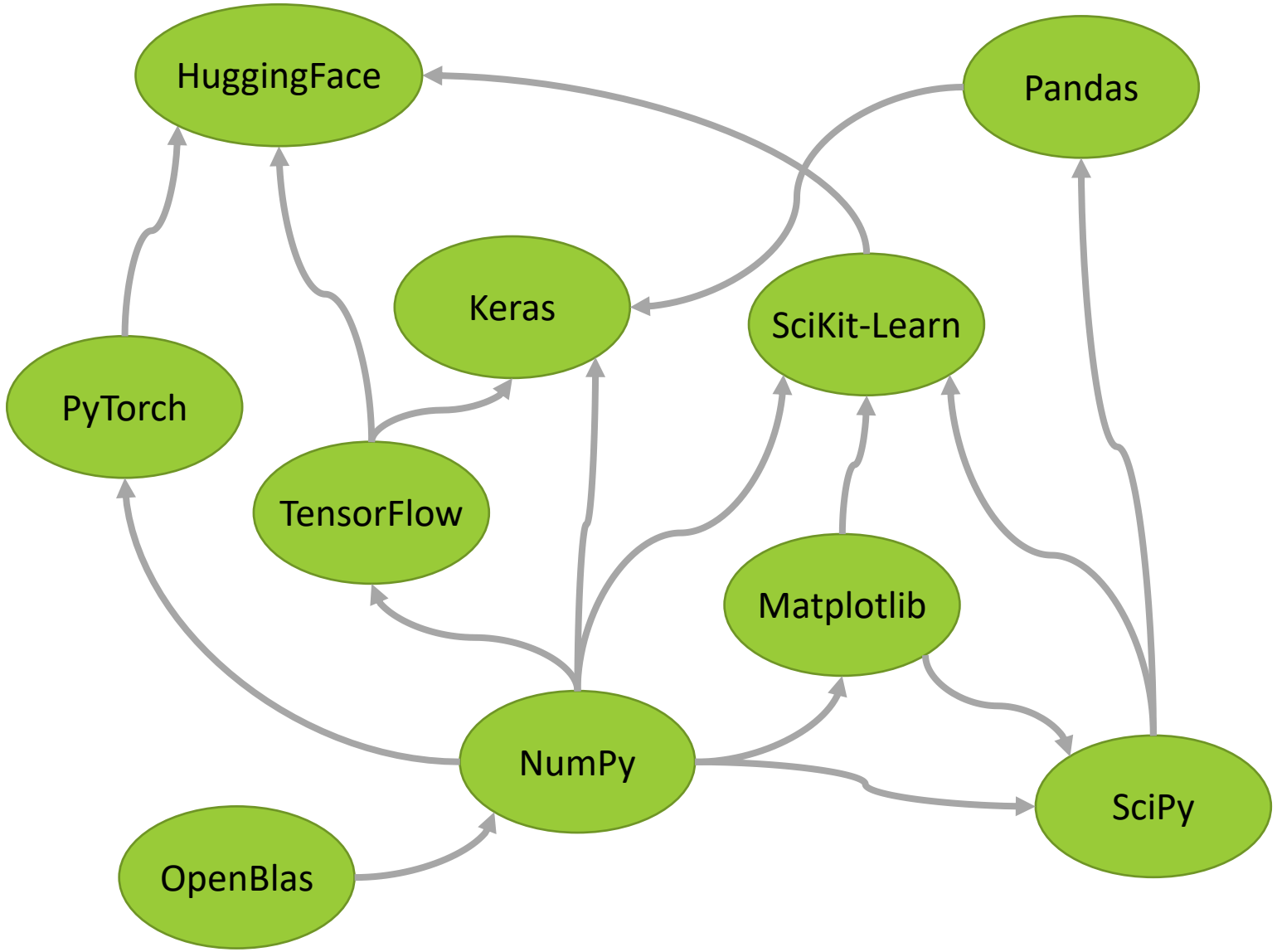
DB Management Systems
Arango Exercises

Joel Klein – jdk514@gwmail.gwu.edu

Graph Traversals

Dependency Graph

- The graph on the right depicts a dependency graph for several popular python packages.
  - Dependencies are packages required to use a given package.
- Each connection depicts that a package is a **dependency**, and this is a directed relationship.
  - Dependencies are one way relationships.



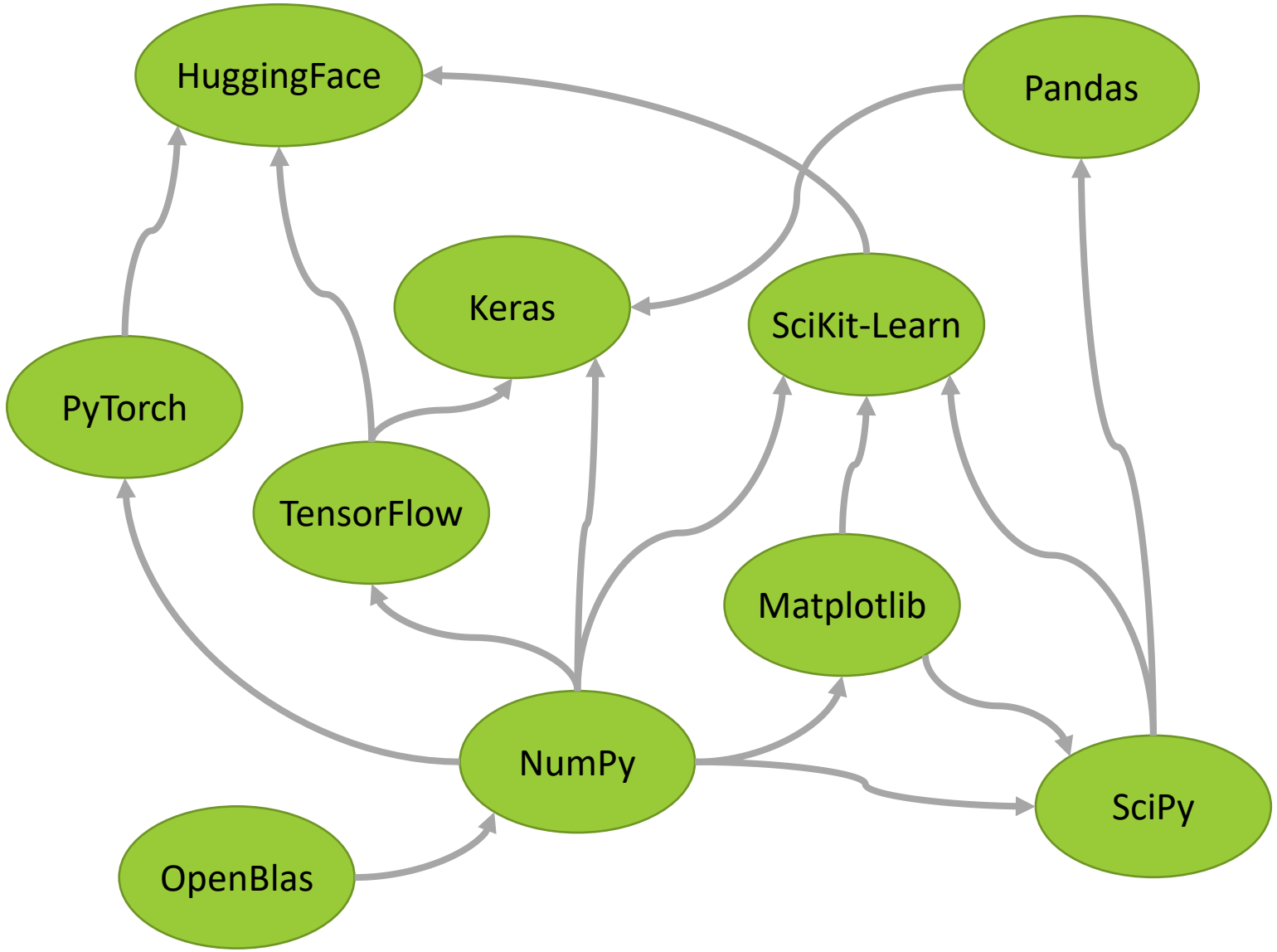
Traversing our Graph

- Let's walk through an example query using our graph:

What packages does **Pandas** depend on?

Or

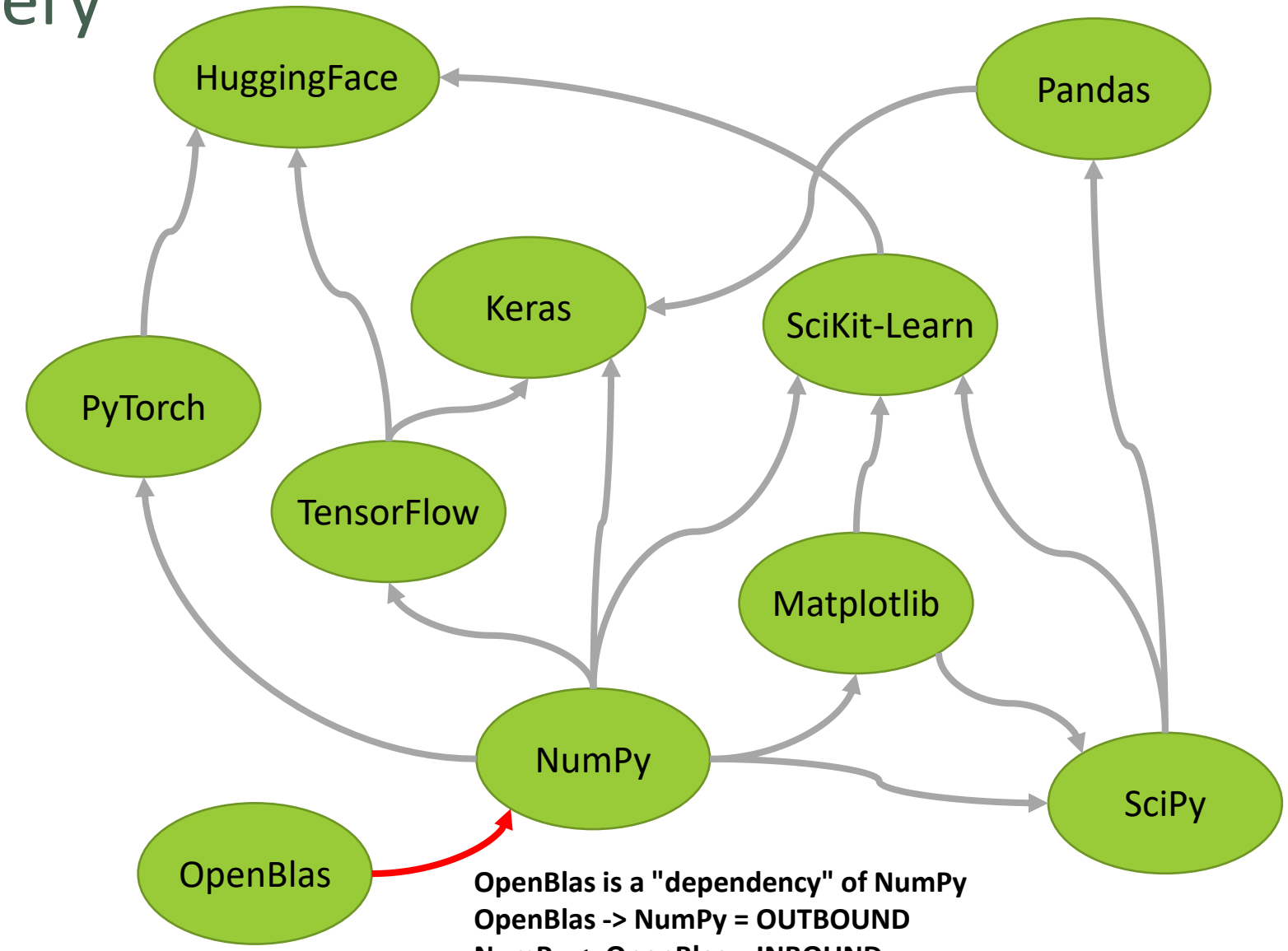
For v, e, p IN 1..5 INBOUND "pkg/Pandas" dependency RETURN v.pkg_name



Breaking Down Our Query

For v, e, p IN 1..5 INBOUND "pkg/Pandas" dependency RETURN v.pkg_name

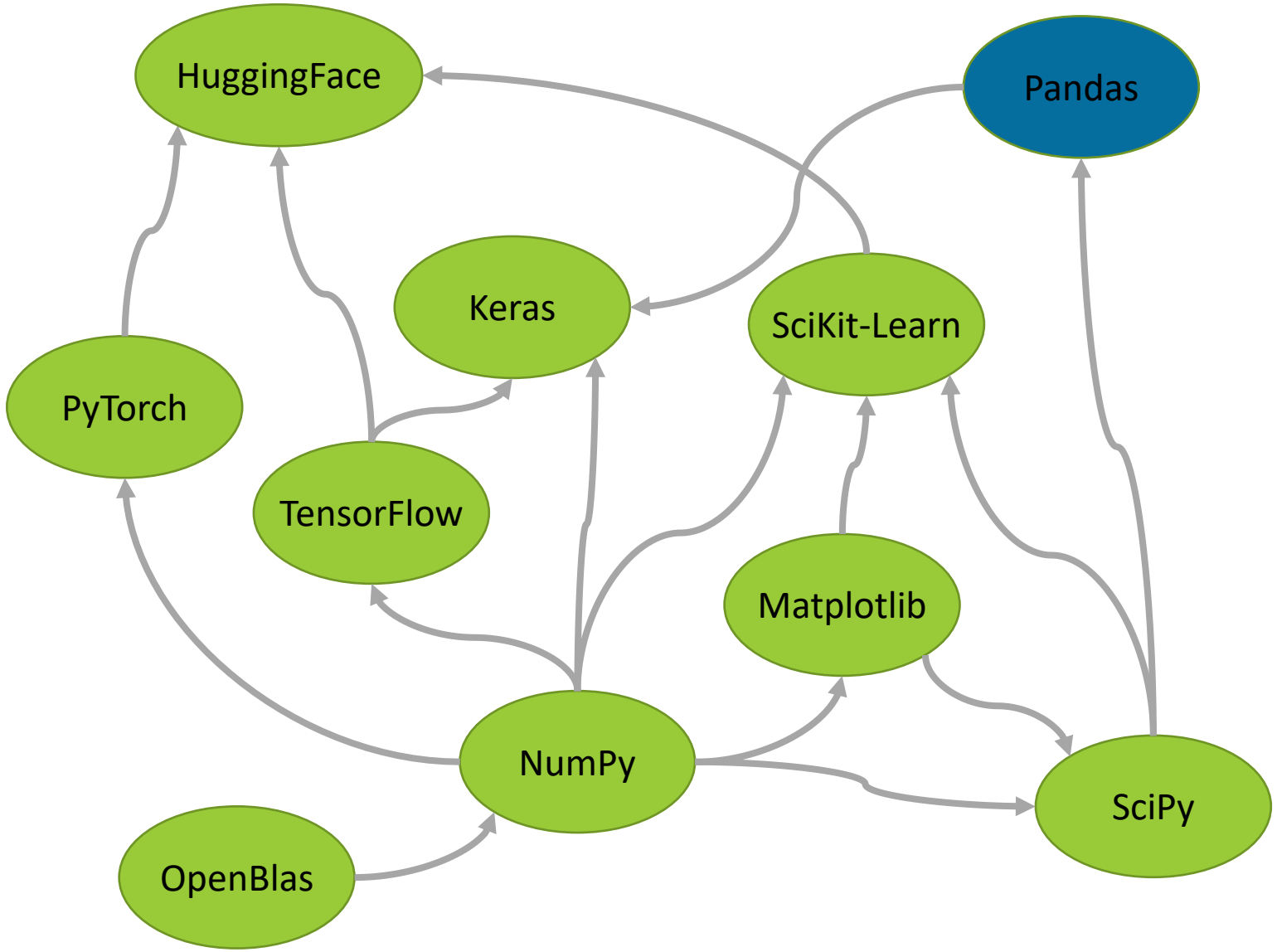
- *v, e, p*: We'll grab each vertex, edge, and path in our traversal
- *1..5*: We'll look at links/jumps 1-5 levels deep
- INBOUND: Our relationships are directed, indicating a node is a dependency of a given package. So, we want to follow inbound links.
- *"pkg/Pandas"*: Using the "pkg" collection, we want to start at the Pandas node.
- *dependency*: Traveling along the dependency edge collection
- *v.pkg_name*: Return the package name for the node/vertex



