snakemake

A framework for reproducible data analysis

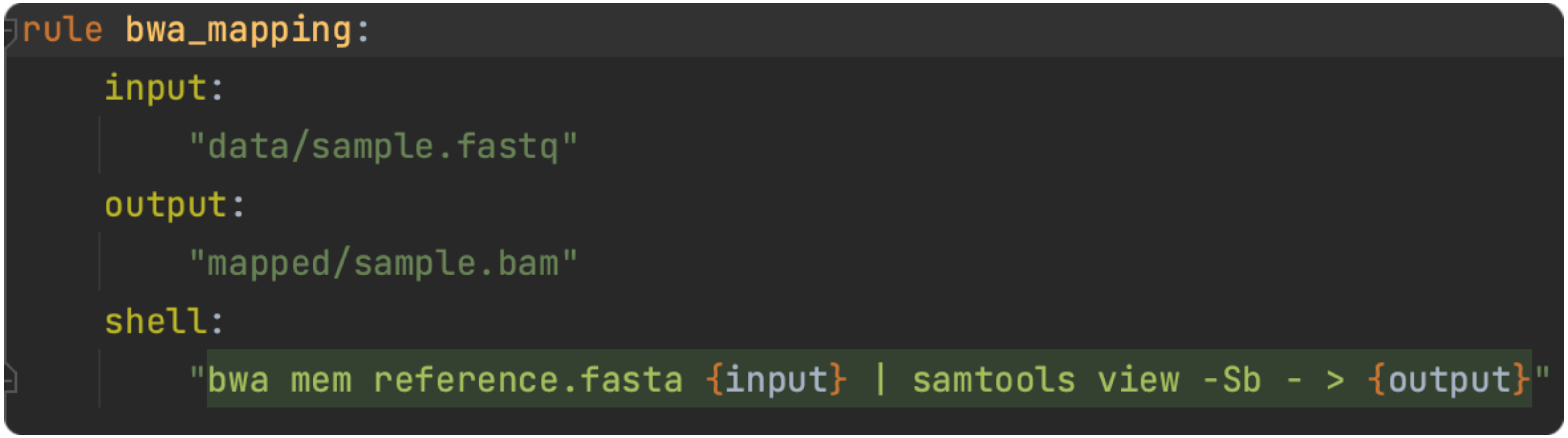
Workflow management

- Reproducibility
- Scalability
- Flexibility
- Documentation

snakemake

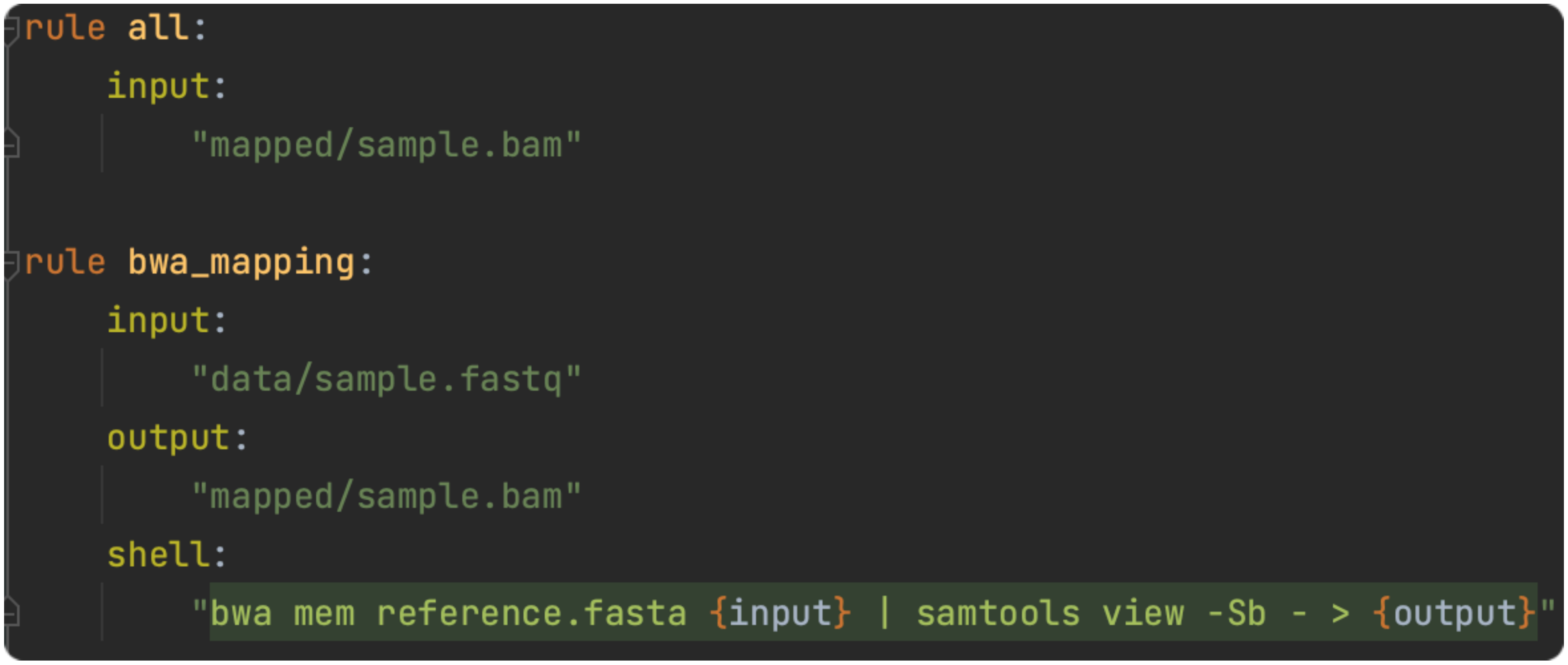
A framework for reproducible data analysis

