Python Basics Cheat Sheet

Here you will find all the Python core concepts you need to know before learning any third-party library.

Data Types

```
Integers (int): 1
Float (float): 1.2
String (str): "Hello World"
Boolean: True/False
List: [value1, value2]
Dictionary: {key1:value1, key2:value2, ...}
```

Numeric Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponent
%	Modulus
//	Floor division

Comparison Operators

==	Equal to
!=	Different
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

String methods

```
string.upper(): converts to uppercase
string.lower(): converts to lowercase
string.title(): converts to title case
string.count('l'): counts how many times "l"
                   appears
string.find('h'): position of the "h" first
                  ocurrance
string.replace('o', 'u'): replaces "o" with "u"
```

Variables

Variable assignment:
```
message_1 = "I'm learning Python"
message_2 = "and it's fun!"
```

String concatenation (+ operator):
```
message_1 + ' ' + message_2
```

String concatenation (f-string):
```
f'{message_1} {message_2}'
```

List

Creating a list:
```
countries = ['United States', 'India',
             'China', 'Brazil']
```

Create an empty list:
```
my_list = []
```

Indexing:
```
>>> countries[0]
United States

>>> countries[3]
Brazil

>>> countries[-1]
Brazil
```

Slicing:
```
>>>countries[0:3]
['United States', 'India', 'China']

>>>countries[1:]
['India', 'China', 'Brazil']

>>>countries[:2]
['United States', 'India']
```

Adding elements to a list:
```
countries.append('Canada')
countries.insert(0,'Canada')
```

Nested list:
```
nested_list = [countries, countries_2]
```

Remove element:
```
countries.remove('United States')
countries.pop(0)#removes and returns value
del countries[0]
```

Creating a new list:
```
numbers = [4, 3, 10, 7, 1, 2]
```

Sorting a list:
```
>>> numbers.sort()
[1, 2, 3, 4, 7, 10]

>>> numbers.sort(reverse=True)
[10, 7, 4, 3, 2, 1]
```

Update value on a list:
```
>>> numbers[0] = 1000
>>> numbers
[1000, 7, 4, 3, 2, 1]
```

Copying a list:
```
new_list = countries[:]
new_list_2 = countries.copy()
```

Built-in Functions

Print an object:
```
print("Hello World")
```

Return the length of x:
```
len(x)
```

Return the minimum value:
```
min(x)
```

Return the maximum value:
```
max(x)
```

Returns a sequence of numbers:
```
range(x1,x2,n) # from x1 to x2
(increments by n)
```

Convert x to a string:
```
str(x)
```

Convert x to an integer/float:
```
int(x)
float(x)
```

Convert x to a list:
```
list(x)
```

Dictionary

Creating a dictionary:
```
my_data = {'name':'Frank', 'age':26}
```

Create an empty dictionary:
```
my_dict = {}
```

Get value of key "name":
```
>>> my_data["name"]
'Frank'
```

Get the keys:
```
>>> my_data.keys()
dict_keys(['name', 'age'])
```

Get the values:
```
>>> my_data.values()
dict_values(['Frank', 26])
```

Get the pair key-value:
```
>>> my_data.items()
dict_items([('name', 'Frank'), ('age', 26)])
```

Adding/updating items in a dictionary:
```
my_data['height']=1.7
my_data.update({'height':1.8,
          'languages':['English', 'Spanish']})
>>> my_data
{'name': 'Frank',
'age': 26,
'height': 1.8,
'languages': ['English', 'Spanish']}
```

Remove an item:
```
my_data.pop('height')
del my_data['languages']
my_data.clear()
```

Copying a dictionary:
```
new_dict = my_data.copy()
```

If Statement

Conditional test:
```
if <condition>:
    <code>
elif <condition>:
    <code>
...
else:
    <code>
```

Example:
```
if age>=18:
    print("You're an adult!")
```

Conditional test with list:
```
if <value> in <list>:
    <code>
```

Loops

For loop:
```
for <variable> in <list>:
    <code>
```

For loop and enumerate list elements:
```
for i, element in enumerate(<list>):
    <code>
```

For loop and obtain dictionary elements:
```
for key, value in my_dict.items():
    <code>
```

While loop:
```
while <condition>:
    <code>
```

Data Validation

Try-except:
```
try:
    <code>
except <error>:
    <code>
```

Loop control statement:
```
break: stops loop execution
continue: jumps to next iteration
pass: does nothing
```

Functions

Create a function:
```
def function(<params>):
    <code>
    return <data>
```

Modules

Import module:
```
import module
module.method()
```

OS module:
```
import os
os.getcwd()
os.listdir()
os.makedirs(<path>)
```

Special Characters

#	Comment
\n	New Line

Boolean Operators

and	logical AND
or	logical OR
not	logical NOT

Boolean Operators (Pandas)