Traversal Step 1

For v, e, p IN 1..5 INBOUND "pkg/Pandas" dependency RETURN v.pkg_name

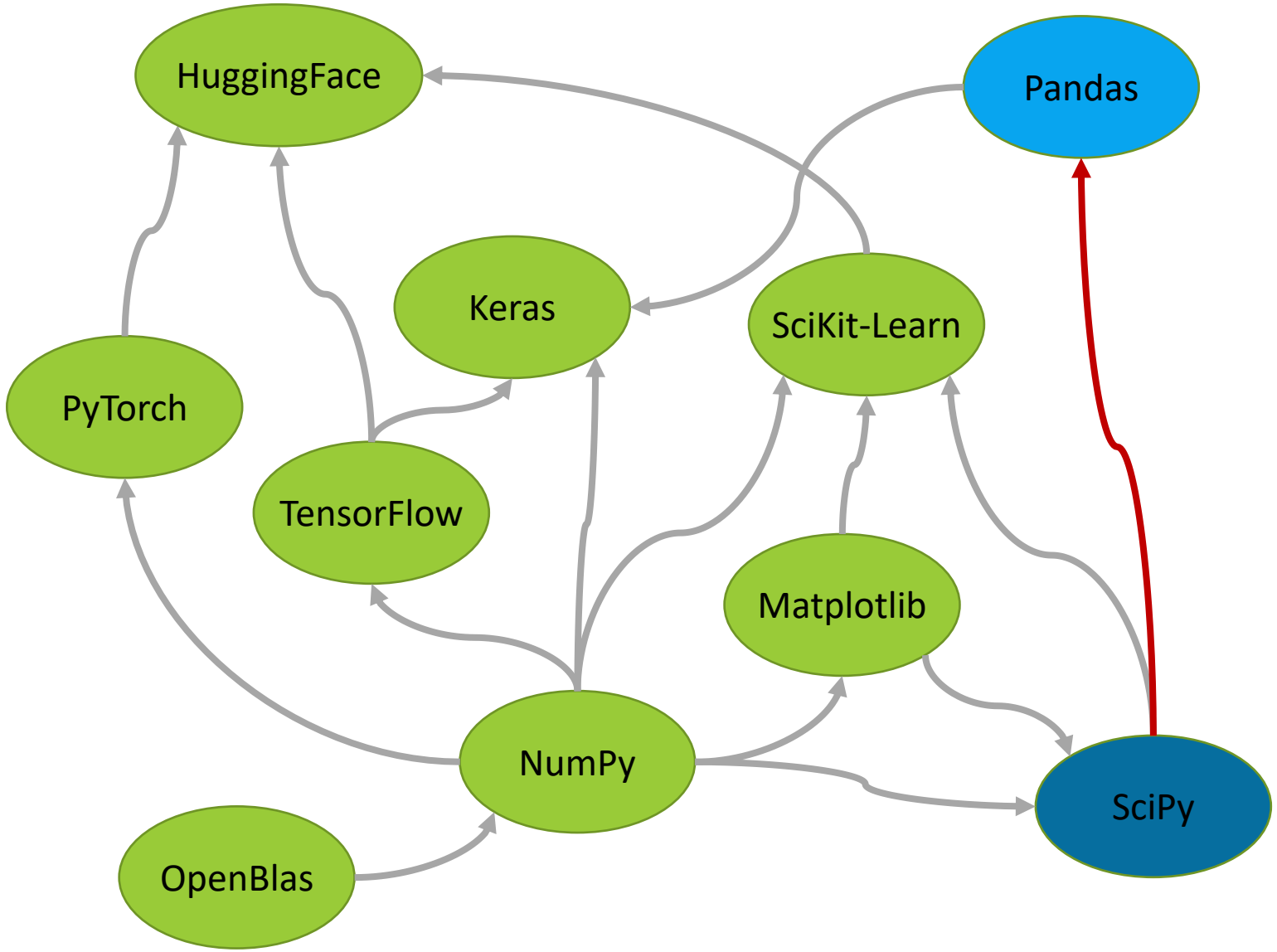
- We Start at the "pkg/Pandas" or Pandas node.



Traversal Step 2

For v, e, p IN 1..5 INBOUND "pkg/Pandas" dependency RETURN v.pkg_name

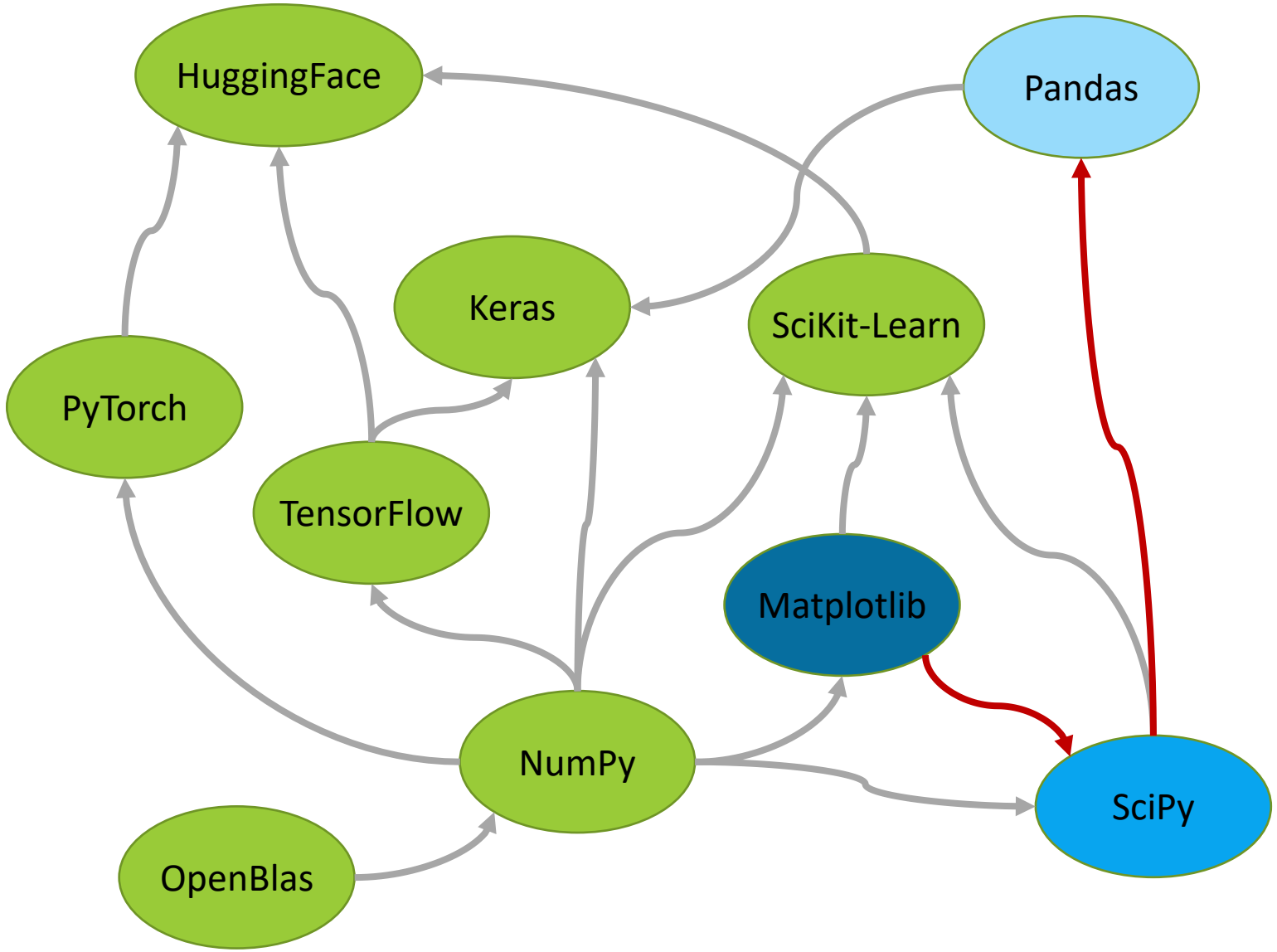
- By default, Arango uses depth-first search or dfs
- So, our first path from Pandas is SciPy



Traversal Step 3

For v, e, p IN 1..5 INBOUND "pkg/Pandas" dependency RETURN v.pkg_name

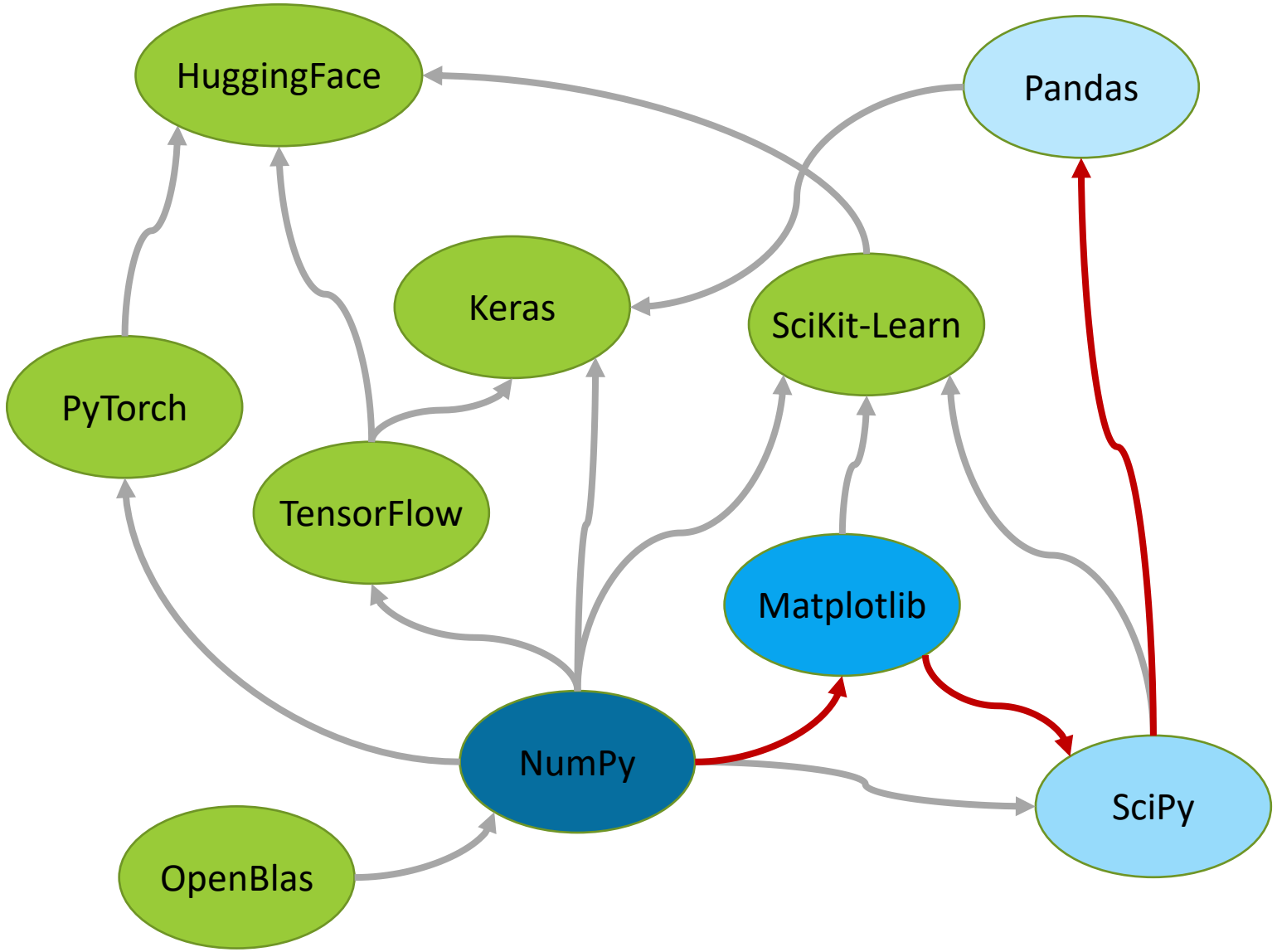
- From SciPy we can go to Matplotlib or NumPy
  - The direction chosen can be defined, but defaults to id ordering (I'm deciding Matplotlib goes first)



