# ASSIGNMENT #2 AND LAB #2

How to solve problems, understand data flow, write tests, and reuse code

# Primary Lessons

01

Understand the data flowing through a program

02

Identify reusable and testable patterns in algorithms

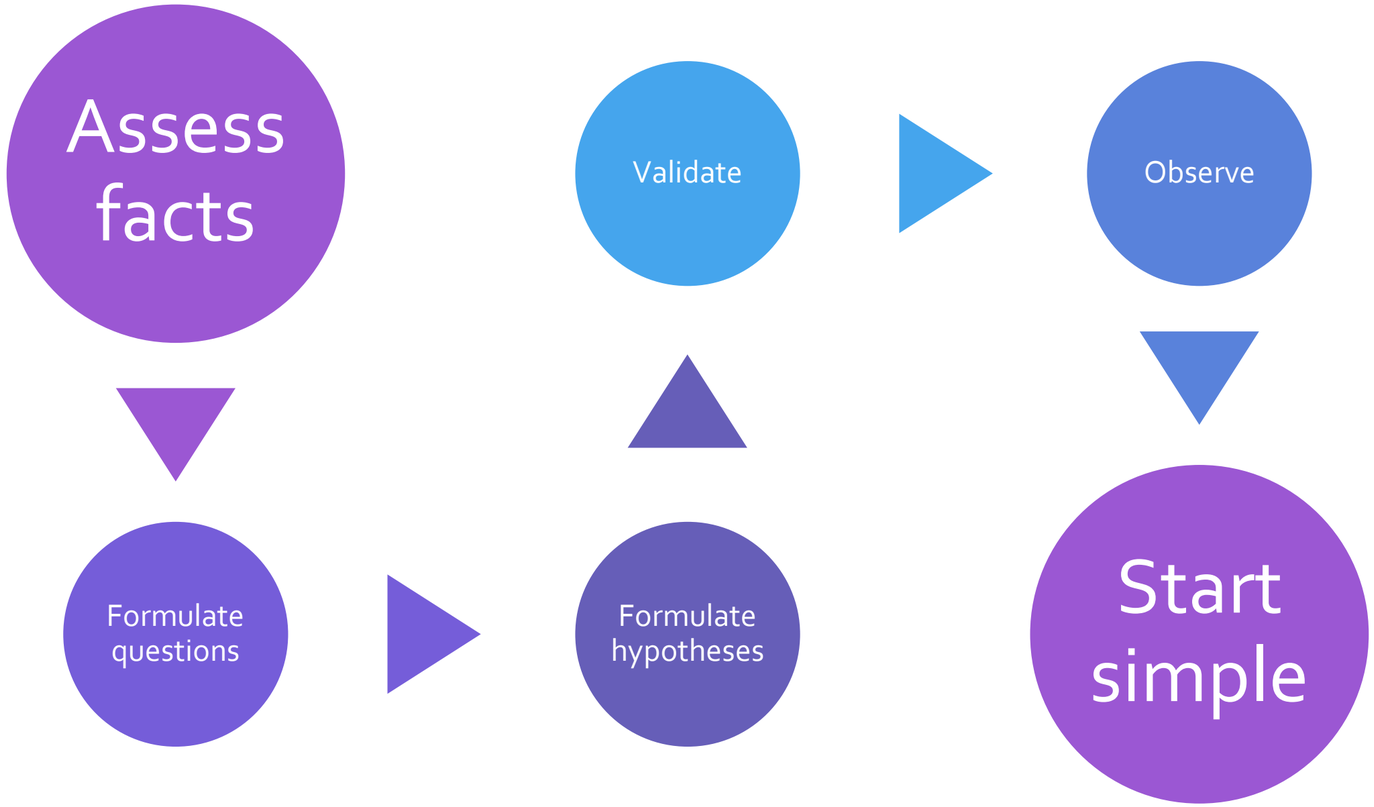
03

Practice writing algorithms

# Secondary Lessons

- Putting code into shared libraries
- Thinking about testing
- Get comfortable with solutions and projects
- Familiarize with the technique of test driven development
- Learn how to make useful console applications
- How to reduce repetition
- ***How to deal with open-ended problem statements!***

# General Advice



# How to make all this easier?

Practice

Have fun

Celebrate small wins

Patience

Kindness

Keep it simple

# Lab #2 Review

- Create a solution
- 3 Console applications
- 1 Class library
- 1 Unit test project
- See: https://github.com/cdiggins/cs321/tree/main/code-examples/cs321/Lab2

# Foreshadowing

The final project (and maybe future assignments) will look similar to this!