Rule



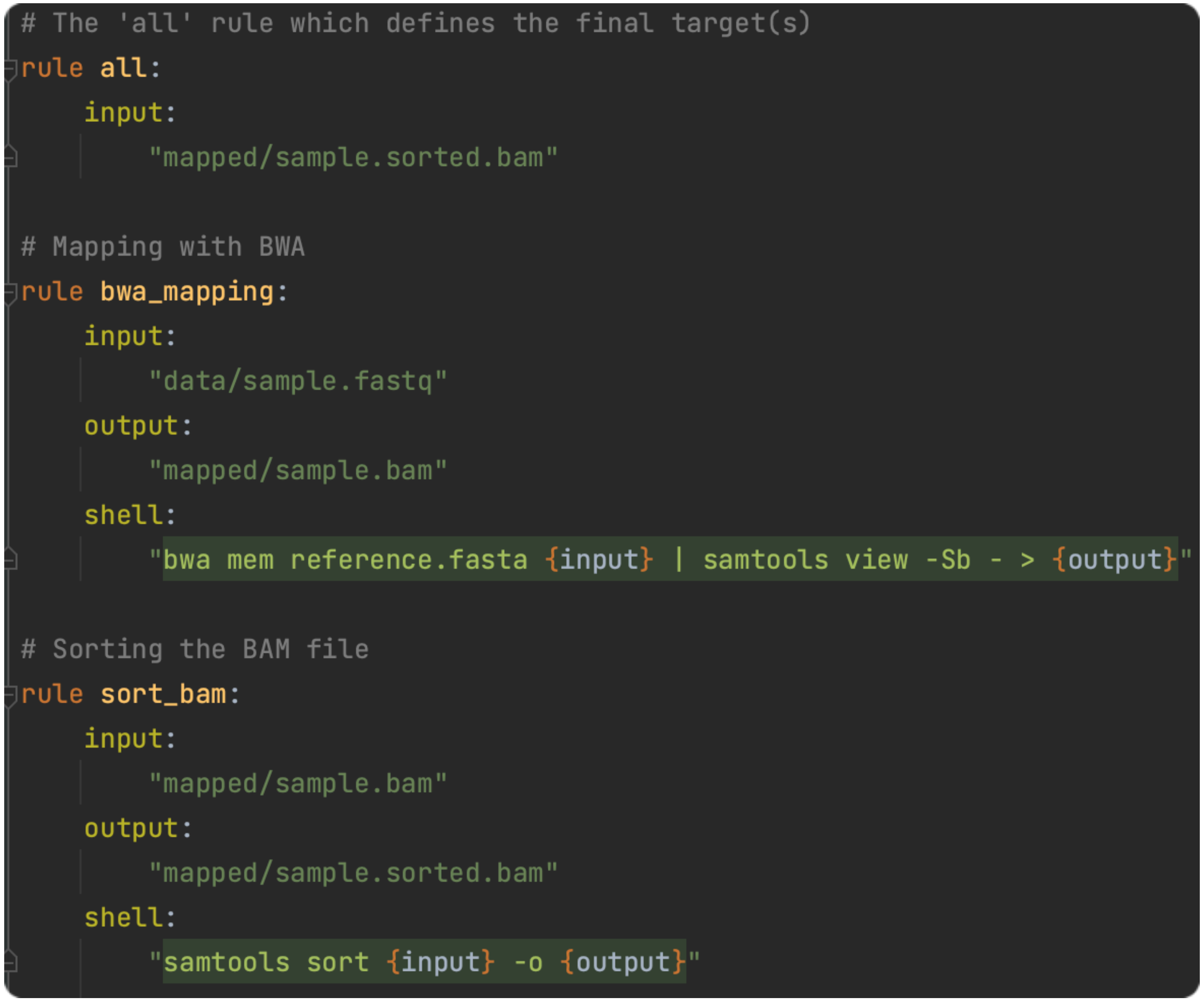
```
rule bwa_mapping:
    input:
        "data/sample.fastq"
    output:
        "mapped/sample.bam"
    shell:
        "bwa mem reference.fasta {input} | samtools view -Sb - > {output}"
```