Traversal Step 4

For v, e, p IN 1..5 INBOUND "pkg/Pandas" dependency RETURN v.pkg_name

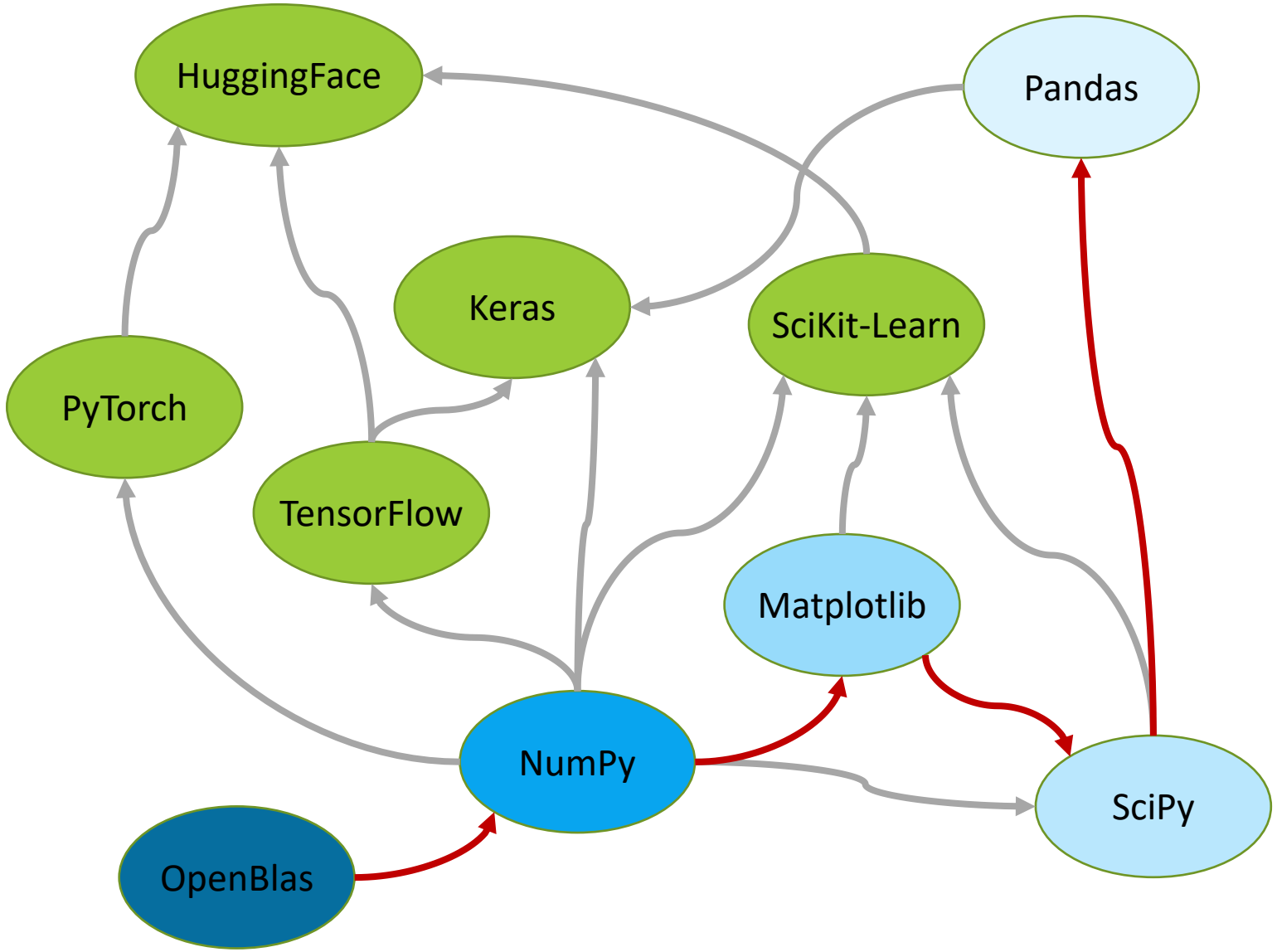
- Since we are depth-first, we'll continue down the dependency line to NumPy



Traversal Step 5

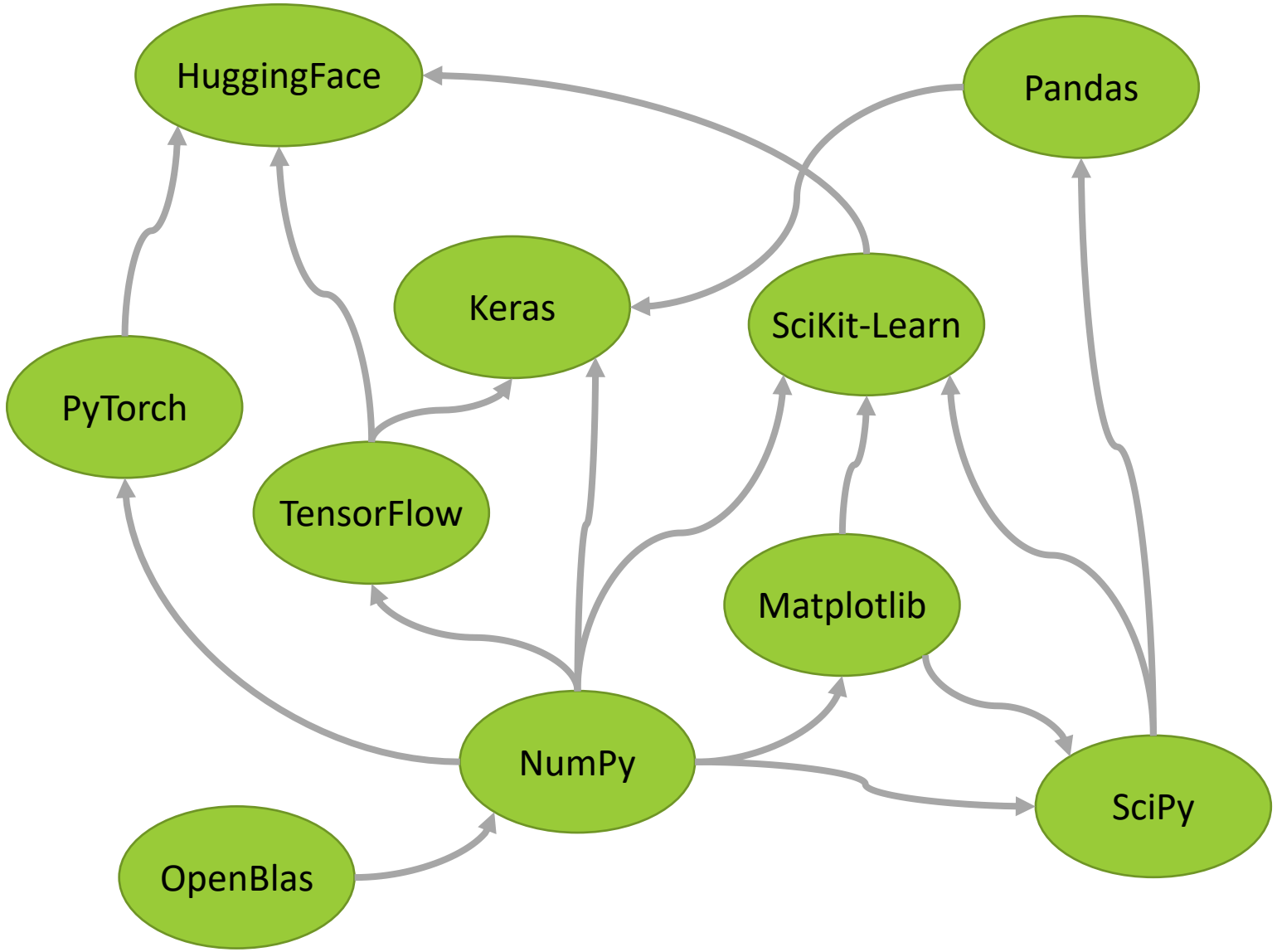
For v, e, p IN 1..5 INBOUND "pkg/Pandas" dependency RETURN v.pkg_name

- Finally, we end at OpenBlas
- Next, we would bubble back up to NumPy, Matplotlib, and SciPy to explore other paths.
  - However, looking at each of these nodes, we can see every dependency was traversed



Exercise 1

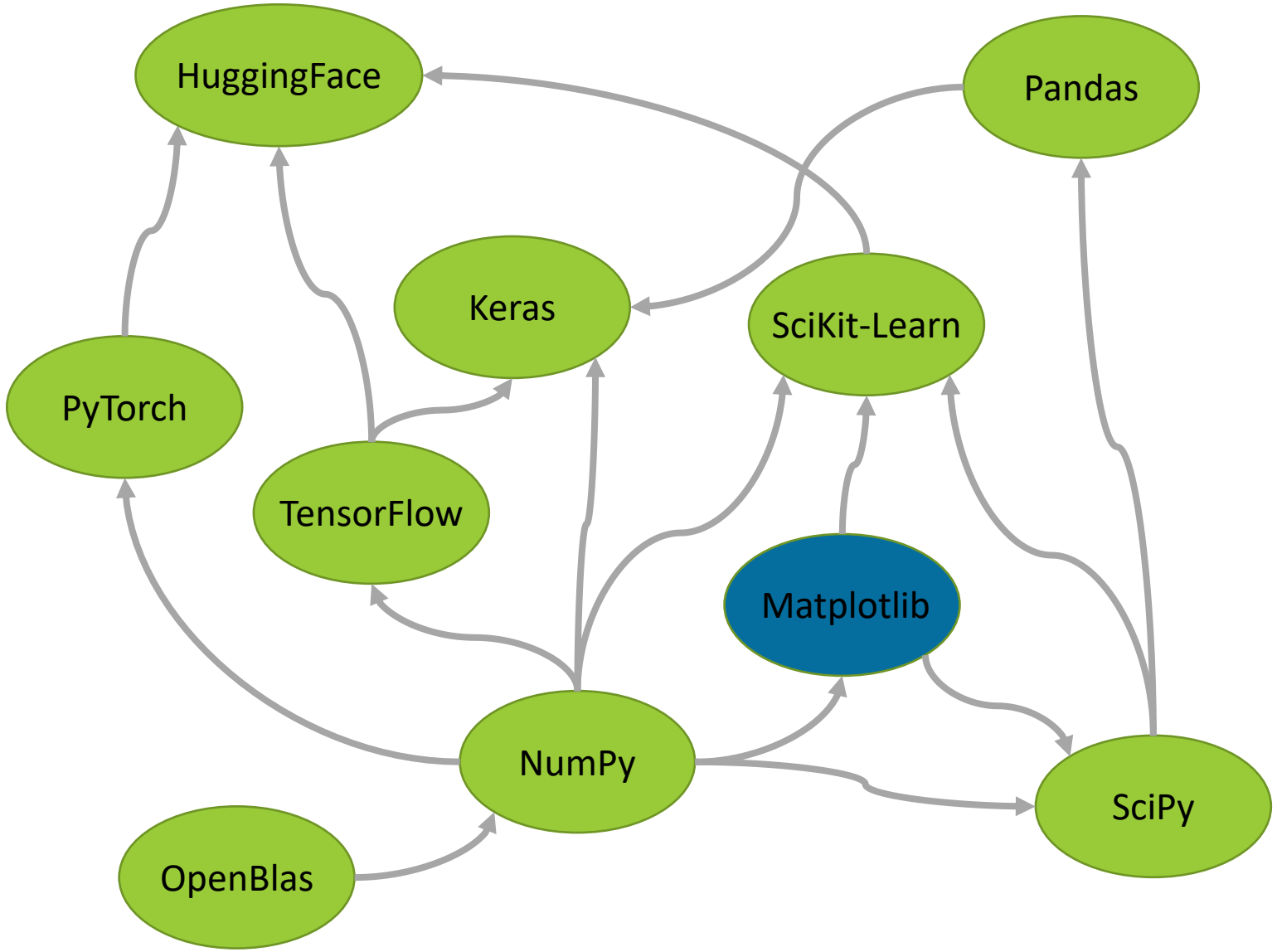
- On slide 12, write the Arango Query to identify all packages that depend on **Matplotlib**
- On the following slides walk through each step of the traversal as demonstrated on the previous slides
  - Feel free to add additional slides if needed



Exercise 1 – Step 1

For v, e, p IN 1..4 OUTBOUND "pkg/Matplotlib" dependency RETURN v.pkg_name

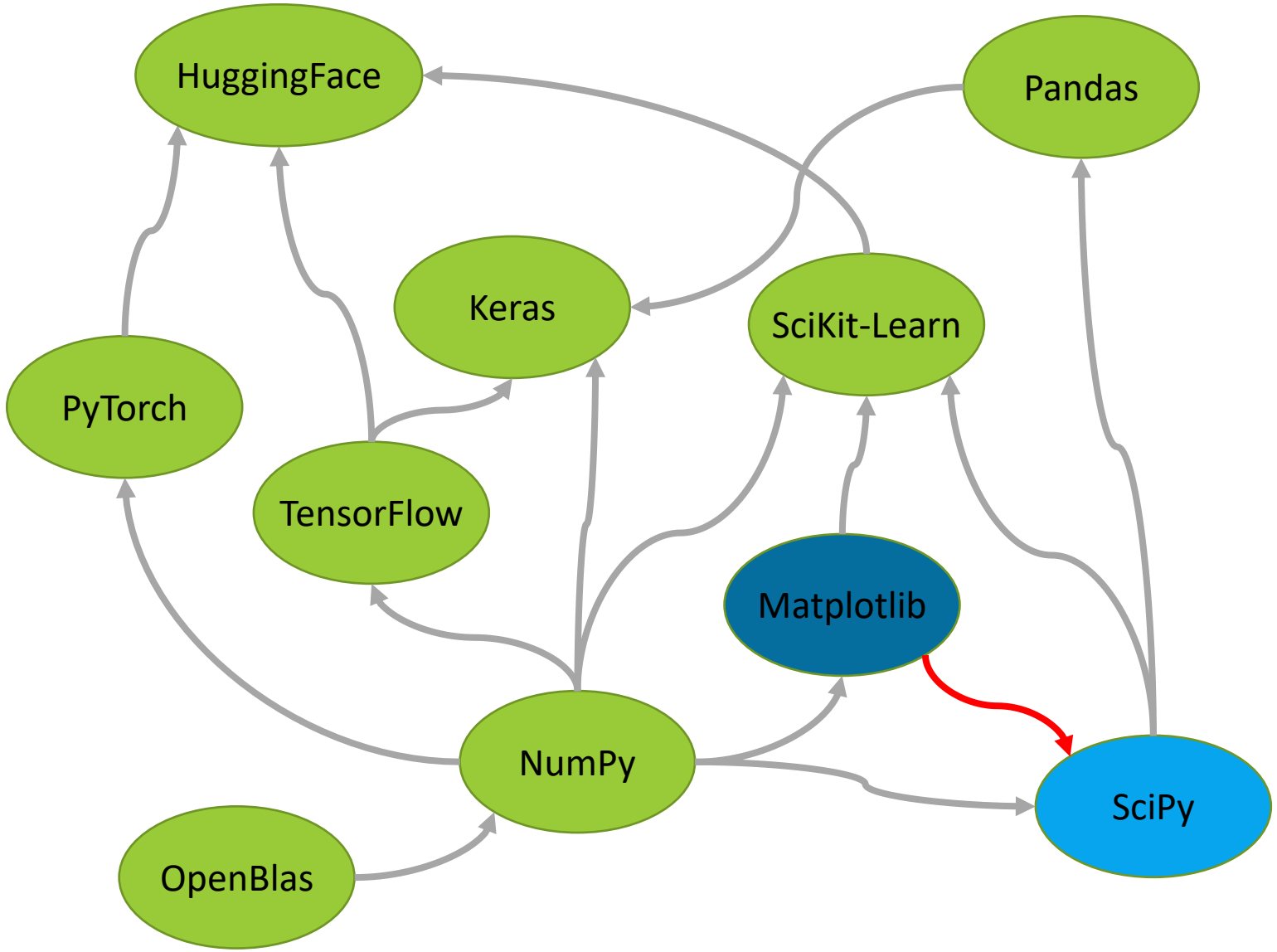
- We Start at the "pkg/Matplotlib" or Matplotlib node.



