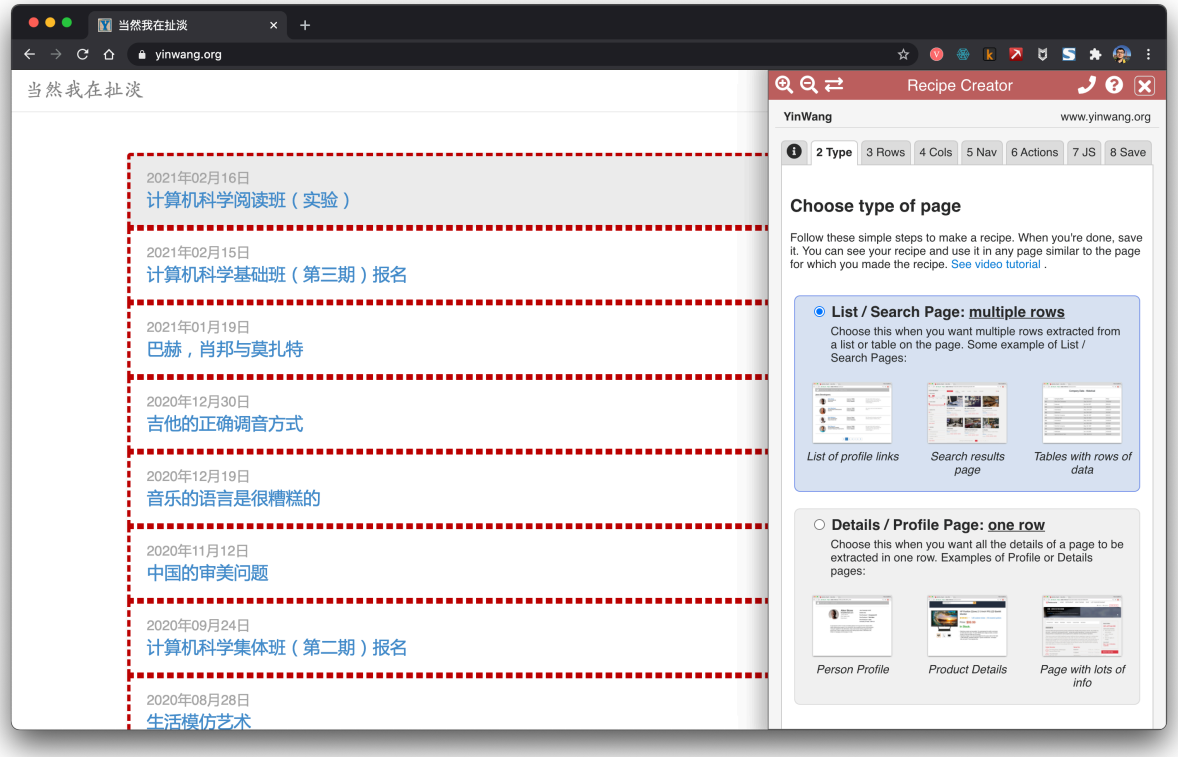
वेबसाइट की सामग्री को स्क्रैप करना

ऐसे बहुत से उपकरण उपलब्ध हैं जो वेबसाइटों की सामग्री को स्क्रैप कर सकते हैं। हालांकि, अगर हम उनका उपयोग करते हैं, तो हम पीछे के प्रक्रिया को बेहतर ढंग से समझ नहीं पाएंगे। अगर काम में जटिल या विशेष वेबसाइटों का सामना होता है, तो उनका उपयोग करने से हमें वांछित परिणाम नहीं मिल सकते हैं। हमें पहियों को फिर से बनाने की आवश्यकता है, ताकि हम उन्हें बेहतर ढंग से सीख सकें और उनका बेहतर उपयोग कर सकें।

आइए पहले से मौजूद कुछ टूल्स पर भी एक नज़र डालें।

डेटा माइनर



□□□□□□ 1: खनिक

Data Miner □□□□□□ पर एक बहुत ही सुविधाजनक एक्सटेंशन है। इसका उपयोग लिंक और सामग्री को आसानी से स्क्रैप करने के लिए किया जा सकता है।