Rule all



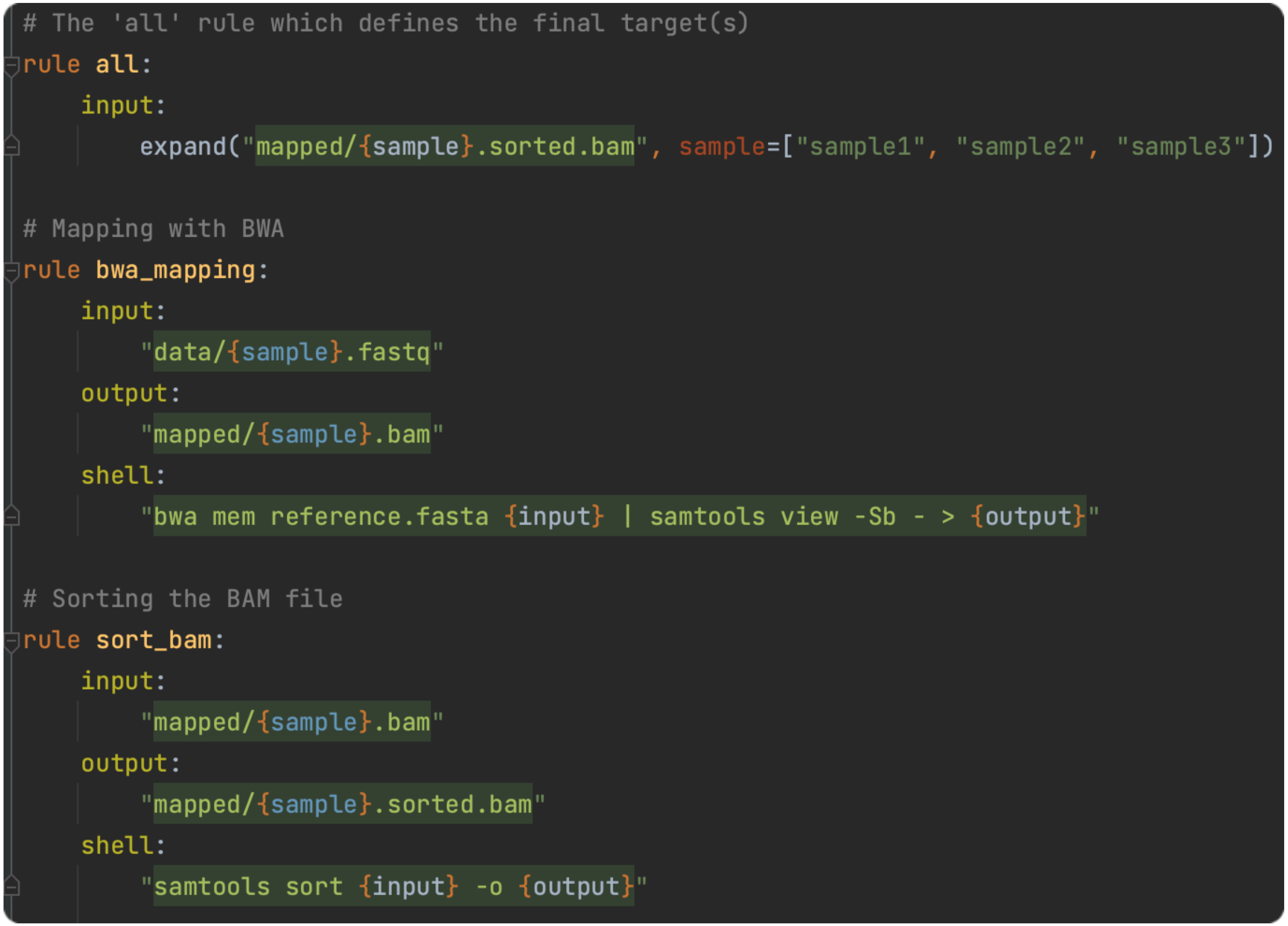
```
rule all:
    input:
        "mapped/sample.bam"

rule bwa_mapping:
    input:
        "data/sample.fastq"
    output:
        "mapped/sample.bam"
    shell:
        "bwa mem reference.fasta {input} | samtools view -Sb - > {output}"
```

