Instruction Runner

Zijian Zhang

University of Toronto

the matter lab

Overview

Objectives

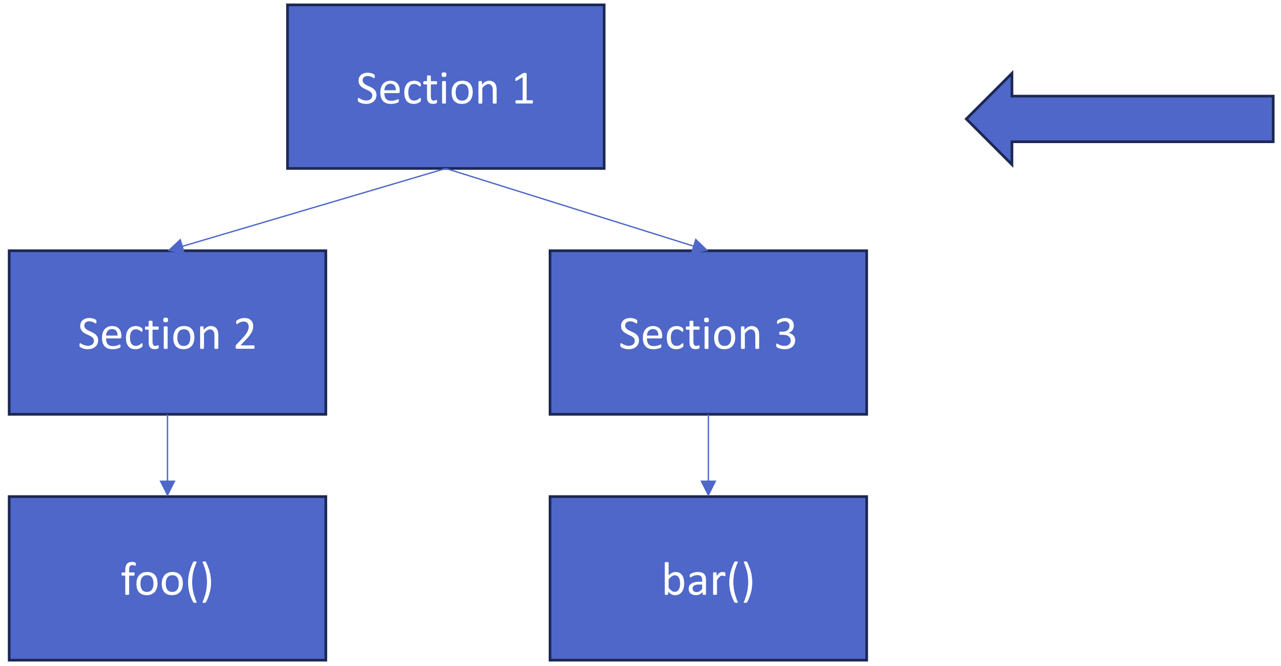
- Convert natural language instruction to Python code
- Generate Python code based on private library of functions
- Automatic fill function parameters

For these goals, we use

- A tool that allows arrange functions in a tree structure
- A beam searcher that searches functions from the tree
- A pipeline generating and running the codes

