To classify books into Mechanical (Mech), Computer Science (CSE), or Electronics (ECE) based on keywords in their titles using NLTK operations and DataFrame operations, here is a step-by-step approach you can follow:

Step 1: Prepare Sample Dataset

Create or use a DataFrame with book titles that need classification.

Step 2: Import Necessary Libraries

You'll need `pandas` for DataFrame operations and `nltk` for text processing.

Step 3: Preprocessing Titles with NLTK

Use NLTK operations for keyword extraction and normalization:

- Tokenization
- Stopwords removal
- Stemming or Lemmatization
- POS tagging (optional for enhanced keyword filtering)
- Lowercasing

Step 4: Define Keywords for Each Category

Define sets of keywords for Mech, CSE, and ECE based on common terms in book titles.

Step 5: Classify Each Book Based on Keywords

For each title, after preprocessing, check if keywords appear from any category and assign the classification.

Step 6: Add the classification back to the DataFrame.

Example Code Outline

```
python
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

# Downloading required NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

# Sample book titles
data = {'title': [
    "Introduction to Thermodynamics for Mechanical
Engineers",
    "Data Structures and Algorithms in Python",
    "Digital Electronics and Microprocessor Design",
    "Fluid Mechanics Fundamentals",
    "Operating System Concepts",
    "Analog Electronics and Circuits",
    "Machine Design and Manufacturing",
    "Artificial Intelligence with Python",
    "Control Systems Engineering",
    "Computer Networks and Security"
]}

df = pd.DataFrame(data)

# NLTK preprocessing functions
stop_words = set(stopwords.words('english'))
ps = PorterStemmer()
lemmatizer = WordNetLemmatizer()

def preprocess_title(title):
    # Tokenize
    tokens = word_tokenize(title.lower())
    # Remove stopwords
    filtered_tokens = [word for word in tokens if
word.isalpha() and word not in stop_words]
    # Stemming or Lemmatization (choose only one; here
Lemmatization)
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word
in filtered_tokens]
    return lemmatized_tokens

# Keyword sets for classification
```

```
mech_keywords = {'thermodynamics', 'fluid', 'mechanics',
'machine', 'manufacturing', 'design', 'control',
'engineering'}
cse_keywords = {'data', 'algorithm', 'operating', 'system',
'artificial', 'intelligence', 'computer', 'network',
'security', 'python'}
ece_keywords = {'electronics', 'digital', 'microprocessor',
'analog', 'circuit'}

def classify_title(title):
    tokens = preprocess_title(title)
    # Check keywords intersection
    if any(word in mech_keywords for word in tokens):
        return 'Mechanical (Mech)'
    elif any(word in cse_keywords for word in tokens):
        return 'Computer Science (CSE)'
    elif any(word in ece_keywords for word in tokens):
        return 'Electronics (ECE)'
    else:
        return 'Unclassified'

df['category'] = df['title'].apply(classify_title)

print(df)
```

Explanation of NLTK operations used:

1. Tokenization with `word_tokenize`
2. Lowercasing (normalization)
3. Stopwords removal with `stopwords.words()`
4. Lemmatization with `WordNetLemmatizer`
5. Alphabetic filtering to remove punctuation and numbers

DataFrame operation:

- Apply the classification function to each title using `apply`.

This code creates a DataFrame, preprocesses each title to extract meaningful keywords, and classifies each title based on the keywords for Mechanical, Computer Science, or Electronics engineering books. Let me know if you want a runnable script or further enhancements!