https://www.goindigo.in/flight-booking.html?linkNav=Flight%7CBook%7CBook

- The defa input can be as follows to get the results in next page -

1. #bar > div.cmp-custom-drop-down > button - <div class="cmp-custom-drop-down "><button role="button" class="cmp-custom-drop-down\_\_btn"><i class="cmp-custom-drop-down\_\_btn\_\_icon skp-iconmoon-icon cmp-custom-drop-down\_\_btn\_\_icon--oneway-trip"></i>One Way</button></div>

2. <div class="cmp-custom-drop-down "><button role="button" class="cmp-custom-drop-down\_\_btn"><i class="cmp-custom-drop-down\_\_btn\_\_icon skp-iconmoon-icon icon-user cmp-custom-drop-down\_\_btn\_\_icon--user"></i>1 Pax </button></div>

3. <div class="faresLabel " name="Special Fares">Special Fares</div>

4. MAA

#container-b9910c1891 > div > div.dynamiccontainer.aem-GridColumn.aem-GridColumn--default--12 > div > div > div > section > div.widget-container > div.widget-container\_\_search-form > div.from-destination - <div class="from-destination"><div class="custom-form-control input-text-field "><input type="text" placeholder="From" class="input-text-field\_\_input" maxlength="500" data-info="null" value=""></div><div class="destination-auto-suggestion-dropdown false "><div class="destination-auto-suggestion-dropdown\_\_close"><button type="button" class="destination-auto-suggestion-dropdown\_\_close\_\_btn"><i class="destination-auto-suggestion-dropdown\_\_close\_\_btn\_\_icon skp-iconmoon-icon"></i></button></div><h2 class="destination-auto-suggestion-dropdown\_\_title">Select Departure City</h2><div class="destination-auto-suggestion-dropdown\_\_search"><i class="destination-auto-suggestion-dropdown\_\_search\_\_icon skp-iconmoon-icon"></i><input class="destination-auto-suggestion-dropdown\_\_search\_\_input" placeholder="Select City" value=""><button type="button" class="destination-auto-suggestion-dropdown\_\_search\_\_clear"><i class="destination-auto-suggestion-dropdown\_\_search\_\_clear\_\_icon skp-iconmoon-icon"></i></button></div><div class="destination-auto-suggestion-dropdown\_\_results"><div class="auto-suggestion-group "><div role="heading" aria-level="3"

5. TRZ

#container-b9910c1891 > div > div.dynamiccontainer.aem-GridColumn.aem-GridColumn--default--12 > div > div > div > section > div.widget-container > div.widget-container\_\_search-form > div.to-destination - document.querySelector("#container-b9910c1891 > div > div.dynamiccontainer.aem-GridColumn.aem-GridColumn--default--12 > div > div > div > section > div.widget-container > div.widget-container\_\_search-form > div.to-destination") - //\*[@id="container-b9910c1891"]/div/div[3]/div/div/div/section/div[1]/div[3]/div[2]/div[1]

<div class="to-destination "><div class="custom-form-control input-text-field "><input type="text" placeholder="To" class="input-text-field\_\_input" maxlength="500" data-info="null" value=""></div><div class="destination-auto-suggestion-dropdown false "><div class="destination-auto-suggestion-dropdown\_\_close"><button type="button" class="destination-auto-suggestion-dropdown\_\_close\_\_btn"><i class="destination-auto-suggestion-dropdown\_\_close\_\_btn\_\_icon skp-iconmoon-icon"></i></button></div><h2 class="destination-auto-suggestion-dropdown\_\_title">Select Arrival City</h2><div class="destination-auto-suggestion-dropdown\_\_search"><i class="destination-auto-suggestion-dropdown\_\_search\_\_icon skp-iconmoon-icon"></i><input class="destination-auto-suggestion-dropdown\_\_search\_\_input" placeholder="Select City" value=""><button type="button" class="destination-auto-suggestion-dropdown\_\_search\_\_clear"><i class="destination-auto-suggestion-dropdown\_\_search\_\_clear\_\_icon skp-iconmoon-icon"></i></button></div><div class="destination-auto-suggestion-dropdown\_\_results"><div class="auto-suggestion-group "><div role="heading" aria-level="3" class="auto-suggestion-group\_\_title skp-iconmoon-icon">Popular Destinations</div><div class="destination-row " aria-pressed="false" tabindex="0"><div class="destination-row\_\_left"><div class="airport-city-country">Agra, IN</div><div class="airport-name">Agra Civil Airport, IAF Arjun Nagar Gate, Kheria</div></div><div class="destination-row\_\_right"><div class="airport-code">AGR</div></div></div><div class="destination-row " aria-pressed="false" tabindex="0"><div class="destination-row\_\_left"><div class="airport-city-country">Aizawl, IN</div><div class="airport-name">Aizawl Airport</div

6.

