Procedural Abstraction

CMPT 145

Copyright Notice

Learning Objectives

After studying this chapter, a student should be able to:

- Define the concept of procedural abstraction.
- Give an example of procedural abstraction, and explain how it helps meet design and implementation goals.
- Describe the components of a function's interface documentation.
- Write interface documentation for simple functions.
- Describe the role of procedural abstraction as part of the implementation process.
- Describe the role of procedural abstraction as part of the design process.
- Apply procedural abstraction during stepwise refinement.

Procedural Abstraction

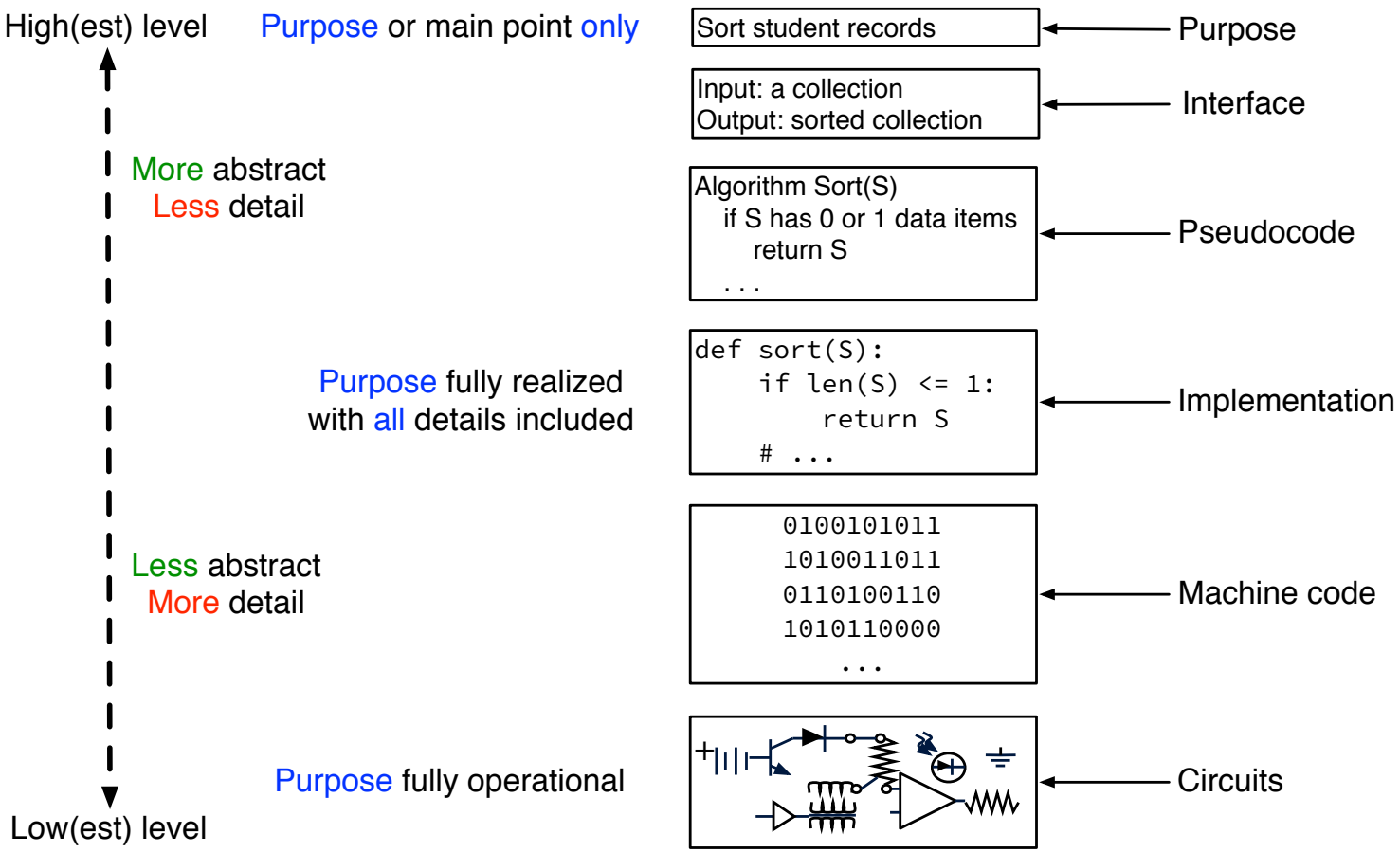
- When we define a function, we are creating a procedural abstraction.
- Allows us to view software from two perspectives:
  - Purpose
    - What is the point of the function? What purpose does the code achieve?
  - Implementation
    - How does the function work? What steps are needed?

Benefits

Procedural abstraction enhances:

- Correctness
- Reusability
- Adaptability

Levels of Procedural Abstraction



Interface Documentation

- A function interface defines a function's input-output relationship.
- All interfaces must be documented.
  - Purpose
    - What does the function do?
  - Pre-Condition(s)
    - Parameters and constraints on them, if any
  - Post-Condition(s)
    - Effects outside of function, if any
  - Return
    - Values returned by the function, if any

When to document your function

- Document your function after it's implemented if:
  - You're working really well; you know exactly what you're doing; and you don't want to break your flow.
  - An assignment describes the function completely.
- Document your function before you implement it if:
  - You are not sure how the function will work.
  - You are designing a bunch of functions that work together.
  - The problem you are solving is not completely defined by someone else.

Demo 0

We will examine some Python code and document its interface using docstrings.

Demo 1

Describe the interface, as a docstring, for this Python function:

```
def positive_evens( numbers ):
    return [x for x in numbers if x % 2 == 0 and x > 0]
```

Demo 2

Define an interface, as a docstring, for this Python function:

```
1 def set_currency( customers,country,currency ):
2     for customer in customers:
3         if customer["country"] == country:
4             customer["currency"] = currency
```

Demo 3

Write an interface (only) for the following abstract purposes:

- Find all the prime factors of a given positive integer $x > 1$
- Remove all the duplicates of a list.
- Microsoft Excel uses letters for column labels. Translate column strings to integers.

Demo 3

Let's refactor the sieve script, and make it a function!

Generalization and Abstraction

- Abstraction hides decisions within a function.
- Generalization exposes decisions to the function caller.
- Combining these two ideas is the essence of procedural abstraction.