Tree-based function search

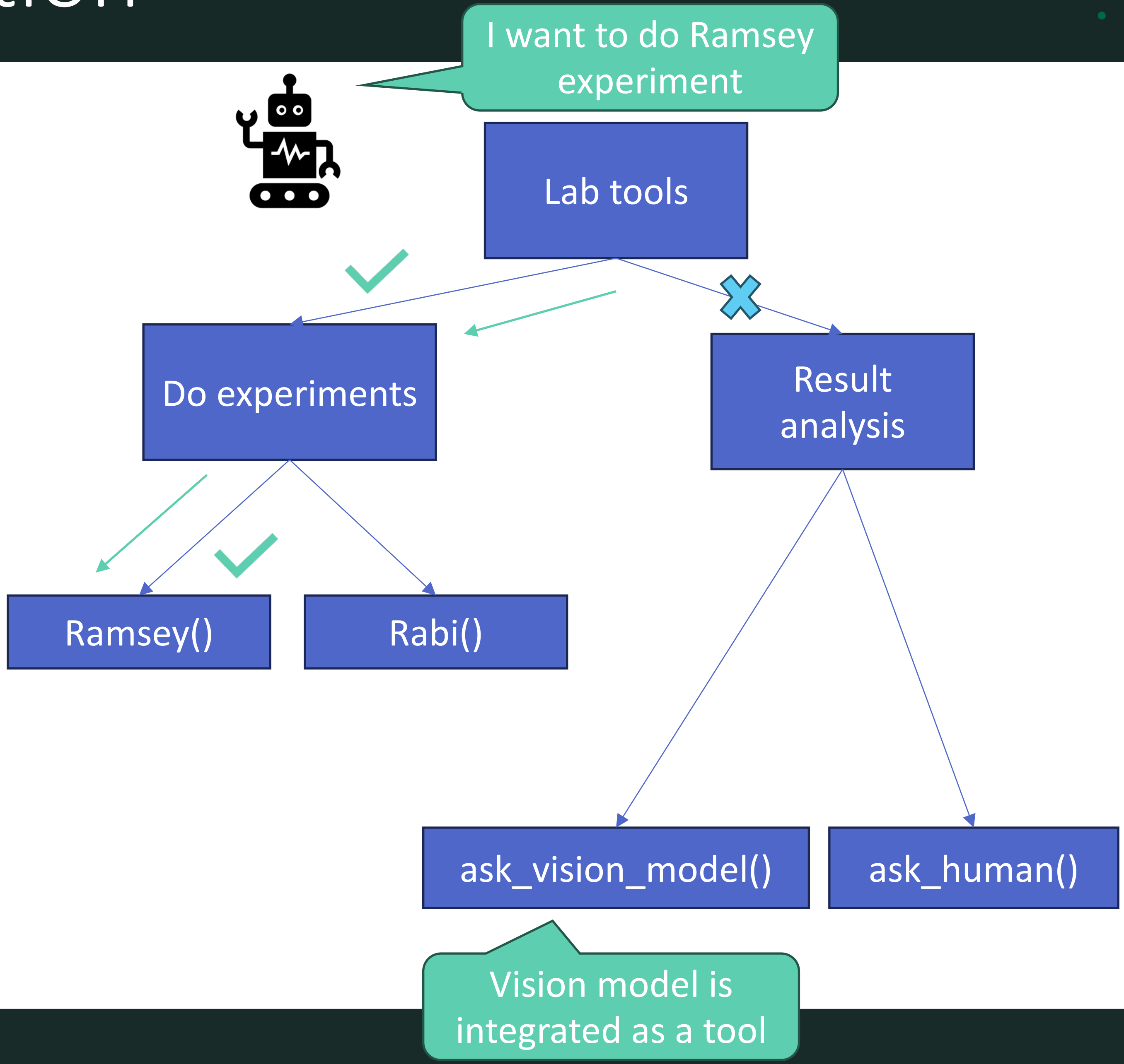
- We developed a Python package Moduler for arranging Python functions into a tree.
- The code in the right can be transformed into the tree in the left.