Exercise 1 – Step 2

For v, e, p IN 1..4 OUTBOUND "pkg/Matplotlib" dependency RETURN v.pkg_name

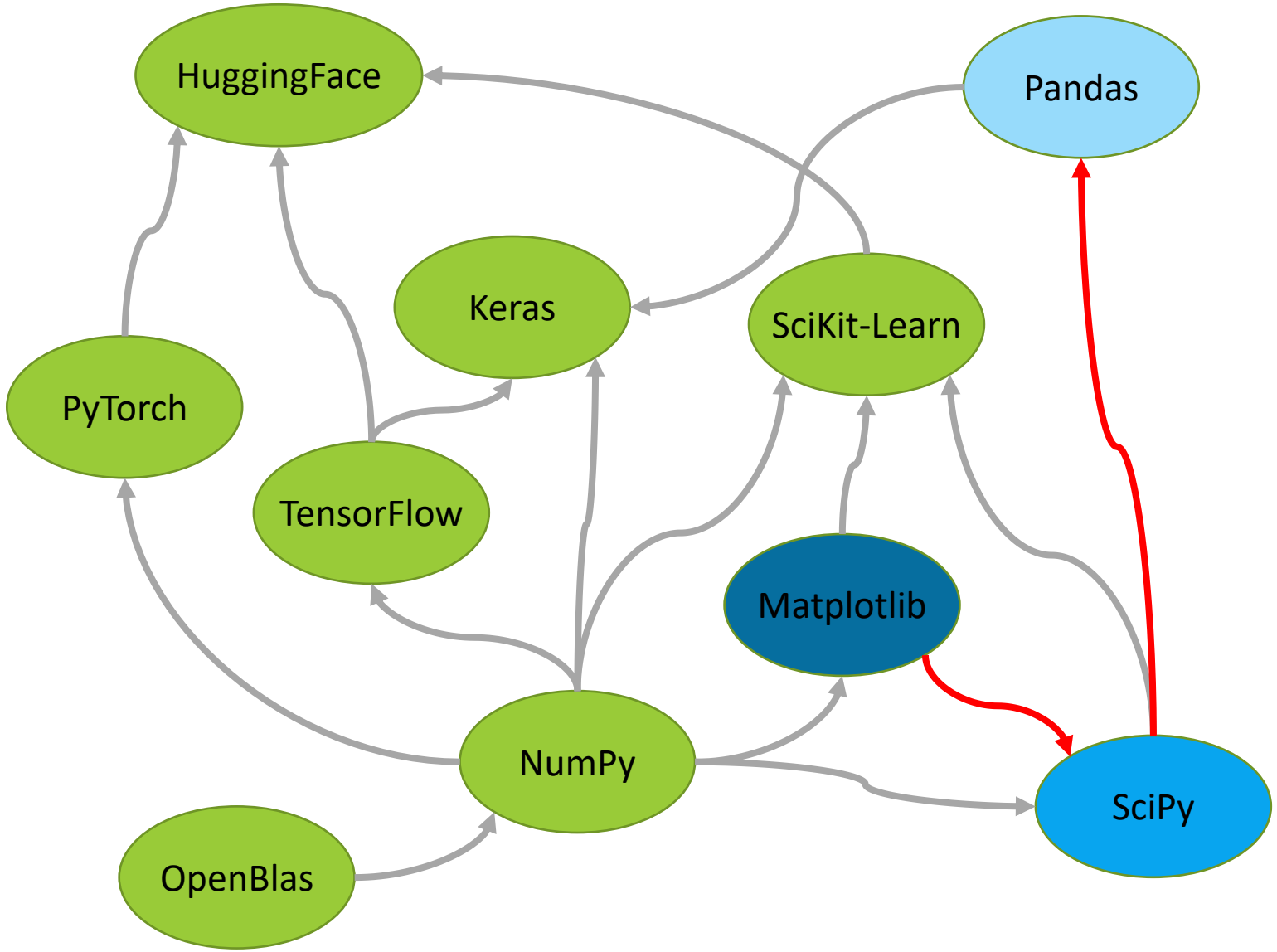
- As we know by default, Arango uses depth-first search or dfs.
- So, our first path from Pandas is SciPy



Exercise 1 – Step 3

For v, e, p IN 1..4 OUTBOUND "pkg/Matplotlib" dependency RETURN v.pkg_name

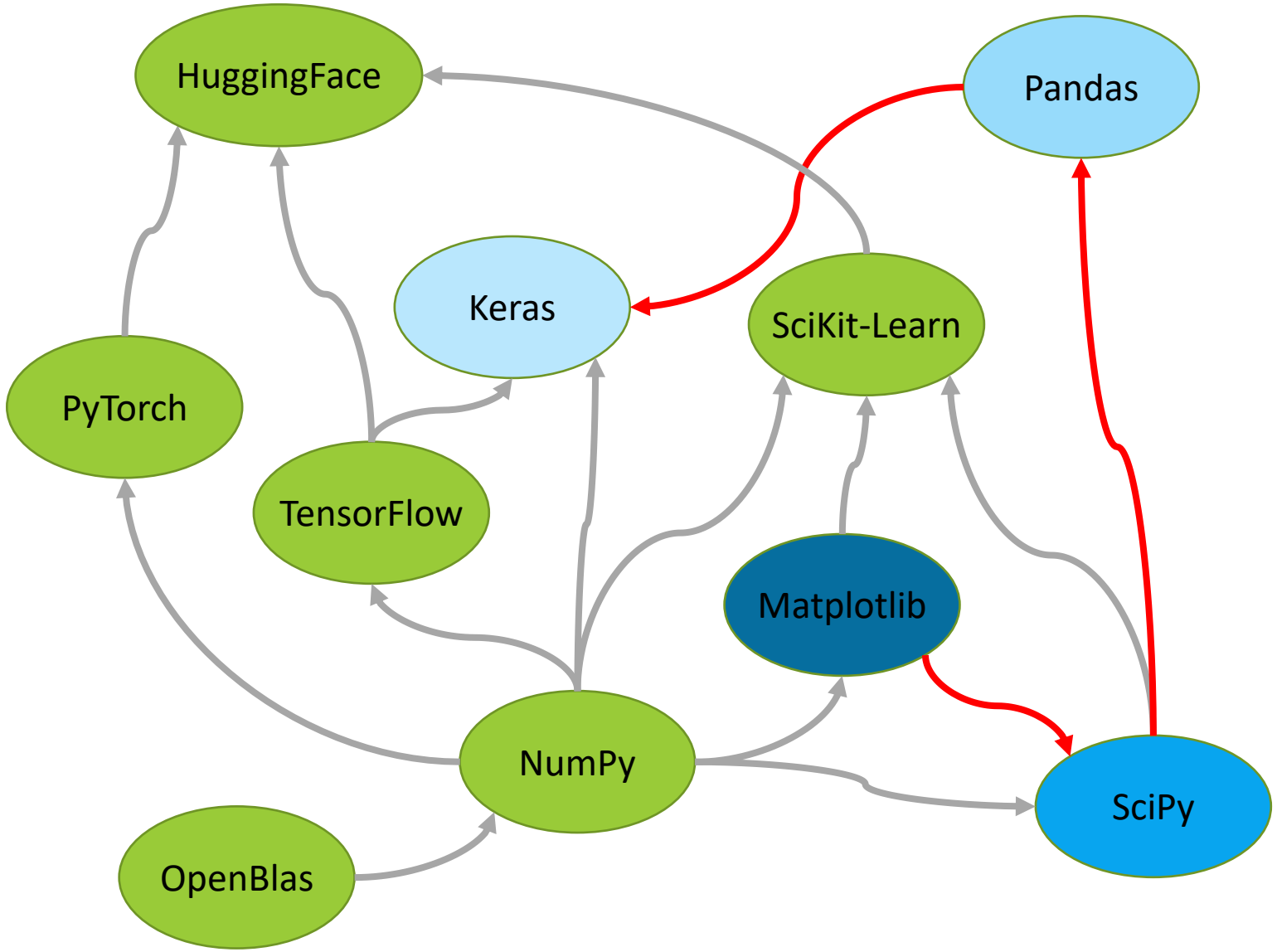
- As we can see from the graph, from SciPy we can go to Pandas or SciKit-Learn.
  - From the previous slides we know that that direction can be chosen by id ordering. So, I'm deciding Pandas goes first.



Exercise 1 – Step 4

For v, e, p IN 1..4 OUTBOUND
"pkg/Matplotlib" dependency
RETURN v.pkg_name

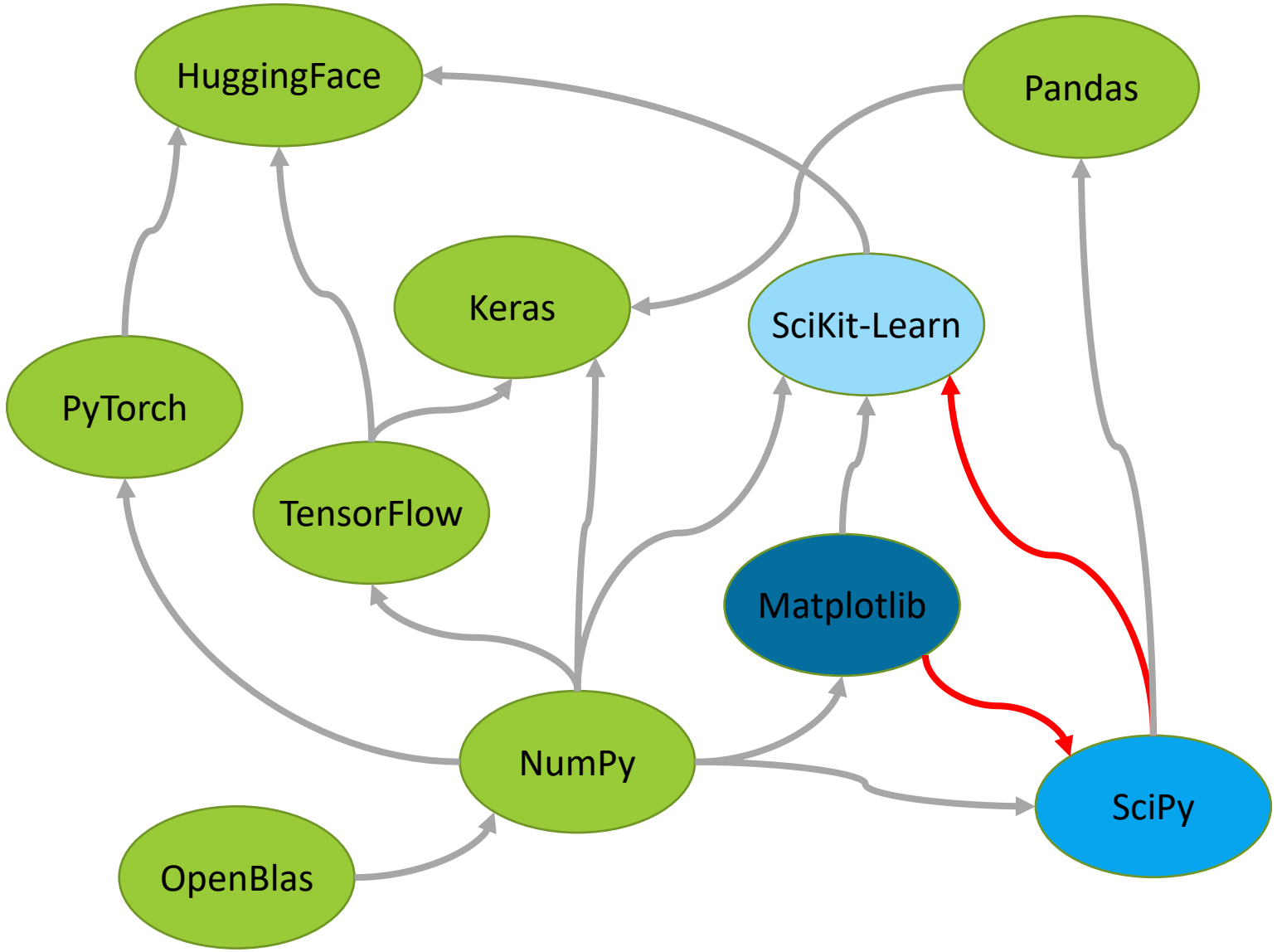
- Finally, we end at Keras by going through the default method, depth-first search.
- Next, we would bubble back up to SciPy to explore another path.



Exercise 1 – Step 5

For v, e, p IN 1..4 OUTBOUND "pkg/Matplotlib" dependency RETURN v.pkg_name

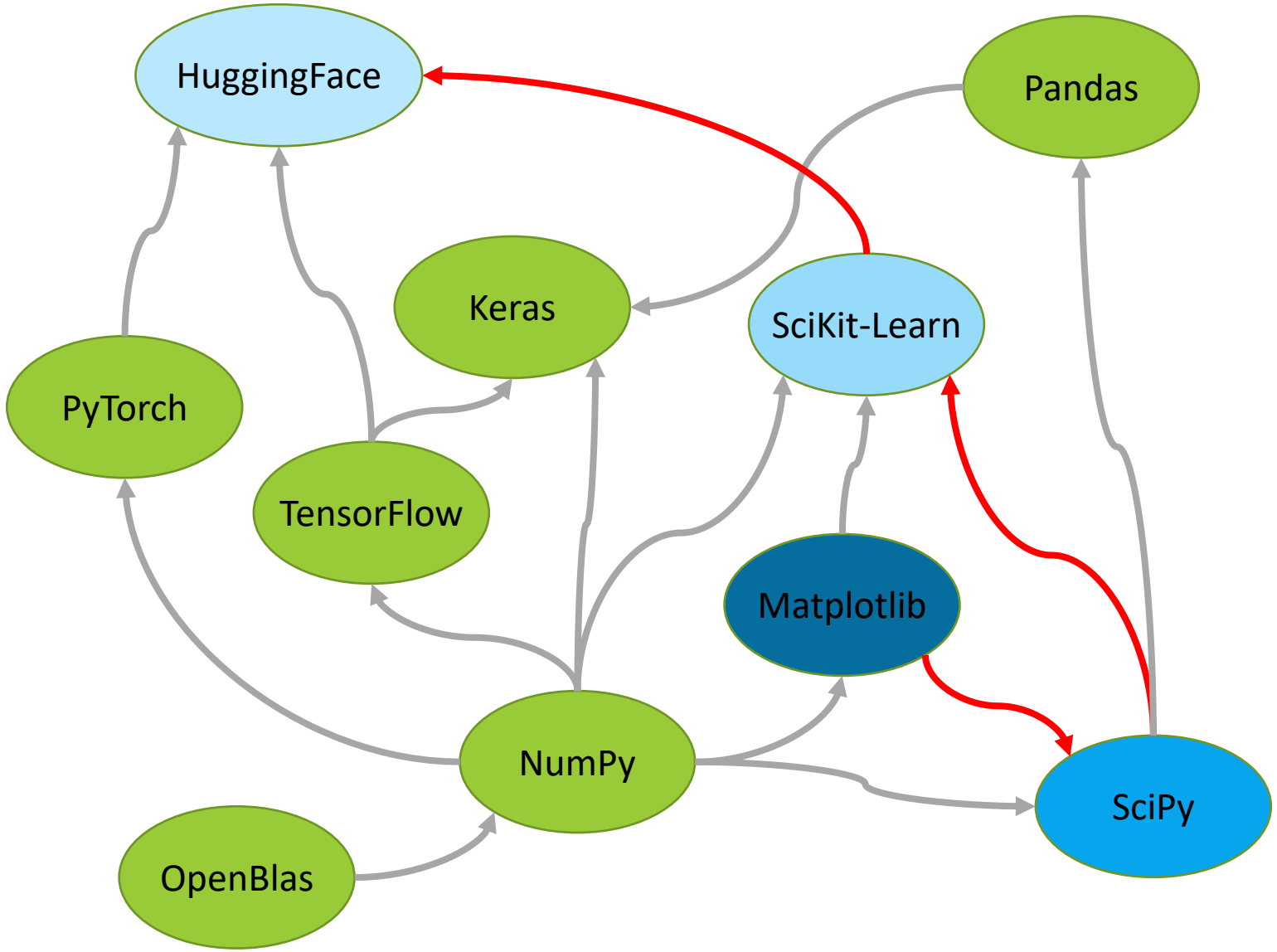
- As we are back to SciPy to start another path, we have SciKit-Learn as the next line.



