Exercise 1:

1. Untie the bread bag
2. Take out the bread
3. Place on clean plate.
4. Add butter on each side
5. Lightly toast the bread in toaster oven (1min)
6. On the bread add
   1. meat (ham, turkey, chicken breast)
   2. Provolone cheese
7. Melt the cheese in toaster (1-2mins)
8. Add the other side of the bread
9. Cut diagonally
10. Ready to serve

Reflection:

9 steps. I revised as I was typing. In the first draft, definitely forgot a few of the steps that didn’t necessarily involve making the sandwich (e.g. place on plate, lightly toast, etc).

Input:

* Turn stove dial
* Stove heats up

ADVANCED:

Ruby on Rails – framework

Ruby – does not need another language to build a website

HTTML, Java, Script

Ruby does not need those languages. It’s the backend and React is the front end.

Ruby: first 4-week

React:

Think about the capstone at week 12-13!!

**Day 3: Patterns and Sequences**

Part A: Patterns and Sequences

Reading & Explanation

1. **What Are Patterns?**
   * Repetitions that follow a predictable rule (e.g., 2, 4, 6, 8…).
   * Recognizing patterns helps **organize** tasks and spot repetition.
2. **Examples:**
   * Even/odd numbers.
   * Weekly schedule repeating Monday to Sunday.

Exercise 1: Completing Sequences

1. **Task:** Fill in 2, 4, 6, 8, \_\_ (answer: 10).
2. **Challenge Variation:** Create your own sequence with a different rule (e.g., +3, ×2). Ask someone to guess the next number.

Exercise 2: Patterns in Your Daily Routine

1. **Task:** Identify a **real-life** pattern (coffee vs. tea, checking email every 2 hours, etc.).
2. **Write it down**: Summarize the repetition (e.g., “Every second day, I jog at 7 AM.”).

**Extra Reflection**

* Tie these patterns to **loops**—if something repeats, a loop can automate it.

Part B: Boolean Logic & Conditionals

Reading & Explanation

1. **Boolean Values:** true / false
2. **Logical Operators:**
   * **AND** (&&): Both conditions must be true.
   * **OR** (||): At least one condition is true.
   * **NOT** (!): Flips true → false or false → true.
3. **Real-World Example:**
   * “I will watch TV if I finish my homework **AND** it’s before 9 PM.”

Real-life pattern: Wake up, walk kaiju (dog), brush teeth, get ready for meeting (if I have one), end work, walk kaiju, sleep

Exercise: Real-World Rules

1. **Task:** Pick a personal rule (e.g., “I order takeout if I have enough money **OR** I’m too tired”).
2. **Rewrite** using &&, ||, ! statements.
3. **Truth Table (Optional):** List all true/false combinations for your conditions, see the outcome.

**Extra Practice**

* Make a scenario that requires **all** conditions to be true (e.g., “I go hiking if it’s the weekend **AND** the weather is nice **AND** I’m not sick”).

Boolean Logic and Conditionals

Exercise: Real-world Rules

1. Task: I pay my bills when every payday
2. Conditions:
   1. I get to buy something if I get paid AND paid my bills.
      1. I don’t get to buy something if I get paid AND paid my bills.
   2. I get to buy something if I get paid OR I am celebrating something.
      1. I don’t get to buy something if I get paid OR I am celebrating something.
   3. I don’t get to buy something if I do NOT get paid
      1. I get to buy something if I get paid
   4. Extra practice:

**Day 4:**

**Reading & Explanation**

1. **What is Pseudocode?**
   * Write logic in **plain English** before dealing with code syntax.
   * Focus on **sequence** of steps (the “big picture”).
2. **Why Use Pseudocode?**
   * **Clarify** your thinking.
   * **Catch** logical issues early.
   * **Communicate** easily with teammates or non-coders.

**Exercise 1: Writing Simple Pseudocode**

**Task:** Plan what you do **before going outside**:

1. Check weather forecast.

2. IF sunny: wear sunglasses.

3. IF raining: take umbrella.

4. IF cold: wear jacket.

5. Go outside.

**Reflection:** Does it handle all weather (snow, hail, strong wind)? Add more conditions if needed.

Task: Prep before going to the beach

1. Charge speaker
2. Pack towels
3. Pack cooler
4. If going to dinner after: pack nice clothes
5. If going to Ritidian: take truck
6. If going to tumon: take floaties
7. Go to store and buy drinks and snacks
8. Go to beach

Reflection:

Does not take into account of timing.