□□□□□□□

getbook एक बहुत ही सुविधाजनक ई-बुक निर्माण उपकरण है।

```
pip install getbook
```

book.json:

```
{
  "uid": "book",
  "title": "Hello World",
  "author": "Armin",
  "chapters": [
    "http://lucumr.pocoo.org/2018/7/13/python/",
    "http://lucumr.pocoo.org/2017/6/5/diversity-in-technology",
  ]
}
```

```
getbook -f ./book.json --mobi
```

इस तरह से कुछ लिंक्स को आसानी से ई-बुक में बदल दिया गया। `Data Miner` और `getbook` का उपयोग करके, एक लिंक्स को क्रॉल करता है और दूसरा उन्हें ई-बुक में बदल देता है, जिससे ई-बुक बनाना बहुत आसान हो जाता है।

फ़ेमन भौतिकी व्याख्यान

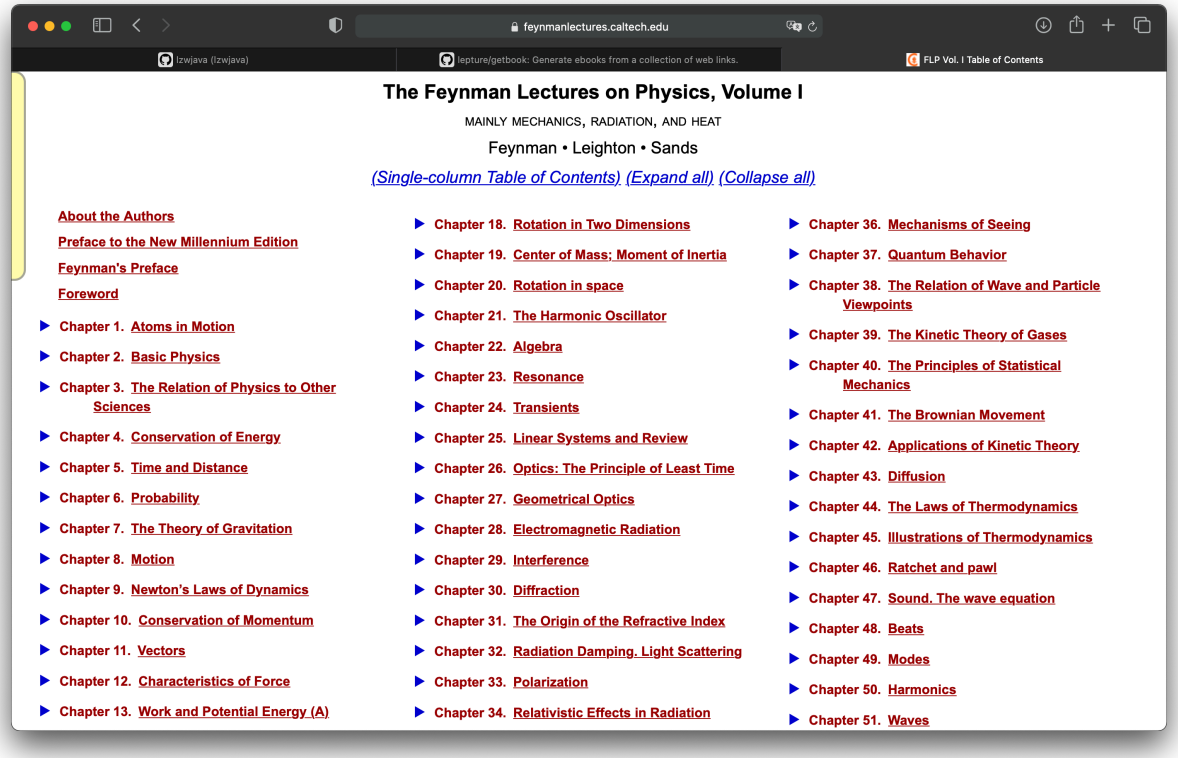
□□ `mathjax` □□□ `html` □□□□□□□□□□□□□□□□□□□□□□□□□□ □□

