Microsoft Power Platform Developer PL-400 Exam - Detailed Crash Course

1. **Dataverse (CDS) Development**

- Create custom tables, columns (types: text, number, choice, lookup, etc.), relationships (1:N, N:1, N:N).
- Business Rules: Apply logic at form level without code.
- Business Process Flows (BPF): Guide users through processes, configurable stages and steps.
- Workflows vs. Power Automate: Use workflows for synchronous server-side logic; use Power Automate for async cloud flows.

2. **Power Apps Development**

- Canvas Apps:
  * Use Power Fx formulas for logic (e.g., Filter(), Lookup(), Patch()).
  * Connectors: Standard vs. Custom, use for external services.
  * Components: Reusable UI logic.
- Model-driven Apps:
  * Driven by Dataverse schema.
  * Use forms, views, dashboards, charts.
  * Customize using form scripting (JavaScript), command bar, and business rules.

3. **Power Automate**

- Cloud Flows:
  * Triggers: Instant, Automated, Scheduled.
  * Actions: HTTP request, Dataverse CRUD operations.
  * Expressions: Use functions like utcNow(), formatDateTime(), length().
- Desktop Flows (RPA):
  * Automate legacy systems using Power Automate Desktop.
  * Use UI elements, recorders, exception handling.
- Business Process Flows:
  * Stage-based process flow, can invoke actions or Power Automate flows.

4. **Extending Platform with Code**

- JavaScript:
  * Use Client API (formContext, Xrm.Page) to manipulate forms.

  * Events: onLoad, onChange, onSave.
- Plugins:
  * C# code triggered on Dataverse messages (Create, Update, Delete, etc.).
  * Registered in PreValidation, PreOperation, PostOperation stages.
  * Use IPluginExecutionContext and IOrganizationService.
- Custom Workflow Activities: C# code for server-side logic used in workflows.
- PCF (PowerApps Component Framework):
  * Custom UI components for Model-Driven and Canvas Apps.
  * Use TypeScript, npm, and CLI for development.
- Custom Connectors:
  * Extend using OpenAPI, create secure connections to APIs.

5. **ALM and DevOps**
- Solutions:
  * Unmanaged (dev), Managed (test/prod).
  * Patches and cloning solutions.
- Source Control:
  * Use Git to track changes to code and solution XML.
- Azure DevOps:
  * Use pipelines with Power Platform Build Tools to export/import solutions, run tests, and deploy.
  * YAML pipelines with service connections and secrets.
- Environment Strategies:
  * Dev, Test, UAT, Prod environments.
  * Use DLP policies to enforce governance.

6. **Azure Integration**
- Azure Functions: Use serverless logic via HTTP trigger.
- Azure Logic Apps: Orchestrate external services and integrate with Power Automate.
- Azure Key Vault: Secure secrets and API keys.
- Event Grid + Service Bus: Event-driven architectures with Dataverse webhooks.

7. **Security and Monitoring**
- Environment Roles and Security Roles in Dataverse.

- Column-level, table-level, and record-level security.
- Azure Active Directory integration for authentication.
- Power Platform Admin Center: Monitor flows, apps, usage.
- Use Azure Monitor, Application Insights for telemetry.

8. **Best Practices and Governance**
- Use naming conventions (prefixes, solution structure).
- Use Environment Variables and Connection References in solutions.
- Minimize use of unmanaged solutions in production.
- Monitor flow usage and API call limits.

9. **Key Power Platform Tools**
- Power Platform CLI (pac): Export/import, unpack/pack solutions.
- Solution Packager: ALM tool to work with solution components in source control.
- Plugin Registration Tool: Register custom plugins.
- Power Apps Test Studio: Automate UI tests.

Study Strategy:
- Practice real use cases using Power Apps, Power Automate, and Dataverse.
- Create custom connectors and PCF controls to reinforce understanding.
- Understand where and why to use plugins vs Power Automate.
- Practice setting up ALM pipelines with DevOps.