One or more applications

One or more test projects

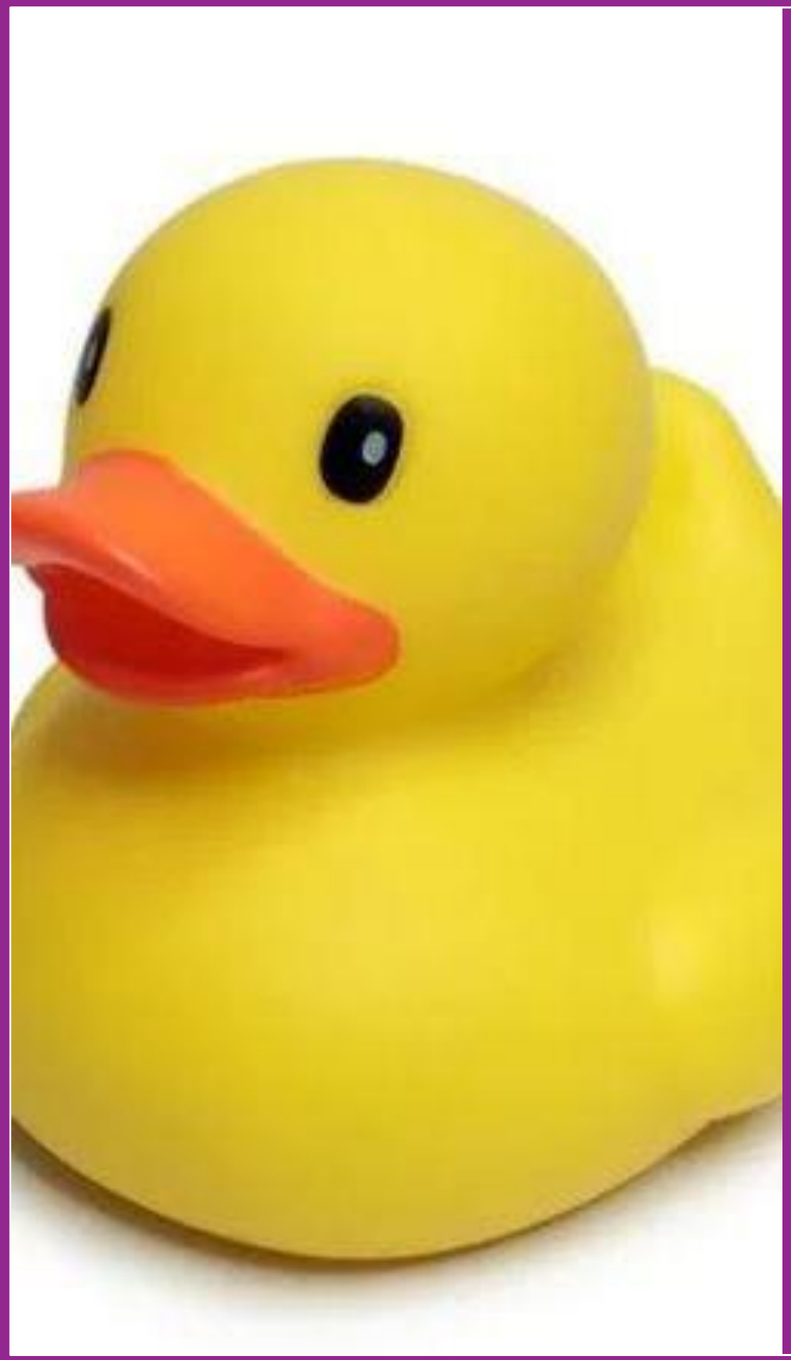
One or more shared code libraries

# Practice thinking and problem solving

***Don’t rely on tools or resources without thinking!***

# Lab 2: What are the questions?

- How to tackle it:
- What are your questions?
  - Is it a misunderstanding or ambiguity of what is requested?
  - Is it an unclear of how to best tackle the problem?
  - Are you unclear on how to best represent data?