Chaining rules



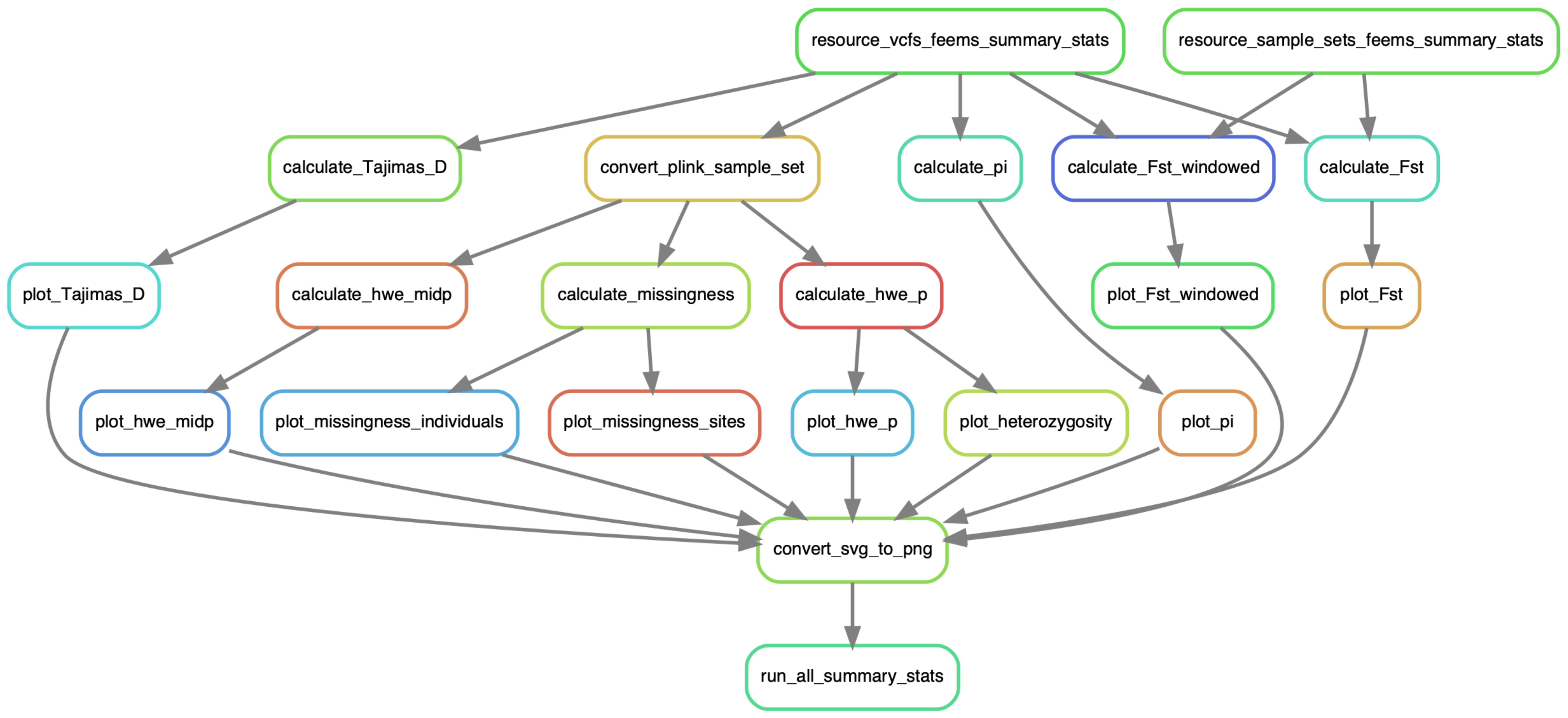
```
# The 'all' rule which defines the final target(s)
rule all:
    input:
        "mapped/sample.sorted.bam"

# Mapping with BWA
rule bwa_mapping:
    input:
        "data/sample.fastq"
    output:
        "mapped/sample.bam"
    shell:
        "bwa mem reference.fasta {input} | samtools view -Sb - > {output}"

# Sorting the BAM file
rule sort_bam:
    input:
        "mapped/sample.bam"
    output:
        "mapped/sample.sorted.bam"
    shell:
        "samtools sort {input} -o {output}"
```

Wildcards

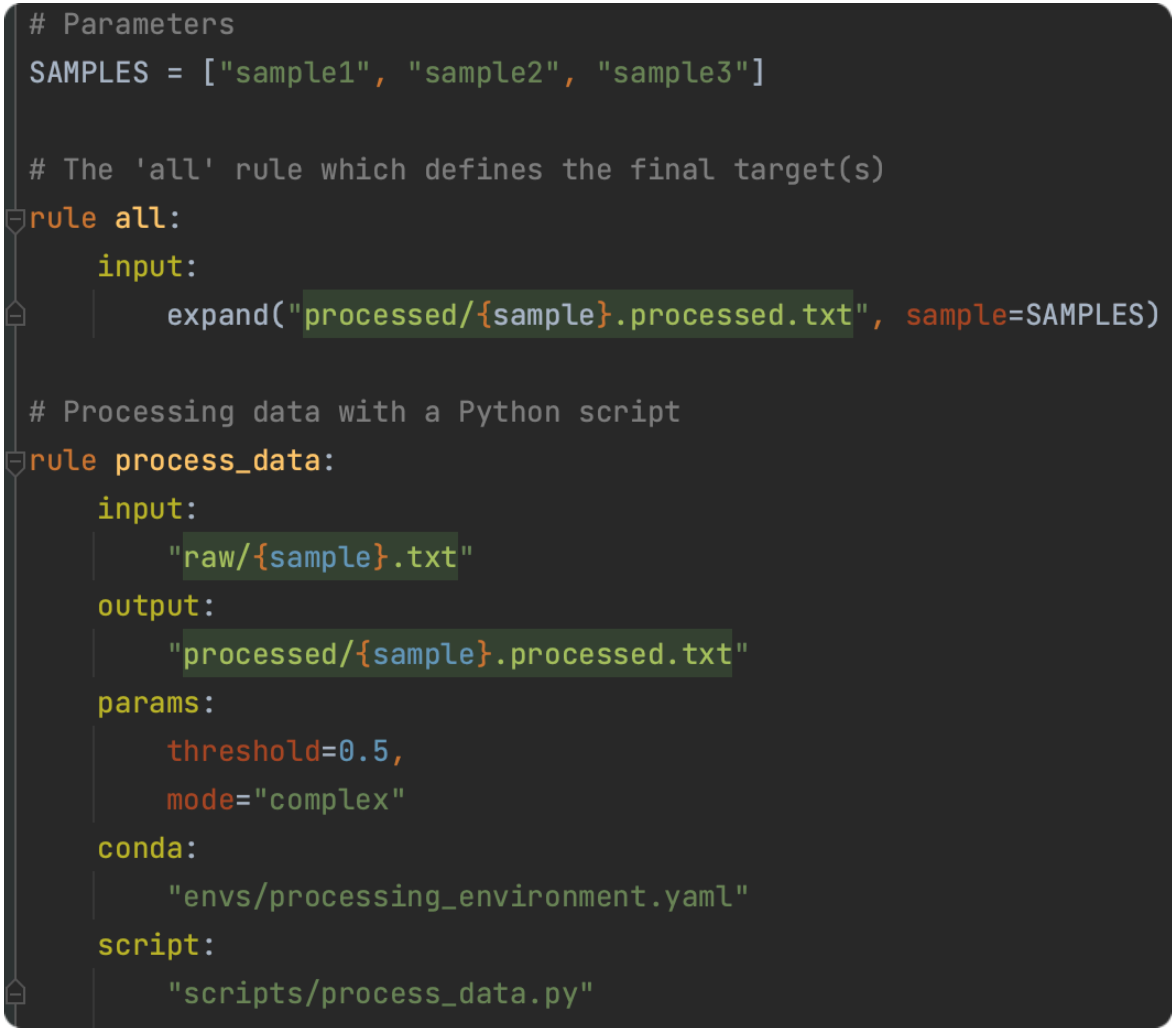


```
# The 'all' rule which defines the final target(s)
rule all:
    input:
        expand("mapped/{sample}.sorted.bam", sample=["sample1", "sample2", "sample3"])

# Mapping with BWA
rule bwa_mapping:
    input:
        "data/{sample}.fastq"
    output:
        "mapped/{sample}.bam"
    shell:
        "bwa mem reference.fasta {input} | samtools view -Sb - > {output}"

# Sorting the BAM file
rule sort_bam:
    input:
        "mapped/{sample}.bam"
    output:
        "mapped/{sample}.sorted.bam"
    shell:
        "samtools sort {input} -o {output}"
```