> □□□□.□□□□□□ — □□□□ प्रोटोकॉल क्लाइंट
>
> स्रोत कोड: □□□/□□□□/□□□□□□.□□
>
> यह मॉड्यूल क्लासेस को परिभाषित करता है जो □□□□ और □□□□□ प्रोटोकॉल के क्लाइंट साइड को लागू करते हैं। यह आमतौर पर सीधे उपयोग नहीं किया जाता है — □□□□□□.□□□□□□□ मॉड्यूल □□□□ और □□□□□ का उपयोग करने वाले □□□□ को हैंडल करने के लिए इसका उपयोग करता है।
>
> यह भी देखें: उच्च-स्तरीय □□□□ क्लाइंट इंटरफेस के लिए □□□□□□□□ पैकेज की सिफारिश की जाती है।

देखा जाए तो `requests` एक उच्च-स्तरीय इंटरफ़ेस है।

```
import requests

def main():
    r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
    print(r.status_code)
```


feynmanlectures.caltech.edu
lzwjava (lzwjava)
lepture/getbook: Generate ebooks from a collection of web links.
FLP Vol. I Table of Contents
The Feynman Lectures on Physics, Volume I
MAINLY MECHANICS, RADIATION, AND HEAT
Feynman • Leighton • Sands
(Single-column Table of Contents) (Expand all) (Collapse all)
About the Authors
Preface to the New Millennium Edition
Feynman's Preface
Foreword
Chapter 1. Atoms in Motion
Chapter 2. Basic Physics
Chapter 3. The Relation of Physics to Other Sciences
Chapter 4. Conservation of Energy
Chapter 5. Time and Distance
Chapter 6. Probability
Chapter 7. The Theory of Gravitation
Chapter 8. Motion
Chapter 9. Newton's Laws of Dynamics
Chapter 10. Conservation of Momentum
Chapter 11. Vectors
Chapter 12. Characteristics of Force
Chapter 13. Work and Potential Energy (A)
Chapter 18. Rotation in Two Dimensions
Chapter 19. Center of Mass; Moment of Inertia
Chapter 20. Rotation in space
Chapter 21. The Harmonic Oscillator
Chapter 22. Algebra
Chapter 23. Resonance
Chapter 24. Transients
Chapter 25. Linear Systems and Review
Chapter 26. Optics: The Principle of Least Time
Chapter 27. Geometrical Optics
Chapter 28. Electromagnetic Radiation
Chapter 29. Interference
Chapter 30. Diffraction
Chapter 31. The Origin of the Refractive Index
Chapter 32. Radiation Damping. Light Scattering
Chapter 33. Polarization
Chapter 34. Relativistic Effects in Radiation
Chapter 36. Mechanisms of Seeing
Chapter 37. Quantum Behavior
Chapter 38. The Relation of Wave and Particle Viewpoints
Chapter 39. The Kinetic Theory of Gases
Chapter 40. The Principles of Statistical Mechanics
Chapter 41. The Brownian Movement
Chapter 42. Applications of Kinetic Theory
Chapter 43. Diffusion
Chapter 44. The Laws of Thermodynamics
Chapter 45. Illustrations of Thermodynamics
Chapter 46. Ratchet and pawl
Chapter 47. Sound. The wave equation
Chapter 48. Beats
Chapter 49. Modes
Chapter 50. Harmonics
Chapter 51. Waves

□□□□□□ 2: □□

मुख्य()

```shell
401

import requests

def main():
    r = requests.get('https://github.com')
    print(r.status_code)
    print(r.text)
```

मुख्य()

```html
200
<html>
  ...
</html>
```

मैंने कोशिश की, और यह साबित हो गया कि `requests` का इंटरफ़ेस काम कर रहा है।

```
     <div class="toc-chapter" id="C03">
        <span class="triangle">
         â ¶
        </span>
        <a class="chapterlink" href="javascript:Goto(1,3)">
         <span class="tag">
          Chapter 3.
         </span>

        </a>
        <div class="sections">
         <a href="javascript:Goto(1,3,1)">
          <span class="tag">
           3-1
          </span>
```

```
</a>
<a href="javascript:Goto(1,3,2)">
 <span class="tag">
  3-2
 </span>

</a>
<a href="javascript:Goto(1,3,3)">
 <span class="tag">
  3-3
 </span>

</a>
<a href="javascript:Goto(1,3,4)">
 <span class="tag">
  3-4
 </span>

</a>
<a href="javascript:Goto(1,3,5)">
 <span class="tag">
  3-5
 </span>

</a>
<a href="javascript:Goto(1,3,6)">
 <span class="tag">
  3-6
 </span>

</a>
<a href="javascript:Goto(1,3,7)">
 <span class="tag">
  3-7
 </span>
```

```
                    ?
          </a>
         </div>
        </div>
```

