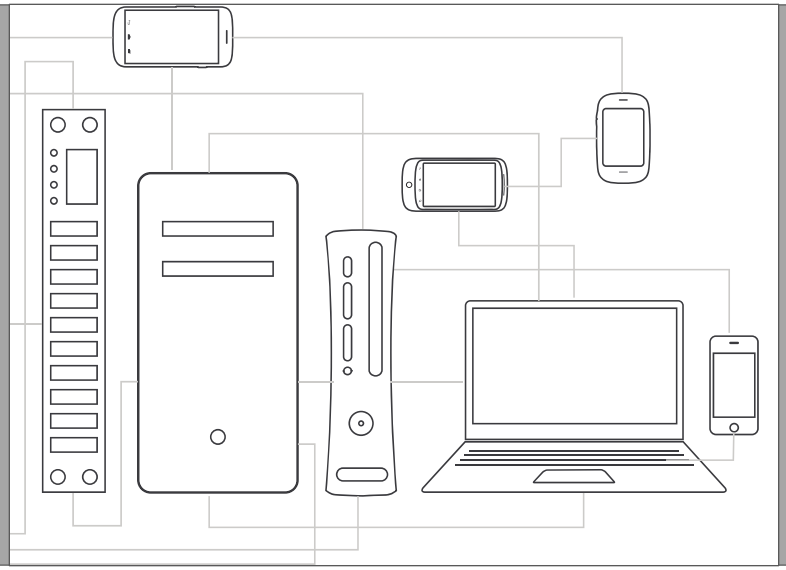
Python Autopsy: Easier Forensics Scripting (not dead snakes)



Richard Cordovano

BASIS TECHNOLOGY

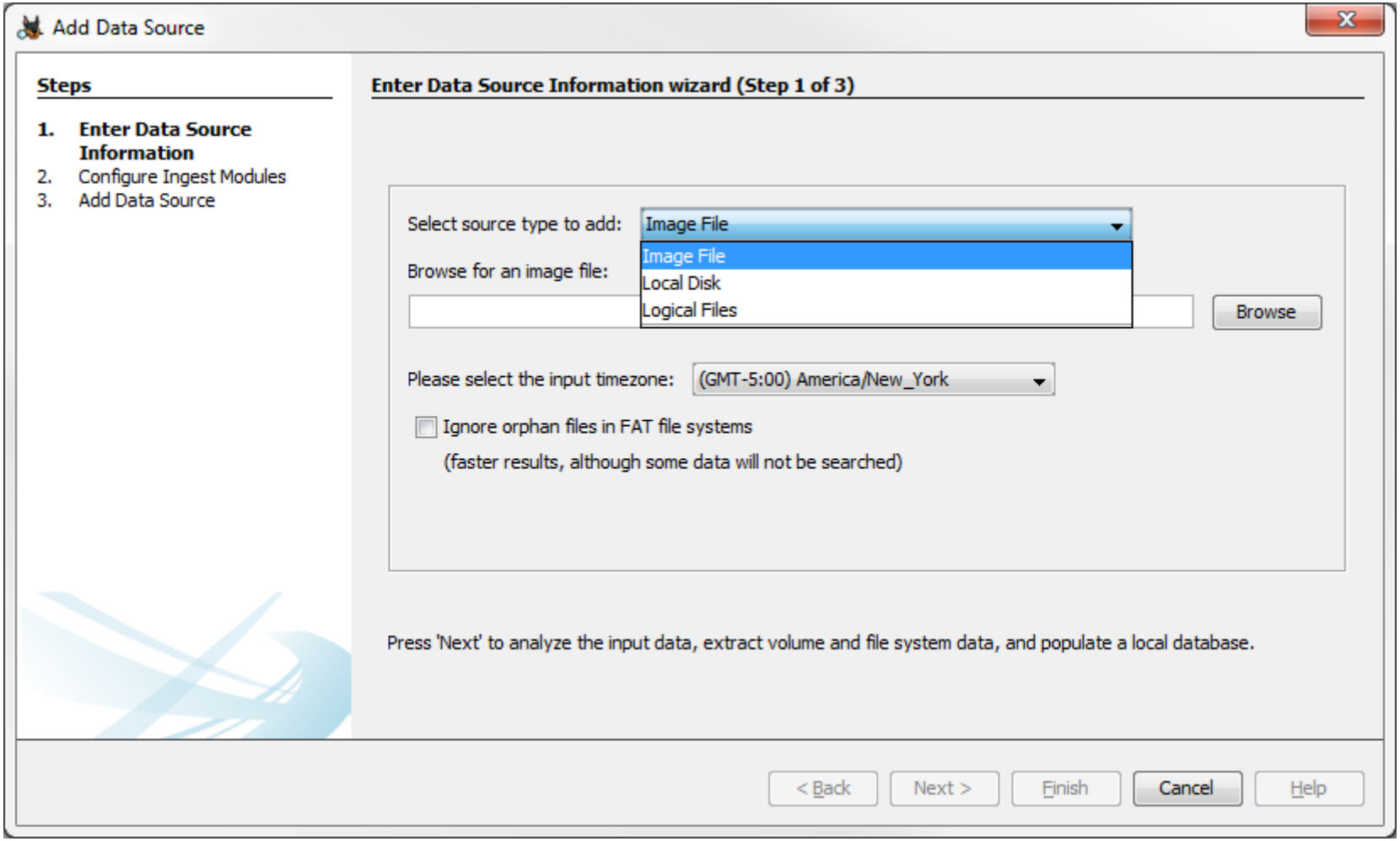
Have you heard about Autopsy?

