Week 9

Pre-Lecture Slides: Top-Down Design

Learning Objectives

ENGINEERING
TEXAS A&M UNIVERSITY

Top-Down Design of Programs

- Create appropriately detailed top-down hierarchies when given a primary (complex) goal
- Explain and give examples of advantages and disadvantages of top-down design
- Define and create trees, roots, nodes, parent, children and leaves in this context
- Utilize a top-down design approach as part of creating a Python program

Create a curriculum…

You are tasked with creating a curriculum for a college degree

- You can require up to 40 courses
- How do you determine which courses should be included in the overall curriculum?

ENGINEERING
TEXAS A&M UNIVERSITY

Curriculum – Option 1:

Start listing classes you think are valuable:

Calculus II	History	Thermodynamics	Maltese Lore	Physics	. . .

ENGINEERING
TEXAS A&M UNIVERSITY

Curriculum – Option 1:

Start listing classes you think are valuable:

Calculus II	History	Thermodynamics	Maltese Lore	Physics	. . .

Problems:

- How do you know you didn't miss an important area?
- How do you ensure the right balance?

ENGINEERING
TEXAS A&M UNIVERSITY

A better option

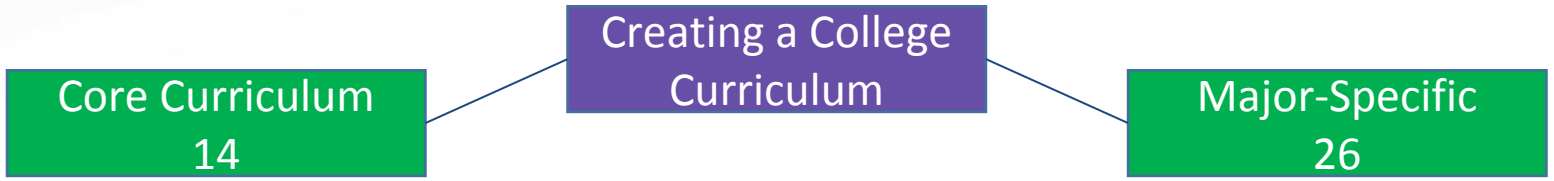
Start with the most general idea.

"Creating a College Curriculum"

ENGINEERING
TEXAS A&M UNIVERSITY

A better option

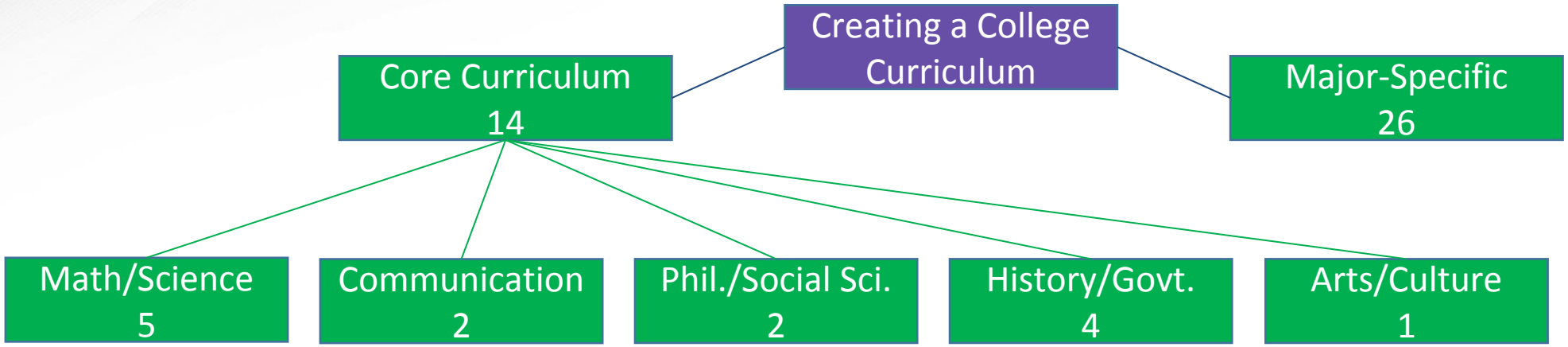
Divide it into the next-most general conceptual units.