Exercise 1 – Step 6

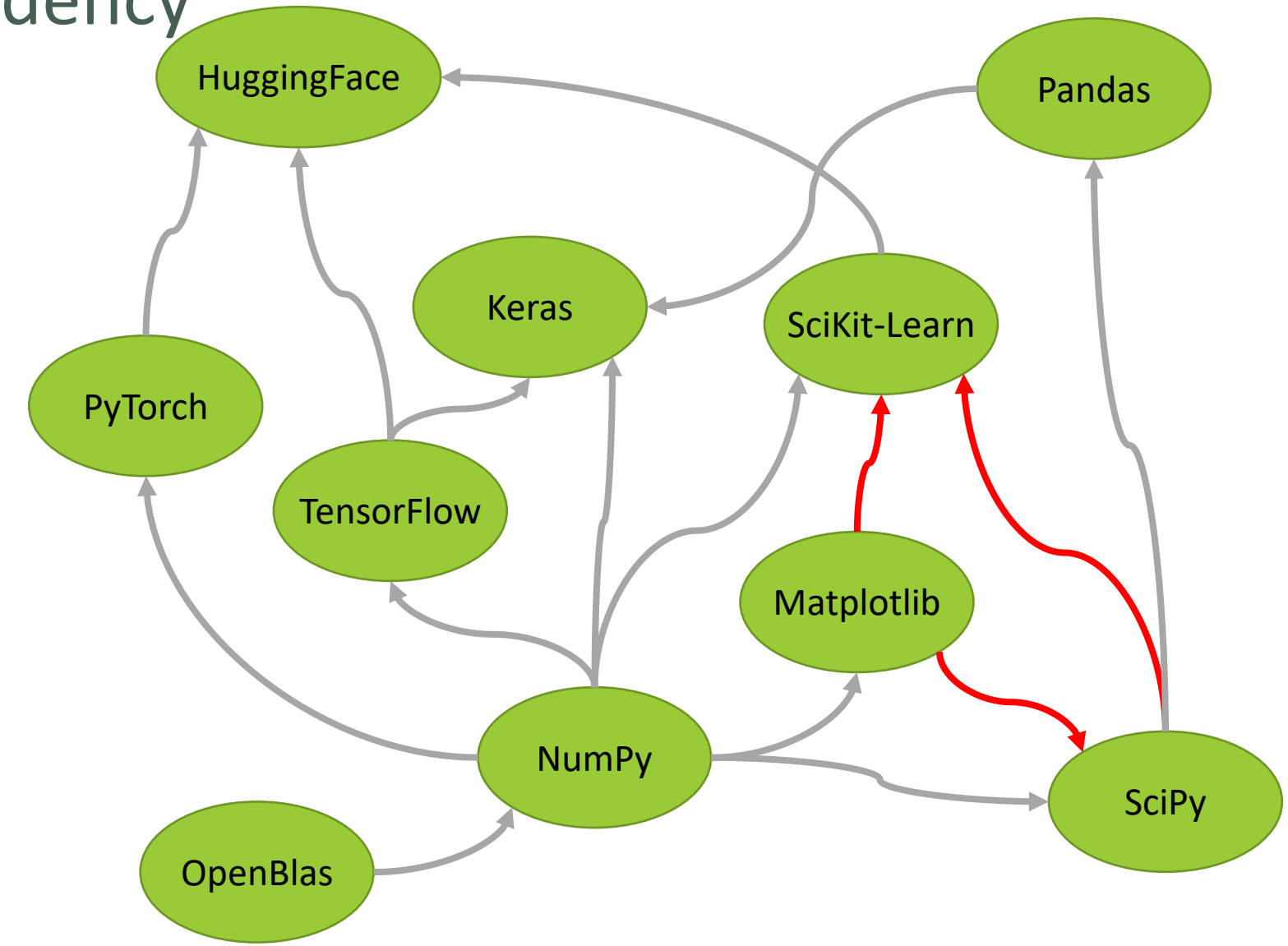
For v, e, p IN 1..4 OUTBOUND "pkg/Matplotlib" dependency RETURN v.pkg_name

- We then finally end up at HuggingFace and the traversal finally ends.



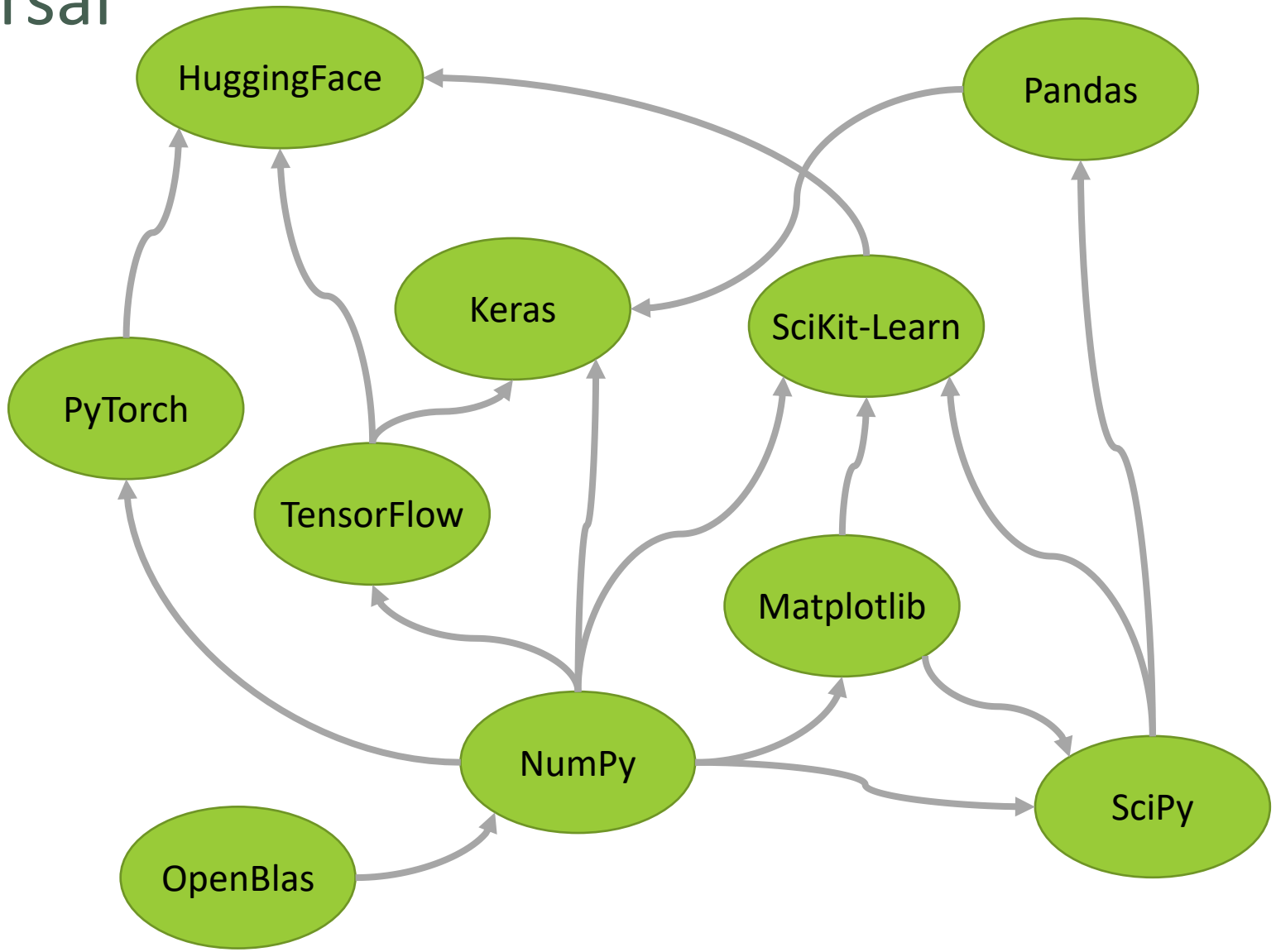
Identifying Dual Dependency

- Looking at the graph, we can see that there are some instances where dependencies double up
  - For example – SciKit-Learn directly depends on Matplotlib, but also gets that dependency from SciPy
- So, how could we identify these "dual dependencies"



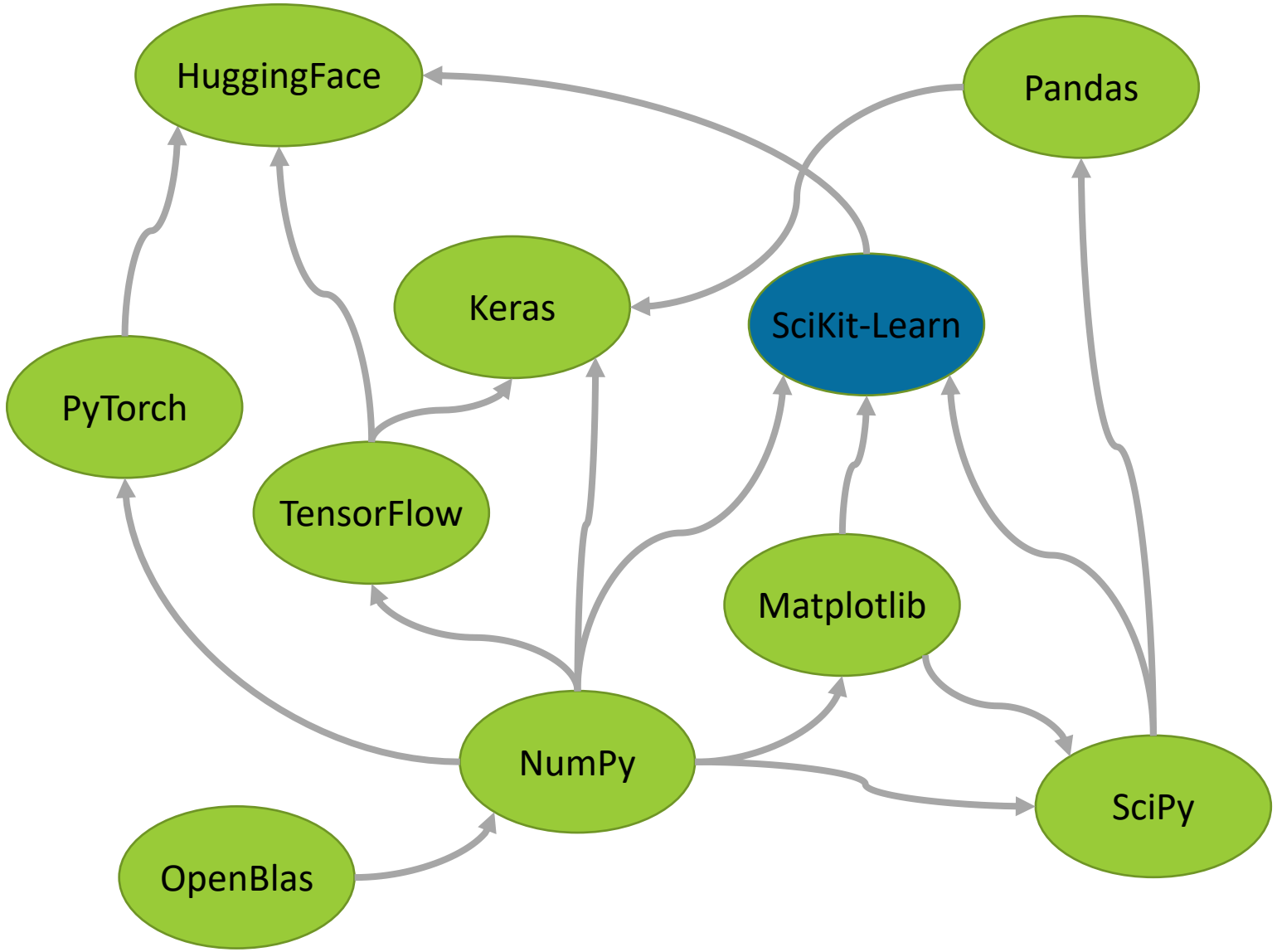
Dual Dependency Traversal

- Traversing the graph, we can see how many times we visit the same node.

