gdown\download.py

```
from __future__ import print_function

import json
import os
import os.path as osp
import re
import shutil
import sys
import tempfile
import textwrap
import time

import requests
import six
import tqdm

from .parse_url import parse_url

CHUNK_SIZE = 512 * 1024  # 512KB
home = osp.expanduser("~")


if hasattr(textwrap, "indent"):
    indent_func = textwrap.indent
else:

    def indent_func(text, prefix):
        def prefixed_lines():
            for line in text.splitlines(True):
                yield (prefix + line if line.strip() else line)

        return "".join(prefixed_lines())


def get_url_from_gdrive_confirmation(contents):
    url = ""
    for line in contents.splitlines():
        m = re.search(r'href="(\/uc\?export=download[^"]+)', lin
e)
        if m:
            url = "https://docs.google.com" + m.groups()[0]
            url = url.replace("&amp;", "&")
            return url
        m = re.search("confirm=([^;&]+)", line)
        if m:
            confirm = m.groups()[0]
            url = re.sub(
                r"confirm=([^;&]+)", r"confirm={}".format(confirm),
url
            )
            return url
        m = re.search('"downloadUrl":"([^"]+)', line)
        if m:
            url = m.groups()[0]
            url = url.replace("\\u003d", "=")
            url = url.replace("\\u0026", "&")
            return url
        m = re.search('<p class="uc-error-subcaption">(.*)</p>'
, line)
        if m:
            error = m.groups()[0]
            raise RuntimeError(error)


def download(
    url, output=None, quiet=False, proxy=None, speed=None,
use_cookies=True
```

gdown_mine\download.py

```
from __future__ import print_function

import json
import os
import os.path as osp
import re
import shutil
import sys
import tempfile
import textwrap
import time

import requests
import six
import tqdm

from .parse_url import parse_url

CHUNK_SIZE = 512 * 1024  # 512KB
home = osp.expanduser("~")


if hasattr(textwrap, "indent"):
    indent_func = textwrap.indent
else:

    def indent_func(text, prefix):
        def prefixed_lines():
            for line in text.splitlines(True):
                yield (prefix + line if line.strip() else line)

        return "".join(prefixed_lines())


def get_url_from_gdrive_confirmation(contents):
    url = ""
    for line in contents.splitlines():
        m = re.search(r'href="(\/uc\?export=download[^"]+)', lin
e)
        if m:
            url = "https://docs.google.com" + m.groups()[0]
            url = url.replace("&amp;", "&")
            return url
        m = re.search("confirm=([^;&]+)", line)
        if m:
            confirm = m.groups()[0]
            url = re.sub(
                r"confirm=([^;&]+)", r"confirm={}".format(confirm), u
rl
            )
            return url
        m = re.search('"downloadUrl":"([^"]+)', line)
        if m:
            url = m.groups()[0]
            url = url.replace("\\u003d", "=")
            url = url.replace("\\u0026", "&")
            return url
        m = re.search('<p class="uc-error-subcaption">(.*)</p>'
, line)
        if m:
            error = m.groups()[0]
            raise RuntimeError(error)


def download(
    url, output=None, quiet=False, proxy=None, speed=None,
use_cookies=True
```

gdown\download.py	gdown_mine\download.py
):	):
"""Download file from URL.	"""Download file from URL.
Parameters	Parameters
----------	----------
url: str	url: str
URL. Google Drive URL is also supported.	URL. Google Drive URL is also supported.
output: str, optional	output: str, optional
Output filename. Default is basename of URL.	Output filename. Default is basename of URL.
quiet: bool	quiet: bool
Suppress terminal output. Default is False.	Suppress terminal output. Default is False.
proxy: str	proxy: str
Proxy.	Proxy.
speed: float	speed: float
Download byte size per second (e.g., 256KB/s = 256 * 1024).	Download byte size per second (e.g., 256KB/s = 256 * 1024).
use_cookies: bool	use_cookies: bool
Flag to use cookies. Default is True.	Flag to use cookies. Default is True.
Returns	Returns
-------	-------
output: str	output: str
Output filename.	Output filename.
"""	"""
url_origin = url	url_origin = url
sess = requests.session()	sess = requests.session()
# Load cookies	# Load cookies
cache_dir = osp.join(home, ".cache", "gdown")	cache_dir = osp.join(home, ".cache", "gdown")
if not osp.exists(cache_dir):	if not osp.exists(cache_dir):
os.makedirs(cache_dir)	os.makedirs(cache_dir)
cookies_file = osp.join(cache_dir, "cookies.json")	cookies_file = osp.join(cache_dir, "cookies.json")
if osp.exists(cookies_file) and use_cookies:	if osp.exists(cookies_file) and use_cookies:
with open(cookies_file) as f:	with open(cookies_file) as f:
cookies = json.load(f)	cookies = json.load(f)
for k, v in cookies:	for k, v in cookies:
sess.cookies[k] = v	sess.cookies[k] = v
if proxy is not None:	if proxy is not None:
sess.proxies = {"http": proxy, "https": proxy}	sess.proxies = {"http": proxy, "https": proxy}
print("Using proxy:", proxy, file=sys.stderr)	print("Using proxy:", proxy, file=sys.stderr)
file_id, is_download_link = parse_url(url)	file_id, is_download_link = parse_url(url)
headers = {	headers = {
"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36" # NOQA	"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36" # NOQA
}	}
while True:	while True:
try:	try:
res = sess.get(url, headers=headers, stream=True)	res = sess.get(url, headers=headers, stream=True)
except requests.exceptions.ProxyError as e:	except requests.exceptions.ProxyError as e:
print("An error has occurred using proxy:", proxy, file=sys.stderr)	print("An error has occurred using proxy:", proxy, file=sys.stderr)
print(e, file=sys.stderr)	print(e, file=sys.stderr)
return	return
# Save cookies	# Save cookies
with open(cookies_file, "w") as f:	with open(cookies_file, "w") as f:
cookies = [	cookies = [
k, v)	k, v)
for k, v in sess.cookies.items()	for k, v in sess.cookies.items()
if not k.startswith("download_warning_")	if not k.startswith("download_warning_")
]	]