Autopsy®

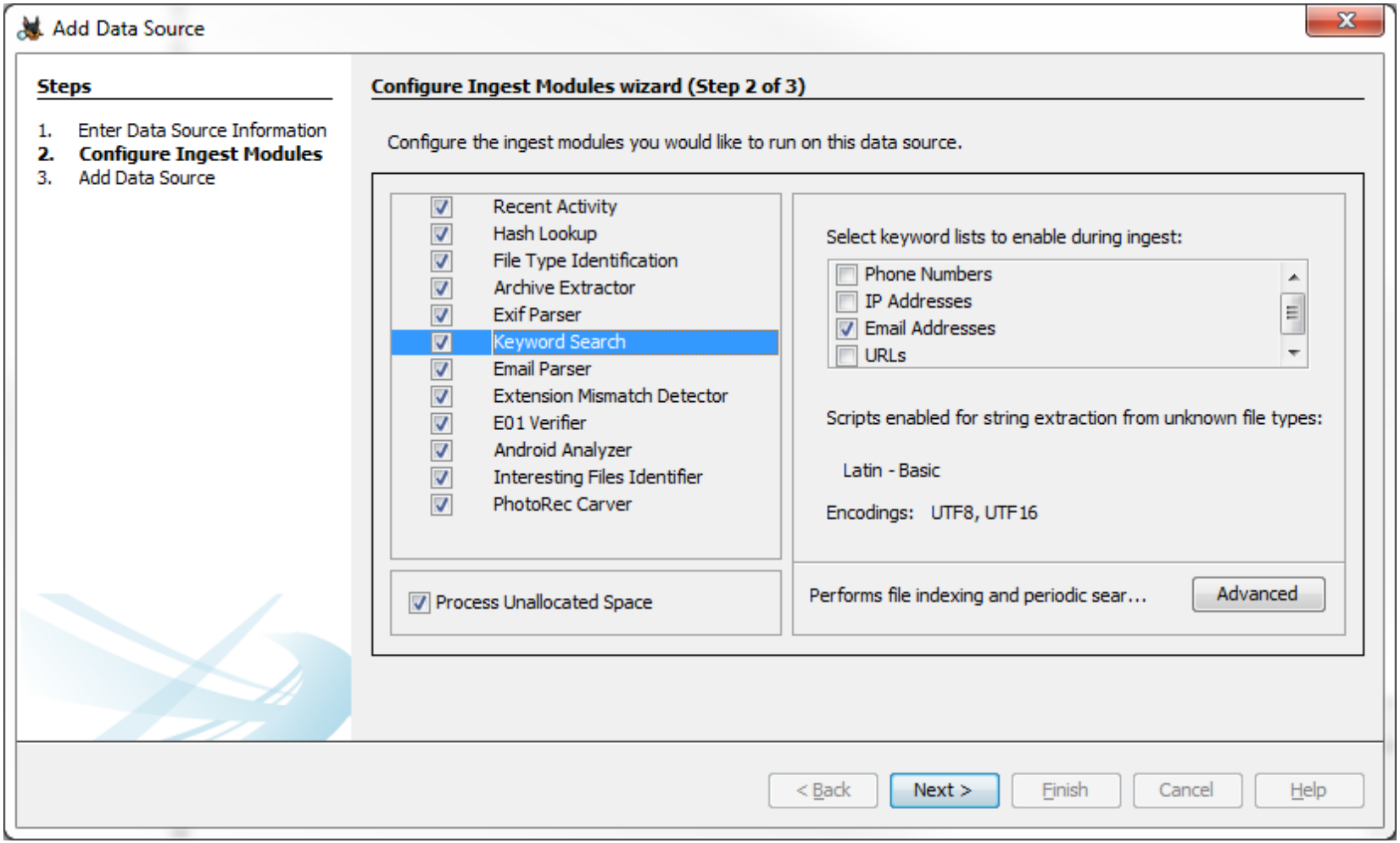
OPEN | EXTENSIBLE | FAST

- An open source desktop digital forensics tool, built on top of the SleuthKit

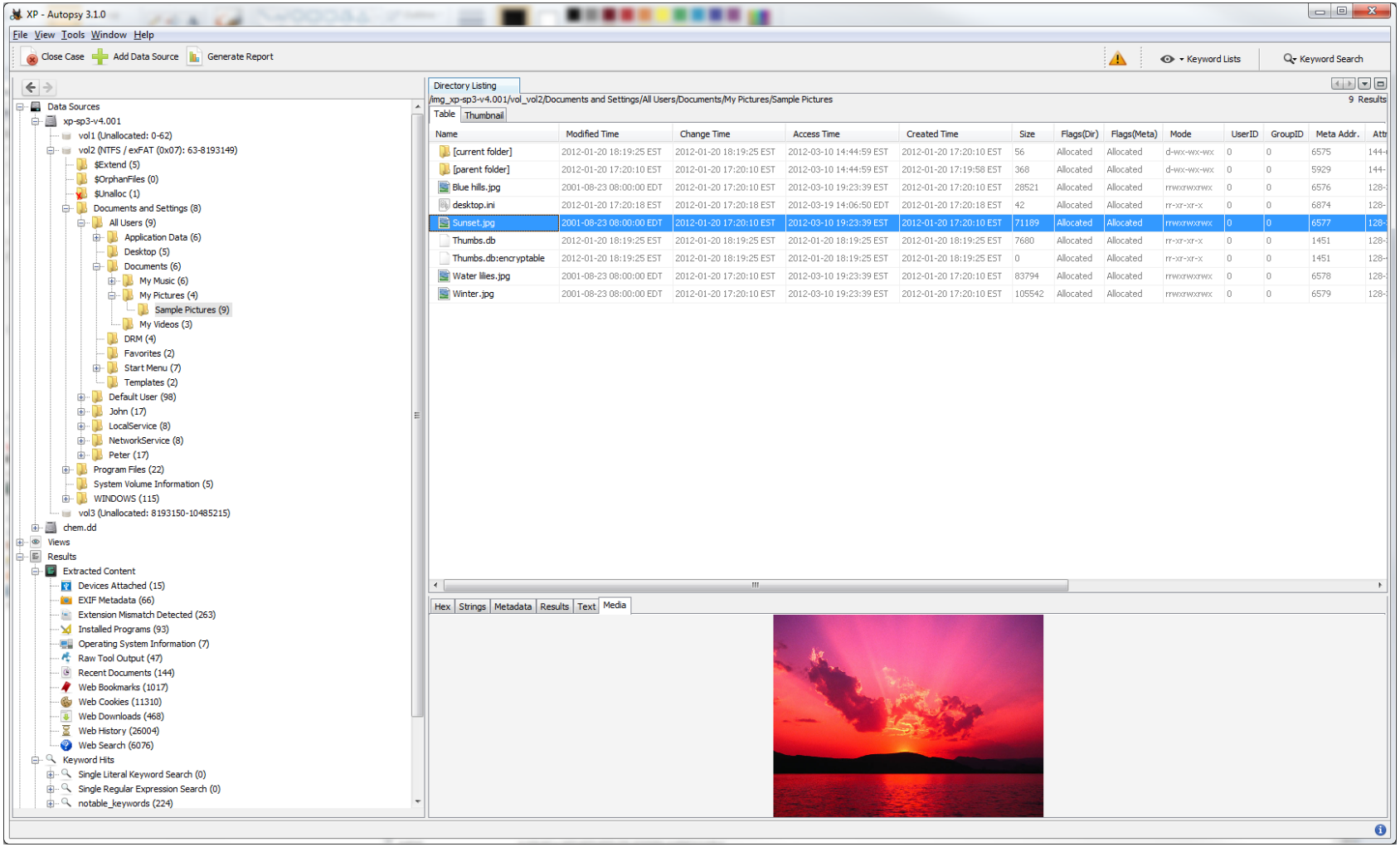
Step 1: Add a data source



Step 2: Analyze it with ingest modules



Step 3: Review the analysis results

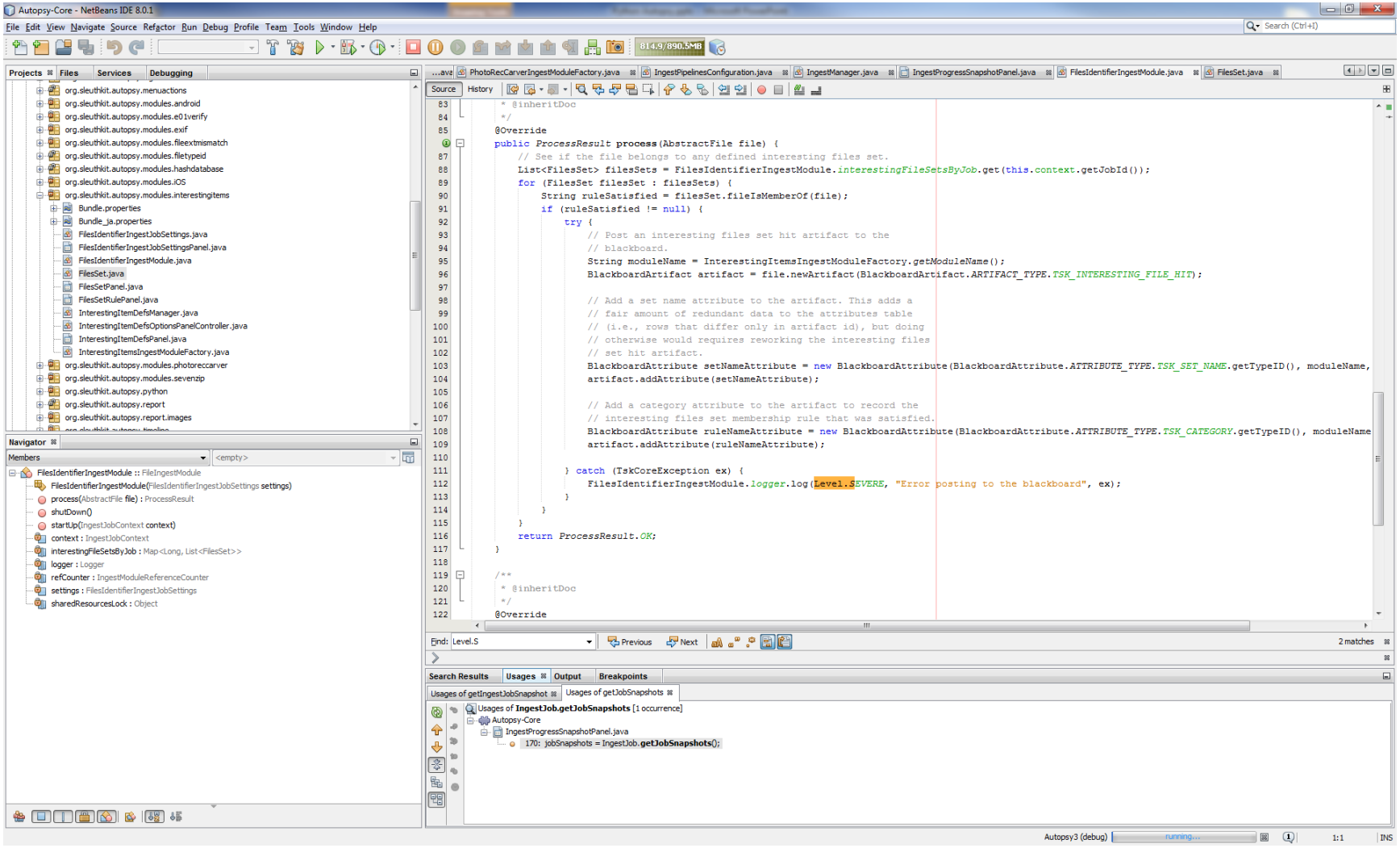


BASIS TECHNOLOGY

Ingest modules can…

- Access every byte of the data source
  - Data source file
  - Files in the data source courtesy of SleuthKit and other modules (archive extractors, carvers)
- Read and write the case database
- Use the blackboard to examine results of other modules and post results for other modules to see
- Submit files they discover (i.e., extracted, carved) for analysis
- So how do I write one?

With…Java?



BASIS TECHNOLOGY

Works for me…does it work for you?

- Are you a professional software developer?
- Do you know Java or have time to learn it?
- Are you prepared to package and distribute your Autopsy plugins as NetBeans modules?

The people want Python!