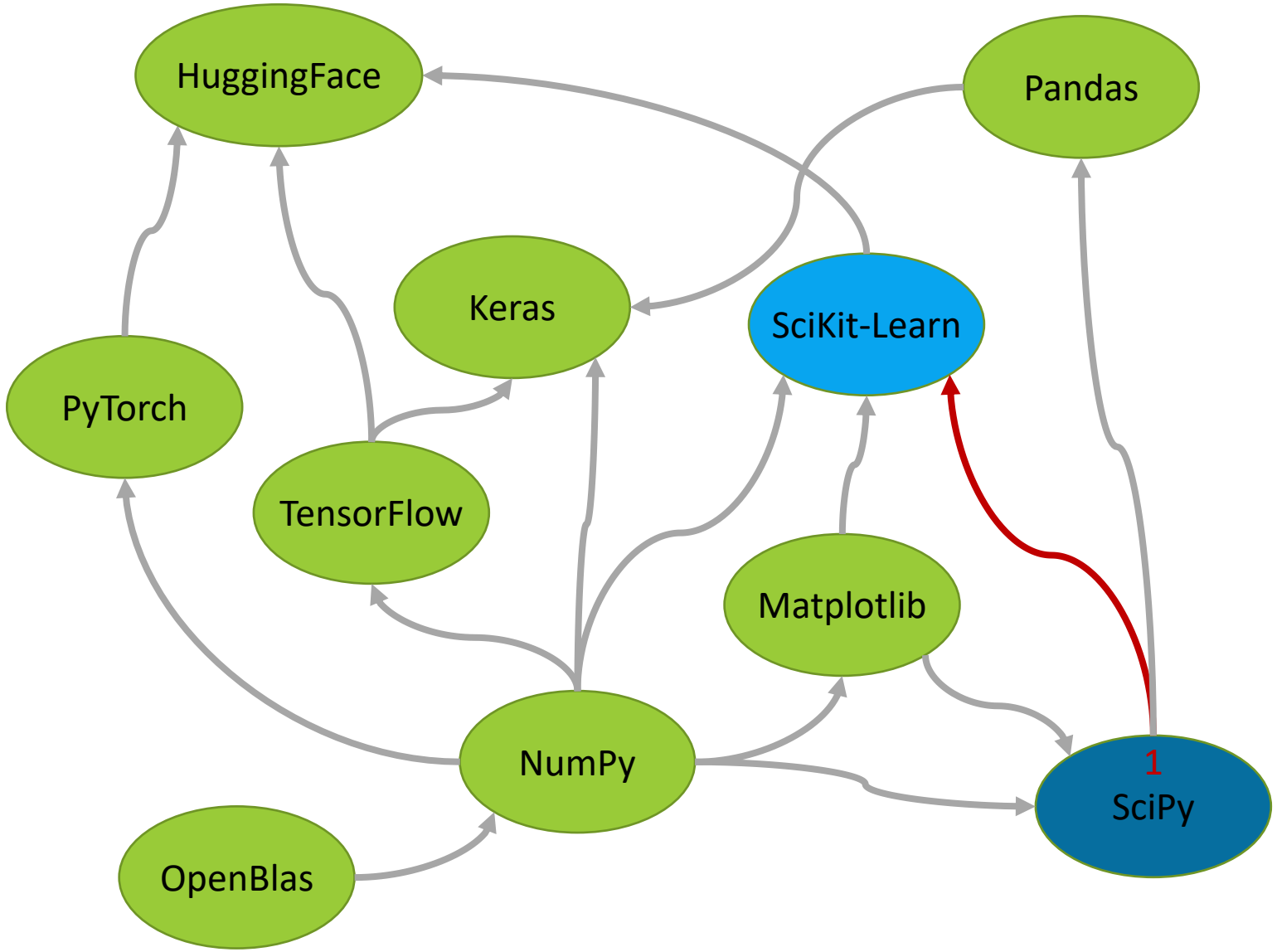


Dual Dependency Traversal Step 1

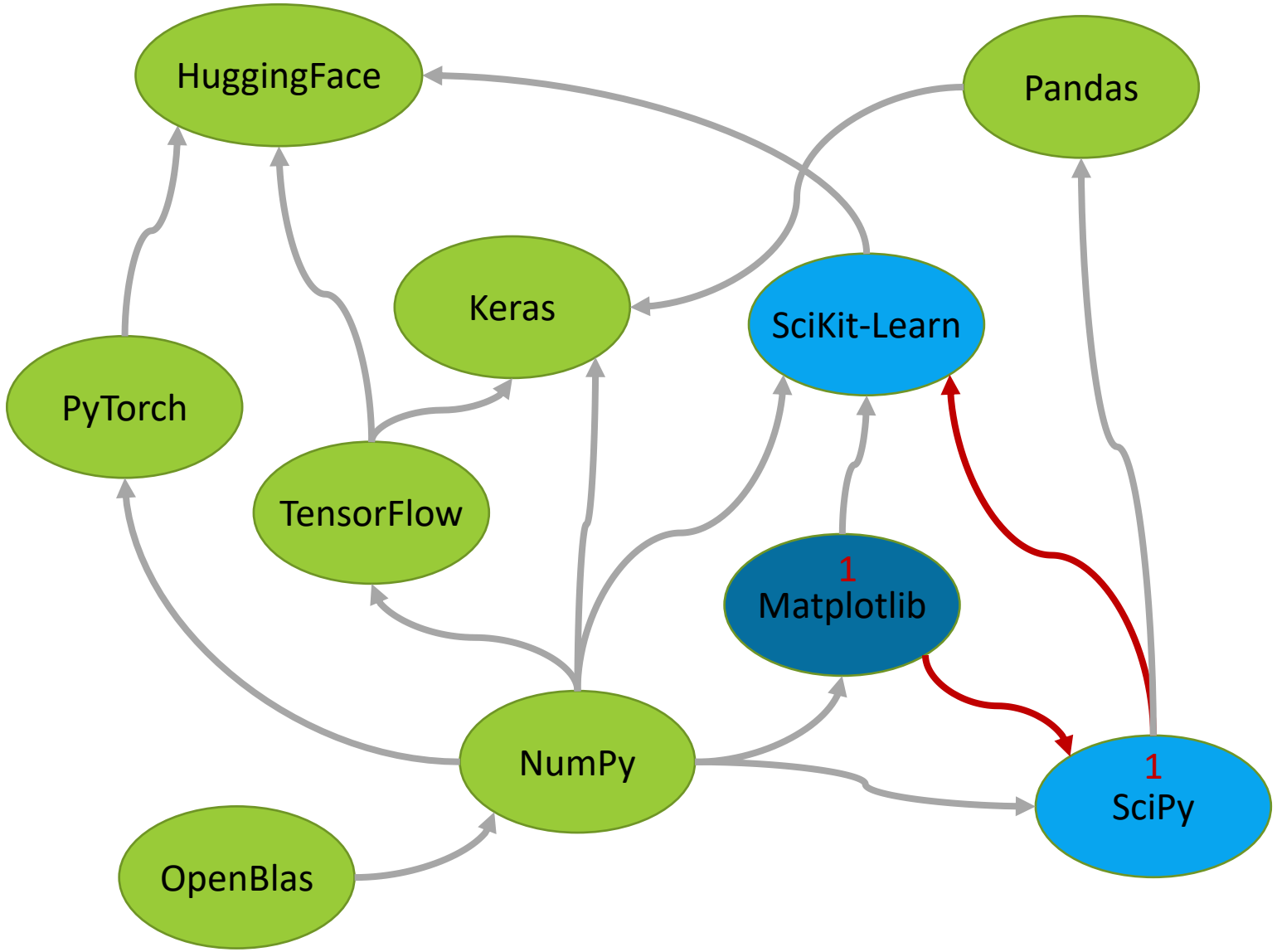
- Using depth-first search we'll count each time we hit a node



