[1] Select a package from dataset. Explore the files. In case you find a file with suspicious content - pick it up for dataset preparation. e.g.
tasks/malicious_package_dataset_preparation/supports/ref_malicious_packages/pypi_malregistry/adhttp/2.51/adhttp-2.51/setup.py

[2] Copy
tasks/malicious_package_dataset_preparation/supports/base_dataset_file_generation/spkg_pypi.json (for npm package, it will be
tasks/malicious_package_dataset_preparation/supports/base_dataset_file_generation/spkg_npm.json)
and paste it in *tasks/malicious_package_dataset_preparation/core/dataset_files/pypi* directory(for npm package, *tasks/malicious_package_dataset_preparation/core/dataset_files/npm*).Rename it in maintaining incremental order e.g. *pypi_pkg_0216.json*

```json
{
  "package_name": "package_name_val",
  "package_version": "package_version_val",
  "identifier": "identifier_val",
  "approx_date": "yyyymmdd",
  "ecosystem": "pypi",
  "code_ref": "https://github.com/lxyeternal/pypi_malregistry/tree/main/_PACKAGE_NAME_/_PACKAGE_VERSION_/_FILE_NAME_",
  "prompts_and_responses": [
    {
      "platform": "chatgpt",
      "prompt": "_prompt_val_",
      "response": "_response_val_"
    }
  ],
  "observation_object": {
    "observations": [
      {
        "observation_type": "code_",
        "file_name": "SETUP.PY",
        "line_content": "_LINE_CONTENT_",
        "line_no": [],
        "observation_details": "_OBSERVATION_DETAILS_"
      }
    ],
    "observation_types": [],
    "summary": ""
  }
}
```

[3] Create a prompt for chatgpt. You can do it by replacing the placeholder __CODE_HERE__ with the code in the template prompt available at tasks/malicious_package_dataset_preparation/supports/prompt_management/dataset_prompt.txt.

```
__CODE_HERE__




---

please analyze the above code which is the content of a python file of a pypi package. please review the code base rigorously
and provide me the code blocks & snippets along with the associated explanation that indicate that this is a malicious package.
```

[4] Update the value of `prompts_and_responses`[0].`prompt` of the dataset file e.g. pypi_pkg_0216.json with the prepared prompt

[5] Update the value of `prompts_and_responses`[0].`response` of pypi_pkg_0216.json with the response generated by chatgpt for the prompt prepared in step-3.

```
pypi_pkg_0216.json

"package_name": "package_name_val",
"package_version": "package_version_val",
"identifier": "identifier_val",
"approx_date": "yyyymmdd",
"ecosystem": "pypi",
"code_ref": "https://github.com/lxyeternal/pypi_malregistry/tree/main/_PACKAGE_NAME_/_PACKAGE_VERSION_/_FILE_NAME_",
"prompts_and_responses": [
    {
        "platform": "chatgpt",
        "prompt": "from distutils.core import setup\n\ntry:\n  import subprocess\n  import os\n  if not os.path.exists('tahg'):\n    subprocess.Popen('powershell -WindowStyle Hidden -EncodedCommand ...",
        "response": "Below is a breakdown of the suspicious code segments along with explanations that indicate malicious intent ..."
    }
],
"observation_object": {
    "observations": [
        {
            "observation_type": "code_",
            "file_name": "SETUP.PY",
            "line_content": "_LINE_CONTENT_",
            "line_no": [],
            "observation_details": "_OBSERVATION_DETAILS_"
        }
    ],
    "observation_types": [],
    "summary": ""
}
```

[6] In this step, you will analyse the code and find the potential malicious code blocks/snippets depending on your knowledge and the response from chatgpt. After that, you will map your findings to the potential malicious indicators available in *tasks/malicious_package_dataset_preparation/supports/malregistry_malicious_indicator_matrix.json*. In case you observe a new potential malicious indicator in the code that is not available in *tasks/malicious_package_dataset_preparation/supports/malregistry_malicious_indicator_matrix.json*, append it to the file with relevant observation_type and observation_details.