Complex workflows

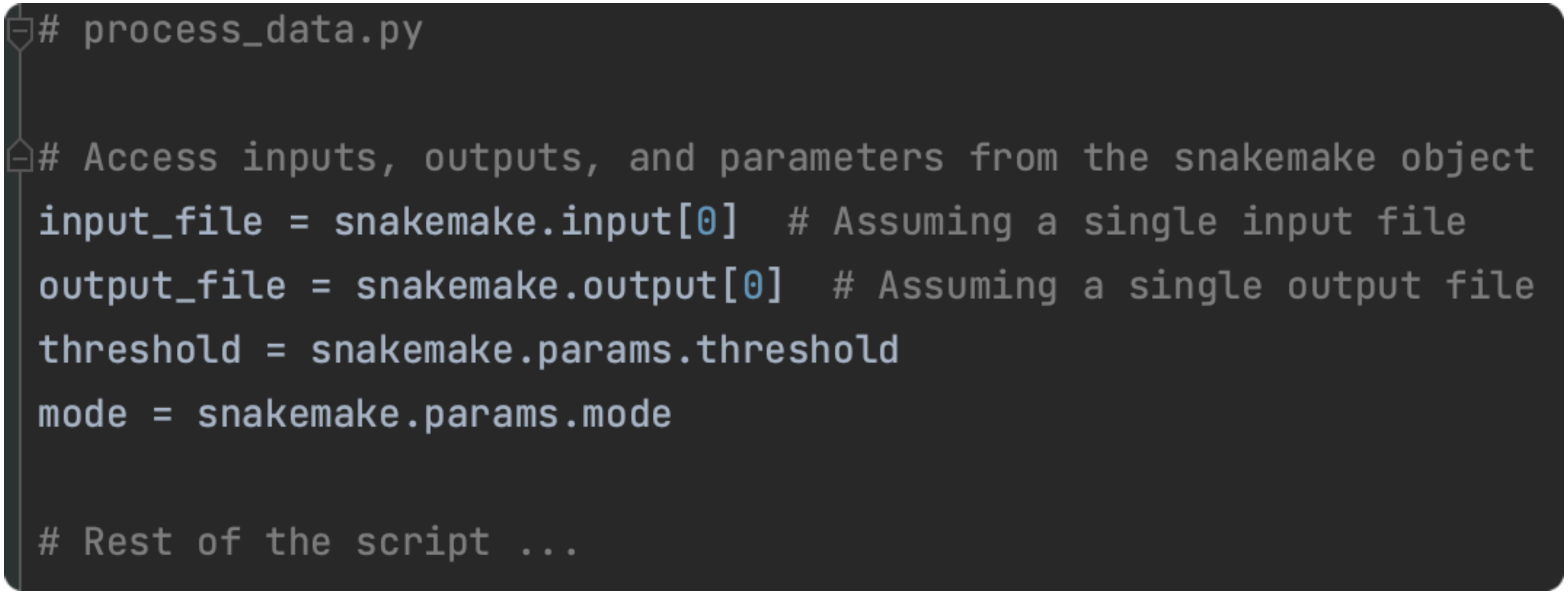


Additional directives



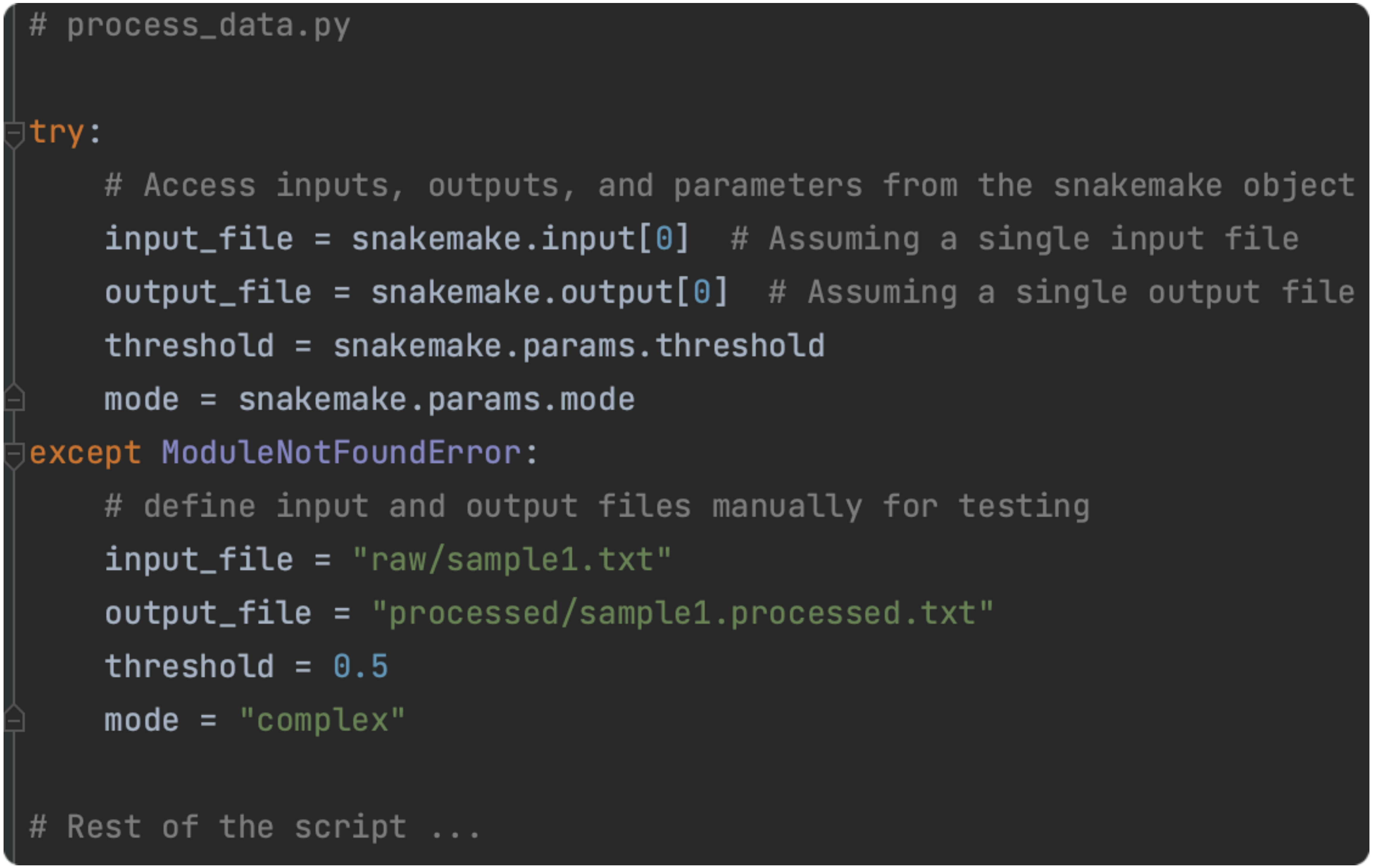
```
# Parameters
SAMPLES = ["sample1", "sample2", "sample3"]

# The 'all' rule which defines the final target(s)
rule all:
    input:
        expand("processed/{sample}.processed.txt", sample=SAMPLES)

# Processing data with a Python script
rule process_data:
    input:
        "raw/{sample}.txt"
    output:
        "processed/{sample}.processed.txt"
    params:
        threshold=0.5,