ENGINEERING
TEXAS A&M UNIVERSITY

A better option

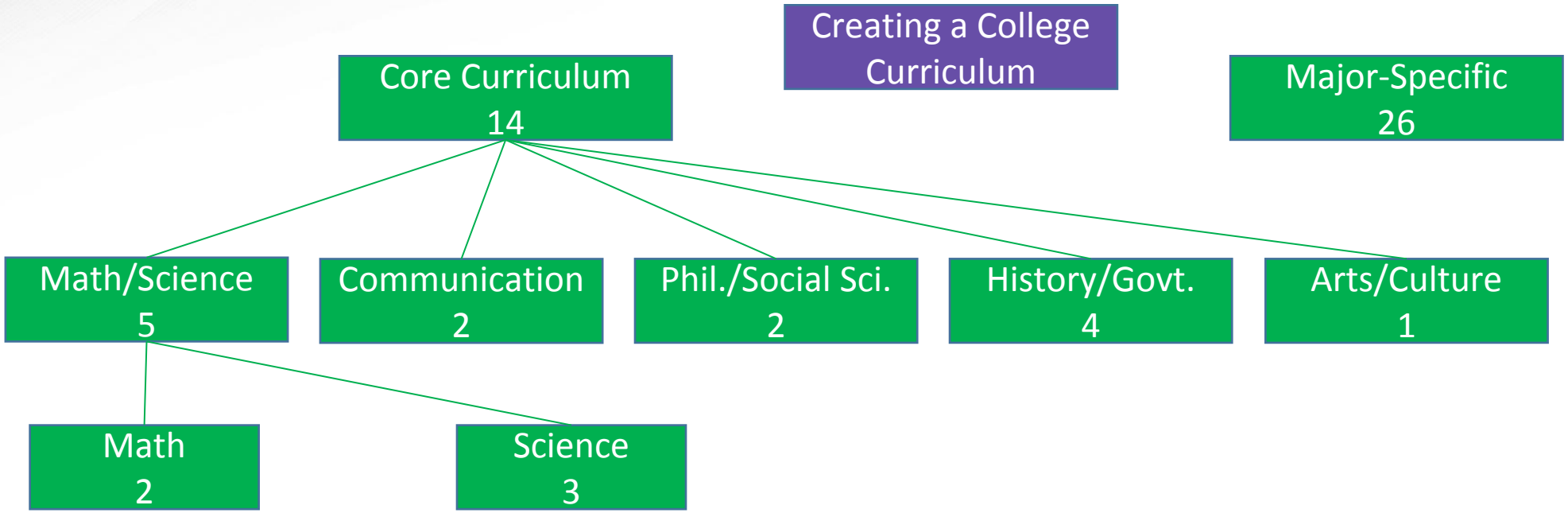
Repeat until the units are obvious (e.g., specific courses).



ENGINEERING
TEXAS A&M UNIVERSITY

A better option

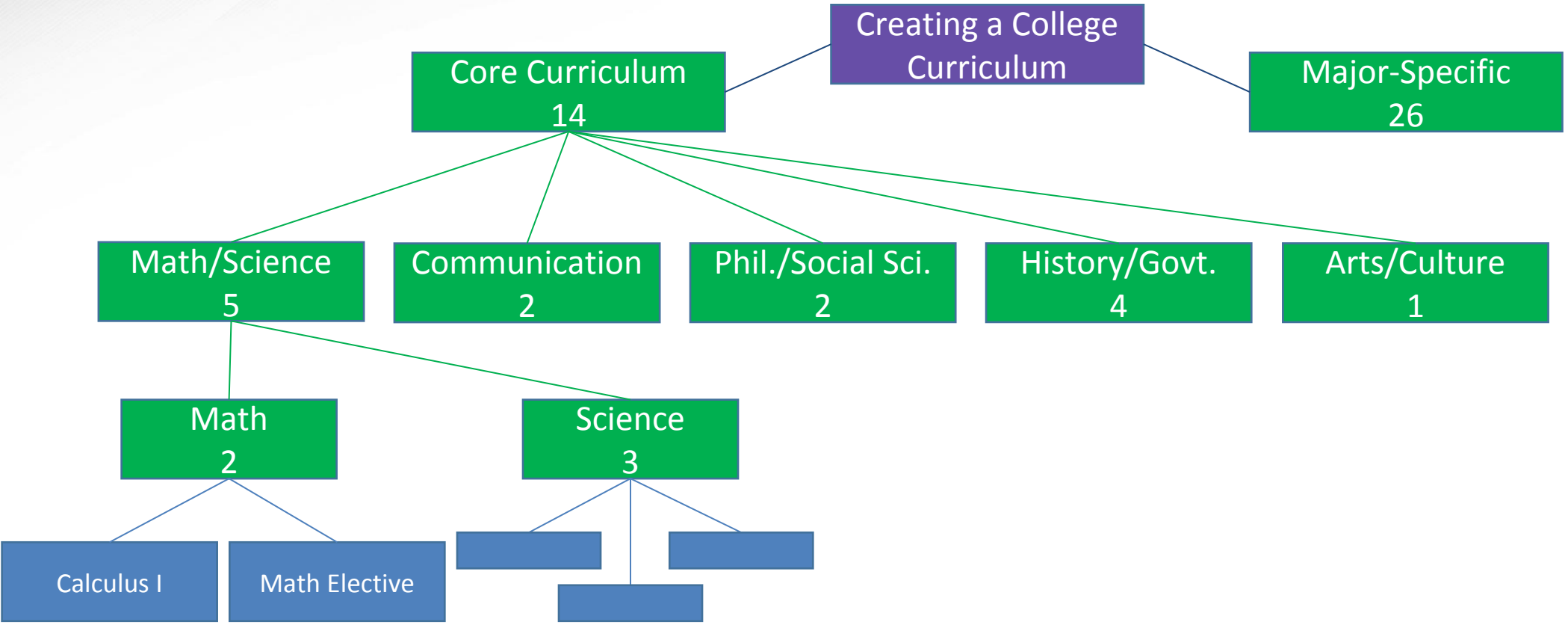
Repeat until the units are obvious (e.g., specific courses).



