INTRODUCTION TO COMPUTATIONAL THINKING & OPTIMIZATION

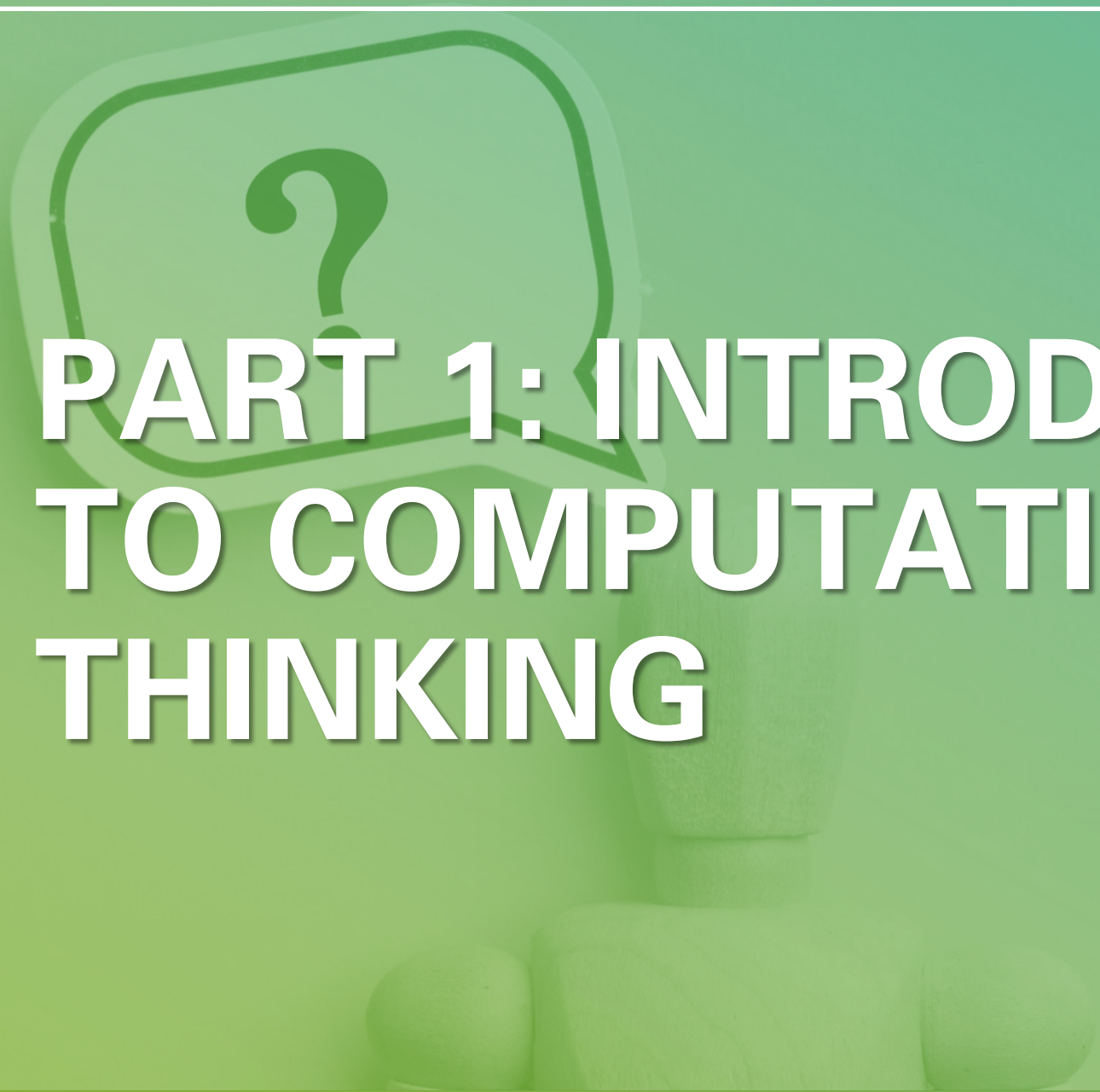
CPE 311 Computational Thinking with Python

Engr. Roman M. Richard
Instructor

Intended Learning Outcomes (ILOs)