**Part B: Recursion vs. Iteration (Conceptual)**

**Reading & Explanation**

1. **Iteration (Loops):**
   * Repeats actions with while or for.
   * Example: Counting down from 5 to 1 with a loop.
2. **Recursion:**
   * A function calls **itself** until a **base case** is reached.
   * Example: Recursively counting down from 5 to 1.

**Exercise 1: Countdown Comparison**

1. **Task A (Iteration):**
2. num = 5
3. WHILE num > 0:
4. print num

num = num - 1

**Task B (Recursion):**

1. def countdown(n):
2. IF n > 0:
3. print(n)
4. countdown(n - 1)
5. ELSE:

print("Done!")

**Reflection:** Which do you find more intuitive? Why?

Recursion seems more intuitive bc it explains hwo a countdown is defined in code.

**Extra Practice**

* **Nested Boxes**: If each box contains another box, recursion is like opening them one by one. Compare a recursive approach vs. an iterative approach for “unpacking” the boxes.

Iteration

* Boxes = 4
* While boxes > 1:
  + Print boxes
  + Boxes = boxes

Recursion

* Def Countdown (n):
  + IF n>1:
    - Print(n)
    - Countdown(n-1)
  + ELSE:
    - Print(“SURPRISE!”)

**Day 5: Control Structures - Loops and Repetition**

Part A: Control Structures – Loops & Repetition

Reading & Explanation

1. **What Are Loops?**
   * Loops let you **repeat** actions multiple times.
   * **While loop:** Repeat until a condition changes.
   * **For loop:** Repeat a fixed number of times.

Exercise 1: Real-Life Loops

1. **Task:** Think of a daily activity involving repetition (e.g., “Drink water until not thirsty.”).
2. **While Loop Parallel**:
3. WHILE still\_thirsty:

drink water

1. **Reflection:** The condition (still\_thirsty) is basically a **true/false** check.

Exercise 2: Loop-Based Routine

1. **Scenario:** “Study until you understand the material.”

WHILE you\_do\_not\_understand:

read textbook

take notes

try practice problems

review mistakes

1. **Challenge Variation:** If you know exactly how many times something repeats (e.g., “Take medicine 3 times a day”), use a **for** loop.

**Additional Practice**

* Brainstorm an activity that repeats a **strict number** of times vs. one that repeats **until** a condition is met.

WHILE:

* While you have cramps:
  + Use hot pad

For:

* For kaiju’s walking schedule
  + Walk him twice a day

Part B: Real-World Algorithms & Efficiency

Reading & Explanation

1. **Real-World Algorithms:**
   * Examples: GPS navigation, search engines, recommendation systems.
   * They handle complex data but rely on the same basic principles of logic and loops.
2. **Why Efficiency Matters:**
   * Slow algorithms cost **time**, **money**, and user **satisfaction**.
   * Example: A GPS app that takes too long to find a route is frustrating.

Exercise: Mapping App Scenario

1. **Task:** Think about how Google Maps (or similar) calculates a route:
   * **Inputs:** Current location, Destination
   * **Process:** Checks roads, traffic, closures, etc.
   * **Output:** Route and arrival time
2. **Reflection:**
   * Which parts might be **optimized** to save time (caching maps, skipping useless routes)?
     1. Skipping tolls (if preferred)
   * How does the app handle **real-time** updates?
     1. Recalculates the map for the faster arrival time.
     2. Maybe calculates it using open/available data from news channels?
     3. Maybe takes into account the number of devices using the same app in the area?
     4. Maybe takes into account the number of devices nearest a cell tower

**Bonus Challenge**

* **Shopping App Example:**
  + Inputs: Past purchases, ratings.
  + Process: Compare preferences with similar users.
  + Output: Product or content recommendations.
  + Where does **efficiency** matter with **huge** data sets?
    - How the data is stored and sorted.