ENGINEERING
TEXAS A&M UNIVERSITY

A better option

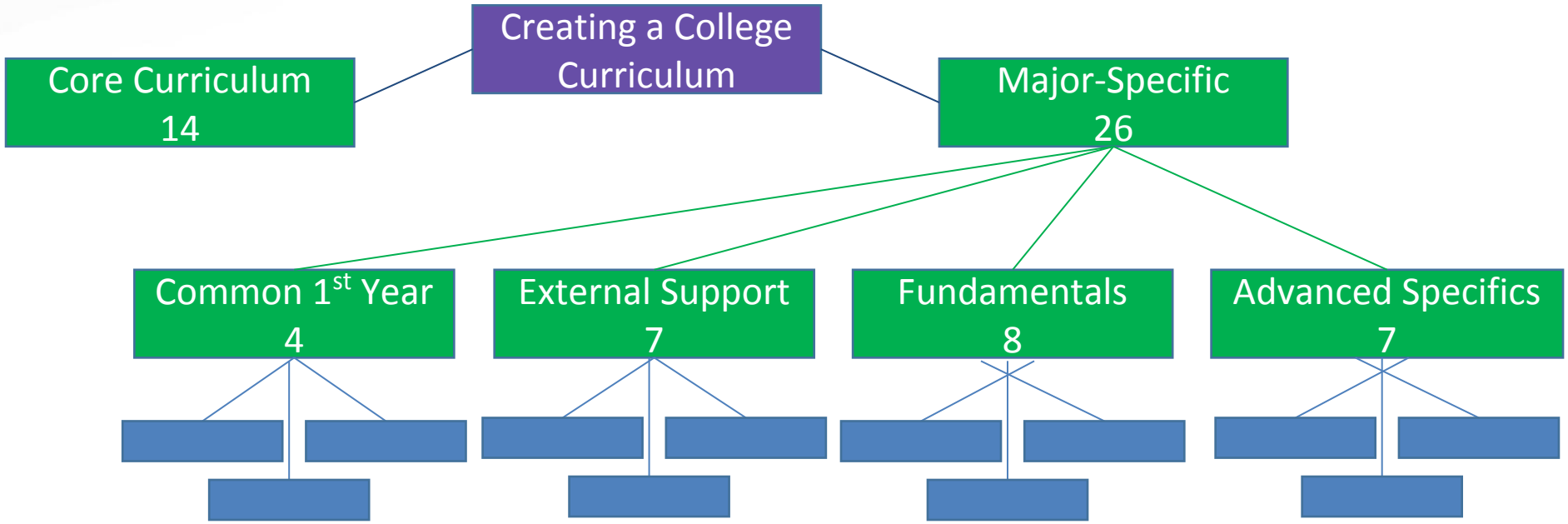
Repeat until the units are obvious (e.g., specific courses).



ENGINEERING
TEXAS A&M UNIVERSITY

A better option

Repeat until the units are obvious (e.g., specific courses).



Some Computing Terminology

Hierarchies like this in computing are usually called “Trees”

- The tree has a “root” at the base
- Individual elements are often called “nodes”
- Nodes have:
  - A “parent” (the node just above; the root has no parent)
  - Possibly “children” (the nodes that descend from it, below)
  - Nodes without children are called “leaves”

Top-Down Hierarchies

This approach is called the top-down approach

- Start with the most general idea (“Create a curriculum”)
- Divide it into the next-most general conceptual unit (“Core”, “Major”)
  - Not everyone will divide these the same; i.e., an alternative set may be “ABET Required”, “State Required”, “University Required”, etc.
- Repeat the process until the unit is obvious (“Course”)