यह डायरेक्टरी पेज में तीसरे अध्याय का `html` कोड है। यहां से हर अध्याय का लिंक निकालना चाहते हैं। `<a href="javascript:Goto(1,3,7)">` से देखा जा सकता है कि यह एक `javascript` का हाइपरलिंक है।

`https://www.feynmanlectures.caltech.edu/I_03.html`

फिर यह पता चला कि प्रत्येक अध्याय का पथ बहुत ही नियमित है। `I_03.html` पहले खंड के तीसरे अध्याय को दर्शाता है।

```
import requests
from bs4 import BeautifulSoup
from multiprocessing import Process

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {url}')
    r = requests.get(url)
    if r.status_code != 200:
        raise Exception(r.status_code)
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1))
        p.start()
        p.join()

main()
```

आगे बढ़ते हुए, हम कोड को स्क्रैप करने के बारे में लिखेंगे। यहाँ `Process` का उपयोग किया गया है।

```
    raise RuntimeError('''
RuntimeError:
```

fork idiom :

```
if __name__ == '__main__':
    freeze_support()
    ...

    "freeze_support()"
```

```python
def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1,))
        p.start()
    p.join()
```

(यह कोड □□□□□□ में एक मल्टीप्रोसेसिंग उदाहरण है जहां 52 प्रक्रियाएं शुरू की जाती हैं, जिनमें से प्रत्येक `scrape` फ़ंक्शन को एक अलग आर्ग्युमेंट के साथ कॉल करती है। अंत में, सभी प्रक्रियाओं के समाप्त होने की प्रतीक्षा की जाती है।)

```
if __name__ == "__main__":
    main()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('  : ', stop - start)
```

यदि □□□□ == “□□□□”:
□□□□()

```shell
scraping https://www.feynmanlectures.caltech.edu/I_01.html
scraping https://www.feynmanlectures.caltech.edu/I_04.html
...
scraping https://www.feynmanlectures.caltech.edu/I_51.html
scraping https://www.feynmanlectures.caltech.edu/I_52.html
  :  9.144841699
```

the most information in the fewest words? I believe it is the *atomic hypothesis* (or the atomic *fact* , or whatever you wish to call it) that *all things are made of atomsâlittle particles that move around in perpetual motion, attracting each other when they are a little distance apart, but repelling upon being squeezed into one another* . In that one sentence, you will see, there is an *enormous* amount of information about the world, if just a little imagination and thinking are applied.

Figure 1â1

To illustrate the power of the atomic idea, suppose that we have a drop of water a quarter of an inch on the side. If we look at it very closely we see nothing but waterâsmooth, continuous water. Even if we magnify it with the best optical microscope availableâroughly two thousand timesâthen the water drop will be roughly forty feet across, about as big as a large room, and if we looked rather closely, we would *still* see relatively smooth waterâbut here and there small football-shaped things swimming back and forth. Very interesting. These are paramecia. You may stop at this

□□□□□□□ 3: □□□

```
<div class="figure" id="Ch1-F1">
        <img src="img/FLP_I/f01-01/f01-01_tc_big.svgz">
        <div class="caption empty">
         <span class="tag">
             1â 1
         </span>
        </div>
</div>
```

```
import requests
from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {url}')
    r = requests.get(url)
    if r.status_code != 200:
        raise Exception(r.status_code)
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('  : ', stop - start)
```

□□□□ □□□□ == “□□□□”:

□□□□()

```python
    imgs = soup.find_all('img')
    for img in imgs:
        print(img)
```