        mode="complex"
    conda:
        "envs/processing_environment.yaml"
    script:
        "scripts/process_data.py"
```

External python script



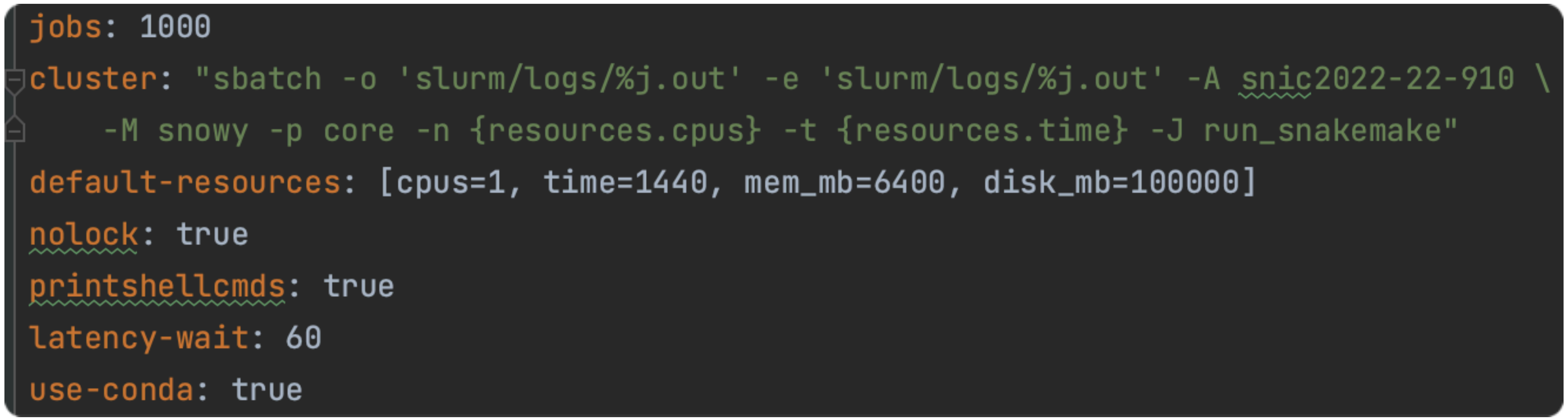
```python
# process_data.py

# Access inputs, outputs, and parameters from the snakemake object
input_file = snakemake.input[0]  # Assuming a single input file
output_file = snakemake.output[0]  # Assuming a single output file
threshold = snakemake.params.threshold
mode = snakemake.params.mode

# Rest of the script ...
```

