```
import os
import re
import string
import requests
from collections import Counter

# Try optional libraries
try:
    import spacy
except ImportError:
    spacy = None

try:
    from transformers import pipeline
except ImportError:
    pipeline = None


def load_text(source):
    """Load text from local file or URL."""
    if source.startswith(("http://", "https://")):
        return requests.get(source).text
    elif os.path.exists(source):
        with open(source, "r", encoding="utf-8") as f:
            return f.read()
    else:
        raise FileNotFoundError(f"Cannot load: {source}")


def title_basic(text, top_n=3):
    """Basic keyword frequency title."""
    text_clean = text.lower().translate(str.maketrans("", "", string.punctuation))
    words = text_clean.split()
    stopwords = {"the", "and", "is", "in", "to", "of", "a", "for", "on", "it", "this", "that"}
    filtered_words = [w for w in words if w not in stopwords]
    common_words = [w for w, _ in Counter(filtered_words).most_common(top_n)]
    return " ".join(word.capitalize() for word in common_words)


def title_spacy(text):
    """spaCy noun-chunk based title."""
    if not spacy:
        return "[spaCy not installed]"
```

```
    nlp = spacy.load("en_core_web_sm")
    doc = nlp(text)
    phrases = [chunk.text for chunk in doc.noun_chunks]
    top_phrases = Counter(phrases).most_common(2)
    return " - ".join(p[0].title() for p in top_phrases)


def title_ai(text):
    """AI summarization title."""
    if not pipeline:
        return "[Transformers not installed]"
    summarizer = pipeline("summarization", model="t5-small")
    result = summarizer(text, max_length=12, min_length=3, do_sample=False)
    return result[0]['summary_text']

def load_text(source):
    """Load text from local file, directory, or URL."""
    if source.startswith(("http://", "https://")):
        return requests.get(source).text

    elif os.path.isdir(source):
        text_content = []
        for root, _, files in os.walk(source):
            for file in files:
                if file.lower().endswith((".txt", ".md", ".py", ".html", ".csv")):
                    try:
                        with open(os.path.join(root, file), "r", encoding="utf-8") as f:
                            text_content.append(f.read())
                    except Exception as e:
                        print(f"Skipping {file}: {e}")
        return "\n".join(text_content)

    elif os.path.exists(source):
        with open(source, "r", encoding="utf-8") as f:
            return f.read()

    else:
        raise FileNotFoundError(f"Cannot load: {source}")

def make_code_friendly(title):
    """Convert title to code-safe string."""
    return re.sub(r'[^A-Za-z0-9_]', '_', title)
```

```
if __name__ == "__main__":
    print("=== Document Title Generator ===")
    source = input("Enter file path or URL: ").strip()
    text = load_text(source)

    print("\nGenerating titles...")

    basic_title = title_basic(text)
    spacy_title = title_spacy(text)
    ai_title = title_ai(text)

    print("\n--- RESULTS ---")
    print(f"1. Basic : {basic_title}")
    print(f"2. spaCy : {spacy_title}")
    print(f"3. AI    : {ai_title}")

    print("\nCode-Friendly Versions:")
    print(f"1. {make_code_friendly(basic_title)}")
    print(f"2. {make_code_friendly(spacy_title)}")
    print(f"3. {make_code_friendly(ai_title)}")

    choice = input("\nWhich title to copy to clipboard? (1-3, skip to cancel): ").strip()
    if choice in {"1", "2", "3"}:
        chosen_title = [basic_title, spacy_title, ai_title][int(choice) - 1]
        code_version = make_code_friendly(chosen_title)
        os.system(f'echo "{code_version}" | termux-clipboard-set')
        print(f"(Copied '{code_version}' to clipboard)")
```