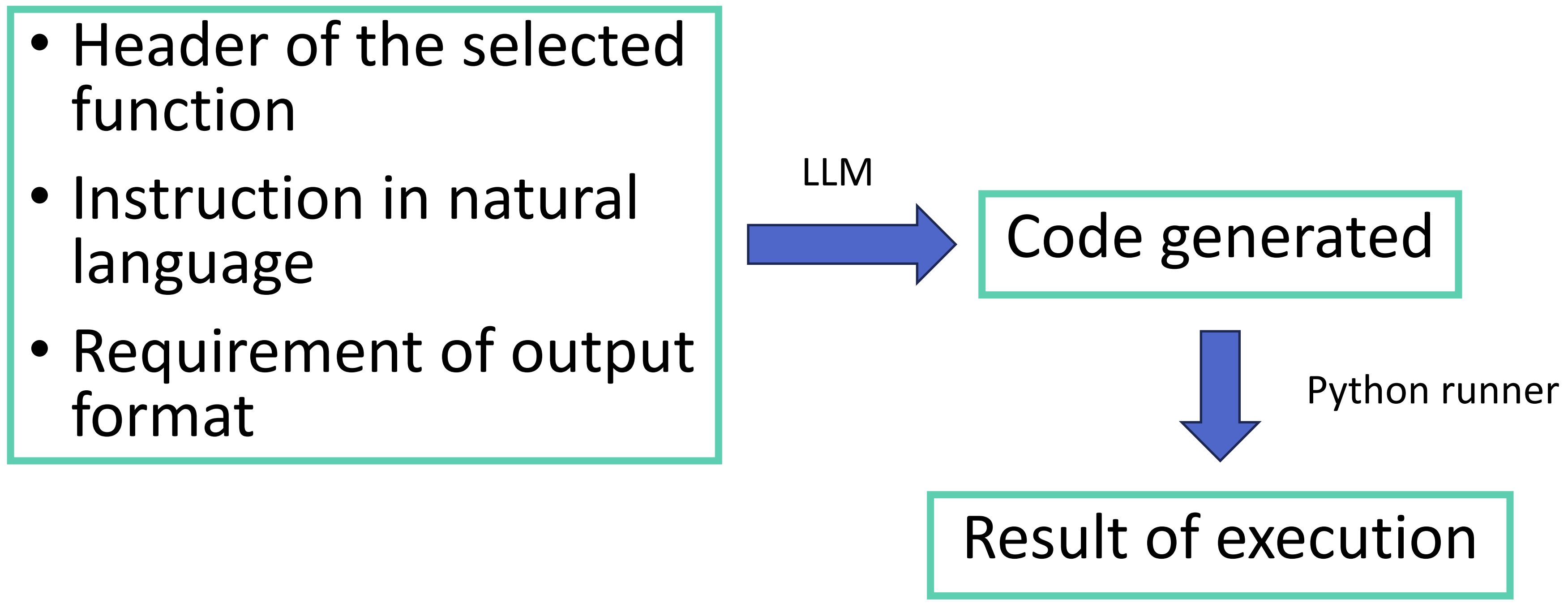
```python
"""
# Section 1
## Section 2
"""


def foo():
    """
    This is a function
    """
    pass



"""
## Section 3
"""


def bar():
    """
    This is another function
    """
    pass
```

Beam search of function

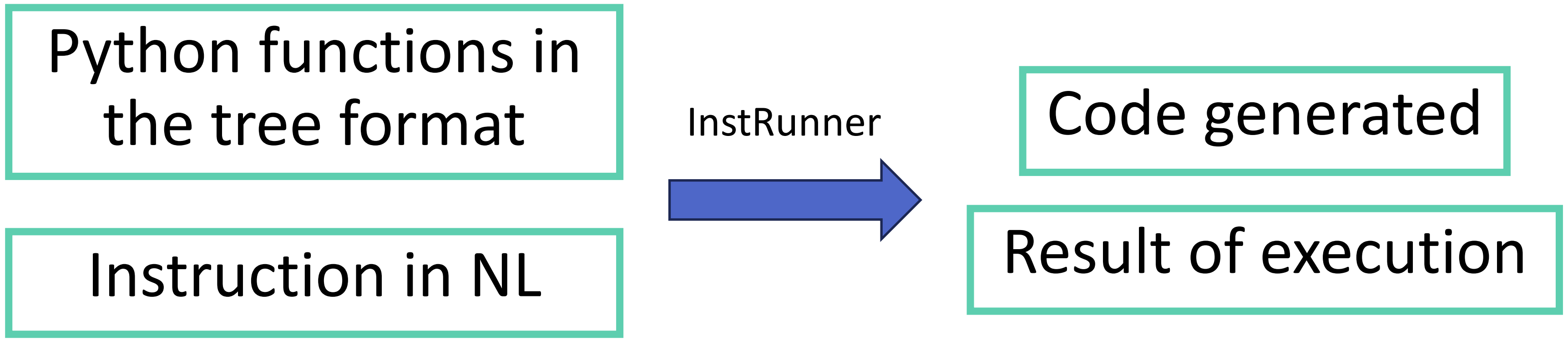
- We create a **summary at each layer** of node based on their contents
- The beam searcher iterate layers from **top to bottom**.
- At each layer, the searcher **select related nodes** based on the instruction (use LLM).
- Irrelevant nodes and their descendants are ignored in the following search.



Fill and run functions


• Header of the selected function
• Instruction in natural language
• Requirement of output format
LLM
Code generated
Python runner
Result of execution

Put everything together



Limitations

- To make the beam search efficient, people must manually arrange the functions based on a certain logic.
- The pipeline work well only on instructions that need only one function.