The end result is a hierarchy

- The individual units should be **coherent** and **distinct**

Top-Down Design

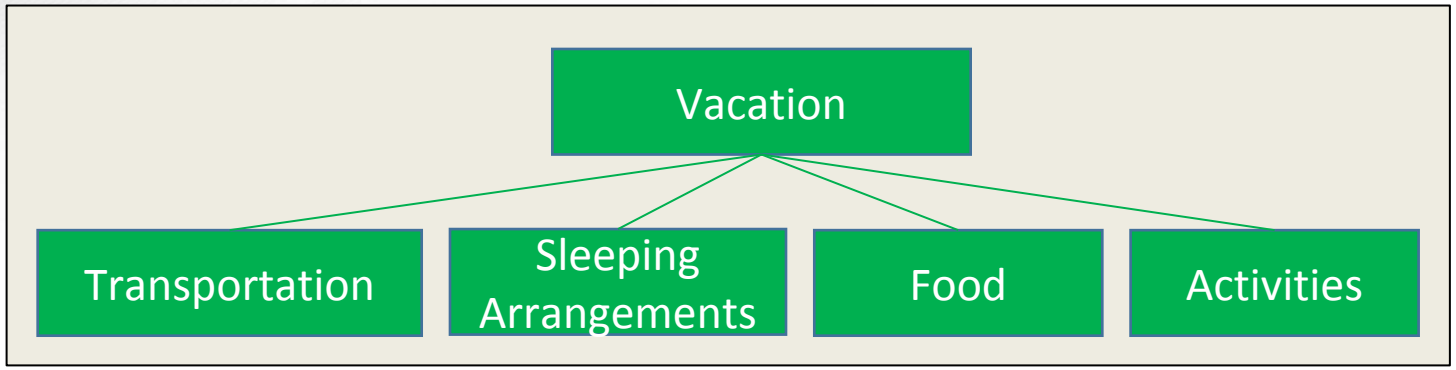
- Top-down design refers to tackling a problem by breaking it down into a hierarchy from the top-most (root) level downward.
- There is not a “right” or “wrong” design
- This method can help solve many problems, including engineering challenges

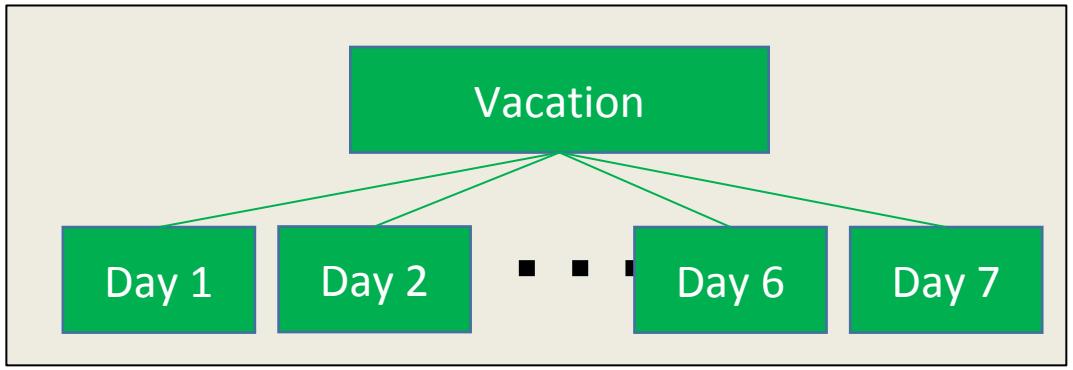
Example/Exercise:

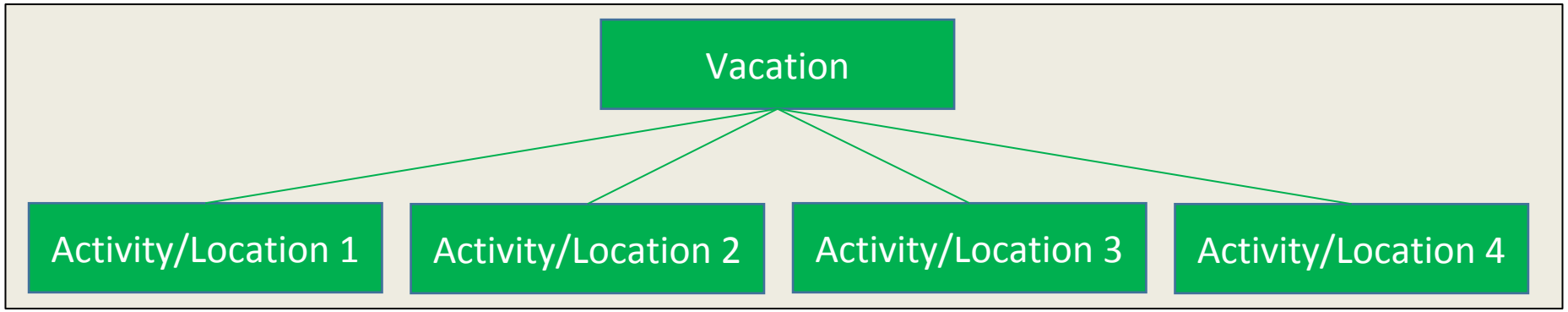
Say you want to plan a week-long vacation (maybe a road trip across the Southwest US, or a beach-hopping trip in Hawaii)

- What do you choose as the first level in a top-down “design”?