यह कोड □□□□□□□ में लिखा गया है और यह एक वेब पेज से सभी `img` टैग्स को ढूंढता है और उन्हें प्रिंट करता है। `soup.find_all('img')` का उपयोग करके सभी `img` टैग्स को एक लिस्ट में एकत्रित किया जाता है, और फिर `for` लूप के माध्यम से प्रत्येक `img` टैग को प्रिंट किया जाता है।

```
    https://www.feynmanlectures.caltech.edu/I_01.html
<img id="TwitLink" src=""/>
<img id="FBLink" src=""/>
<img id="MailLink" src=""/>
<img id="MobileLink" src=""/>
<img id="DarkModeLink" src=""/>
<img id="DesktopLink" src=""/>
<img src="img/camera.svg"/>
<img src="img/FLP_I/f01-00/f01-00.jpg"/>
<img data-src="img/FLP_I/f01-01/f01-01_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-02/f01-02_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-03/f01-03_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-04/f01-04_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-05/f01-05_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-06/f01-06_tc_big.svgz"/>
<img class="first" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_a.svgz"/>
<img class="last" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_b.svgz"/>
<img data-src="img/FLP_I/f01-08/f01-08_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-09/f01-09_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-10/f01-10_tc_big.svgz"/>
```

□□□□□://□□□.□□□□□□□□□□□□□□□□□.□□□□□□□□.□□□/□□□/□□□_□/□01-01/□01-01_□□_□□□.□□□□

आपके पास इस संसाधन तक पहुंचने की अनुमति नहीं है।

□□□□□□□/2.4.38 (□□□□□□□) सर्वर □□□.□□□□□□□□□□□□□□□□□.□□□□□□□□.□□□ पर पोर्ट 443 पर चल रहा है

```shell
% pip install selenium
Collecting selenium
  Using cached selenium-3.141.0-py2.py3-none-any.whl (904 kB)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/site-packages (from selenium) (1.24.1
```

```
Installing collected packages: selenium
Successfully installed selenium-3.141.0

export CHROME_DRIVER_HOME=$HOME/dev-env/chromedriver
export PATH="${PATH}:${CHROME_DRIVER_HOME}"
```

(यह कोड ब्लॉक है, इसे अनुवादित नहीं किया जाना चाहिए।)

```
% chromedriver -h
   : chromedriver [   ]
```

विकल्प –□□□□=□□□□ सुनने के लिए पोर्ट –□□□-□□□□=□□□□ □□□ सर्वर पोर्ट –□□□-□□□□=□□□□ सर्वर लॉग को □□□□□□ के बजाय फ़ाइल में लिखें, लॉग स्तर को □□□□ तक बढ़ाएं –□□□-□□□□□=□□□□□ लॉग स्तर सेट करें: □□□, □□□□□, □□□□, □□□□□□□, □□□□□□, □□□ –□□□□□□□ विस्तृत लॉग (–□□□-□□□□□=□□□ के समान) –□□□□□□ कुछ भी लॉग न करें (–□□□-□□□□□=□□□ के समान) –□□□□□□-□□□ लॉग फ़ाइल को फिर से लिखने के बजाय जोड़ें –□□□□□□□□□□ (प्रायोगिक) विस्तृत लॉग और लंबी स्ट्रिंग्स को छोटा न करें ताकि लॉग को दोबारा चलाया जा सके। –□□□□□□□ संस्करण संख्या प्रिंट करें और बाहर निकलें –□□□-□□□□ कमांड के लिए आधार □□□ पथ उपसर्ग, उदाहरण के लिए □□/□□□ –□□□□□□□□-□□□□□□□□□ लॉग में पठनीय टाइमस्टैम्प जोड़ें –□□□□□□-□□□□□□-□□□□ ब्राउज़र से लॉग दिखाएं (अन्य लॉगिंग विकल्पों को ओवरराइड करता है) –□□□□□□□-□□□ दूरस्थ □□ पतों की अनुमति सूची जो □□□□□□□□□□□□ से कनेक्ट करने की अनुमति देते हैं, अल्पविराम से अलग किए गए

```python

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located
```

