XPATH QUICK REFERENCE

Axes, predicates, functions, operators, node selection

Syntax

Path Expressions

`/`	Root node (absolute path start)
`/bookstore/book`	Direct child selection
`//book`	Select all book nodes anywhere
`.`	Current context node
`..`	Parent of current node
`@lang`	Attribute named lang
`node()`	Any node of any type
`*`	Any element node
`@*`	Any attribute

Basic Examples

```
/html/body/div          # absolute path to <div>
//input[@type='text']   # all text inputs
//div[@class='main']/*  # children of div.main
//a/@href               # all href attributes
```

Combining Paths

```
//book/title | //book/price   # union of two paths
//h1 | //h2 | //h3           # multiple element types
```

Axes

Axis Directions

`child::`	Direct children (default axis)
`parent::`	Direct parent
`ancestor::`	All ancestors to root
`ancestor-or-self::`	Ancestors + current node
`descendant::`	All descendants
`descendant-or-self::`	Descendants + current node
`following::`	All nodes after current in document
`following-sibling::`	Siblings after current
`preceding::`	All nodes before current in document
`preceding-sibling::`	Siblings before current
`self::`	Current node only
`attribute::`	Attributes of current node
`namespace::`	Namespace nodes

Axis Examples

```
//div/child::p              # <p> children of <div>
//td/parent::tr             # <tr> parent of <td>
//h2/following-sibling::p   # <p> after <h2>
//li/ancestor::ul           # <ul> containing <li>
```

Predicates

Filtering with Predicates

```
//book[1]                   # first book element
//book[last()]              # last book element
//book[position() < 3]      # first two books
//book[@lang='en']          # books with lang="en"
//book[price > 30]          # books with price > 30
```

Predicate Patterns

`[n]`	Element at position n (1-based)
`[last()]`	Last element
`[last()-1]`	Second to last
`[@attr]`	Has attribute
`[@attr='val']`	Attribute equals value
`[element]`	Has child element
`[element='text']`	Child element contains text
`[not(@attr)]`	Does not have attribute

Chained Predicates

```
//div[@class='list']//a[1]          # first <a> in div.list
//input[@type='text'][@name='q']    # AND condition
//book[price>30][@lang='en']        # multiple conditions
```

Functions

String Functions

`contains(s, sub)`	True if s contains sub
`starts-with(s, pre)`	True if s starts with pre
`string-length(s)`	Length of string
`normalize-space(s)`	Trim and collapse whitespace
`concat(a, b, ...)`	Join strings
`substring(s, pos, len)`	Extract substring (1-based)
`translate(s, from, to)`	Character-by-character replace

Numeric Functions

`sum(node-set)`	Sum of numeric values
`count(node-set)`	Number of nodes
`floor(n)`	Round down
`ceiling(n)`	Round up
`round(n)`	Round to nearest integer
`number(val)`	Convert to number

Function Examples

```
//div[contains(@class, 'active')]
//a[starts-with(@href, 'https')]
//p[string-length(text()) > 100]
//ul[count(li) > 5]
```

Operators

Comparison Operators

`=`	Equal
`!=`	Not equal
`<`	Less than
`<=`	Less than or equal
`>`	Greater than
`>=`	Greater than or equal

Logical & Arithmetic

`and`	Logical AND
`or`	Logical OR
`not()`	Logical NOT (function)
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`div`	Division
`mod`	Modulus
`|`	Union of node sets

Operator Examples

```
//book[price > 20 and price < 50]
//item[@type='a' or @type='b']
//span[not(contains(@class, 'hidden'))]
```

Node Tests

Node Types

`node()`	Any node (element, text, comment, PI)
`text()`	Text node only
`comment()`	Comment node only
`processing-instruction()`	Processing instruction node
`*`	Any element node
`@*`	Any attribute node
`element-name`	Element with specific name

Node Test Examples

```
//p/text()              # text content of <p>
//div/comment()         # comments inside <div>
//body/node()           # all child nodes of <body>
//div/*                 # all element children of <div>
```

Boolean Functions

`true()`	Boolean true
`false()`	Boolean false
`boolean(expr)`	Convert to boolean
`not(expr)`	Negate boolean
`lang(code)`	True if node's lang matches

Abbreviations

Short vs Long Form

`(none)`	child:: (default axis)
`@`	attribute::
`//`	/descendant-or-self::node()/
`.`	self::node()
`..`	parent::node()
`[n]`	[position()=n]

Abbreviated Examples

```
# These pairs are equivalent:
child::div              →  div
attribute::href         →  @href
/descendant-or-self::node()/p  →  //p
self::node()            →  .
parent::node()          →  ..
```

Common Abbreviated Patterns

```
//div[@id='main']       # div with id="main"
//table//td             # all <td> in any <table>
../sibling              # sibling via parent
.//span                 # span descendants of context
```

Common Patterns

Web Scraping / Testing

```
//input[@name='username']          # form input by name
//button[text()='Submit']          # button by text
//div[contains(@class, 'error')]   # element by partial class
//a[contains(@href, 'login')]      # link by partial href
```

Conditional Selection

```
//div[@class='item'][.//span[@class='price']]
//tr[td[1]='Active']                # row where 1st cell = Active
//*[contains(text(), 'Warning')]   # any element with text
```

XPath in Python (lxml)

```
from lxml import html
tree = html.fromstring(page_content)
links = tree.xpath('//a/@href')
titles = tree.xpath('//h2/text()')
```

XPath in Selenium

```
driver.find_element(By.XPATH, "//input[@id='search']")
driver.find_elements(By.XPATH, "//li[@class='result']")
```