Solve Optimization Problems using Computational Thinking

Utilize 4 elements on Computational Thinking to solve problems



PART 1: INTRODUCTION TO COMPUTATIONAL THINKING

Coding =/= Writing Program

Thinking & Analysis

Writing Program

Presentation

Coding

3 Levels of Coding

Presentation

Expressed

Middle

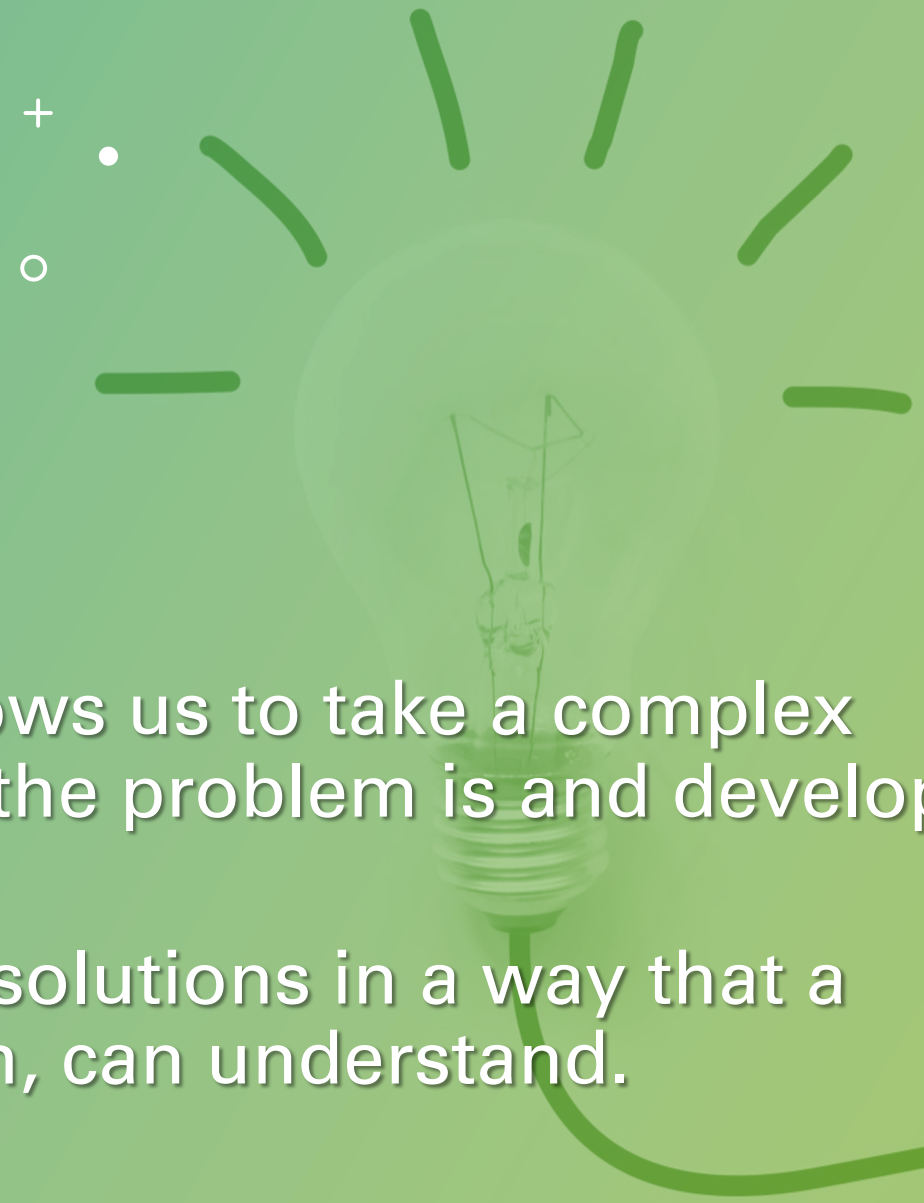
Executed

Based

Computational Thinking

Thought

What is Computational Thinking?



- Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions.
- We can then present these solutions in a way that a computer, a human, or both, can understand.

COMPUTATIONAL THINKING

Solving problems effectively with or without a computer.

There are four key techniques to computational thinking…

Decomposition

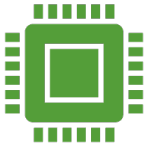
Breaking down a complex problem or system into smaller, more manageable parts.

Abstraction

Focusing on more important information only, ignoring irrelevant detail

Pattern Recognition

Looking for similarities among and within problems.



Algorithms

Developing a step-by-step solution to the problem, or the rules to be followed, to solve the problem

- Each **cornerstone** is as important as the others.
- They are like legs on a table - *if one leg is missing, the table will probably collapse*.

Decomposition

- Breaking the problem down into smaller parts means that each smaller problem can be examined in more detail.

Example Task:

Solve the crime with decomposition. Look at the picture carefully. A crime has been committed. A diamond has been stolen. **How could this complex problem of the committed crime be solved by decomposing it into smaller, simpler details individually?**


Similarities
Time
Location
WANTED
JEWEL THIEF
DIAMOND
The Crime
Witness
Evidence

Pattern Recognition

- To find patterns in problems, we look for things that are the same (or very similar) in each problem.

There are two different methods for baking the cakes in the picture.

Can you recognize any similar patterns?