<div class="react-datepicker\_\_input-container"><input type="text" placeholder="Travel Dates" class="" value="30 Nov 2024"></div>

7.

<span class="custom-button\_\_label">Search Flight </span>

#container-b9910c1891 > div > div.dynamiccontainer.aem-GridColumn.aem-GridColumn--default--12 > div > div > div > section > div.widget-container > div.widget-container\_\_search-form > button > span

<div class="skyplus-chip recommended-chip" role="presentation"><div class="skyplus-chip-filled-col-secondary-light-size-xs-bg-system-success-txtcol-system-success"><span>Best Value</span></div></div>

The input value for departure city is - Chennai(MAA)

& destination - Tiruchirappalli(TRZ)  
  
  
The search result page is https://www.goindigo.in/book/flight-select.html and kindly try capturing the single result with the selectors as follows please -  
  
<div class="srp-flight-carousel-container\_\_item rounded-1"><div class="d-flex justify-content-between align-items-center"><div class="skyplus-chip recommended-chip" role="presentation"><div class="skyplus-chip-filled-col-secondary-light-size-xs-bg-system-success-txtcol-system-success"><span>Best Value</span></div></div><p class="tags-small text-tertiary d-flex justify-content-center align-items-center srp\_\_flight-num"><img height="16px" width="16px" class="codeshare-icon" alt="CS" src="https://www.goindigo.in/content/dam/s6common/in/en/assets/codeshare/images/icATR-new.jpg"> 6E 7051</p></div><div class="mt-6 d-flex align-items-center gap-8"><div><p class="sh4 time">20:50</p><p class="airport price-regular-stroked text-decoration-none text-secondary">MAA, T1</p></div><div class="journey-lap flex-grow-1 srp\_\_journey-lap"><i class="sky-icons icon-Flight journey-lap-flight-icon md"></i><div class="journey-lap-line"></div><div class="d-flex gap-1 flex-column align-items-center"><div class="skyplus-text text-color body-small-regular">01h 15m</div><div class="skyplus-text text-color body-small-regular"></div></div><div class="journey-lap-line"></div><div class="journey-lap-dot"></div></div><div><p class="sh4">22:05</p><p class="airport price-regular-stroked text-decoration-none text-secondary">TRZ</p></div></div><div class="line-bg my-8"></div><div class="d-flex justify-content-between align-items-center"><div><h4 class="body-light text-tertiary">Saver fare</h4><h4 class="body-medium-medium text-primary-main">₹3,216</h4><div class="loyalty-points--wrapper"><div class="loyalty-points loyalty-recommended-card-points " tabindex="0" aria-describedby="points-tooltip">+ Earn 276 IndiGo BluChips</div></div></div><div class="skyplus-button "><button type="button" class="skyplus-button--filled skyplus-button--filled-primary skyplus-button--medium body-medium-medium py-3 px-6" aria-label="Select Best Value Flight starting at ₹3,216">Select</button></div></div></div>

Scraping a flight booking site is a complex project, as many sites have protections in place to prevent automated data extraction. Before beginning, there are several technical and ethical considerations you should be familiar with:

### 1. \*\*Understanding Legal and Ethical Concerns\*\*

- \*\*Terms of Service (ToS):\*\* Many websites, including flight booking sites, explicitly prohibit web scraping in their ToS. Scraping against these terms can lead to legal consequences.

- \*\*API Usage:\*\* Some sites provide official APIs that grant access to their data legally and efficiently, often requiring registration and approval for access. Check if the site offers this option first.

### 2. \*\*Core Web Scraping Concepts\*\*

- \*\*HTML Parsing:\*\* Learn HTML basics and how to identify the elements containing the information you need. Knowledge of XPath or CSS selectors helps in targeting specific HTML elements.

- \*\*HTTP Requests:\*\* Familiarize yourself with HTTP methods (GET, POST) to understand how to request and submit data to a server. Tools like Python's `requests` library are essential here.

- \*\*JavaScript Handling:\*\* Many modern sites use JavaScript to dynamically load content, which might require a headless browser (e.g., Puppeteer, Selenium) for scraping.

### 3. \*\*IP Management and Rate Limiting\*\*

- \*\*IP Rotation:\*\* To avoid blocking, IP rotation and proxy management are often necessary for large-scale scraping.

- \*\*Respecting Rate Limits:\*\* Limit request rates to avoid server overload and detection. Ideally, add random time delays between requests.

