1.borrow

# Affichage de la liste mise à jour des livres
library.list_books
```

Gestion des exceptions :

Ajoutons une exception personnalisée pour gérer les cas où un utilisateur essaie d'emprunter un livre déjà emprunté.

```
ruby

class AlreadyBorrowedError < StandardError
  def initialize(book)
    super("#{book.title} is already borrowed.")
  end
end

# Dans la classe Book, ajoutons une vérification avant d'autoriser l'emprunt :
class Book
  # ...

  def borrow
    raise AlreadyBorrowedError.new(self) if borrowed?

    @borrowed = true
  end

  # ...
end
```

Utilisation de l'exception dans le script principal :

```
ruby

begin
  # Tentative d'emprunt du même livre
  book1.borrow
rescue AlreadyBorrowedError => e
```

```
  puts "Error: #{e.message}"
end
```

Cet exemple introduit quelques concepts avancés de Ruby, tels que l'utilisation de modules, la gestion des exceptions, et l'extension des fonctionnalités d'une classe existante. Pratiquez ces concepts en modifiant et en étendant cet exemple pour renforcer votre compréhension.