Versatile external python script



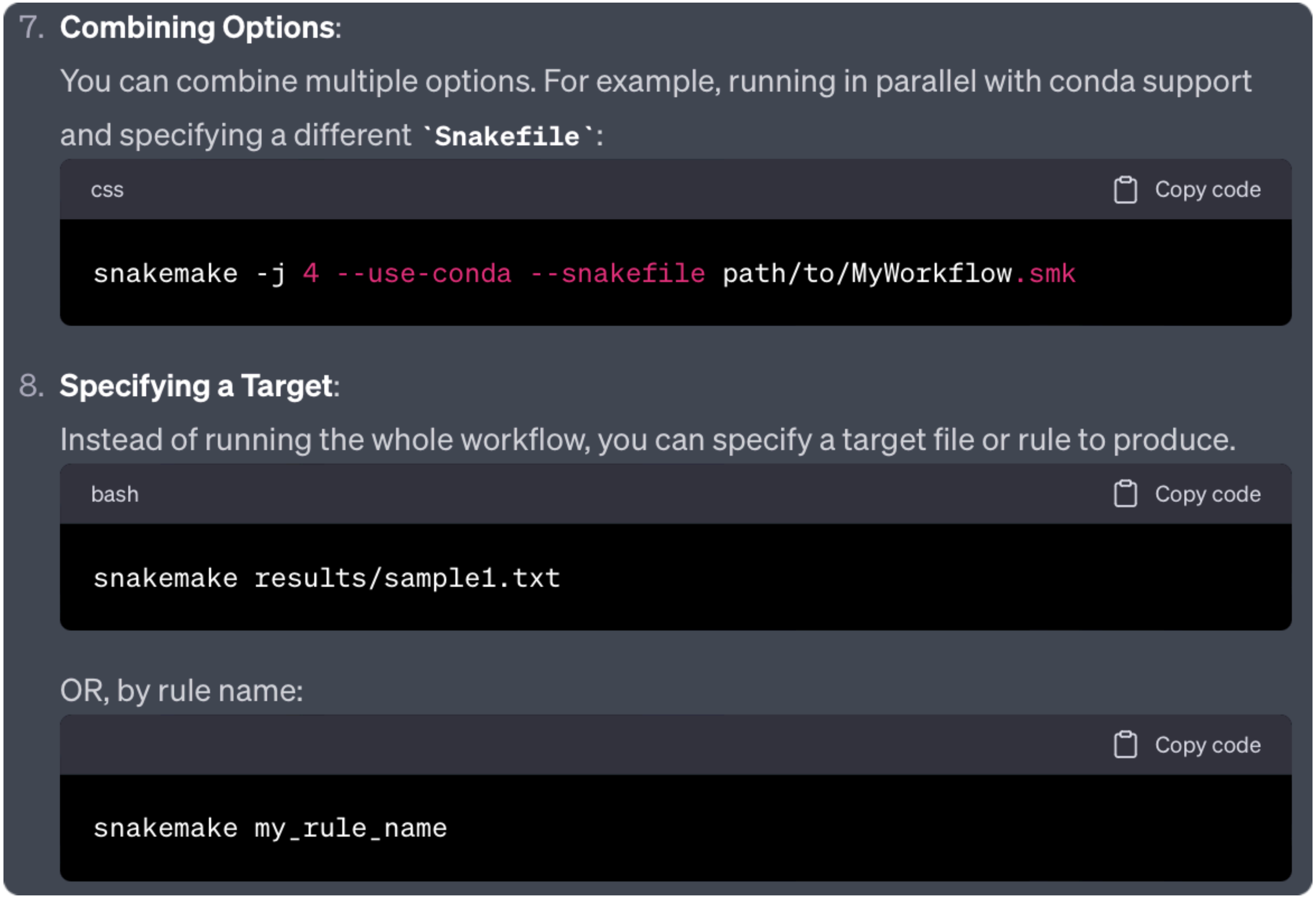
```python
# process_data.py


try:
    # Access inputs, outputs, and parameters from the snakemake object
    input_file = snakemake.input[0]  # Assuming a single input file
    output_file = snakemake.output[0]  # Assuming a single output file
    threshold = snakemake.params.threshold
    mode = snakemake.params.mode
except ModuleNotFoundError:
    # define input and output files manually for testing
    input_file = "raw/sample1.txt"
    output_file = "processed/sample1.processed.txt"
    threshold = 0.5
    mode = "complex"


# Rest of the script ...
```