# Learn to be specific

Computers are unlike people

They don’t understand context

Compiler messages are very helpful

Are there words you don’t understand?

Well worded questions are easy to answer

What doesn’t work?

What did you expect?

What do you want to do?

# Kinds of Errors

- Type system errors
- Syntax errors
- Semantic errors
- Typographical errors
- Arithmetic errors
- Misspelling
- Forgotten braces
- Computational errors
- Off by one errors

# Question to ask yourself

- What are the requirements?
- What is unclear?
- What is ambiguous?
- What are the possible interpretations?
- What is the data flow?
- What data structures do I need?
- What algorithms do I need?
- What can I test?

# How to make coding easy

PRACTICE

… THEN …

PRACTICE SOME MORE

# Shared Code?

What are common patterns in the transformations or procedures?

What makes sense to test?

# Types of Tests (informal)

### Smoke test
- Does it do what I expect with basic input

### Comprehensive test
- Does it meet properties across a range of inputs

### Performance test
- Does the algorithm meet performance criteria

### Edge case test
- Does it work with rare or degenerate input

### Limit test
- Uncover operational constraints

### Manual test
- Tests which are run manually

### Fuzz test
- Tested with pseudo-random data

### Regression test
- Assure that fixed bugs don't reoccur

### Integration test
- Does the algorithm work in conjunction with other components

# Patterns for the Common Library

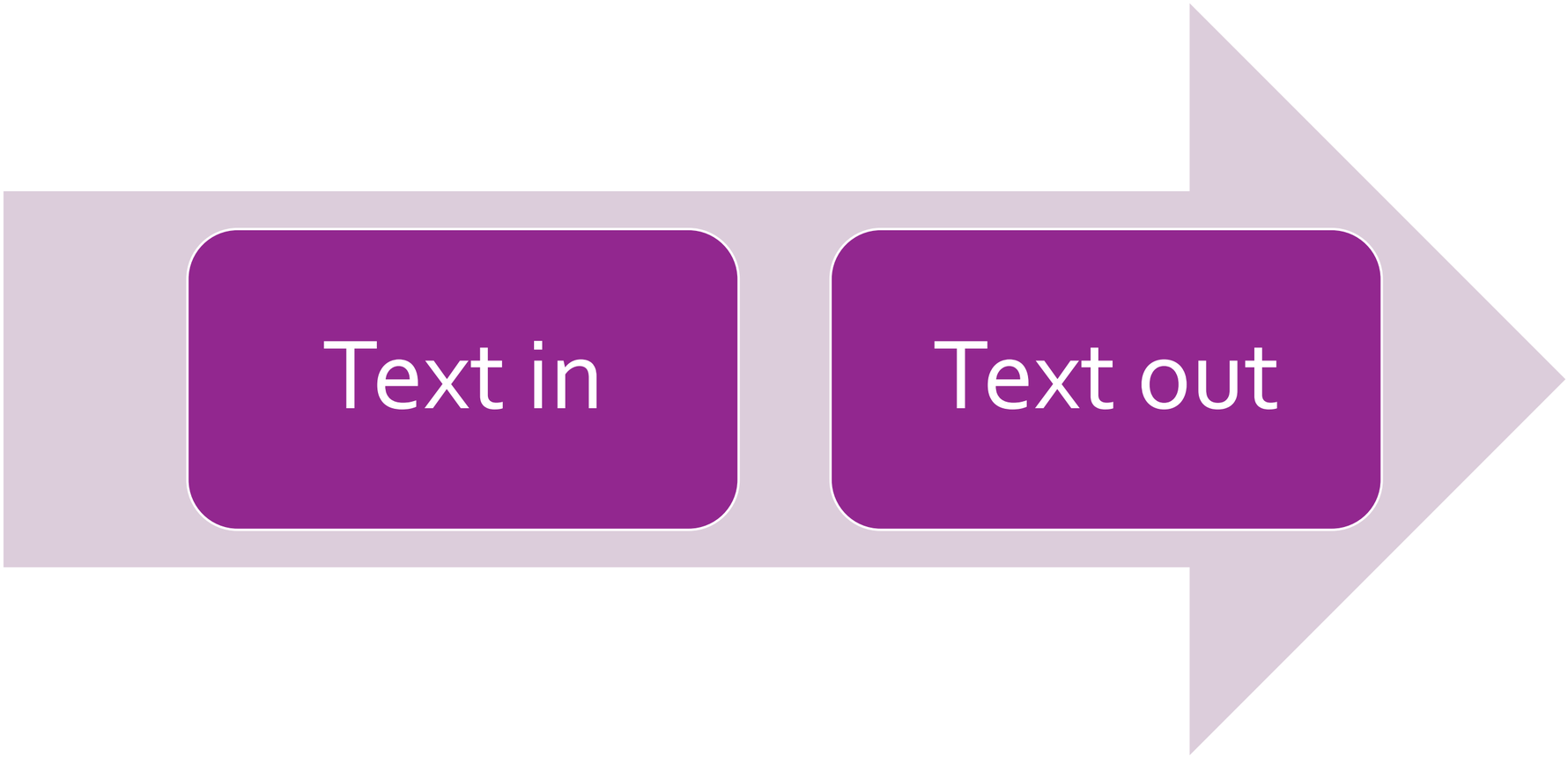
- Data comes from either standard input or a file
- Data is described in many cases as “lines”
- All programs process command line arguments
- All programs have to output a help
- All programs write to the standard output