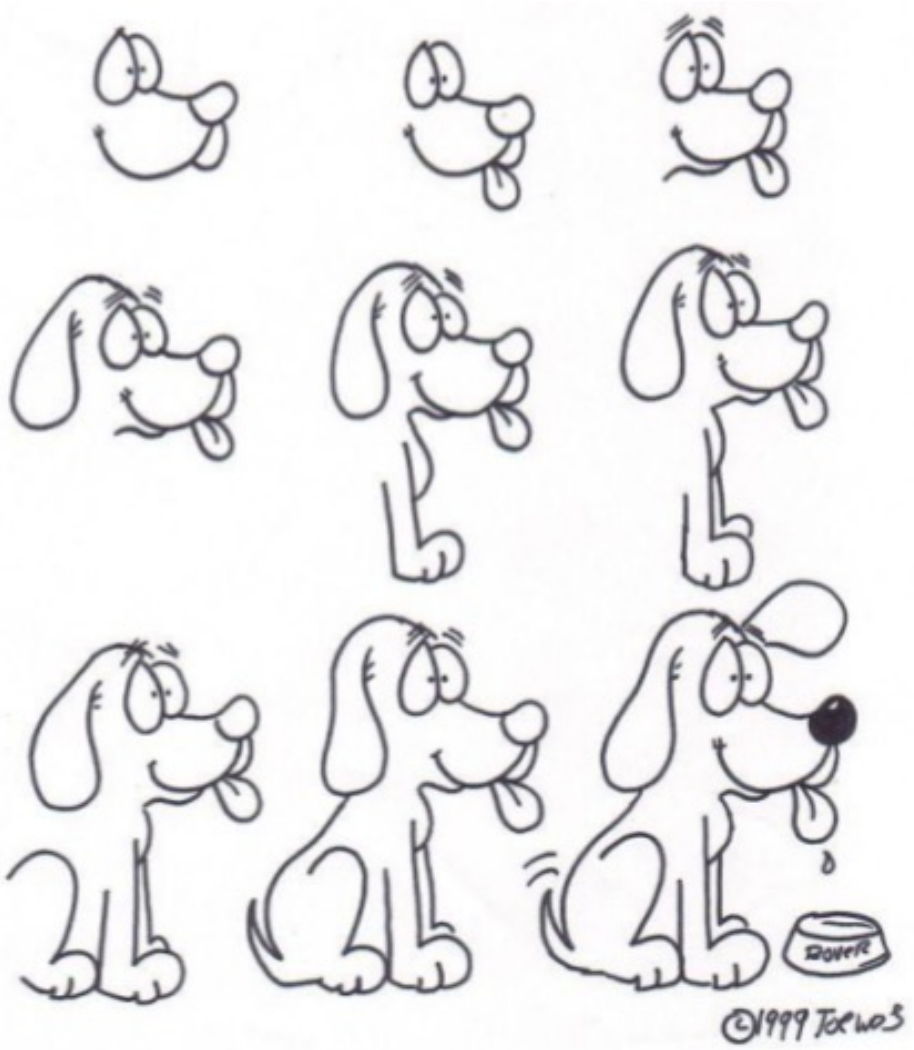
Abstraction

- Removing unnecessary detail.

Abstraction is identifying what is important and leaving out details we do not need.

Question for the class:
Is it possible to learn to drive a car without knowing how all the components work?

Once we have recognized patterns in our problems, we use abstraction to gather the general characteristics and to filter out of the details we do not need to solve our problem.

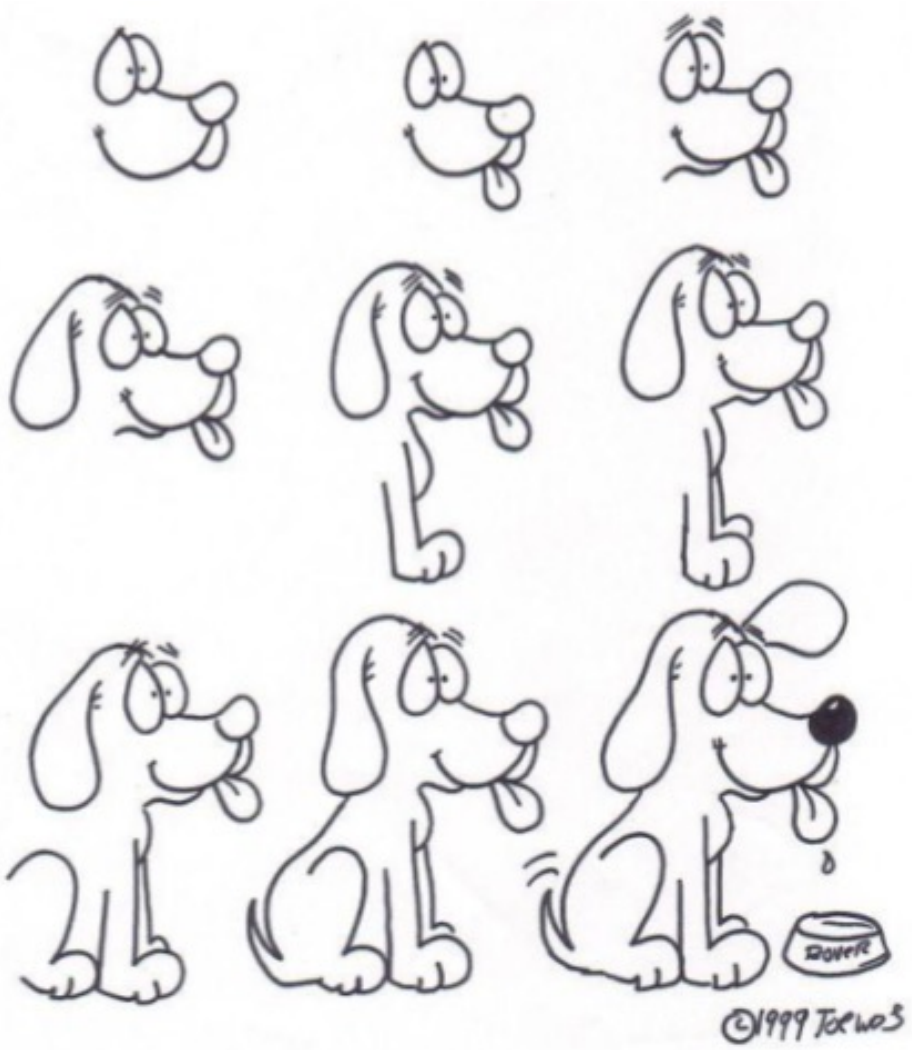


Example:

When drawing a dog, which of the following characteristics could be ignored?

Dogs run quickly.
Dogs have paws.
Dog have a nose.

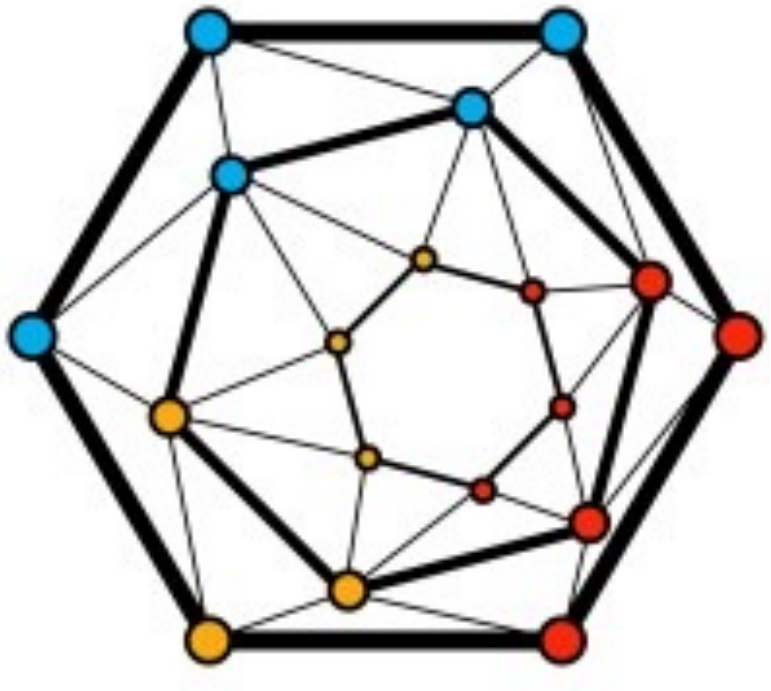
Once we have recognized patterns in our problems, we use abstraction to gather the general characteristics and to filter out of the details we do not need to solve our problem.