यह कोड □□□□□□□□ वेब ड्राइवर का उपयोग करके वेब ऑटोमेशन के लिए आवश्यक मॉड्यूल्स को इम्पोर्ट करता है। इसमें `webdriver` वेब ब्राउज़र को नियंत्रित करने के लिए, `By` वेब एलिमेंट्स को ढूंढने के लिए, `Keys` कीबोर्ड इनपुट सिम्युलेट करने के लिए, `WebDriverWait` विशिष्ट शर्तों की प्रतीक्षा करने के लिए, और `presence_of_element_located` यह सुनिश्चित करने के लिए कि एक एलिमेंट पेज पर मौजूद है, शामिल हैं।

```
with webdriver.Chrome() as driver:
    wait = WebDriverWait(driver, 10)
    driver.get("https://google.com/ncr")
    driver.find_element(By.NAME, "q").send_keys("cheese" + Keys.RETURN)
    first_result = wait.until(presence_of_element_located((By.CSS_SELECTOR, "h3>div")))
    print(first_result.get_attribute("textContent"))
```

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located
import urllib
```

यह कोड □□□□□□□□□ और □□□□□□□ लाइब्रेरी का उपयोग करता है। □□□□□□□□□ एक वेब ब्राउज़र ऑटोमेशन टूल है, जिसका उपयोग वेब एप्लिकेशन को टेस्ट करने और वेब स्क्रैपिंग के लिए किया जाता है। □□□□□□□ लाइब्रेरी का उपयोग □□□ को खोलने और डेटा को पढ़ने के लिए किया जाता है।

```
def main():
    driver = webdriver.Chrome()
    wait = WebDriverWait(driver, 10)
    driver.get("https://www.feynmanlectures.caltech.edu/I_01.html")
    elements = driver.find_elements(By.TAG_NAME, "img")
    # print(dir(elements[0]))
    print(driver.page_source)
    i = 0
    for element in elements:
        # src = element.get_attribute('src')
        element.screenshot(f'images/{i}.png')
        i +=1
    driver.close()
main()
```

```
from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit
from pathlib import Path
from selenium import webdriver
from selenium.webdriver.common.by import By

def img_path(chapter):
    return f'./chapters/{chapter}/img'
```

```
def img_name(url):
    splits = url.split('/')
    last = splits[len(splits) - 1]
    parts = last.split('.')
    name = parts[0]
    return name
```

यह फ़ंक्शन `img_name` एक □□□ लेता है और उस □□□ से छवि का नाम निकालता है। यह □□□ को '/' से विभाजित करता है, अंतिम भाग को लेता है, और फिर उसे '.' से विभाजित करके छवि का नाम प्राप्त करता है। अंत में, यह छवि का नाम वापस करता है।

```
def download_images(driver: webdriver.Chrome, chapter):
    path = img_path(chapter)
    Path(path).mkdir(parents=True, exist_ok=True)

    elements = driver.find_elements(By.TAG_NAME, "img")
    for element in elements:
        src = element.get_attribute('src')
        name = img_name(src)
        element.screenshot(f'{path}/{name}.png')
```

□□□□_□□□□□ = '□□□□□□□□/5.0 (□□□□□□□□□□; □□□□□ □□□ □□ □ 10_15_6) □□□□□□□□□□□□/605.1.15 (□□□□□□, □□□□ □□□□□) □□□□□□□□/14.0.3 □□□□□□□/605.1.15'

```
def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    driver = webdriver.Chrome()
    driver.get(url)
    page_source = driver.page_source
    Path(f'./chapters/{chapter_str}').mkdir(parents=True, exist_ok=True)
    print(f'scraping {url}')

    download_images(driver, chapter_str)

    soup = BeautifulSoup(page_source, features='lxml')
```

```
    imgs = soup.find_all('img')
    for img in imgs:
        if 'src' in img.attrs or 'data-src' in img.attrs:
            src = ''
            if 'src' in img.attrs:
                src = img.attrs['src']
            elif 'data-src' in img.attrs:
                src = img.attrs['data-src']
                del img.attrs['data-src']
            name = img_name(src)
            img.attrs['src'] = f'img/{name}.png'

    f = open(f'./chapters/{chapter_str}/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

    driver.close()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(2)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('  : ', stop - start)
```

□□□□ == "□□□□":
□□□□()