# Data Flow

What data is going into each part of a program?

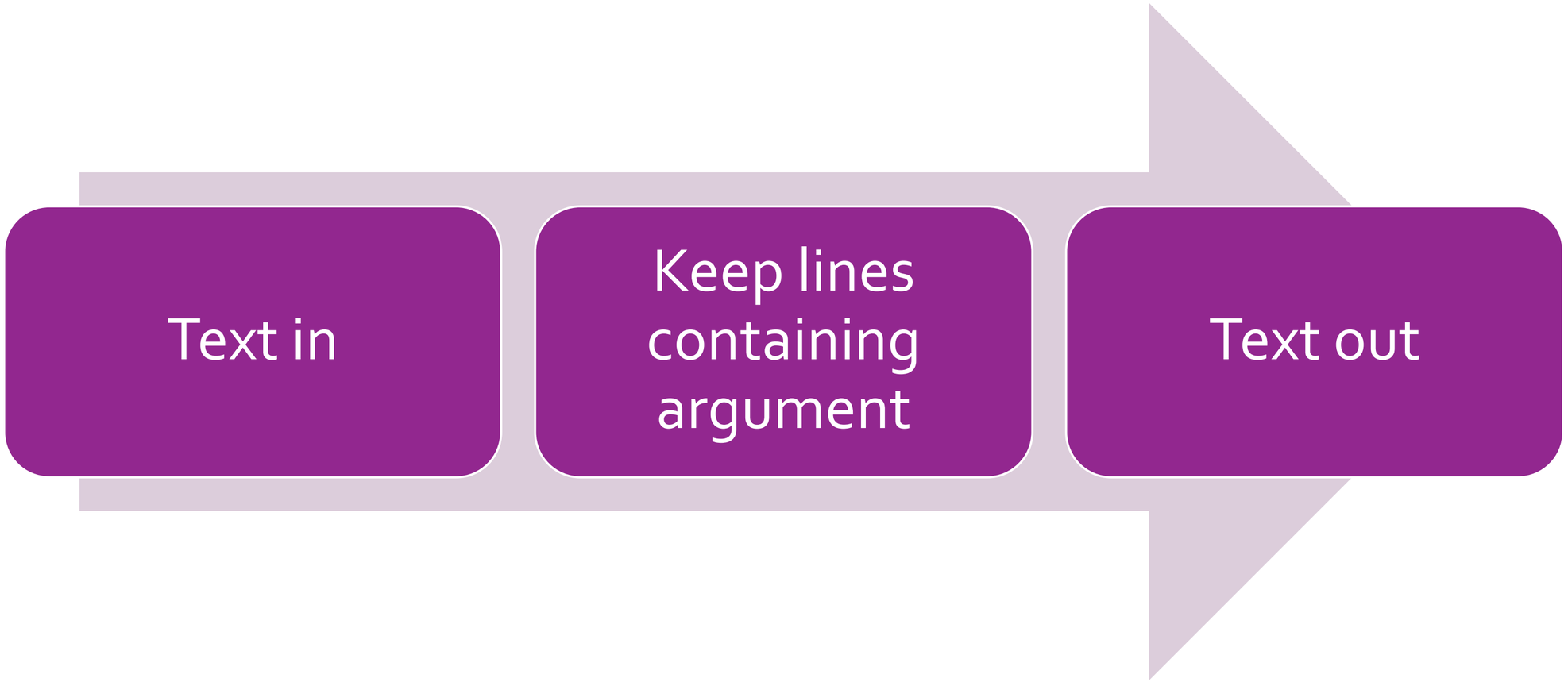
How is the data is transformed, ordered, filtered, and/or aggregated?