Example:

When drawing a dog, which of the following characteristics could be ignored?

Dogs run quickly.
Dogs have paws.
Dog have a nose.



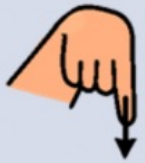
This is the final part of Computational Thinking before we can plan out a solution to our problem otherwise known as making an **ALGORITHM.**

Making toast

1

Put 1 or 2 slices of bread into the toaster.

2



Press the button on the toaster down.

3

Wait until the bread pops up.

4

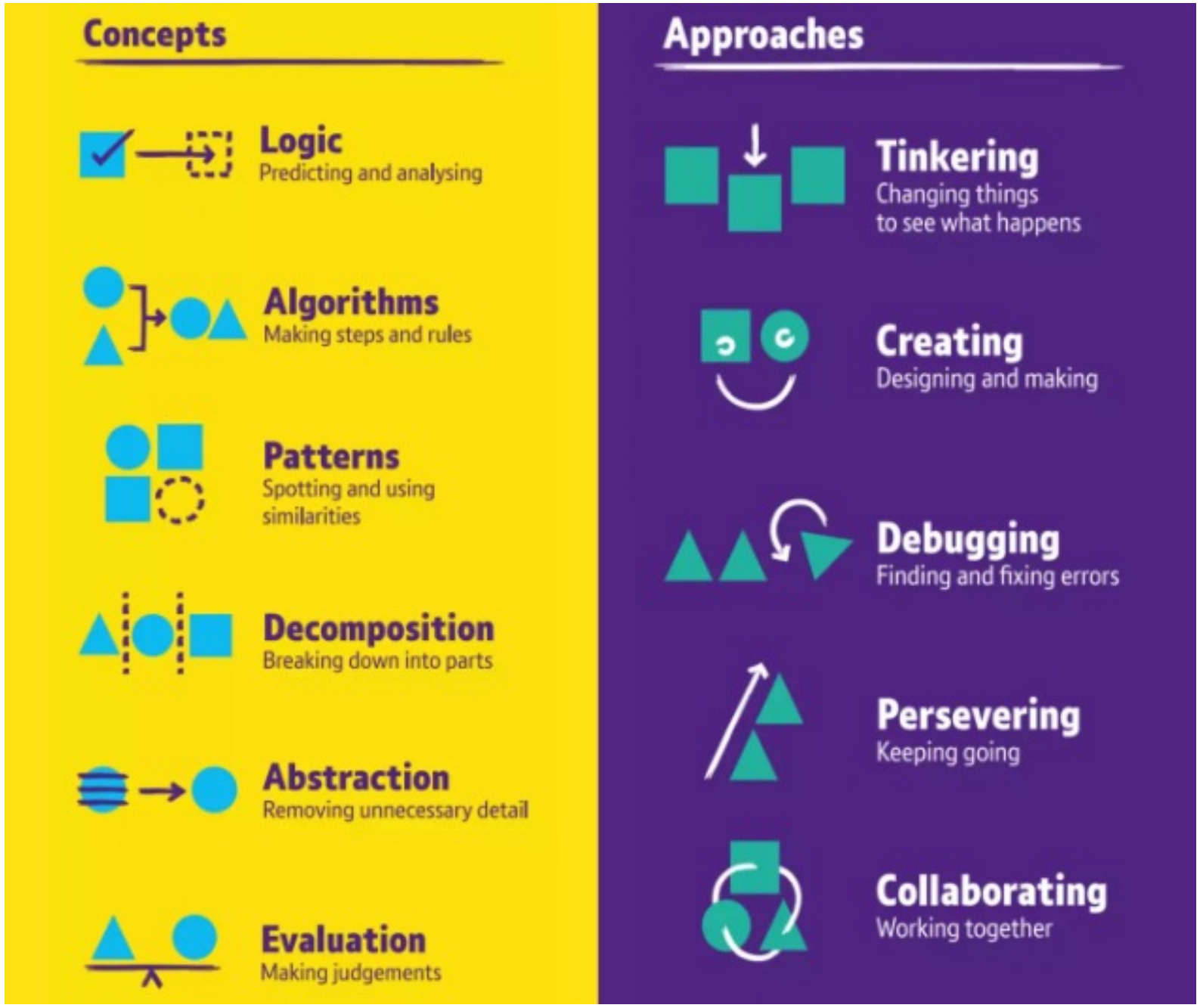
1 Wait a minute until it cools.

5

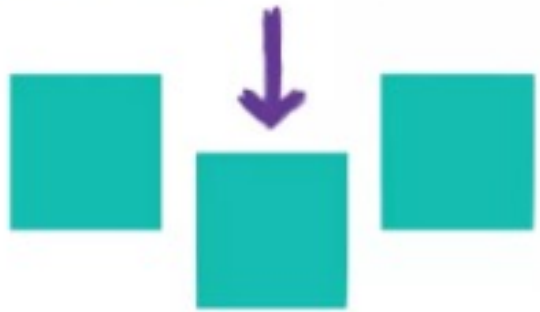
Put toast on a plate.

6

Spread butter on the toast.