gdown\download.py	gdown_mine\download.py
json.dump(cookies, f, indent=2)	json.dump(cookies, f, indent=2)
if "Content-Disposition" in res.headers:	if "Content-Disposition" in res.headers:
# This is the file	# This is the file
break	break
if not (file_id and is_download_link):	if not (file_id and is_download_link):
break	break
# Need to redirect with confirmation	# Need to redirect with confirmation
try:	try:
url = get_url_from_gdrive_confirmation(res.text)	url = get_url_from_gdrive_confirmation(res.text)
except RuntimeError as e:	except RuntimeError as e:
print("Access denied with the following error:')	print("Access denied with the following error:')
error = "\n".join(textwrap.wrap(str(e)))	error = "\n".join(textwrap.wrap(str(e)))
error = indent_func(error, "\t")	error = indent_func(error, "\t")
print("\n", error, "\n", file=sys.stderr)	print("\n", error, "\n", file=sys.stderr)
print(	print(
"You may still be able to access the file from the browser:",	"You may still be able to access the file from the b rowser:",
file=sys.stderr,	file=sys.stderr,
)	)
print("\n\t", url_origin, "\n", file=sys.stderr)	print("\n\t", url_origin, "\n", file=sys.stderr)
return	return
if url is None:	if url is None:
print("Permission denied:", url_origin, file=sys.stderr)	print("Permission denied:", url_origin, file=sys.stderr)
print(	print(
"Maybe you need to change permission over "	"Maybe you need to change permission over "
"'Anyone with the link'?",	"'Anyone with the link'?",
file=sys.stderr,	file=sys.stderr,
)	)
return	return
if file_id and is_download_link:	if file_id and is_download_link:
m = re.search('filename="(.*)"', res.headers["Content-Di sposition"])	m = re.search('filename="(.*)"', res.headers["Content-Di sposition"])
filename_from_url = m.groups()[0]	filename_from_url = m.groups()[0]
else:	else:
filename_from_url = osp.basename(url)	filename_from_url = osp.basename(url)
if output is None:	if output is None:
output = filename_from_url	output = filename_from_url
output_is_path = isinstance(output, six.string_types)	output_is_path = isinstance(output, six.string_types)
if output_is_path and output.endswith(osp.sep):	if output_is_path and output.endswith(osp.sep):
if not osp.exists(output):	if not osp.exists(output):
os.makedirs(output)	os.makedirs(output)
output = osp.join(output, filename_from_url)	output = osp.join(output, filename_from_url)
if not quiet:	if not quiet:
print("Downloading...", file=sys.stderr)	print("必要なファイルをダウンロードしています",file= sys.stderr)
print("From:", url_origin, file=sys.stderr)	
print(	
"To:",	
osp.abspath(output) if output_is_path else output,	
file=sys.stderr,	
)	
if output_is_path:	if output_is_path:
tmp_file = tempfile.mktemp(	tmp_file = tempfile.mktemp(
suffix=tempfile.template,	suffix=tempfile.template,
prefix=osp.basename(output),	prefix=osp.basename(output),
dir=osp.dirname(output),	dir=osp.dirname(output),
)	)

gdown\download.py	gdown_mine\download.py
f = open(tmp_file, "wb")	f = open(tmp_file, "wb")
else:	else:
tmp_file = None	tmp_file = None
f = output	f = output
try:	try:
total = res.headers.get("Content-Length")	total = res.headers.get("Content-Length")
if total is not None:	if total is not None:
total = int(total)	total = int(total)
if not quiet:	if not quiet:
pbar = tqdm.tqdm(total=total, unit="B", unit_scale=True)	pbar = tqdm.tqdm(total=total, unit="B", unit_scale=True)
t_start = time.time()	t_start = time.time()
for chunk in res.iter_content(chunk_size=CHUNK_SIZE):	for chunk in res.iter_content(chunk_size=CHUNK_SIZE):
f.write(chunk)	f.write(chunk)
if not quiet:	if not quiet:
pbar.update(len(chunk))	pbar.update(len(chunk))
if speed is not None:	if speed is not None:
elapsed_time_expected = 1.0 * pbar.n / speed	elapsed_time_expected = 1.0 * pbar.n / speed
elapsed_time = time.time() - t_start	elapsed_time = time.time() - t_start
if elapsed_time < elapsed_time_expected:	if elapsed_time < elapsed_time_expected:
time.sleep(elapsed_time_expected - elapsed_time)	time.sleep(elapsed_time_expected - elapsed_time)
if not quiet:	if not quiet:
pbar.close()	pbar.close()
if tmp_file:	if tmp_file:
f.close()	f.close()
shutil.move(tmp_file, output)	shutil.move(tmp_file, output)
except IOError as e:	except IOError as e:
print(e, file=sys.stderr)	print(e, file=sys.stderr)
return	return
finally:	finally:
sess.close()	sess.close()
try:	try:
if tmp_file:	if tmp_file:
os.remove(tmp_file)	os.remove(tmp_file)
except OSError:	except OSError:
pass	pass
return output	return output