- Python is already familiar to many working in the digital forensics domain and lots of Python scripts are available for reuse
- Jython could be used as a code bridge between Java and Python to support:
  - A simple development environment, all you would need is a text editor
  - Faster development: change code and rerun without shutting down Autopsy
  - Easier module installation
- You got it!

Getting started: one simple decision

- What kind of ingest module do you want to make?
  - Data source level module if you want to analyze the data source file or a subset of files in the data source
  - File level module if you want to analyze many or all files in the data source

Finishing up: two things to do

- Write a few lines of script for an ingest module factory to make instances of your module for Autopsy
- Write as much script as you want inside your module to do your custom analysis

Ingest module factory skeleton

```
class SampleJythonIngestModuleFactory(IngestModuleFactoryAdapter):

    def getModuleDisplayName(self):
        return "Sample Jython Ingest Module"

    def getModuleDescription(self):
        return "A sample Jython ingest module"

    def getModuleVersionNumber(self):
        return "1.0"

    def isFileIngestModuleFactory(self):
        return True

    def createFileIngestModule(self, ingestOptions):
        return SampleJythonFileIngestModule()
```

Data Source ingest module skeleton

```
class DataSourceIngestModuleSkeleton(DataSourceIngestModule):

    def startUp(self, context):
        pass

    def process(self, dataSource, progressBar):

        return IngestModule.ProcessResult.OK;
```

File Ingest Module Skeleton

```
class FileIngestModuleSkeleton(FileIngestModule):

    def startUp(self, context):
        pass

    def process(self, file):

        return IngestModule.ProcessResult.OK;

    def shutDown(self):
        pass
```

Let’s make an ingest module!

- We’ll make it simple, let’s find all files with “ebola” in the name and post them to the blackboard
- We only want some of the files, so we want to make a data source ingest module (or do we?)

Ebola Finder module factory

```
class EbolaFileFinderFactory(IngestModuleFactoryAdapter):

    def getModuleDisplayName(self):
        return "Ebola File Finder"

    def getModuleDescription(self):
        return "Finds files that have 'ebola' in the file name."

    def getModuleVersionNumber(self):
        return "1.0"

    def isDataSourceIngestModuleFactory(self):
        return True

    def createDataSourceIngestModule(self, ingestOptions):
        return EbolaFileFinder()
```

How about this?

```
def process(self, dataSource, progressBar):

    # Get the file manager service
    autopsyCase = Case.getCurrentCase()
    sleuthkitCase = autopsyCase.getSleuthkitCase()
    services = Services(sleuthkitCase)
    fileManager = services.getFileManager()

    # Get files with "ebola" in name and post them to the blackboard.
    files = fileManager.findFiles(dataSource, "%malaria%")
    for file in files:
        art = file.newArtifact(
            BlackboardArtifact.ARTIFACT_TYPE.TSK_INTERESTING_FILE_HIT)
        att = BlackboardAttribute(
            BlackboardAttribute.ATTRIBUTE_TYPE.TSK_SET_NAME.getTypeID(),
            "Ebola File Finder", "Ebola Files")
        art.addAttribute(att)

    return IngestModule.ProcessResult.OK;
```