Concepts
Logic
Predicting and analysing
Algorithms
Making steps and rules
Patterns
Spotting and using similarities
Decomposition
Breaking down into parts
Abstraction
Removing unnecessary detail
Evaluation
Making judgements
Approaches
Tinkering
Changing things to see what happens
Creating
Designing and making
Debugging
Finding and fixing errors
Persevering
Keeping going
Collaborating
Working together

Tinkering



Experimenting and playing

Creating

Designing and making

Debugging

Finding and fixing errors

Persevering

Keeping going

Collaborating

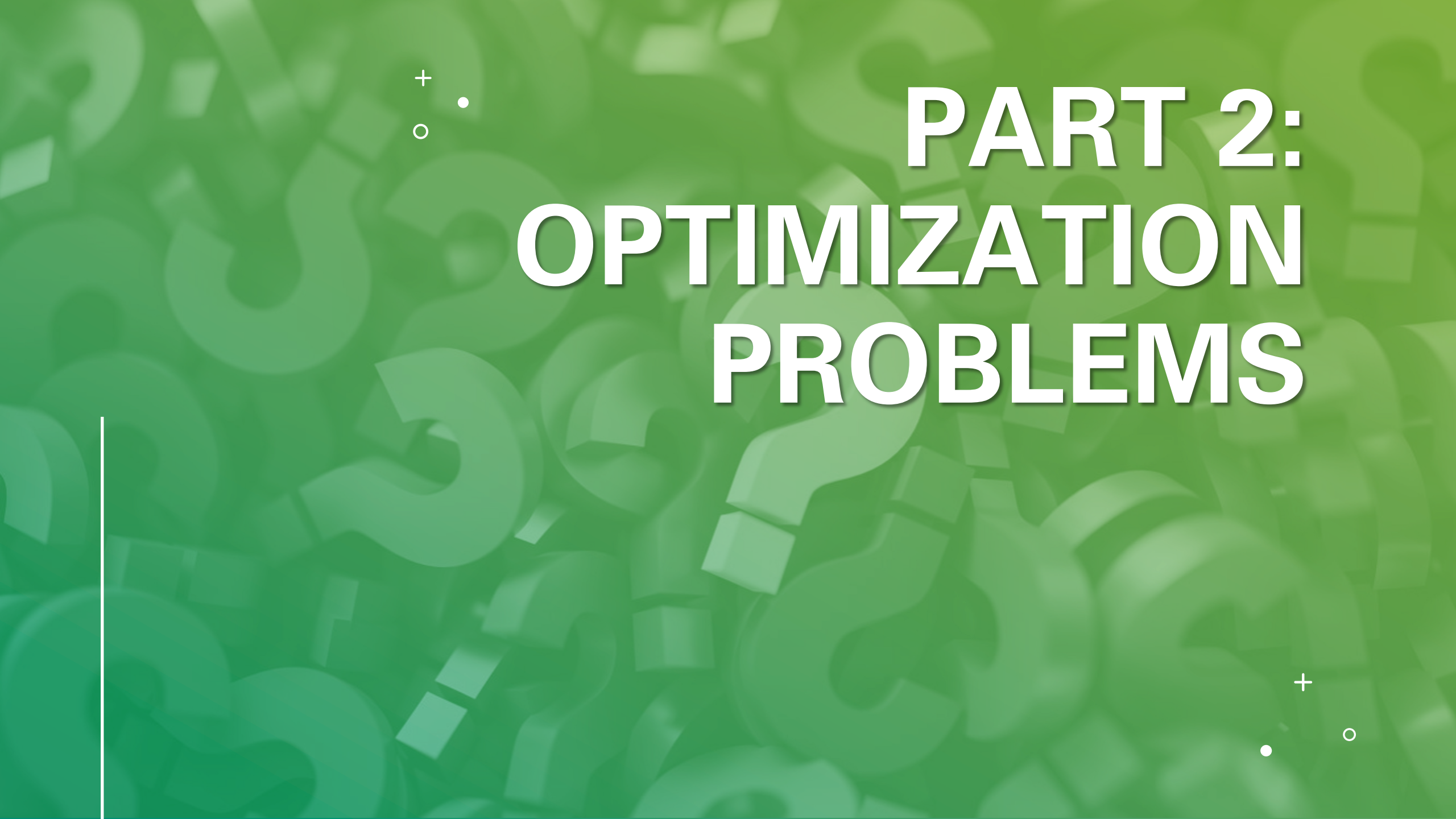
Working together

YOU HAVE NOW HAD A BASIC INTRODUCTION INTO COMPUTATIONAL THINKING. IT'S TIME TO TEST YOUR KNOWLEDGE!

Find a partner and get ready to solve the given problem (~5 mins)

Team Challenge

- Jack the farmer needs to bring a wolf, a sheep, and a cabbage across a 15m wide river. The beige wooden boat is tiny and can only one passenger at a time.
- If he leaves the wolf with the sheep, the wolf eats the sheep. If he leaves the sheep with the cabbage, the sheep eats the cabbage.
- How can he bring all three safely across the river?


PART 2:
OPTIMIZATION
PROBLEMS

Computational Models

- Using computation to help understand the world in which we live
- Experimental devices that help us to understand something that has happened or to predict the future.
  - **Optimization models**
  - Statistical models
  - Simulation models

What is an optimization model?

An objective function that is to be maximized or minimized.

