Daksh Programming Exercises & Challenges

Phase 0 Practice Workbook

Purpose

A structured practice workbook for the finalized and implemented Phase 0 Daksh language. Exercises progress from basic syntax to combined programs and mini-projects.

Scope

This workbook uses Phase 0 concepts implemented in the web interpreter: variables, data types, arithmetic, conditions, loops, functions, operators, lists, objects, input/output, scope, modules, errors, and the implemented standard library.

How to use this workbook

1. Write each program yourself before looking for help.
2. Run small tests frequently and test boundary values.
3. Use the Daksh Error & Debugging Guide when something fails.
4. Move to the next level after you are comfortable with the current one.

Level 1 — Basics

01. Hello Daksh

Task: Use **tell** to print a greeting and three separate messages.

Concepts: tell, output

02. Personal Profile

Task: Create variables for a name, age, city, and score. Print each value.

Concepts: set, strings, integers, decimals

03. Multiple Declaration

Task: Declare three or more variables in one **set** statement and print them.

Concepts: multiple variables

04. Constants

Task: Create a constant value and print it. Then intentionally try to modify it and observe the runtime error.

Concepts: const, errors

05. Dynamic Value

Task: Store a number, then assign a string and a boolean to the same variable and print each value.

Concepts: dynamic typing

06. Type Conversion

Task: Convert a numeric string to an integer and a number to text.

Concepts: integer, text

07. Type Inspection

Task: Create values of several types and inspect them using **type()**.

Concepts: type, data types

08. Arithmetic Calculator

Task: Create two numbers and print sum, difference, product, quotient, and remainder.

Concepts: arithmetic

09. Power Calculator

Task: Calculate a number raised to a power using ******.

Concepts: exponent

10. Expression Practice

Task: Build expressions using parentheses and mixed arithmetic operators. Verify precedence.

Concepts: expressions, precedence

Level 2 — Conditions & Logic

11. Positive or Negative

Task: Check whether a number is positive, negative, or zero.

Concepts: if, else if, else

12. Even or Odd

Task: Determine whether an integer is even or odd using the remainder operator.

Concepts: %, conditions

13. Largest of Two

Task: Compare two numbers and print the larger value.

Concepts: comparisons

14. Largest of Three

Task: Find the largest among three numbers.

Concepts: multiple conditions

15. Grade Checker

Task: Given an average score, print a grade category using multiple conditions.

Concepts: comparison, else if

16. Age Category

Task: Classify a person's age into sensible categories.

Concepts: conditions

17. Logical Eligibility

Task: Use **&&** and **||** to decide whether multiple conditions are satisfied.

Concepts: logical operators

18. Boolean Negation

Task: Create a boolean condition and use **!** to test the opposite result.

Concepts: logical NOT

19. Number Range

Task: Check whether a number lies inside a chosen inclusive range.

Concepts: comparison, &&

20. Mini Login Check

Task: Use two expected values and conditions to decide whether both supplied values match.

Concepts: strings, &&, conditions

Level 3 — Loops

21. Count 1 to 10

Task: Print numbers from 1 through 10 using a range **for** loop.

Concepts: for, range

22. Countdown

Task: Print a countdown using a loop and decrementing values.

Concepts: loop, --

23. While Counter

Task: Print 1 through 10 using **while**.

Concepts: while, ++

24. Sum of Numbers

Task: Calculate the sum of numbers from 1 through 100.

Concepts: for, arithmetic

25. Multiplication Table

Task: Print the multiplication table of a chosen number.

Concepts: loops, arithmetic

26. Skip Multiples

Task: Print a range while using **continue** to skip selected values.

Concepts: continue

27. Stop at Target

Task: Search through numbers and use **break** when a target is reached.

Concepts: break

28. Nested Loop Grid

Task: Use nested loops to produce a small row/column pattern.

Concepts: nested loops

29. Factorial

Task: Calculate factorial of a positive integer using a loop.

Concepts: loop, multiplication

30. Number Analyzer

Task: For a chosen range, calculate useful counts such as even and odd numbers.

Concepts: loops, conditions

Level 4 — Lists & Objects

31. Print a List

Task: Create a list and print the list plus selected indexes.

Concepts: lists, indexing

32. List Traversal

Task: Use **foreach** to print every item in a list.

Concepts: foreach

33. List Sum

Task: Calculate the sum of numeric list items.

Concepts: foreach, arithmetic

34. Find Maximum

Task: Find the largest value in a numeric list.

Concepts: list traversal, conditions

35. Search List

Task: Check whether a chosen value exists in a list.

Concepts: list, contains

36. List Mutation

Task: Add an item, remove an item, and print the resulting list.

Concepts: list.add, list.remove

37. First and Last

Task: Retrieve the first and last elements with the list standard library.

Concepts: list.first, list.last

38. Student Object

Task: Create an object containing a student's name, average, and grade. Print its properties.

Concepts: objects, property access

39. Nested Data

Task: Create an object containing another object and a list. Access deep values.

Concepts: nested objects/lists

40. Nested Assignment

Task: Modify an object property and a list item through nested assignment.

Concepts: assignment, property/index access

Level 5 — Functions

41. Greeting Function

Task: Create a function that accepts a name and prints a greeting.

Concepts: function, parameter

42. Add Function

Task: Create a function that accepts two numbers and returns their sum.

Concepts: return

43. Average Function

Task: Create a function that accepts a list and returns its average.

Concepts: function, list, return

44. Grade Function

Task: Create a function that accepts an average and returns a grade.