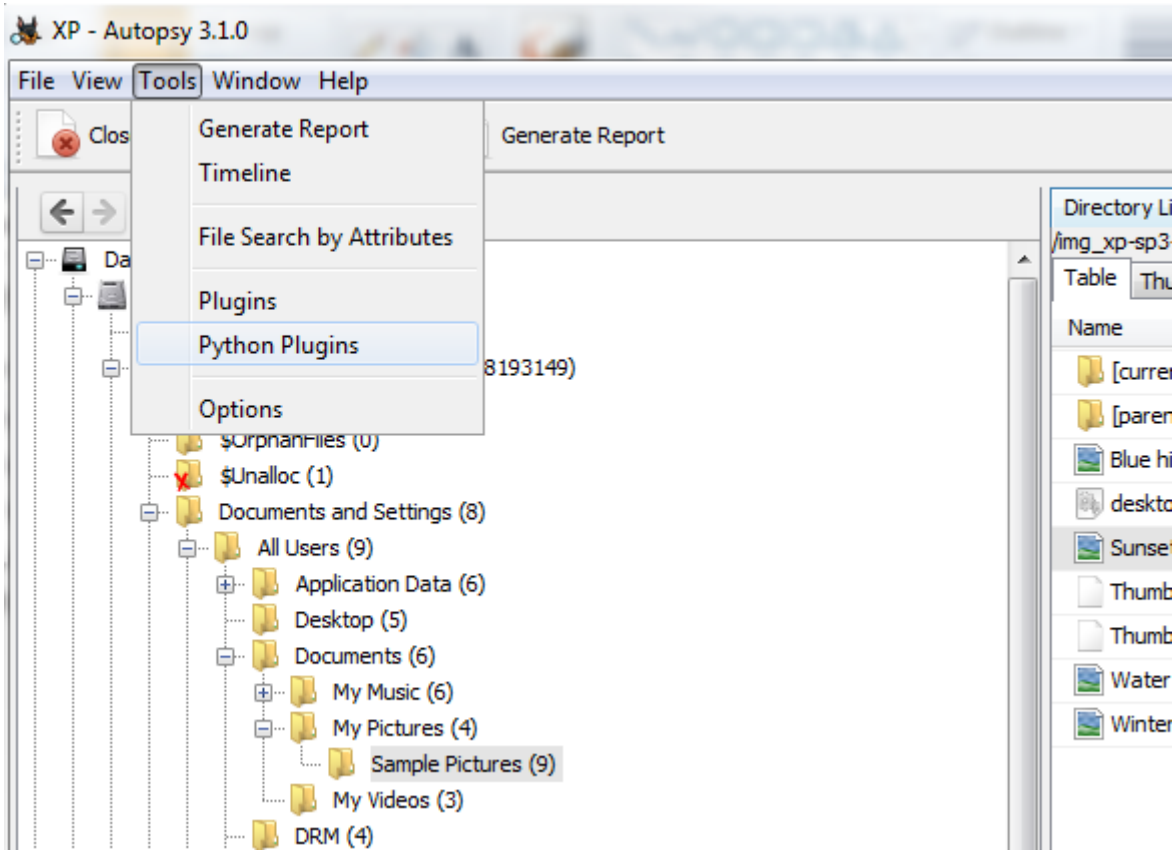
Better!

```
def process(self, file):
    # If the file name has "ebola" in it, post it to the blackboard.
    if file.getName().find("ebola") != -1:
        art = file.newArtifact(
            BlackboardArtifact.ARTIFACT_TYPE.TSK_INTERESTING_FILE_HIT)
        att = BlackboardAttribute(
            BlackboardAttribute.ATTRIBUTE_TYPE.TSK_SET_NAME.getTypeID(),
            "Ebola File Finder", "Text Files")
        art.addAttribute(att)

    return IngestModule.ProcessResult.OK
```