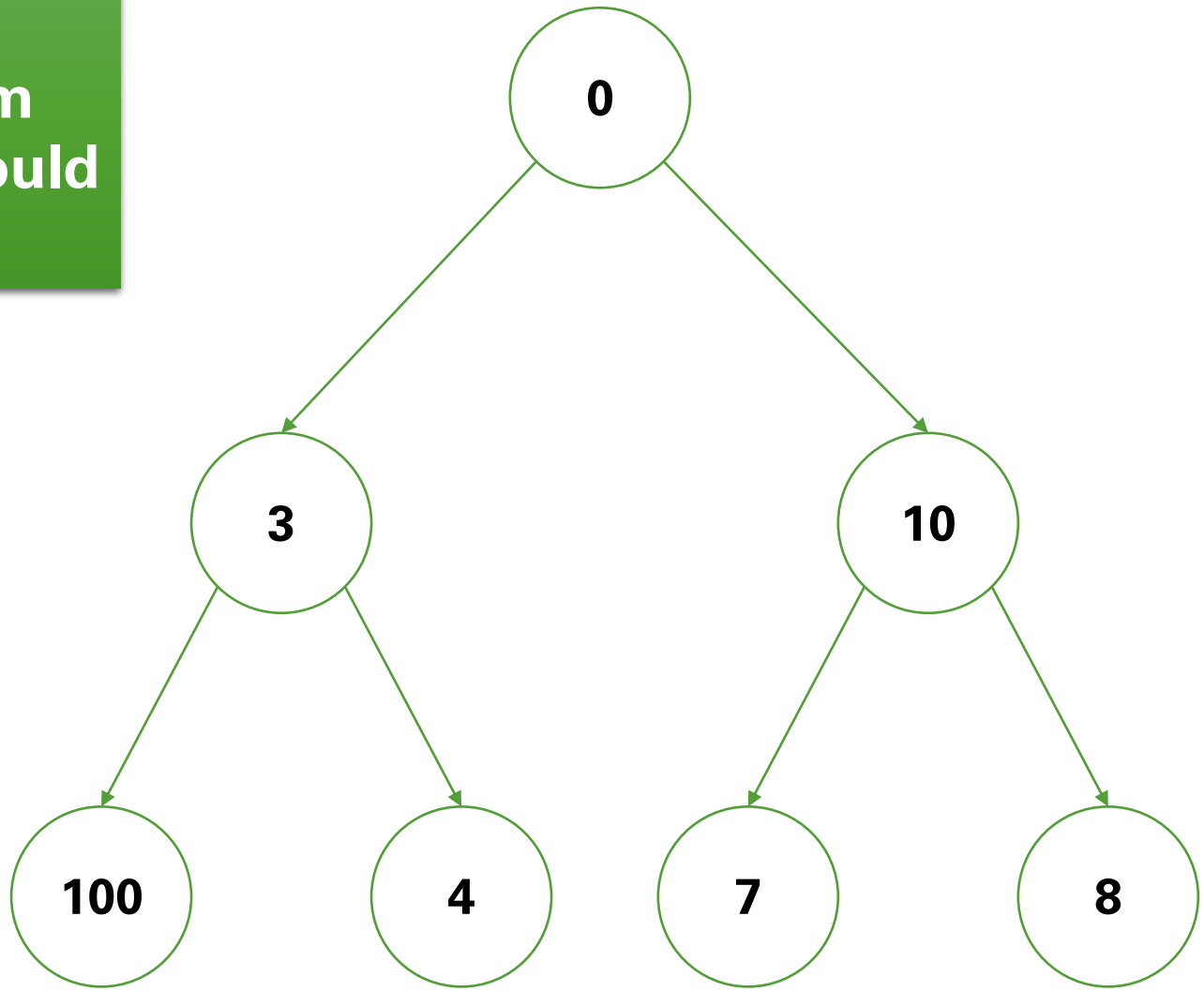
- Example: Minimize time spent traveling from A to B.

A set of constraints (possibly empty) that must be honored.

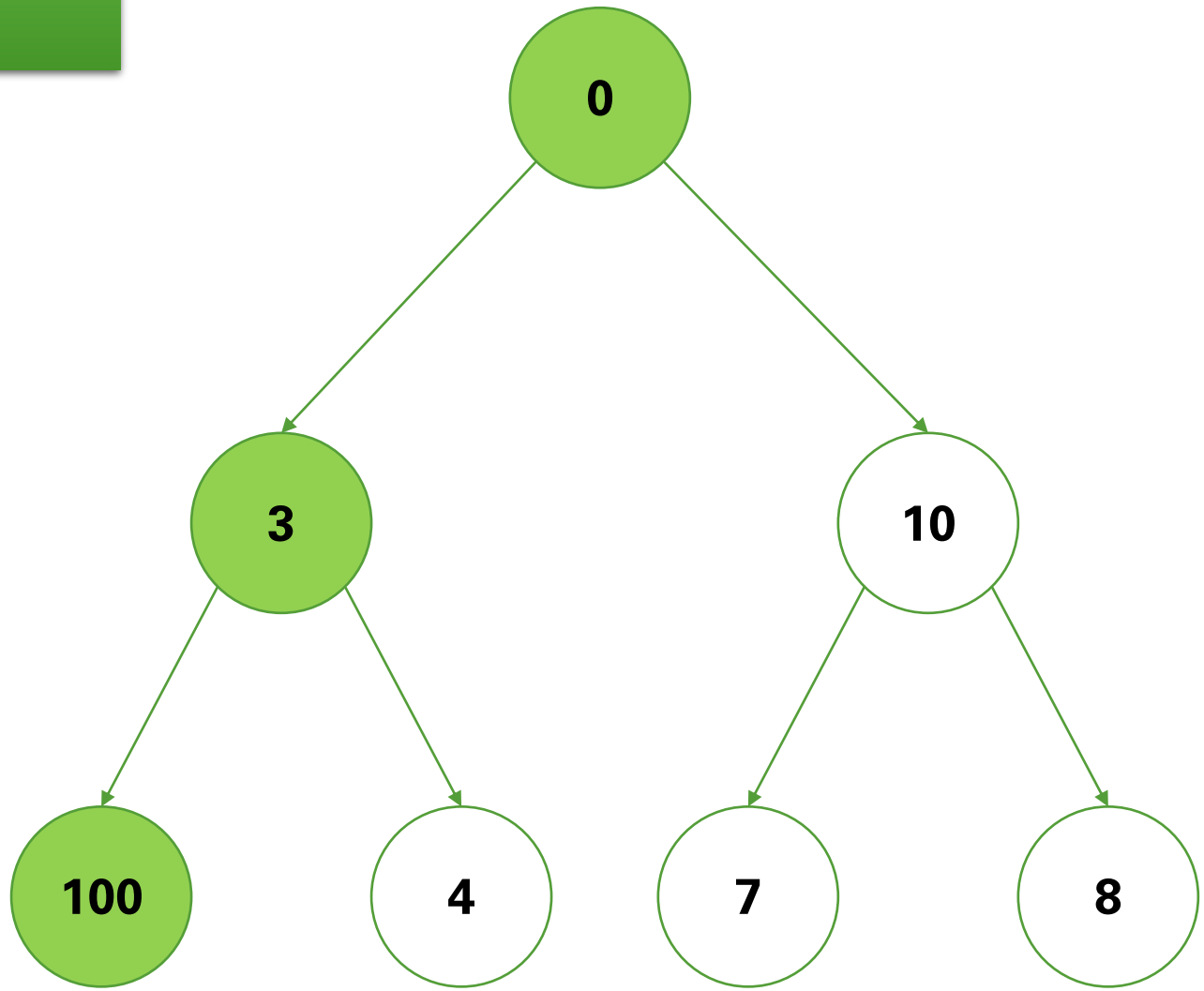
- Examples:
  - You cannot spend more than PHP 150.00.
  - You must be at school by 7:15AM.