Concepts: conditions, return

45. Function + Loop

Task: Create a function that processes a range using a loop and returns a result.

Concepts: function, loop

46. Function + List

Task: Create a function that searches a list and returns whether a value exists.

Concepts: function, list

47. Recursive Countdown

Task: Write a recursive function that counts down to a stopping condition.

Concepts: recursion

48. Recursive Factorial

Task: Implement factorial recursively.

Concepts: recursion, return

49. Student Processor

Task: Create a function that receives a name and marks list, computes an average, and returns a student object.

Concepts: functions, lists, objects

50. Reusable Calculator

Task: Create separate functions for common arithmetic operations and call them from one program.

Concepts: multiple functions

Level 6 — Standard Library Practice

51. Math Toolkit

Task: Use math functions for square root, absolute value, rounding, minimum, and maximum.

Concepts: math

52. String Cleaner

Task: Trim a name, convert it to uppercase, and replace a selected substring.

Concepts: string

53. String Validator

Task: Use contains, startsWith, and endsWith to validate a text value.

Concepts: string

54. List Utility

Task: Use length, contains, first, last, add, and remove in one program.

Concepts: list

55. Random Picker

Task: Generate random numbers and choose a random item from a list.

Concepts: random

56. Date Reporter

Task: Print today's date, current time, day name, month name, and numeric date parts.

Concepts: time

57. Date Comparison

Task: Use addDays/removeDays plus isBefore/isAfter to compare dates.

Concepts: time

58. Formatted Date

Task: Format a date/time using a chosen pattern.

Concepts: time.format

59. System Reporter

Task: Print useful information exposed by the system module.

Concepts: system

60. File Workflow

Task: Use the browser-safe file APIs to read or save user-selected content where supported.

Concepts: file, browser APIs

Level 7 — Integration Challenges

61. Student Result System

Task: Input a student name and marks, calculate total and average, determine a grade, and store the result as an object.

Concepts: input, functions, lists, objects

62. Shopping Bill

Task: Store item names and prices, calculate a total, and apply a condition-based discount.

Concepts: lists/objects, loops, conditions

63. Number Statistics

Task: Process a list and calculate total, average, minimum, maximum, even count, and odd count.

Concepts: lists, loops, math

64. Text Analyzer

Task: Analyze a string using length, case conversion, contains, startsWith, endsWith, and replacement.

Concepts: string

65. Random Quiz

Task: Choose questions or options from a list using random.choice and track a score.

Concepts: random, lists, conditions

66. Attendance Checker

Task: Represent students and attendance information, then report eligible and ineligible students.

Concepts: objects, loops, conditions

67. Inventory Tracker

Task: Maintain products with quantities, update values, and report low-stock items.

Concepts: objects, lists, nested assignment

68. Expense Tracker

Task: Store expenses, calculate totals, and classify spending using conditions.

Concepts: lists, objects, functions

69. Date Planner

Task: Use the time module to calculate future dates and compare planned dates.

Concepts: time, functions

70. Utility Dashboard

Task: Combine math, string, list, random, time, and system functionality in one program.

Concepts: standard library integration

Level 8 — Advanced Mini-Projects

71. Student Management Program

Task: Build a small student manager: store students, calculate results, search records, and update properties.

Concepts: functions, lists, objects, loops

72. Shopping Cart

Task: Create a cart workflow with products, quantities, totals, and a final bill.

Concepts: lists, objects, arithmetic

73. Expense Manager

Task: Record expenses, calculate summaries, and classify spending.

Concepts: functions, lists, objects

74. Number Game

Task: Generate a random target and let the user repeatedly guess until it is found.

Concepts: random, input, while, break

75. Quiz Game

Task: Create a multi-question quiz with a score, conditions, loops, and reusable functions.

Concepts: functions, lists, input

76. Student Report Generator

Task: Process several students and produce result objects with names, averages, and grades.

Concepts: functions, foreach, objects

77. Personal Utility Toolkit

Task: Create reusable functions for calculations, strings, lists, and date operations.

Concepts: functions, standard library

78. Data Processing Pipeline

Task: Accept a list of records, transform values, validate data, and produce summaries.

Concepts: objects, lists, functions, errors

79. File-Based Utility

Task: Create a browser-safe utility that selects/reads data and saves or downloads generated results.

Concepts: file, functions, strings

80. Final Phase 0 Challenge

Task: Build a complete small application combining input, variables, conditions, loops, functions, lists/objects, error-aware handling, and at least three standard-library modules.

Concepts: full Phase 0 integration

Challenge Rules & Self-Assessment

Beginner: Exercises 1–20 focus on core syntax and expressions.

Intermediate: Exercises 21–50 combine loops, collections, functions, and control flow.

Library: Exercises 51–60 focus on the implemented standard-library APIs.

Integration: Exercises 61–70 combine several Phase 0 concepts in one program.

Advanced: Exercises 71–80 are open-ended mini-project challenges.

Completion Checklist

Area	Completed
Variables & data types	■
Arithmetic & expressions	■
Conditions & logical operators	■
Loops & loop control	■
Lists & objects	■
Functions & recursion	■
Input / output	■
Scope & reassignment	■
Standard library	■
Error handling & debugging	■
Integration mini-project	■

Phase 0 boundary: These exercises do not require unfinished/future syntax such as ternary expressions, pattern matching, bitwise operators, lambdas, finalized custom-error syntax, or a native compiler/runtime.

Practice principle: A program is not complete just because it produces expected output. Test normal inputs, boundary values, invalid inputs, and relevant runtime errors.