Vacation – 3 options







Top-Down Program Design

We'll use top-down design as a way of organizing many of our programs

- Break the problem into individual "large" steps
- Break those into smaller steps
- Stop once the code is "obvious" from the description
- Typically, this is once implementing a concept will take only a few lines of code (on the order of 1 - 10 lines of code)
- Can turn the nodes into comments to help show structure

Example

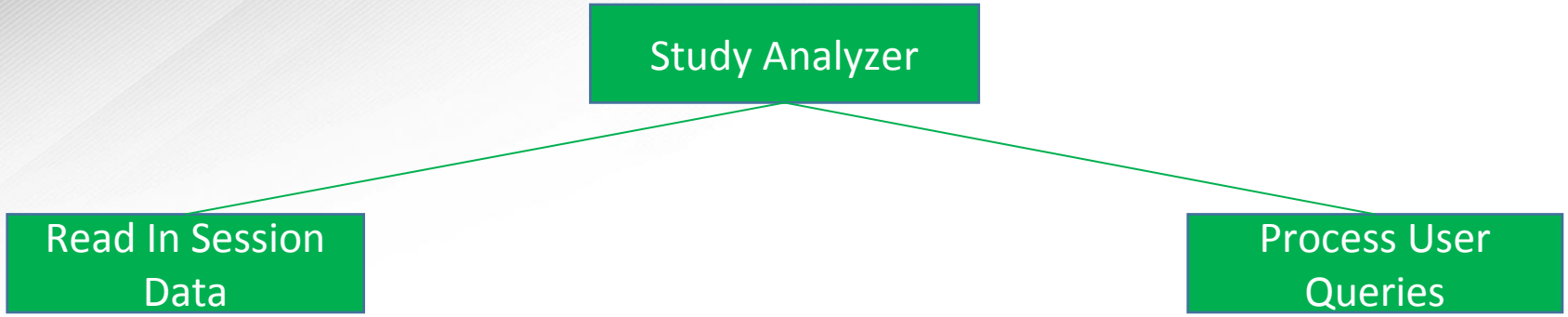
Let's write a program to track how long we studied for various tests.

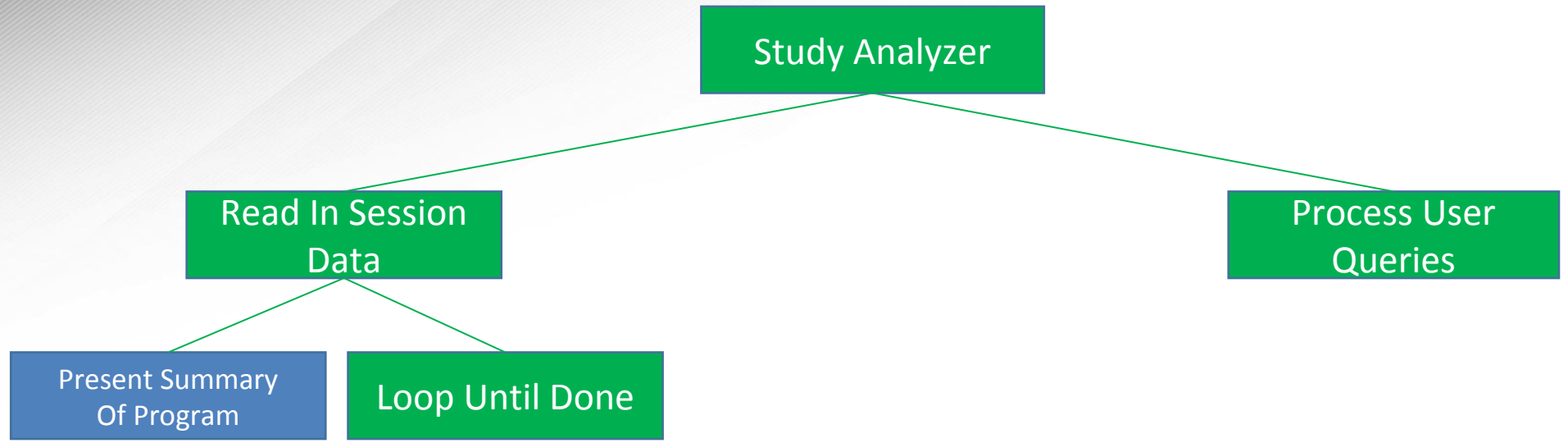
We want to record:

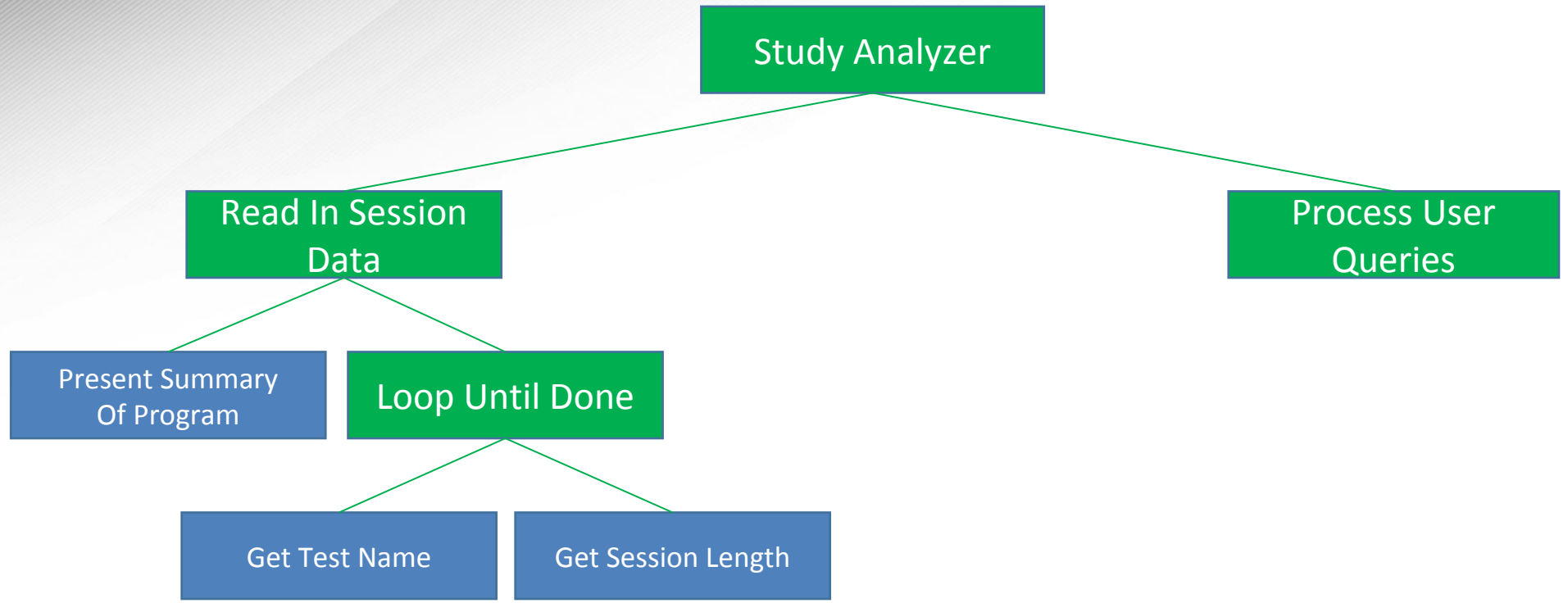
- which test we studied for, and
- the length of time of each study session

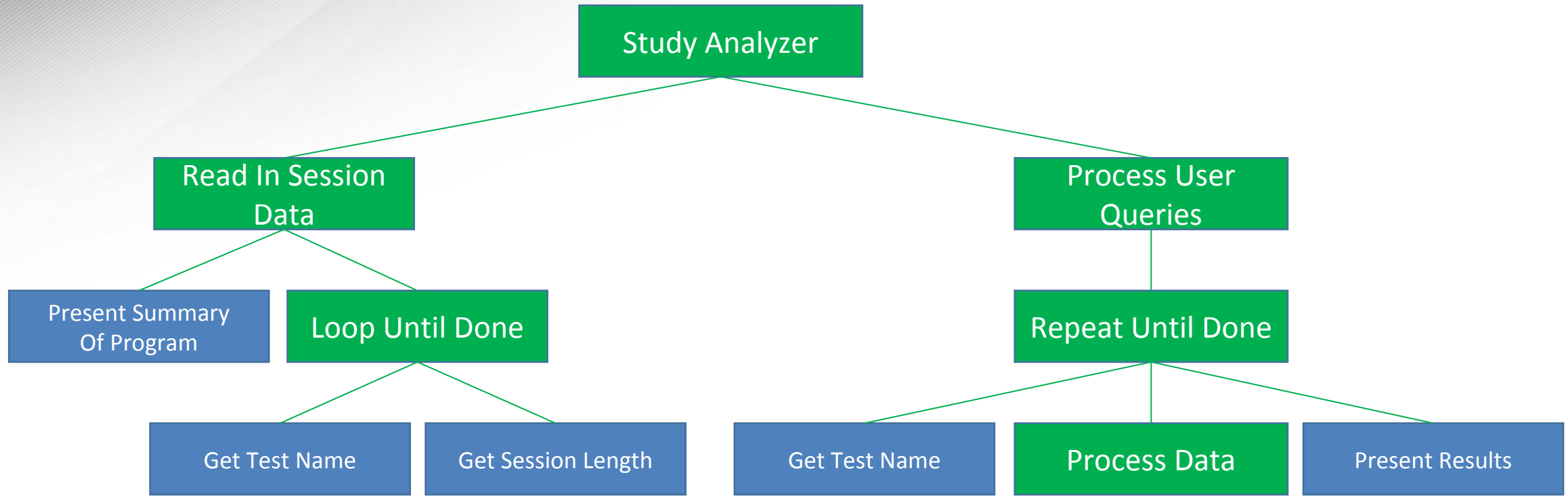
We want an output of:

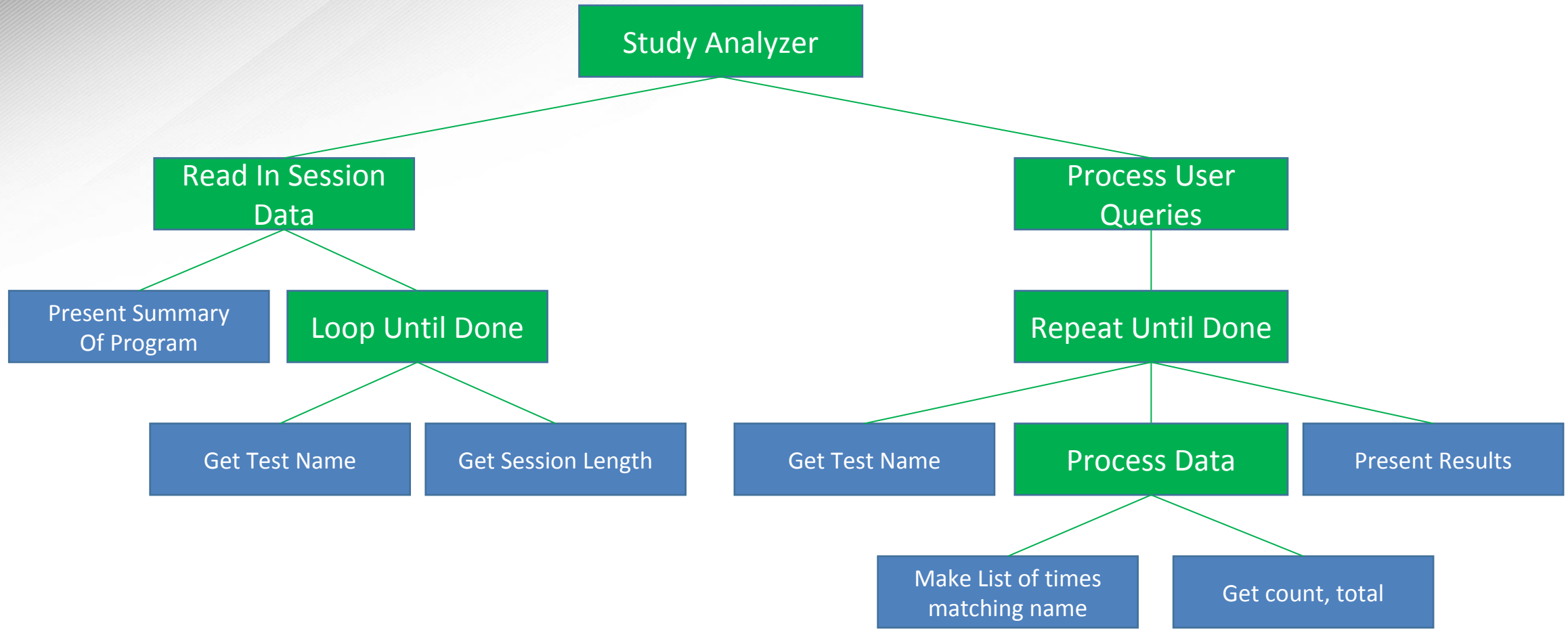
- the total time studied for a given test


Study Analyzer
Read In Session Data
Process User Queries


Study Analyzer
Read In Session Data
Process User Queries
Present Summary Of Program
Loop Until Done


Study Analyzer
Read In Session Data
Process User Queries
Present Summary Of Program
Loop Until Done
Get Test Name
Get Session Length


Study Analyzer
Read In Session Data
Process User Queries
Present Summary Of Program
Loop Until Done
Repeat Until Done
Get Test Name
Get Session Length
Get Test Name
Process Data
Present Results


Study Analyzer
Read In Session Data
Process User Queries
Present Summary Of Program
Loop Until Done
Repeat Until Done
Get Test Name
Get Session Length
Get Test Name
Process Data
Present Results
Make List of times matching name
Get count, total