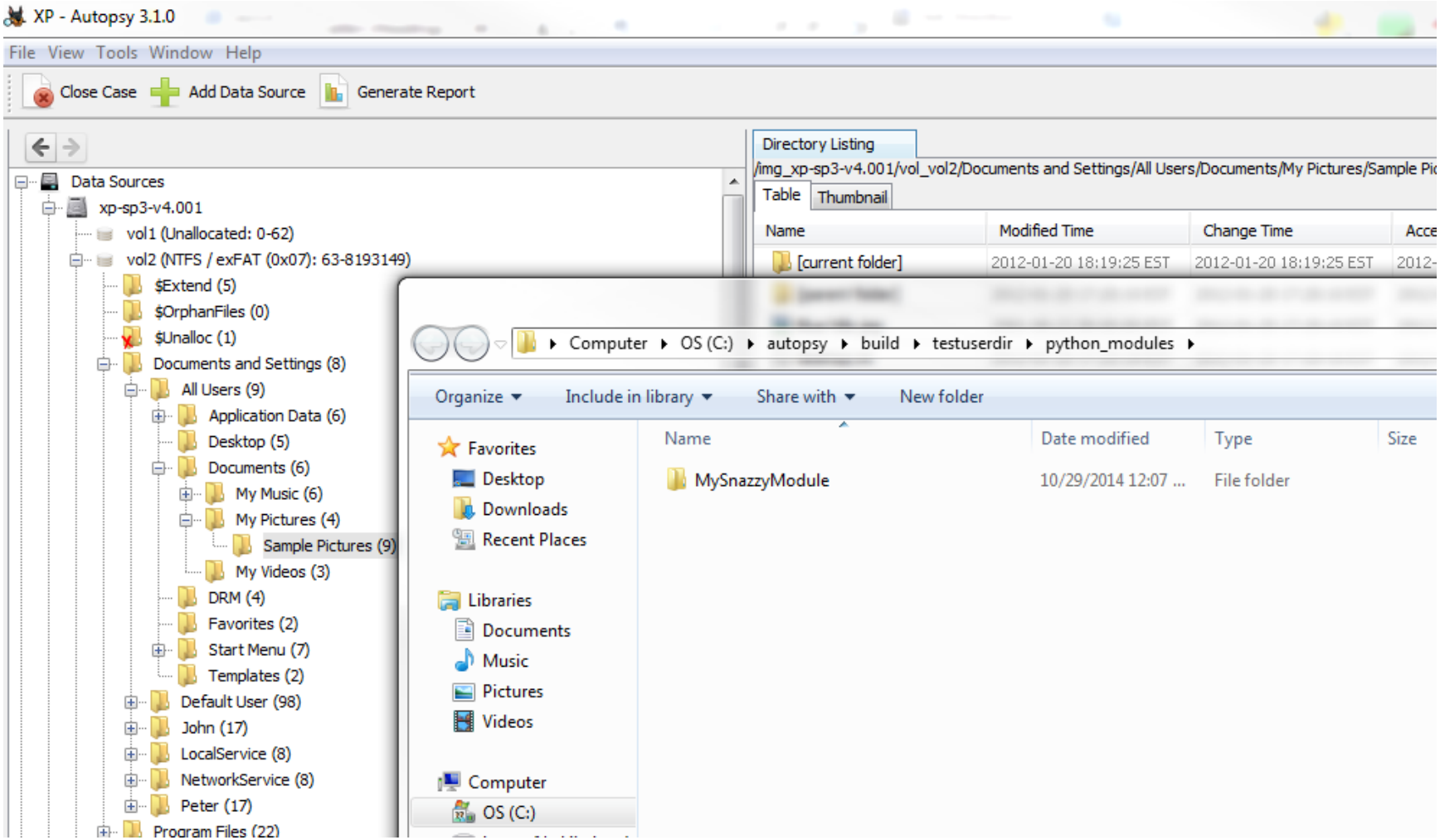
Ebola Finder file module factory

```
class EbolaFileFinderFactory(IngestModuleFactoryAdapter):

    def getModuleDisplayName(self):
        return "Ebola File Finder"

    def getModuleDescription(self):
        return "Finds files that have 'ebola' in the file name."

    def getModuleVersionNumber(self):
        return "1.0"

    def isFileIngestModuleFactory(self):
        return True

    def createFileIngestModule(self, ingestOptions):
        return EbolaFileFinder()
```

