Top-down design and modularity

- functions can be used to design programs that are more easily understood and easier to maintain/modify in the future
- the function takes a (possibly complex) block of code and gives it an intuitive name/parameter list
- in the future, to execute the block of code we simply call the function
- allows us to think of that block of code as an abstract task: we don't need to worry about how the code inside it runs/works, we just need to call it

Impact on readability, maintenance

- when reading the caller function, we just see the intuitive task/function name, avoid the gory details
- when reading the called function, we just see the details for carrying out the task, no need to see details about who calls it and why
- if we want to change what the caller does before/after, that has no impact on the called function
- if we want to change the details of how the called function carries out its task, that has no impact on the caller (assuming the called function still works)

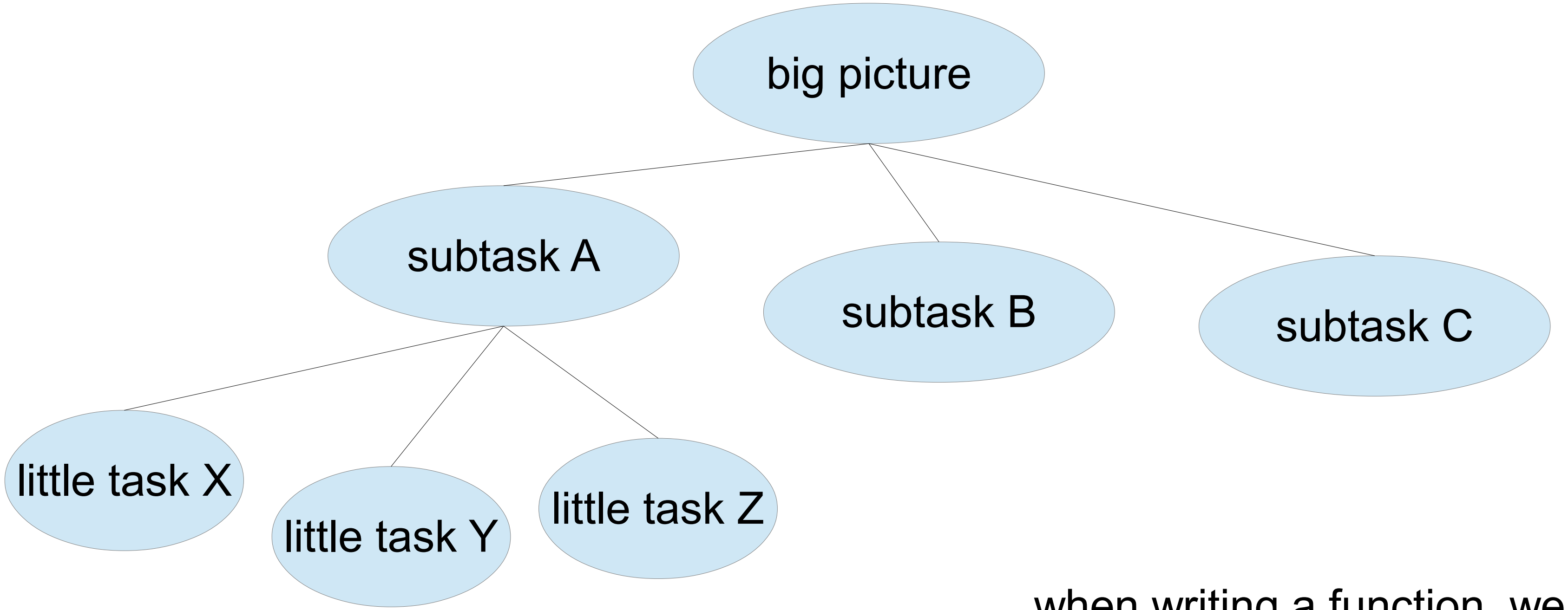
Modularity, compartmentalization

- functions are a first step towards modular design: creating a program of interlocking parts where the internal details of each part are hidden from the other parts
- we agree on the role of each part: name, parameters, return value, and the task that part should complete
- we can update any part (rewrite its internals) without impacting any others as long as the part still ultimately accomplishes its objectives
- allows us to design, implement, and modify the different parts independently

Top down design

- When given a complex problem that we want to develop a program for, use the decomposition into parts one layer at a time
- divide the overall program into several key tasks, have a function for each (main will call those)
- divide each of the key tasks into smaller tasks, e.g. divide task A into smaller tasks X,Y,Z, then have the function for task A call the functions for tasks X,Y,Z
- keep decomposing until the tasks look simple enough to solve as single functions

Gives heirarchical design



when writing a function, we only focus on what it needs to do/call, not any details about what's happening inside any other function