### 4. \*\*Data Extraction and Parsing Tools\*\*

- \*\*Libraries and Frameworks:\*\* Use libraries like BeautifulSoup or lxml for HTML parsing and extraction. If JavaScript content is an issue, tools like Selenium or Puppeteer are helpful.

- \*\*Regex Usage:\*\* Regular expressions can extract specific patterns from text, like dates, prices, and other data formats.

### 5. \*\*Handling Captchas and Anti-Bot Mechanisms\*\*

- Flight sites often use captchas, JavaScript checks, and other mechanisms to detect and block bots. Familiarize yourself with anti-bot strategies, but remember that bypassing certain protections may be illegal or against the site's policies.

### 6. \*\*Data Storage and Management\*\*

- \*\*Database Knowledge:\*\* Be prepared to store extracted data efficiently, typically using databases like MySQL or NoSQL databases if you’re scraping large amounts of information.

- \*\*Data Cleaning:\*\* Ensure your data is clean, formatted, and consistent for later analysis or use.

### 7. \*\*Automation & Scripting Basics\*\*

- \*\*Python or JavaScript Knowledge:\*\* Both languages are commonly used for web scraping tasks. Python’s libraries, like `requests` and `BeautifulSoup`, are particularly popular.

### 8. \*\*Exploring Alternative Data Sources\*\*

- \*\*Flight Aggregator APIs:\*\* Consider paid or free APIs from services like Amadeus, Skyscanner, or Google Flights, which provide comprehensive data legally.

Understanding these concepts prepares you to handle the complexities of scraping, including potential ethical and legal implications.  
  
  
  
HTML parsing is one of the fundamental skills needed for web scraping, as it allows you to extract data from web pages effectively. Here’s an expanded look at HTML parsing and the tools and techniques you can use to target specific elements on a page:

### Understanding HTML Parsing

\*\*HTML parsing\*\* is the process of reading the HTML code of a web page to identify and extract specific elements or data. Every web page is essentially a document made up of nested HTML elements such as tags, attributes, classes, and IDs. These elements form a \*\*DOM (Document Object Model)\*\* tree structure, making it possible to navigate through the elements and extract information.

### Key Concepts in HTML Parsing

1. \*\*HTML Structure Basics\*\*

- HTML documents are organized in a tree structure with nested tags like `<html>`, `<body>`, `<div>`, `<table>`, `<span>`, etc.

- Each tag may contain attributes such as `class`, `id`, `href`, and `src`, which can be used to uniquely identify and target elements.

2. \*\*Identifying Target Elements\*\*

- Elements on a webpage can be identified by their \*\*tag names\*\* (like `div`, `p`, `span`, etc.), \*\*attributes\*\* (like `id`, `class`, `name`), and \*\*hierarchical structure\*\*.

- A good approach is to inspect the page using browser developer tools (usually accessible by right-clicking and selecting "Inspect" or pressing `F12`) to examine the structure and locate the elements of interest.

3. \*\*CSS Selectors\*\*

- CSS selectors are patterns used to select specific elements based on tag names, classes, IDs, attributes, or a combination of these. Some examples include:

- `div`: Selects all `<div>` tags.

- `.class-name`: Selects elements with a specific class.

- `#id-name`: Selects the element with a specific ID.

- `a[href]`: Selects all `<a>` tags that contain the `href` attribute.

- `div > p`: Selects all `<p>` tags that are direct children of `<div>` tags.

- CSS selectors are supported by most HTML parsing libraries, allowing you to directly target elements without needing to traverse the entire document.

4. \*\*XPath Selectors\*\*

- \*\*XPath\*\* (XML Path Language) is a more powerful method for selecting elements, particularly useful when dealing with nested elements.

- XPath syntax allows for more complex queries and supports:

- Absolute paths, e.g., `/html/body/div`.

- Relative paths, e.g., `//div[@class='example']`, which selects all `<div>` tags with the class `example`.

- Conditional expressions, e.g., `//a[contains(@href, 'example')]`, which selects all `<a>` tags where the `href` attribute contains the word "example."

- XPath is especially helpful when a website's structure is complex, and CSS selectors are not sufficient to precisely locate elements.

### Tools and Libraries for HTML Parsing

To parse HTML, you can use libraries in popular programming languages like Python. Here are a few key ones:

- \*\*BeautifulSoup (Python)\*\*: An excellent library for parsing HTML and XML documents, often used with the `requests` library to retrieve the HTML content of web pages.

- Example of selecting elements using CSS selectors:

```python

from bs4 import BeautifulSoup

import requests

# Load page content

page = requests.get('https://example.com')

soup = BeautifulSoup(page.content, 'html.parser')

# Select all elements with a specific class

elements = soup.select('.class-name')

for element in elements:

print(element.text)

```

- \*\*lxml (Python)\*\*: Another powerful library for HTML parsing, lxml offers full support for XPath expressions.