- Many problems of real importance can be formulated as an optimization problem.
- Reducing a seemingly new problem to an instance of a well-known problem allows one to use pre-existing methods for solving them.
- Solving optimization problems is computationally challenging.

If I wanted to get the maximum value as I traverse the graph from the root node, how would you do it?

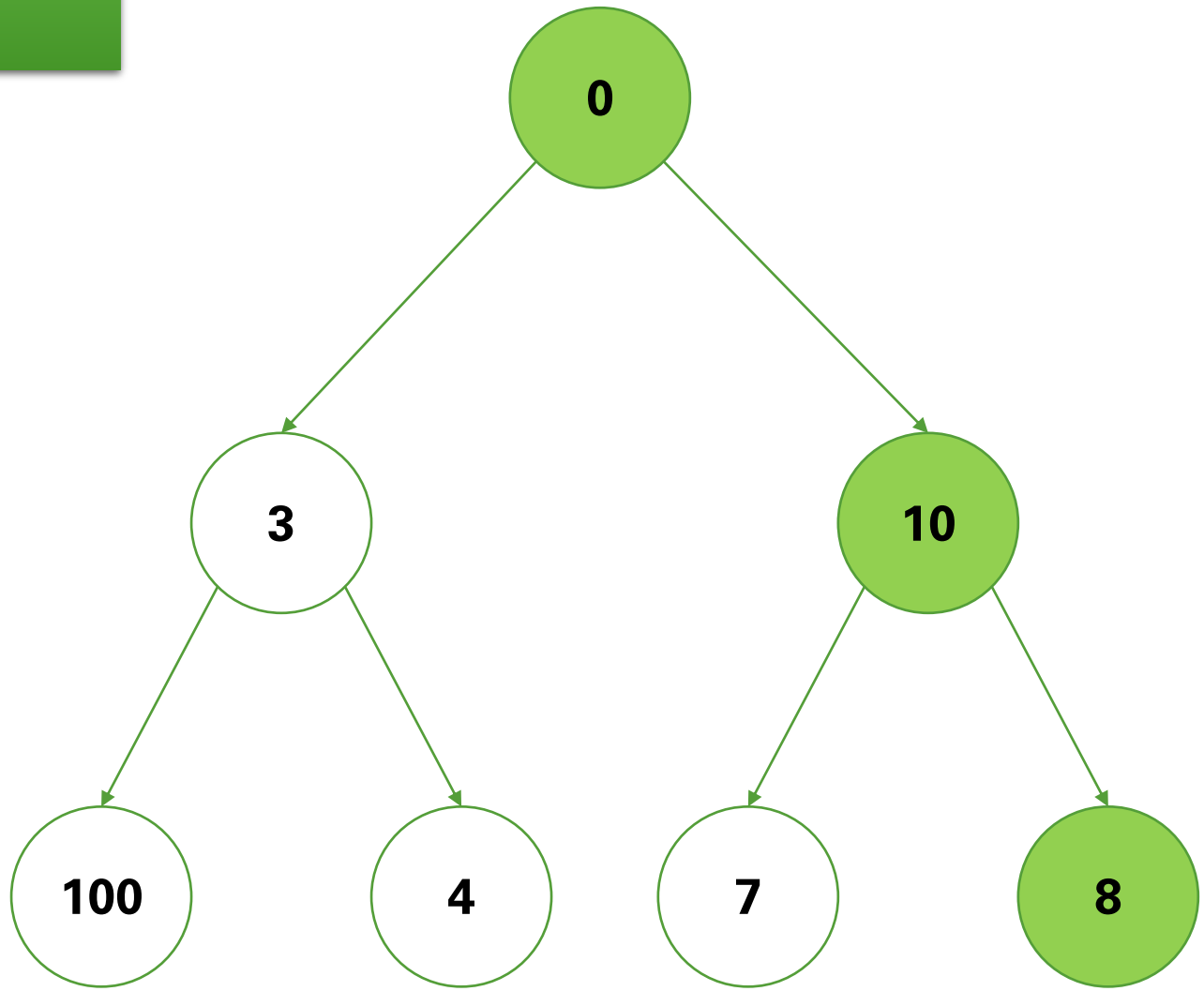


This is the optimal solution.



A **greedy algorithm** is often practical approach to finding a pretty good approximate solution to an optimization problem.

This is the greedy solution.