```shell
    https://www.feynmanlectures.caltech.edu/I_01.html
    https://www.feynmanlectures.caltech.edu/I_02.html
  :  21.478510914999998

 errpipe_read, errpipe_write = os.pipe()
OSError: [Errno 24]
```

```
 % ulimit a
ulimit:       : a
lzw@lzwjava feynman-lectures-mobi % ulimit -a
-t: CPU    ( )
-f:        (  )
-d:           (    )
-s:       (    )             8192
-c:           (  )         0
-v:       (    )
-l:             (    )
-u:                              2784
-n:                           256

12
download_images
12
mathjax2svg
latexs 128
make_svg 0
insert_svg 0
make_svg 1
insert_svg 1
make_svg 2
insert_svg 2
make_svg 3
insert_svg 3
convert
```

(यह कोड ब्लॉक को हिंदी में अनुवाद करने की आवश्यकता नहीं है क्योंकि यह एक प्रोग्रामिंग कोड है और इसे अपरिवर्तित रहना चाहिए।)

```
12
download_images
12
mathjax2svg
latexs 0
latexs 0
convert
```

```
  :  11.369145162

% grep --include=\*.html -r '\$' *
43/I_43.html:       $T$     ,               , $N$,
43/I_43.html:            $T$
43/I_43.html:               $1/\tau$          ,
43/I_43.html:$\tau$                   $\tau$
43/I_43.html:   $60$        ;   $\tau$
43/I_43.html:  $\tau$ (    )

    E21018:                    ,    Mobi               : <In earlier chapters >    : /private/var/folder
     W28001: Kindle               CSS                  CSS            : 'max-width'    : /private/var/fo
     W29004:                  : <span amzn-src-id="985">    : /private/var/folders/_3/n3b7dq8x6652drmx6_d
     W29004:                  : <p amzn-src-id="975">    : /private/var/folders/_3/n3b7dq8x6652drmx6_d3t3

    W14001:            ,           :  /private/var/folders/_3/n3b7dq8x6652drmx6_d3t3bh0000gr/T/97c9cb4d
    W14001:            ,           :  /private/var/folders/_3/n3b7dq8x6652drmx6_d3t3bh0000gr/T/97c9cb4d
    W14001:            ,           :  /private/var/folders/_3/n3b7dq8x6652drmx6_d3t3bh0000gr/T/97c9cb4d

<span class="disabled" href="#Ch1-F1">
          1-1
</span>

'OEBPS/84b8b4179175f097be1180a10089107be75d7d85.svg'   1264x1011
'OEBPS/23a4df37f269c8ed43f54753eb838b29cff538a1.svg'   1264x259
Traceback (most recent call last):
  File "runpy.py", line 194, in _run_module_as_main
  File "runpy.py", line 87, in _run_code
  File "site.py", line 39, in <module>
  File "site.py", line 35, in main
  File "calibre/utils/ipc/worker.py", line 216, in main
  File "calibre/gui2/convert/gui_conversion.py", line 41, in gui_convert_override
  File "calibre/gui2/convert/gui_conversion.py", line 28, in gui_convert
  File "calibre/ebooks/conversion/plumber.py", line 1274, in run
  File "calibre/ebooks/conversion/plugins/mobi_output.py", line 214, in convert
  File "calibre/ebooks/conversion/plugins/mobi_output.py", line 237, in write_mobi
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 55, in __call__
```

```
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 142, in rasterize_spine
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 152, in rasterize_item
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 185, in rasterize_external
  File "calibre/ebooks/oeb/base.py", line 1092, in bytes_representation
  File "calibre/ebooks/oeb/base.py", line 432, in serialize
TypeError: 'NoneType'

% kindlepreviewer feynman-lectures-on-physics-volumn-1.epub -convert


  -
1/1   ( )
                         ! : /Users/lzw/projects/feynman-lectures-mobi/feynman-lectures-on-physics-volum
  -
   /      /Users/lzw/projects/feynman-lectures-mobi/output


              ...
                      ...
AZW3          ...
                 ...
                                  ,        ...
KF8
    KF8              ...
         aid
         aid
         aid
                    ...
             ...
```