```
from bs4 import BeautifulSoup
from pycountry import countries
import re
from scrapy.spiders import Spider
from scrapy.selector import Selector

from extractors.direct_downloads import extract_with_request
from utils.helpers import clean_str


class WikiManufacturerSpider(Spider):
    name = "WikiManufacturerSpider"

    def __init__(self, *args, **kwargs):
        """
        Extracts supplemental aircraft input data from Wikipedia's list of aircraft manufacturers

        :param args: see scrapy.Spider for details
        :param kwargs: see scrapy.Spider for details
        """

        super(WikiManufacturerSpider, self).__init__(*args, **kwargs)

        # assign remote sources to scrape supplemental input data from
        url = "https://en.wikipedia.org/wiki/List_of_aircraft_manufacturers"
        soup = BeautifulSoup(extract_with_request(url, ".html", save=False), "html.parser")
        self.start_urls = [url + ":_" + tag.text for tag in soup.find("dd").find_all()]

    def parse(self, response, **kwargs):
        """
        Handles callback behavior for extracting supplemental aircraft input data from remote sources

        :param response: http response to extract supplemental aircraft input data from
        :param kwargs: see scrapy.Spider for details
        :return: list of transformed supplemental aircraft input data extracted from remote sources
        """

        # assign storage for extracted supplemental input data
        manufacturers = []

        # loop through unordered lists
        for ul in response.css(".mw-parser-output").xpath("./ul").getall():
            # loop through list items of unordered list in focus
            for li in Selector(text=ul).xpath(".//li").getall():
                # assign text of list item in focus
                text = Selector(text=li).xpath(".//text()").getall()
                # drop irrelevant portion of text and assign list of candidate manufacture names
                text = ''.join([x for x in text]).split(" – ")[0].split(",")

                # initialize storage for candidate manufacture names in text
                candidates = [text[0]]
                # remove date-related elements of candidate manufacture names and format the remaining
                candidates.extend([clean_str(x) for x in text[1:] if not any(char.isdigit() for char in x)])

                i = 0
                # loop through candidate manufacture names
                while i < len(candidates):
                    # assign candidate manufacture name in focus
                    candidate = candidates[i]

                    if " (" in candidate:
                        # assign parenthetical text of manufacture name in focus
                        paren = re.search(r"\(([^)]+)", candidate).group(1)

                        if not paren.isnumeric() and not self.check_is_country(paren):
                            # handle when manufacture name in focus is not date-related or a country
                            candidates.append(paren)

                        # drop parenthetical text of manufacture name in focus
                        candidate = candidate.replace("(" + paren + ")", "")

                        if self.check_is_country(candidate):
                            # handle when manufacture name in focus is a country not enclosed in parentheses
                            candidate = ""

                        candidates[i] = clean_str(candidate)
                        i += 1

                    # remove invalidated names from candidate manufacture names
                    candidates = [x for x in candidates if x is not None]

                    # update extracted supplemental input data
                    manufacturers.append({
                        "name": candidates[0],
                        "aliases": candidates[1:]
                    })

        return manufacturers

    @staticmethod
    def check_is_country(string):
        """
        Determines whether a string represents the name of a country

        :param (str) string: country name to check
        :return: 'True' if 'string' argument is a country name
        """

        if string == "US" or string == "USA":
            # handle corner-cases where the United States of America is abbreviated
            return True

        if countries.get(name=string) is not None:
            return True
        else:
            return False
```