Knapsack and Bin-packing Problems

CC BY Mike Mozart

CC BY JOADL

CC BY Tm

Knapsack Problem

- You have limited strength, so there is a maximum weight knapsack that you can carry
- You would like to take more stuff than you can carry
- How do you choose which stuff to take and which to leave behind?
- Two variants
  - 0/1 knapsack problem
  - Continuous or fractional knapsack problem

Our least favorite knapsack problem…

CC BY File Upload Bot (Magnus Manske)

CC BY Zantastik~commonswiki

CC BY JOADL

CC BY ZooFari

CC BY maebmjj

CC BY Gorivero

0/1 Knapsack Problem

- Each item is represented by a pair **<value, weight>**
- The knapsack can accommodate items with a total weight of no more than **w**.
- A vector, **L**, of length **n**, represents the set of available items. Each element of the vector is an item.
- A vector, **V**, of length **n**, is used to indicate whether items are taken or not.
  - If V[i] = 1, item L[i] is taken.
  - If V[i] = 0, item L[i] is not taken.

0/1 Knapsack Problem (Formalized)

- Find a value of V that maximizes:

$$\sum_{i=0}^{n-1} V[i] * I[i].value$$

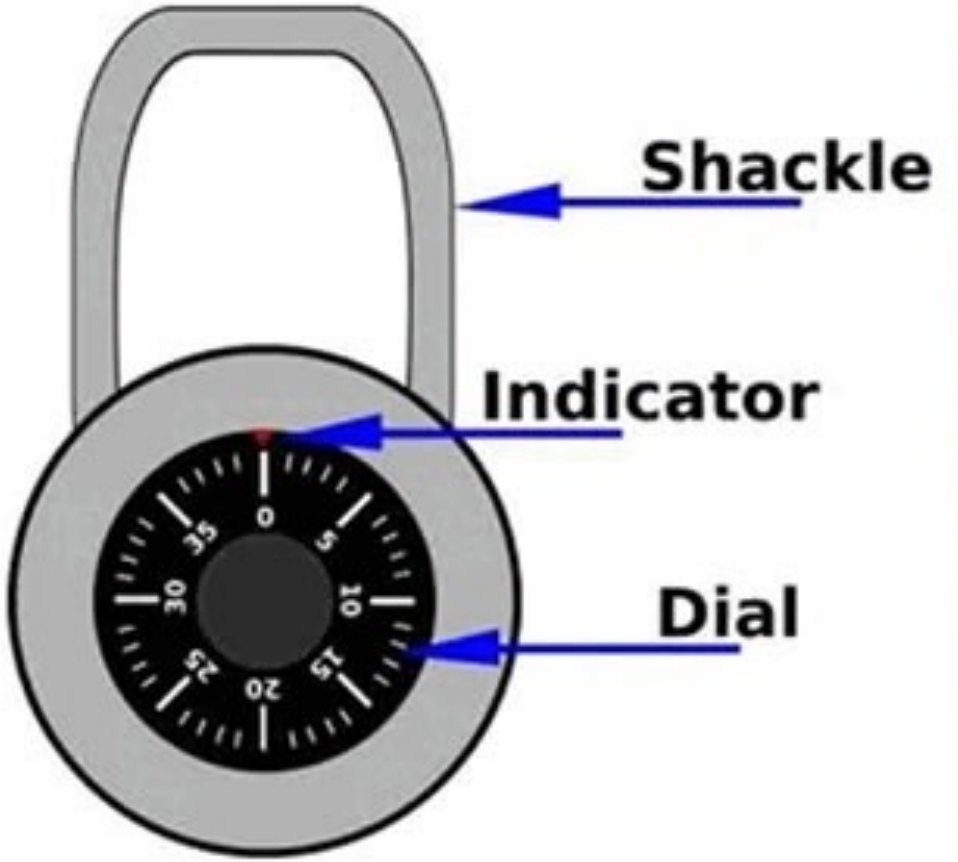
- Subject to the constraint that:

$$\sum_{i=0}^{n-1} V[i] * I[i].weight \leq w$$

Brute Force Algorithm

- Step 1: Enumerate all possible combinations of items. That means, generating all subsets of the set of subjects. This is called the **power set**.
- Step 2: Remove all the combinations whose total units exceeds the allowed weight.
- Step 3: From the remaining combinations, choose any one whose value is the largest.

Often Not Practical



- Recall
  - A vector, V, of length n is used to indicate whether or not items are taken.
    - If V[i] = 1, item I[i] is taken. If V[i] = 0, item I[i] is not taken.
- How many possible values can V have? 2^n bits.
- So, if there are 100 items to choose from, the power set's size is 1,267,650,600,228,229,401,496,703,205,376

Are we just being pedantic?

- Alas, no.
- 0/1 Knapsack Problem is inherently ***exponential***.
- But don’t despair!

0/1 Knapsack Problem is Inherently Exponential

01

Give up

02

Approximate Solution

03

Exact solution that is often fast

Greedy Algorithm

- The Greedy Algorithm is a **practical alternative**.
- While the knapsack is not “full”, put the “best” available items in the knapsack.

- What is the best item?
  - Might be the most valuable, the least expensive, or the highest value/units.

An Example

Consider the following situation:

1. You are about to sit down for a meal.
2. You know how much you value different foods.
3. But you have a calorie budget of not more than 800 calories.

Determine: What can you eat?

Example

- Given the menu:

Food	Wine	Beer	Pizza	Burger	Fries	Coke	Apple	Donut
Value	89	90	30	50	90	79	90	10
Calories	123	154	258	354	365	150	95	195

- Find what you can decide to order.

Procedure:

1. Create a Food Class
2. Create a function that can build the menu of food based on the given example.
3. Implement the greedy algorithm.
4. Create a function to test the greedy algorithm.
5. Use the greedy algorithm.

Assessment Tasks (ATs)

- Module 1: Hands-on Activity 1.1 Optimization and Knapsack Problem
- Quiz on Computational Thinking