# Type – via Standard Input

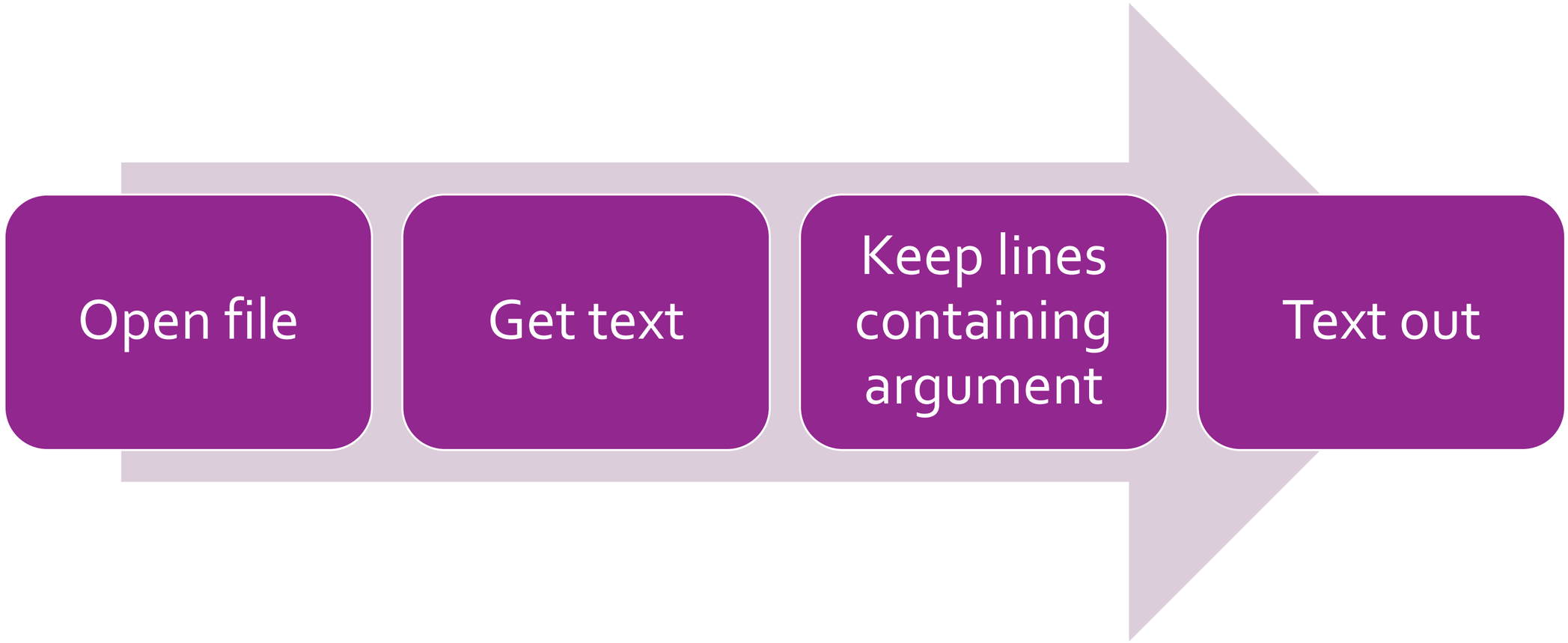


# Type – via Command Line Argument

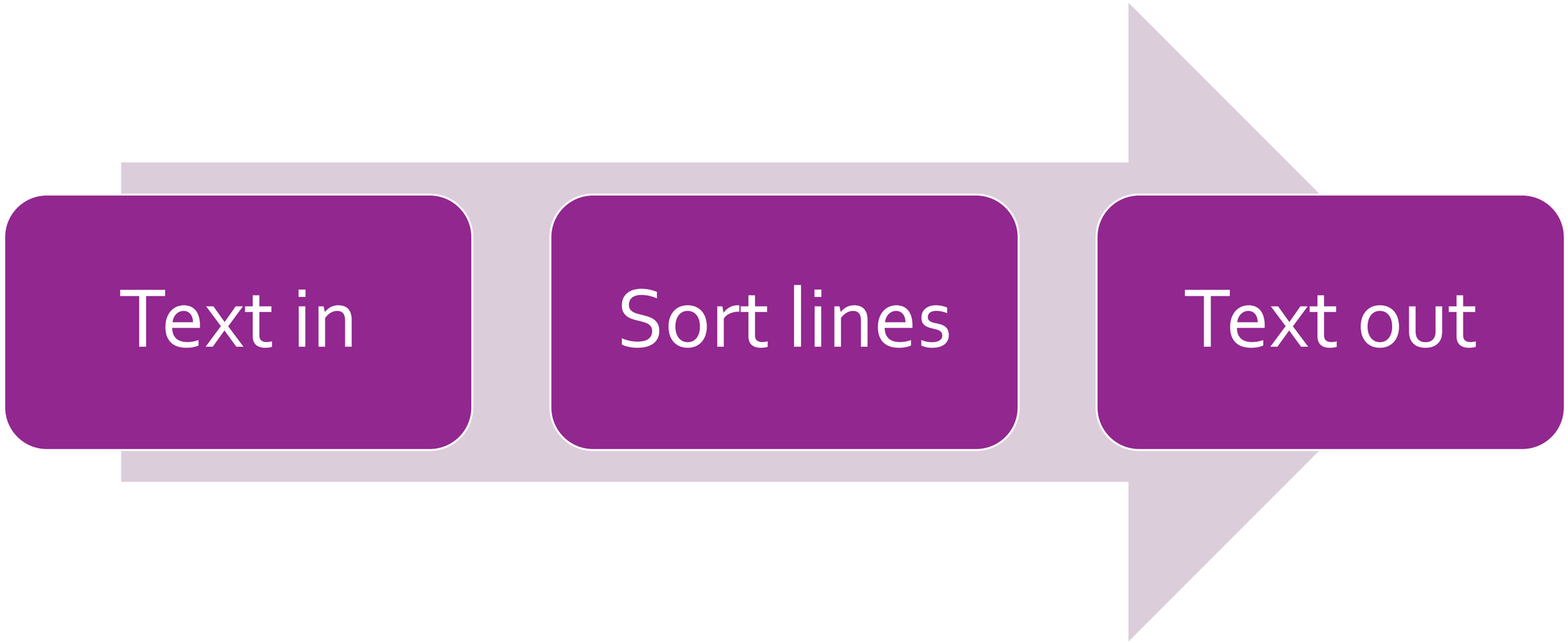
# Find – via Standard Input



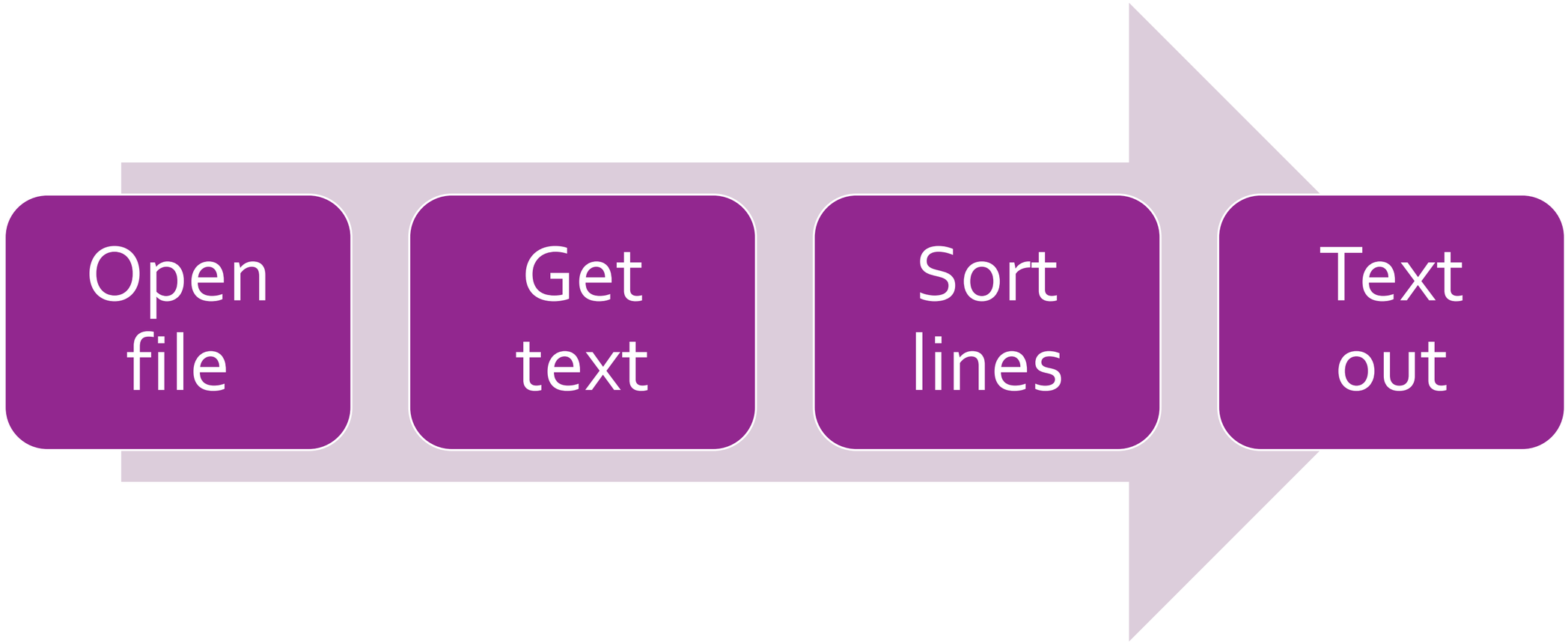
# Find – via Command Line Argument

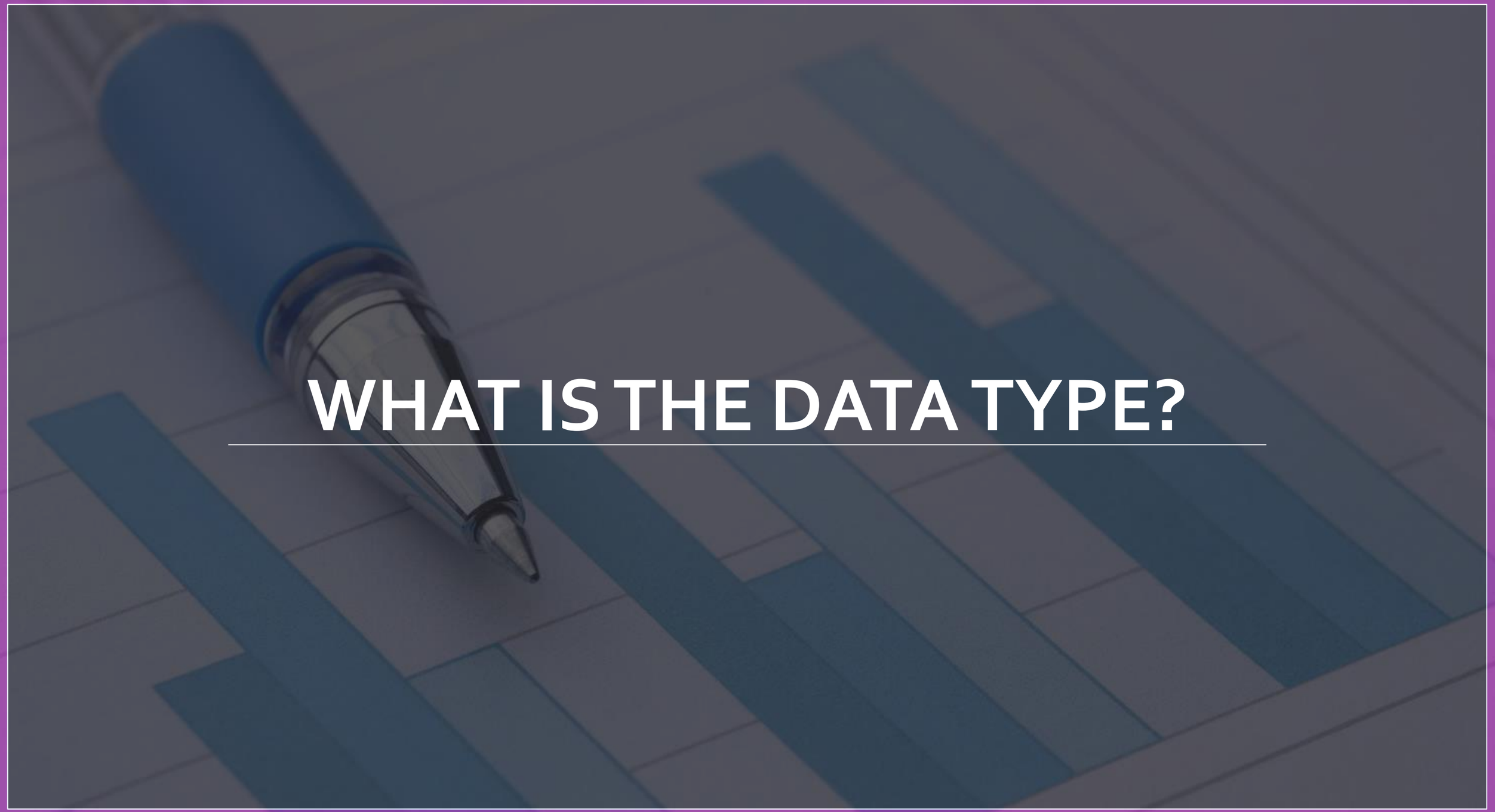


# Sort – via Standard Input



# Sort – via Command Line Argument



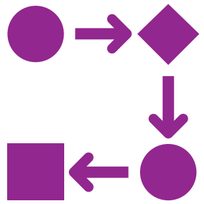


# WHAT IS THE DATA TYPE?

# Describe the data

Look for keywords in requirements, like “text” and “lines”

# How do we represent it?



Before we answer … we need to know what we are doing with it.

How we process data affects the representation

# What do we do to the data?

Input

Output

Filter – “That contains”

Sort – “In sorted order”

# How do we represent it?

Array<byte>

List<byte>

IEnumerable<byte>

string

Array<char>

List<char>

IEnumerable<char>

string[]

List<string>

IEnumerable<string>

Other ideas?

# Notice: Foreach works on Streams

- Streams of data are like collections
- Data might only be able to be visited once
- You might never get to the end of it (can be infinite stream)
- You can still “foreach” over it
- So IEnumerable describes streams as well as collections

# No Right Answer

There is no single best answer → Whatever makes your life easier → Expect to make changes later

# Assignment #2 Review

- Fill out the implementation to as much as you can
- Remember, wrong is better than nothing
- See Microsoft reference source implementation of List.cs
- Search StackOverflow.com
- Read the tests, they hold clues
- Look at the helper functions
- Consider writing your own.

# Assignment Advice

01

Find ways to make things work

02

Use system libraries at the beginning and remove them

03

Refactor afterwards

04

Solve what you can first and come back

05

Look for patterns

06

Think about arrays/list as values (data)

07

Look at the reference implementation