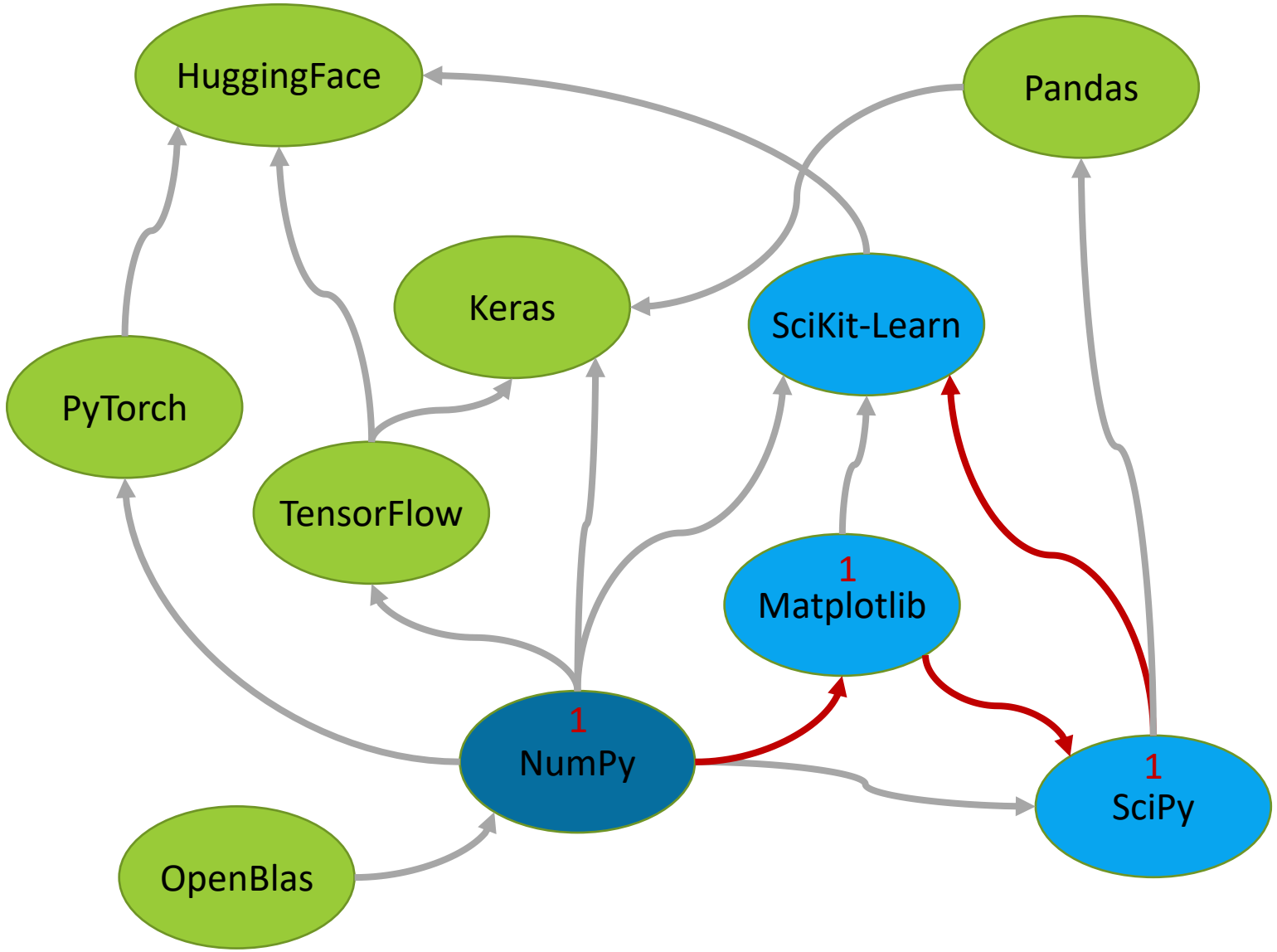
Dual Dependency Traversal
Step 2



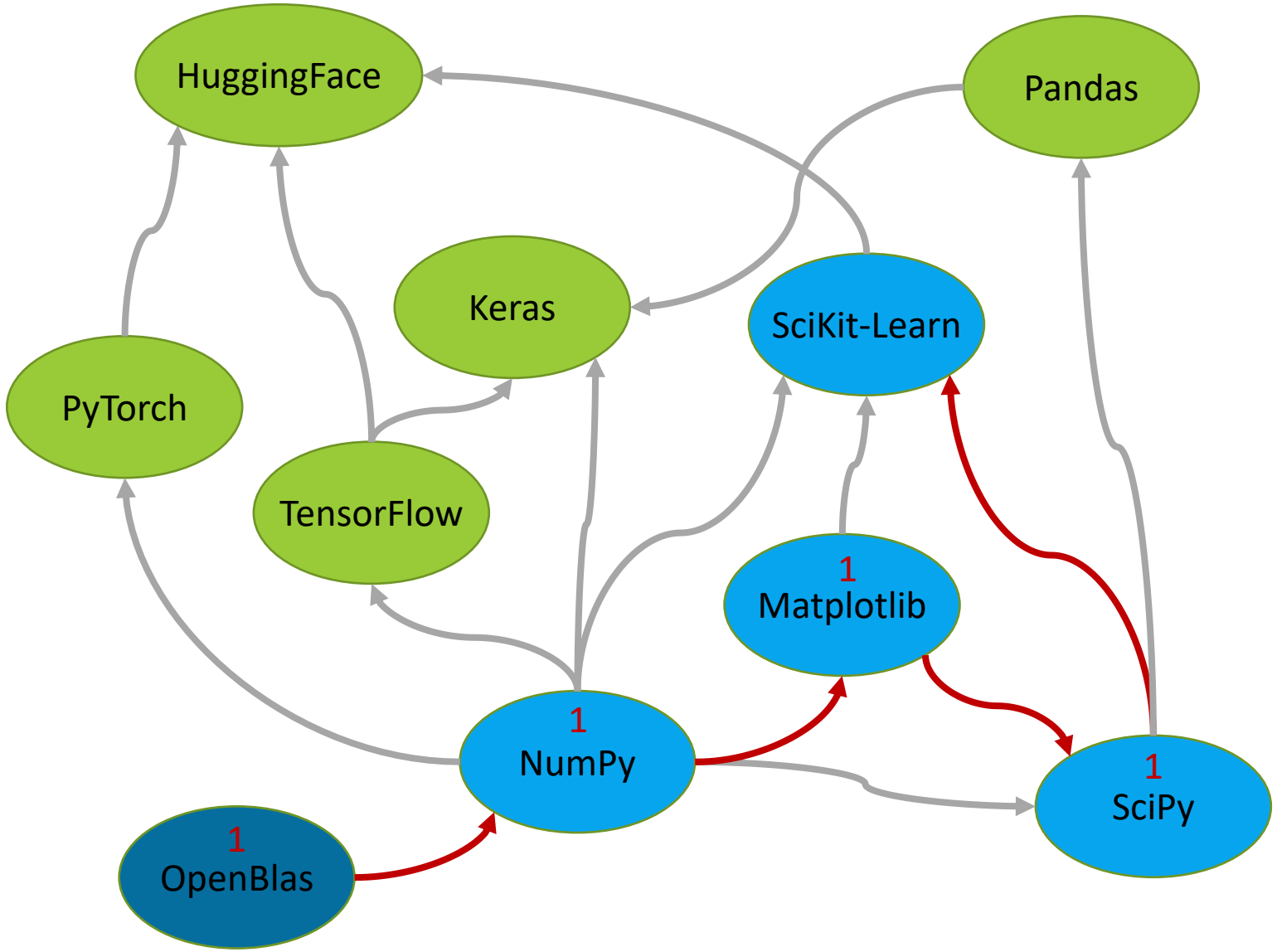
Dual Dependency Traversal
Step 3



Dual Dependency Traversal
Step 4

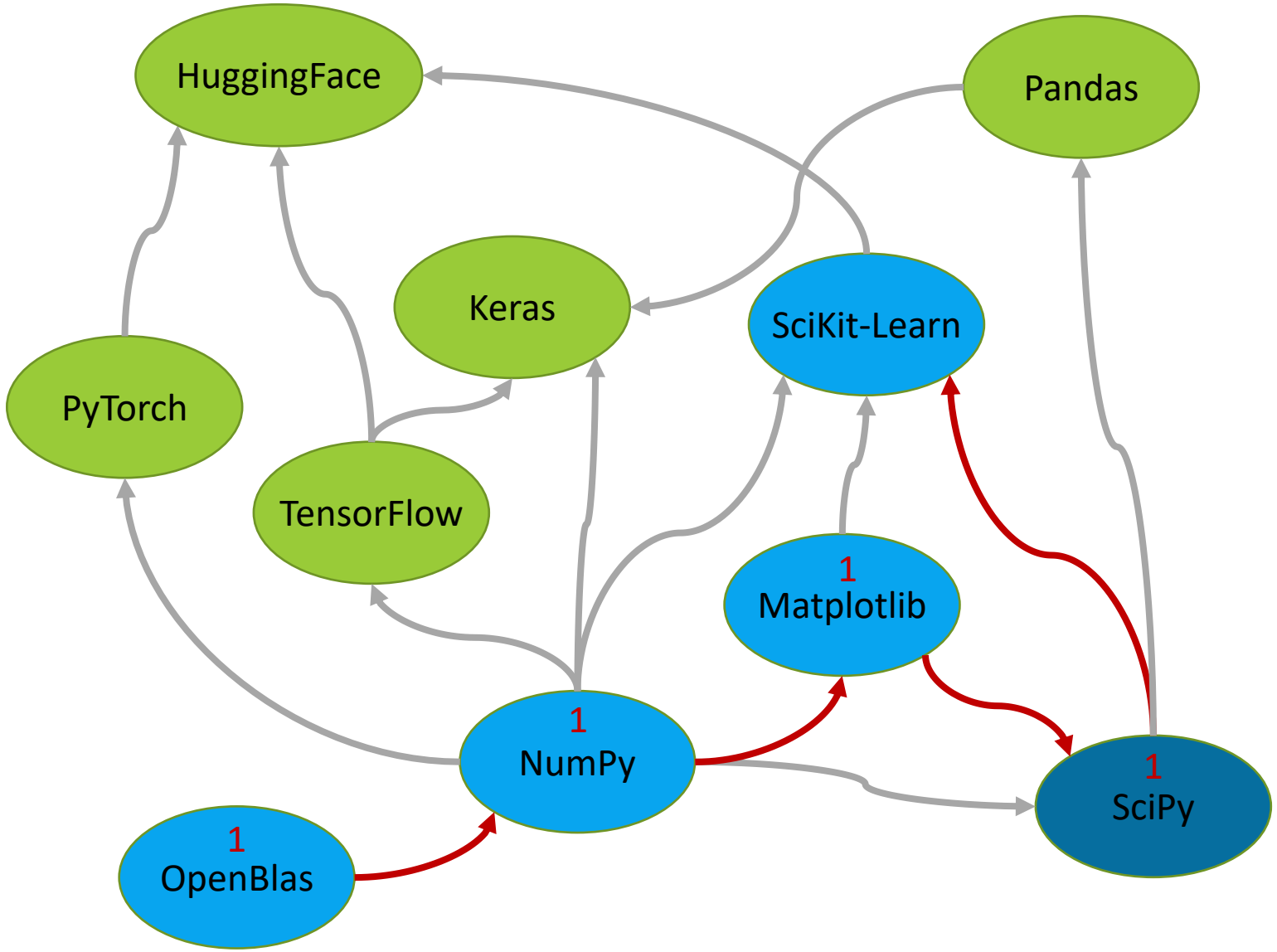


Dual Dependency Traversal
Step 5



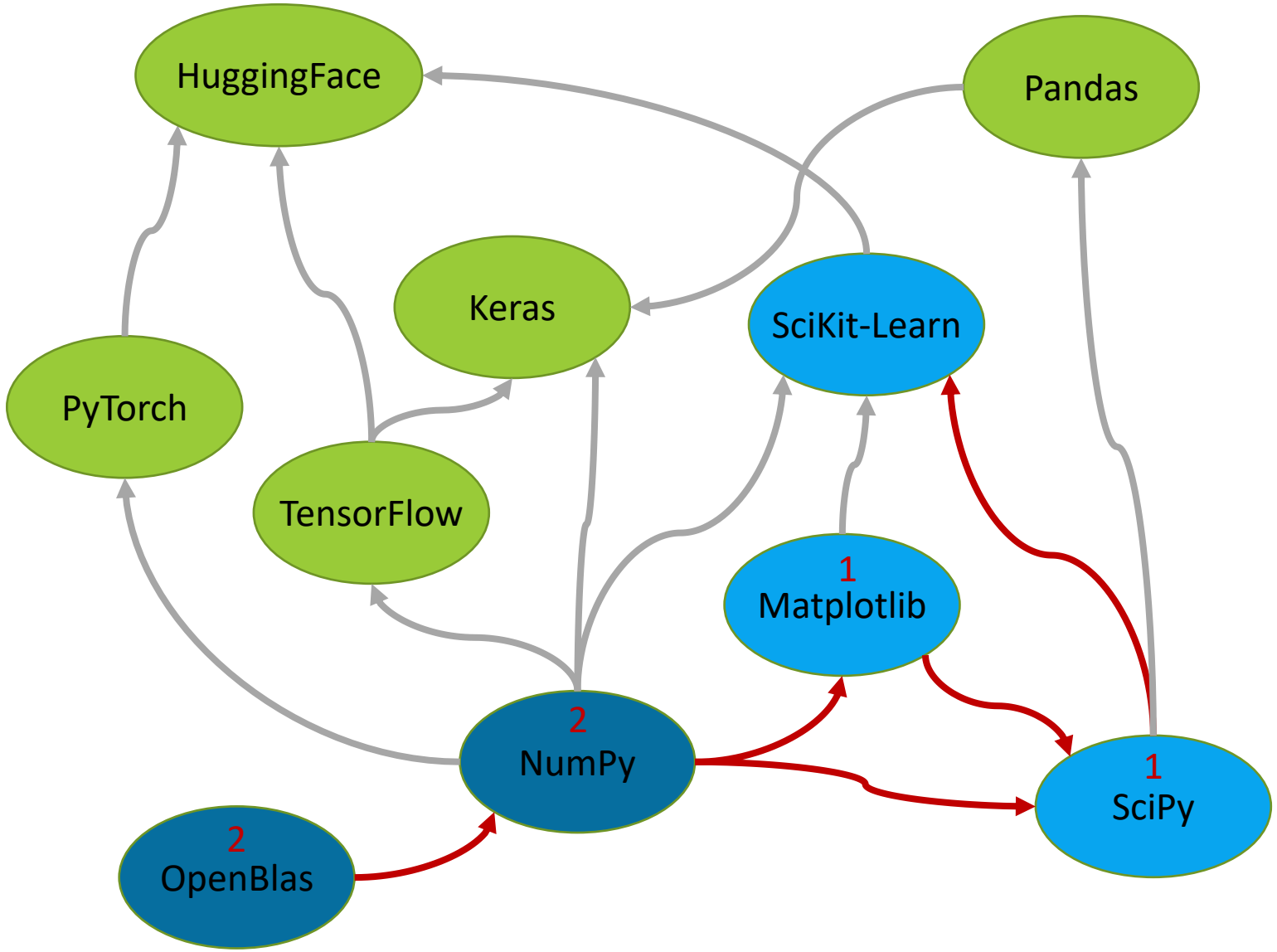
Dual Dependency Traversal Step 6

- Given we are using depth-first search, we'll bubble back up to each node to ensure we traveled every path.
  - We'll skip right to SciPy though, as we can see Matplotlib, NumPy, and OpenBlas don't have untraversed dependencies



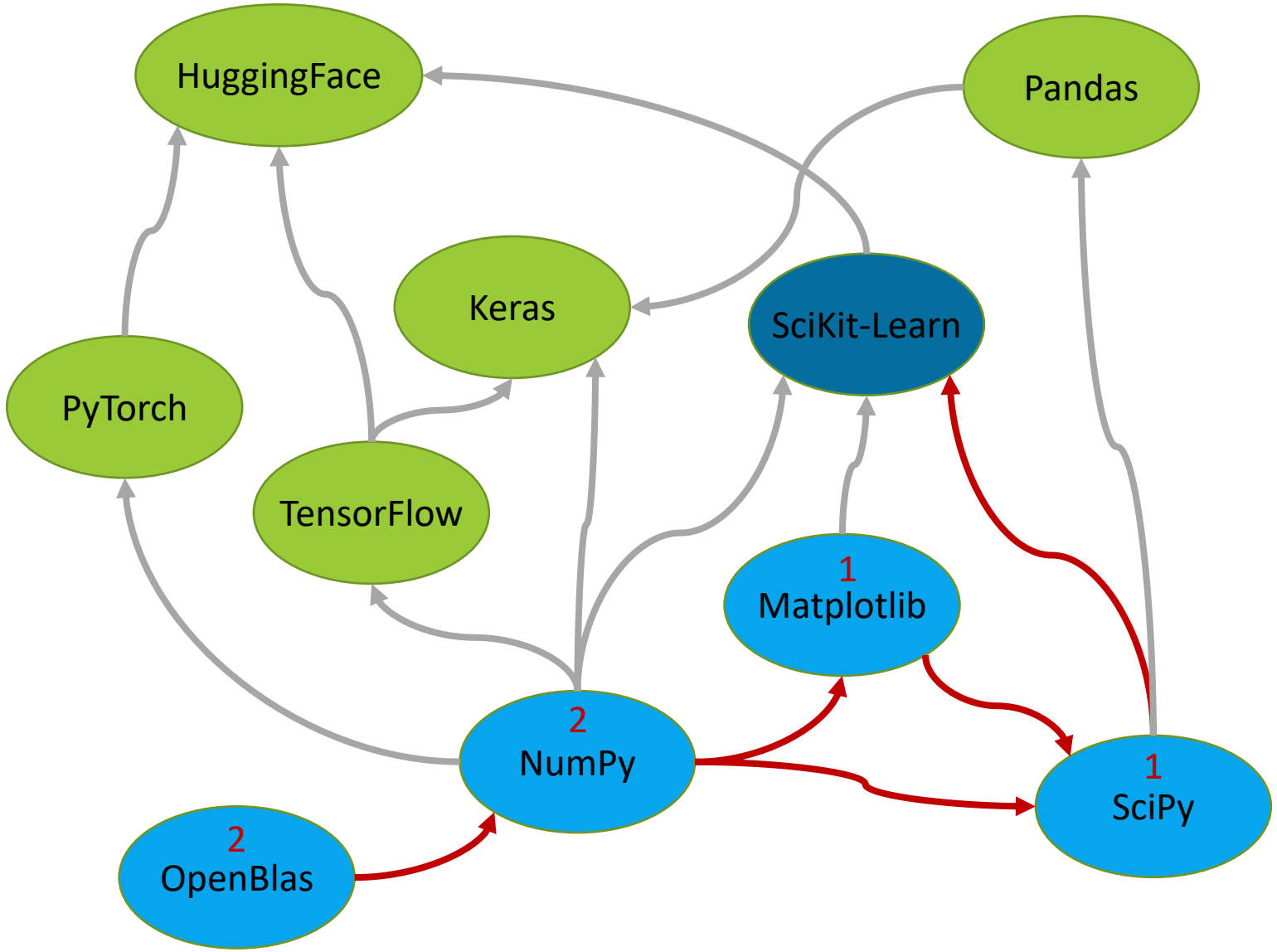
Dual Dependency Traversal Step 7

- From here, we can travel from SciPy <- NumPy, as we haven't traversed that path yet.
- This marks the second time we've seen NumPy
  - In addition, we would also travel down to OpenBlas again (depending on vertex/path uniqueness)