Tools -> Python Plugins



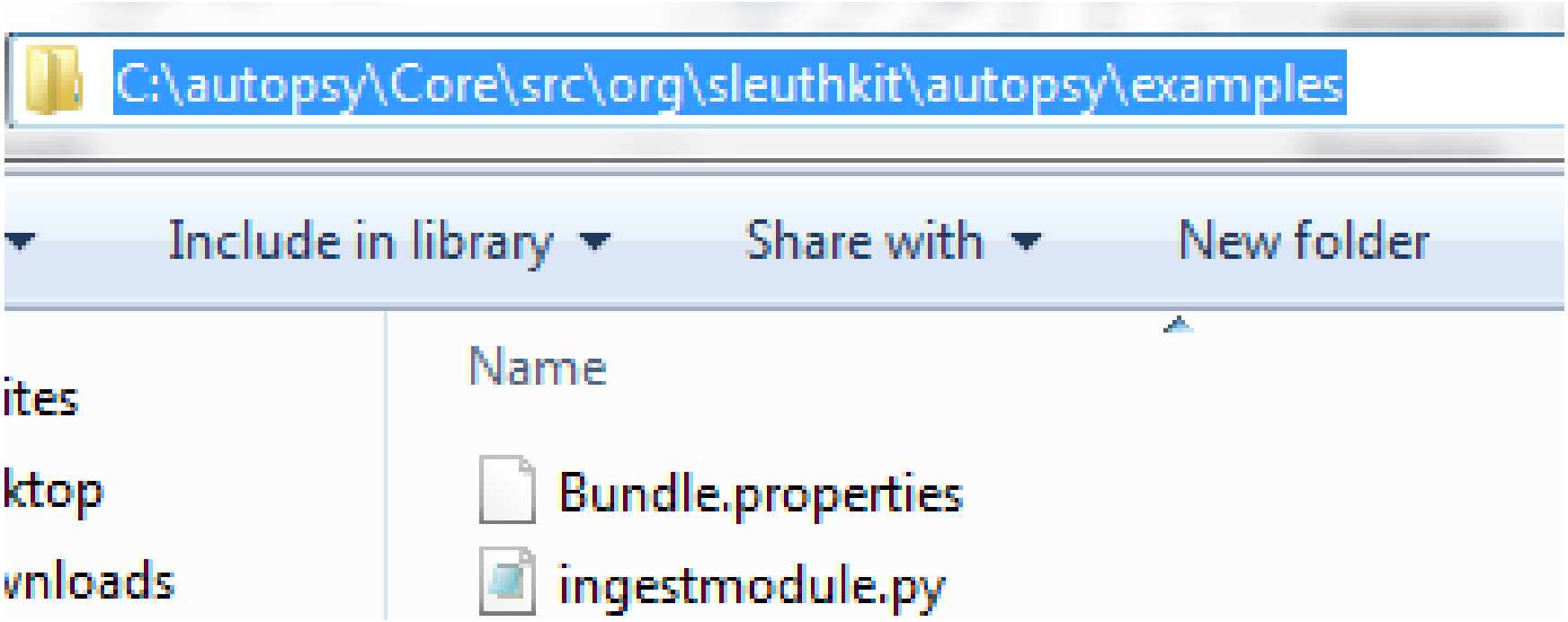
Drop!



Resources: SleuthKit Wiki

- http://www.sleuthkit.org/autopsy/docs/api-docs/3.1/mod_dev_py_page.html
- http://www.sleuthkit.org/autopsy/docs/api-docs/3.1/platform_page.html
- http://www.sleuthkit.org/autopsy/docs/api-docs/3.1/mod_ingest_page.html
- http://www.sleuthkit.org/autopsy/docs/api-docs/3.1/mod_report_page.html

Resources: Sample in source



Getting file bytes

```
# Read the contents of the file.
inputStream = ReadContentInputStream(file)
buffer = jarray.zeros(1024, "b")
len = inputStream.read(buffer)
while (len != -1):
    len = inputStream.read(buffer)
```

The End (Questions?)

BASIS TECHNOLOGY