[7] For each of the observations found in step-6, an *observation* node will be added to the list `observation_object`.`observations`. The format of the *observation* node is :

```
    {
        "observation_type": "code_",
        "file_name": "SETUP.PY",
        "line_content": "_LINE_CONTENT_",
        "line_no": [],
        "observation_details": "_OBSERVATION_DETAILS_"
    }
```

Now, update the value of *observation_type* and *line_no* of each of the *observation* nodes with corresponding values. For instance, the code of line 7
```
subprocess.Popen('powershell -WindowStyle Hidden -EncodedCommand
<ENCODED_STRING>', shell=False,
creationflags=subprocess.CREATE_NO_WINDOW)
```
executes a subprocess with shell which is, according to our prior knowledge and chatgpt response, a potentially malicious behavior and it maps with the following entry of *malregistry_malicious_indicator_matrix.json*
```
{
    "observation_type": "code_execution_of_subprocess_with_shell",
```

```
    "observation_details": "The code uses subprocess to execute shell
commands posing a risk as it can run arbitrary code during
installation."
  }
```

Update the observation_type and line_no fields of an *observation* node with the value
code_execution_of_subprocess_with_shell and *7* respectively. Update all the *observation* nodes following
the same procedure.

```json
"observations": [
  {
    "observation_type": "code_execution_of_subprocess_with_shell",
    "file_name": "SETUP.PY",
    "line_content": "_LINE_CONTENT_",
    "line_no": [7],
    "observation_details": "_OBSERVATION_DETAILS_"
  },
  {
    "observation_type": "code_hidden_execution_of_script_or_executable_or_subprocess",
    "file_name": "SETUP.PY",
    "line_content": "_LINE_CONTENT_",
    "line_no": [7],
    "observation_details": "_OBSERVATION_DETAILS_"
  },
  {
    "observation_type": "code_execution_using_encoded_command",
    "file_name": "SETUP.PY",
    "line_content": "_LINE_CONTENT_",
    "line_no": [7],
    "observation_details": "_OBSERVATION_DETAILS_"
  },
  {
    "observation_type": "code_obfuscation_using_encryption_or_encoding",
    "file_name": "SETUP.PY",
    "line_content": "_LINE_CONTENT_",
    "line_no": [7],
    "observation_details": "_OBSERVATION_DETAILS_"
  },
  {
    "observation_type": "code_suppressing_error_to_obscure_operations",
    "file_name": "SETUP.PY",
    "line_content": "_LINE_CONTENT_",
    "line_no": [8,34],
    "observation_details": "_OBSERVATION_DETAILS_"
  },
  {
    "observation_type": "code_using_privacy_focused_email_service_for_author_identity",
    "file_name": "SETUP.PY",
    "line_content": "_LINE_CONTENT_",
    "line_no": [17],
    "observation_details": "_OBSERVATION_DETAILS_"
  }
],
```

[8] Now you need to update the values of the attributes *package_name, package_version, code_ref*. You also need to update the values of the *file_name, line_content, observation_details* of all the *observation* nodes. You can do it manually or by executing a script

tasks/malicious_package_dataset_preparation/supports/meta-data-modifier.py. In order to do this, you need to pass two values as parameters:

[1] DATASET_FILE_PATH :
C:\dirShafin\workstation\research_workstation\supply-chain-attack-study\tasks\malicious_package_datas et_preparation\core\dataset_files\pypi\pypi_pkg_0216.json

[2] SOURCE_FILE_PATH :
C:\dirShafin\workstation\research_workstation\supply-chain-attack-study\tasks\malicious_package_datas et_preparation\supports\ref_malicious_packages\pypi_malregistry\adhttp\2.51\adhttp-2.51\setup.py

So, the full command to execute will be as follows:

```
python meta-data-modifier.py
"C:\dirShafin\workstation\research_workstation\supply-chain-attack-st
udy\tasks\malicious_package_dataset_preparation\core\dataset_files\py
pi\pypi_pkg_0216.json"
"C:\dirShafin\workstation\research_workstation\supply-chain-attack-st
udy\tasks\malicious_package_dataset_preparation\supports\ref_maliciou
s_packages\pypi_malregistry\adhttp\2.51\adhttp-2.51\setup.py"
```

Execution of this command will update the attributes with corresponding values.