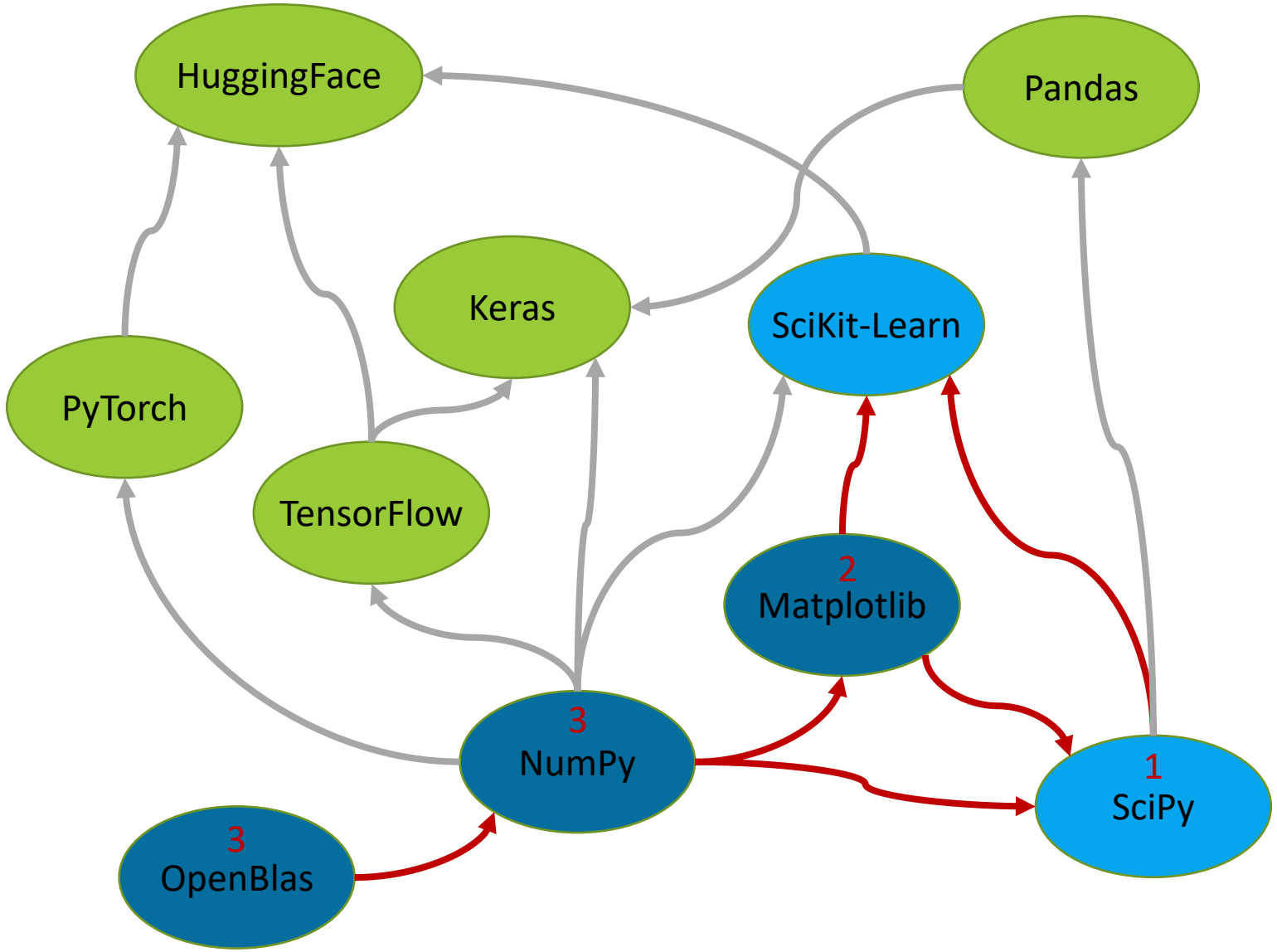
Dual Dependency Traversal Step 8

- At this point we would bubble back up to SciKit-Learn

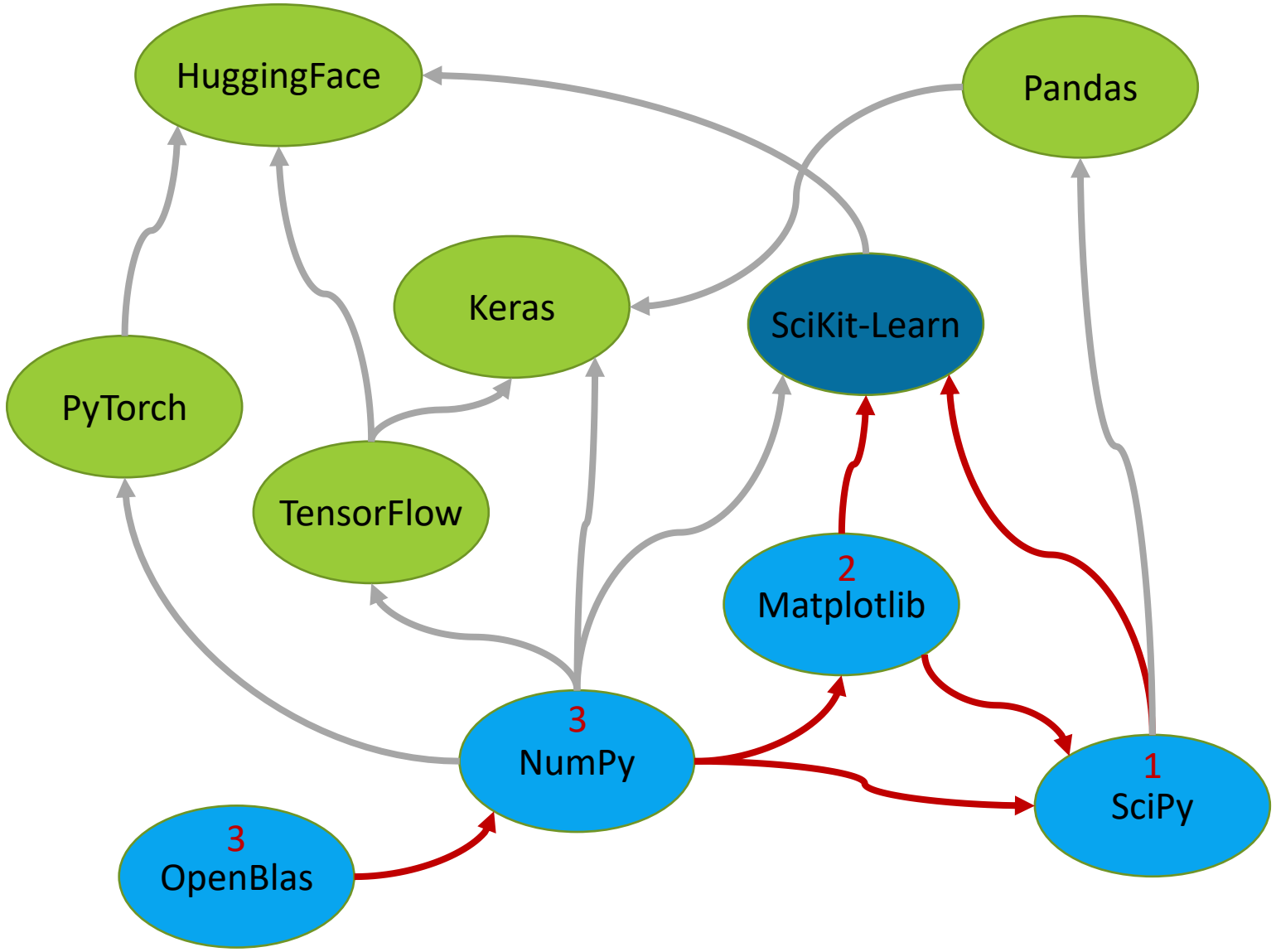


Dual Dependency Traversal Step 9

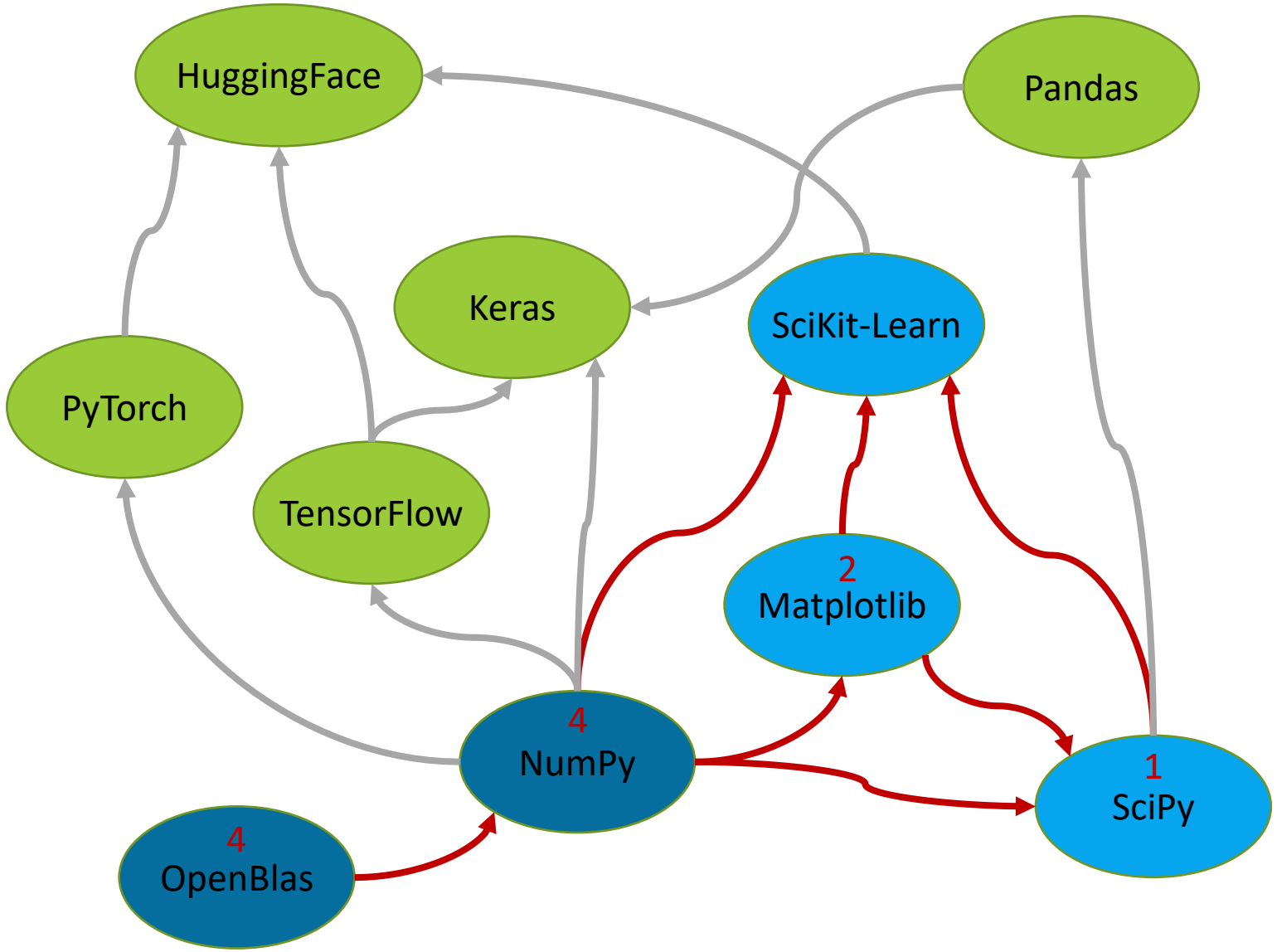
- From here we would continue traversing down Matplotlib and NumPy, given we haven't gone down those paths yet.
  - Likewise, these traversals will also hit NumPy and OpenBlas again



Dual Dependency Traversal
Step 10

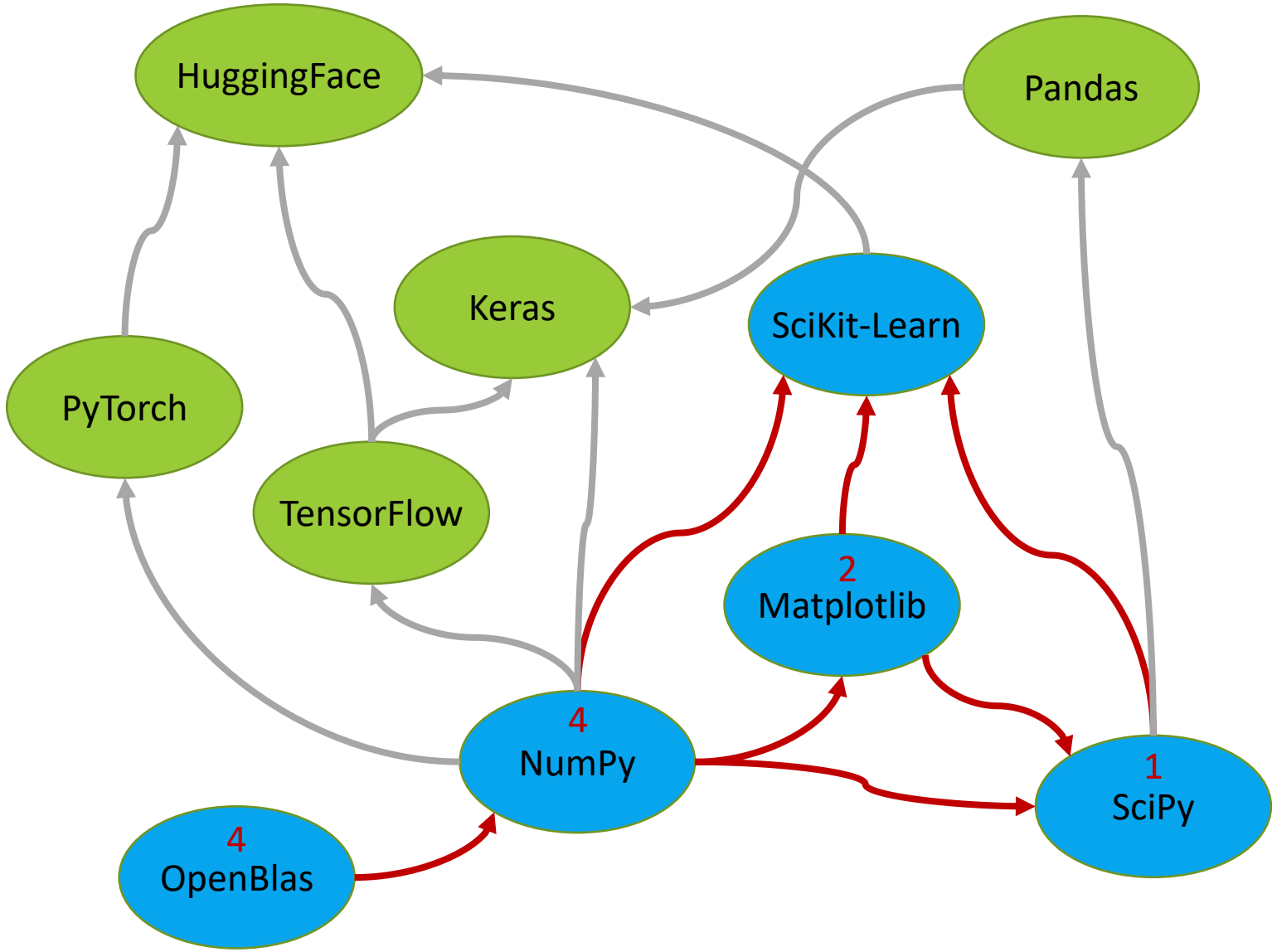


Dual Dependency Traversal
Step 10



Dual Dependency Traversal Step 10

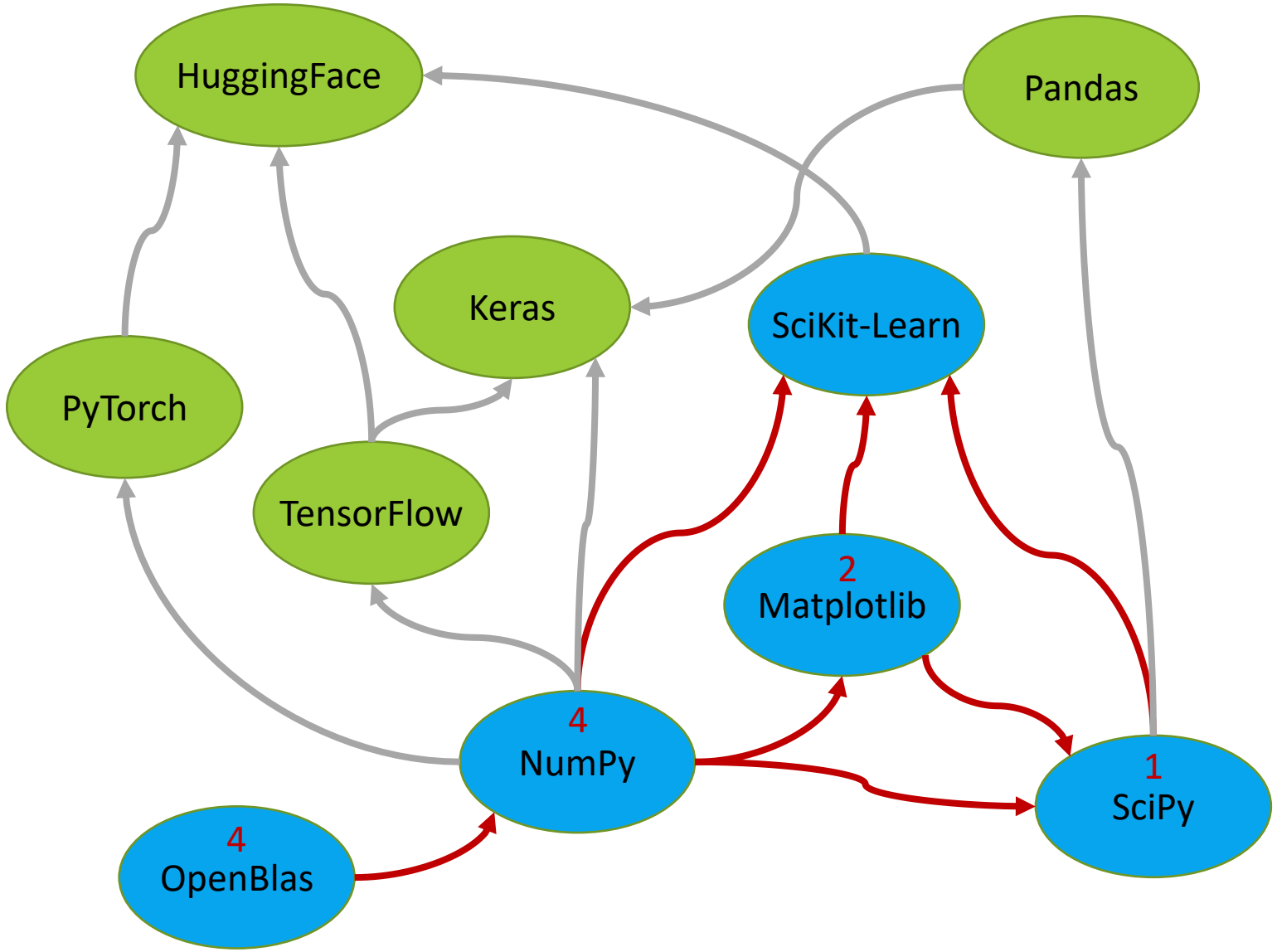
- With these results, we can see that Matplotlib, NumPy, and OpenBlas are all accessible through more than one path
  - This means that there is another dependency that already accounts for that package
- Looking at this, we can see SciKit-Learn could just depend on SciPy



Exercise 2

- Write an Arango query to identify all dual dependencies for SciKit-Learn
  - Refer to the next slide for some hints

```
FOR v, e, p IN 1..5 OUTBOUND
"pkg/SciKit-Learn" dependency
COLLECT pkgs = v.pkg_name WITH
COUNT INTO times_seen
FILTER times_seen > 1
RETURN {pkgs, times_seen}
```



Example Traversal in Twitter Data

```
FOR v, e, p IN 1..2 OUTBOUND "users/44196397" friends
COLLECT ids = v._id WITH COUNT INTO times_seen
RETURN {ids, times_seen}
```

- This query finds all the friends (OUTBOUND) relationships starting at "users/44106397" or Elon Musk
  - Using **COLLECT ids = v._id** we are grouping all vertexes by the user id
  - Using **WITH COUNT INTO times_seen** we are counting the number of times each id has been seen (much like our dependency example)

End Slide

DBMS for Data Analytics