- Example of using XPath:

```python

from lxml import html

import requests

# Load page content

page = requests.get('https://example.com')

tree = html.fromstring(page.content)

# Select elements using XPath

elements = tree.xpath('//div[@class="example"]')

for element in elements:

print(element.text\_content())

```

- \*\*Selenium (Python)\*\*: A headless browser that can interact with JavaScript-rendered content, which is sometimes necessary for dynamically loaded elements.

### Example: Extracting Data with CSS and XPath Selectors

Let’s say you want to extract the flight prices from a sample flight booking site. Here’s how you might approach it:

1. \*\*Inspect the Page\*\*:

- Open the site in a browser, inspect the flight prices, and find the tags, classes, or attributes associated with them.

- Suppose the prices are in `<span>` tags with the class `price`.

2. \*\*Write Code Using CSS Selectors\*\*:

```python

from bs4 import BeautifulSoup

import requests

page = requests.get('https://sample-flight-booking.com')

soup = BeautifulSoup(page.content, 'html.parser')

# Using CSS selector to get prices

prices = soup.select('span.price')

for price in prices:

print(price.text)

```

3. \*\*Using XPath Selectors\*\*:

```python

from lxml import html

import requests

page = requests.get('https://sample-flight-booking.com')

tree = html.fromstring(page.content)

# Using XPath to get prices

prices = tree.xpath('//span[@class="price"]')

for price in prices:

print(price.text\_content())

```

### Tips for Effective HTML Parsing

- \*\*Combine CSS and XPath selectively\*\*: Start with CSS selectors for simpler structures, and use XPath for more complex, hierarchical selections.

- \*\*Use browser tools\*\*: Inspect and test CSS and XPath selectors in the browser’s console to ensure they return the expected elements.

- \*\*Error Handling\*\*: Websites may change their structure over time, so be prepared to update your selectors if your code stops working.

Learning to effectively parse HTML and using CSS/XPath selectors will make your web scraping more accurate and efficient, especially when dealing with complex sites like flight booking pages.

Many modern websites use JavaScript to load content dynamically, meaning the content isn't immediately available in the HTML source code when the page first loads. Instead, JavaScript runs in the browser to fetch and render content asynchronously (after the page is initially loaded). This can create a challenge for web scraping, as simply fetching the HTML from the server might not provide the full content.

To scrape data from such sites, tools like \*\*headless browsers\*\* are useful. A \*\*headless browser\*\* is essentially a web browser without a graphical user interface (GUI), meaning it can load and interact with web pages just like a normal browser but operates in the background without displaying anything visually.

Two popular tools for this are:

### 1. Puppeteer

- \*\*Puppeteer\*\* is a Node.js library that provides a high-level API to control Chrome or Chromium over the DevTools Protocol.

- It allows you to open a browser instance, navigate to a URL, wait for JavaScript to load, and then interact with the page (like clicking buttons or filling forms).

- Puppeteer can take screenshots, extract text, and handle advanced interactions, making it ideal for scraping content from sites that rely heavily on JavaScript.

### 2. Selenium

- \*\*Selenium\*\* is another tool that can automate browser actions. It supports multiple browsers (Chrome, Firefox, Edge, etc.) and is commonly used for testing, but it’s also widely used in scraping.

- Selenium can load dynamic content, interact with page elements, and execute JavaScript. Unlike Puppeteer, which is specific to Node.js and Chrome, Selenium works with multiple programming languages (Python, Java, etc.) and browsers.

### Why Headless Browsers Are Useful for Scraping

- \*\*Rendering JavaScript\*\*: Headless browsers can execute JavaScript just like a user’s browser would, ensuring that all dynamically loaded content appears.

- \*\*Interactivity\*\*: They allow interaction with page elements, such as clicking buttons, filling forms, and scrolling.

- \*\*Waiting for Content\*\*: With headless browsers, you can wait for specific elements to load before scraping, making them reliable for pages with loading delays.

### Example Workflow

1. \*\*Start a Headless Browser\*\*: Launch Puppeteer or Selenium in headless mode.

2. \*\*Navigate to the Page\*\*: Direct the browser to the desired URL.

3. \*\*Wait for Content\*\*: Instruct the browser to wait for specific elements to load or for JavaScript to finish executing.

4. \*\*Extract Data\*\*: Use methods to find and retrieve the data from the fully rendered page.

5. \*\*Close the Browser\*\*: End the headless browser session to free resources.

This approach ensures you capture the complete, rendered content of a web page, enabling successful scraping of data from sites that use JavaScript extensively.