Now for coding

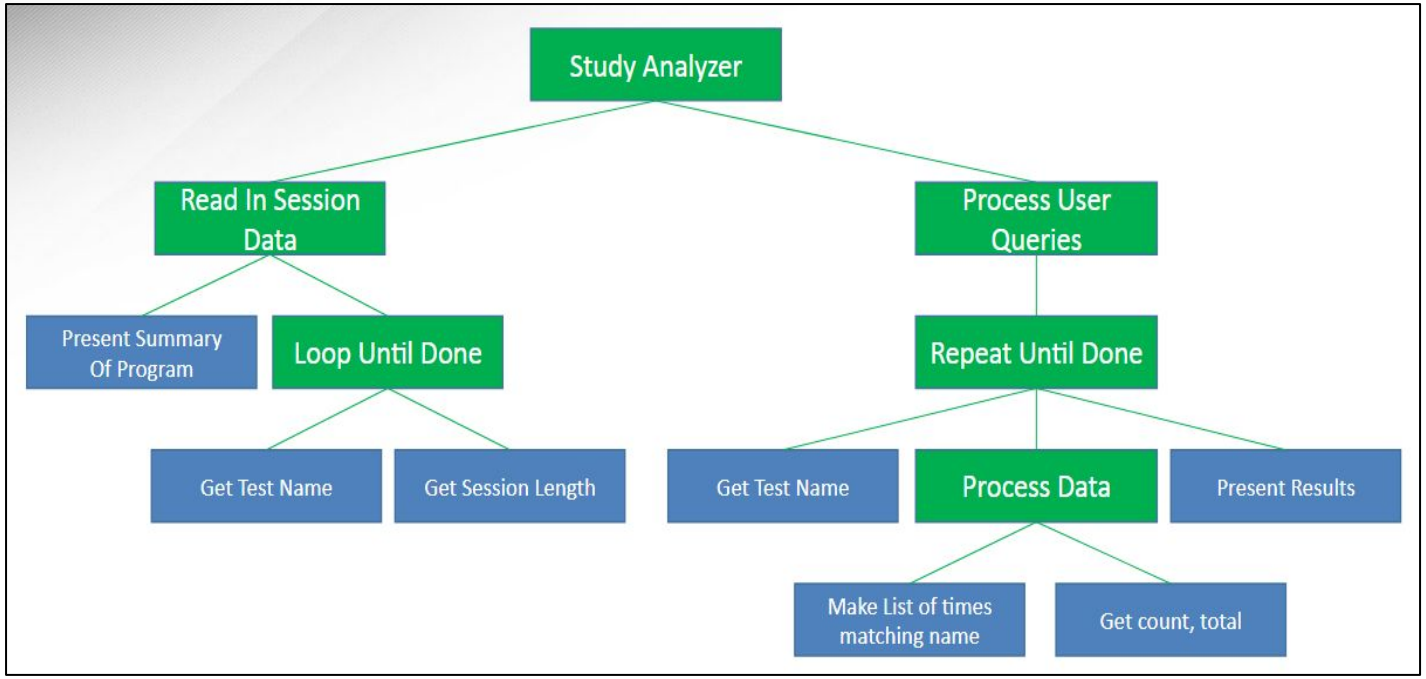
First, convert the nodes to comments

- We're again attempting to have our planning stage help our implementation stage be easier, faster and more straight-forward.

Then, filling in the details should be "obvious"

- 'Nodes' are like headings, and each 'leaf' is a placeholder for 1-10 lines of code. Here, you code each 'leaf'.
- Expect to make some adjustments to your plan.

```
#########################
### STUDY ANALYZER   ###
#########################

#### Read In Session Data ####
# Present Summary / Purpose of Program

# Loop Until Done Entering Study Sessions
    # Get Test Name
    # Get Session Length

#### Process User Queries ####
# Loop Until Done Entering Test Names
    # Get Test Name

    # Process Data
    # Make List of times that match test name
    # Get statistics

    # Present Results
```



```
##########################
### STUDY ANALYZER   ###
##########################

#### Read In Session Data ####
# Present Summary / Purpose of Program
print("This is a program to let you find the amount of time you studied for various tests.")

# Loop Until Done Entering Study Sessions
    # Get Test Name
    # Get Session Length

#### Process User Queries ####
# Loop Until Done Entering Test Names
    # Get Test Name

    # Process Data
    # Make List of times that match test name
    # Get statistics

    # Present Results
```

```
# I broke apart the code for the presentation slides #

# Loop Until Done Entering Study Sessions
more_to_enter = True
names = []
lengths = []
while more_to_enter:
    # Get Test Name
    test_name = input("Enter which test you studied for.  Enter NONE to stop: ")
    if test_name == "NONE":
        more_to_enter = False
    # Get Session Length
    if more_to_enter:
        study_length = int(input("Enter how many minutes you studied in this session: "))
        names.append(test_name)
        lengths.append(study_length)

#### Process User Queries ####
# Loop Until Done Entering Test Names
    # Get Test Name

    # Process Data
    # Make List of times that match test name
    # Get statistics

    # Present Results
```

```
# I broke apart the code for the presentation slides #

#### Process User Queries ####
# Loop Until Done Entering Test Names
more_to_enter = True
while more_to_enter:
    # Get Test Name
    test_name = input("Which test do you want data for?  Enter NONE to stop: ")
    if test_name == "NONE":
        more_to_enter = False
        break

    # Process Data
    # Make List of times that match test name
    studylengths = []
    for i in range(len(names)):
        if (test_name == names[i]):
            studylengths.append(lengths[i])
    # Get statistics
    num_sessions = len(studylengths)
    total_time = 0
    for i in studylengths:
        total_time += i

    # Present Results
    print("You studied for the",test_name,"test in",num_sessions,"sessions, for a total of",total_time,"minutes")
```

Advantages and Disadvantages of Hierarchies

Think of two advantages

- How does a top-down approach help in the ‘real world’

Think of two disadvantages

- What are some drawbacks to this approach?

Write these down and bring them to class.