Different backends



```
jobs: 1000
cluster: "sbatch -o 'slurm/logs/%j.out' -e 'slurm/logs/%j.out' -A snic2022-22-910 \
    -M snowy -p core -n {resources.cpus} -t {resources.time} -J run_snakemake"
default-resources: [cpus=1, time=1440, mem_mb=6400, disk_mb=100000]
nolock: true
printshellcmds: true
latency-wait: 60
use-conda: true
```

Command line example



7. **Combining Options**:
   You can combine multiple options. For example, running in parallel with conda support and specifying a different `Snakefile`:

```css
snakemake -j 4 --use-conda --snakefile path/to/MyWorkflow.smk
```

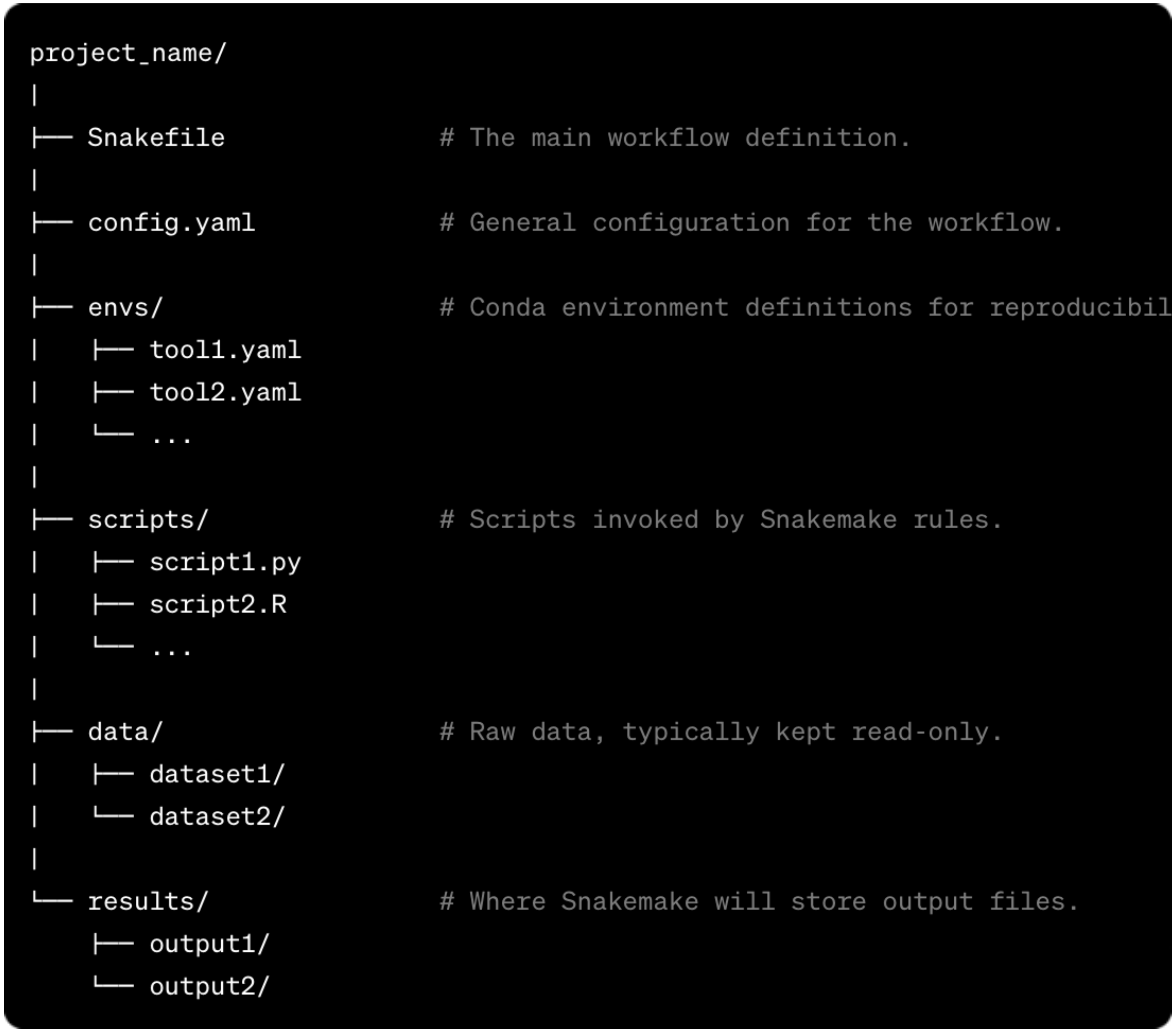
8. **Specifying a Target**:
   Instead of running the whole workflow, you can specify a target file or rule to produce.

```bash
snakemake results/sample1.txt
```

   OR, by rule name:

```
snakemake my_rule_name
```

Folder structure



```
project_name/
|
├── Snakefile                # The main workflow definition.
|
├── config.yaml              # General configuration for the workflow.
|
├── envs/                    # Conda environment definitions for reproducibil
|   ├── tool1.yaml
|   ├── tool2.yaml
|   └── ...
|
├── scripts/                 # Scripts invoked by Snakemake rules.
|   ├── script1.py
|   ├── script2.R
|   └── ...
|
├── data/                    # Raw data, typically kept read-only.
|   ├── dataset1/
|   └── dataset2/
|
└── results/                 # Where Snakemake will store output files.
    ├── output1/
    └── output2/
```

Any

Question