&	logical AND
|	logical OR
~	logical NOT

Below there are my guides, tutorials and complete Data Science course:
- Medium Guides
- YouTube Tutorials
- Data Science Course (Udemy)

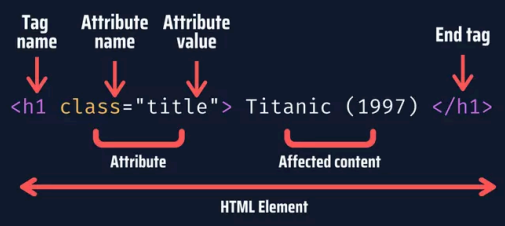
Made by Frank Andrade frank-andrade.medium.com

Web Scraping Cheat Sheet

Web Scraping is the process of extracting data from a website. Before studying Beautiful Soup and Selenium, it's good to review some HTML basics first.

HTML for Web Scraping

Let's take a look at the HTML element syntax.

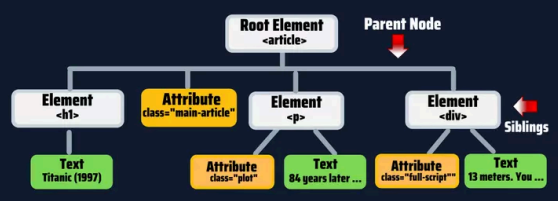


This is a single HTML element, but the HTML code behind a website has hundreds of them.

HTML code example

```
<article class="main-article">
 <h1> Titanic (1997) </h1>
 <p class="plot"> 84 years later ... </p>
 <div class="full-script"> 13 meters. You ... </div>
</article>
```

The HTML code is structured with "nodes". Each rectangle below represents a node (element, attribute and text nodes)



- "Siblings" are nodes with the same parent.
- A node's children and its children's children are called its "descendants". Similarly, a node's parent and its parent's parent are called its "ancestors".
- it's recommended to find element in this order.
  a. ID
  b. Class name
  c. Tag name
  d. Xpath

Beautiful Soup

Workflow

Importing the libraries
```
from bs4 import BeautifulSoup
import requests
```

Fetch the pages
```
result=requests.get("www.google.com")
result.status_code #get status code
result.headers #get the headers
```

Page content
```
content = result.text
```

Create soup
```
soup = BeautifulSoup(content,"lxml")
```

HTML in a readable format
```
print(soup.prettify())
```

Find an element
```
soup.find(id="specific_id")
```

Find elements
```
soup.find_all("a")
soup.find_all("a","css_class")
soup.find_all("a",class_="my_class")
soup.find_all("a",attrs={"class":
                         "my_class"})
```

Get inner text
```
sample = element.get_text()
sample = element.get_text(strip=True,
                          separator= ' ')
```

Get specific attributes
```
sample = element.get('href')
```

XPath

We need to learn XPath to scrape with Selenium or Scrapy.

XPath Syntax

An XPath usually contains a tag name, attribute name, and attribute value.

```
//tagName[@AttributeName="Value"]
```

Let's check some examples to locate the article, title, and transcript elements of the HTML code we used before.

```
//article[@class="main-article"]
//h1
//div[@class="full-script"]
```

XPath Functions and Operators

XPath functions

```
//tag[contains(@AttributeName, "Value")]
```

XPath Operators: and, or

```
//tag[(expression 1) and (expression 2)]
```

XPath Special Characters

/	Selects the children from the node set on the left side of this character
//	Specifies that the matching node set should be located at any level within the document
.	Specifies the current context should be used (refers to present node)
..	Refers to a parent node
*	A wildcard character that selects all elements or attributes regardless of names
@	Select an attribute
()	Grouping an XPath expression
[n]	Indicates that a node with index "n" should be selected

Selenium

Workflow

```
from selenium import webdriver
web="www.google.com"
path='introduce chromedriver path'
driver = webdriver.Chrome(path)
driver.get(web)
```

Find an element

```
driver.find_element_by_id('name')
```

Find elements

```
driver.find_elements_by_class_name()
driver.find_elements_by_css_selector
driver.find_elements_by_xpath()
driver.find_elements_by_tag_name()
driver.find_elements_by_name()
```

Quit driver

```
driver.quit()
```

Getting the text

```
data = element.text
```

Implicit Waits

```
import time
time.sleep(2)
```

Explicit Waits

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.ID,
'id_name'))) #Wait 5 seconds until an element is clickable
```

Options: Headless mode, change window size

```
from selenium.webdriver.chrome.options import Options
options = Options()
options.headless = True
options.add_argument('window-size=1920x1080')
driver=webdriver.Chrome(path,options=options)
```

Below there are my guides, tutorials and complete web scraping course:

- Medium Guides
- YouTube Tutorials
- Web Scraping Course (Udemy)

Made by Frank Andrade frank-andrade.medium.com

Task	Code
Import JSON Module	`import json`
Read JSON from a String	`data = json.loads(json_string)`
Write JSON to a String	`json_string = json.dumps(data, indent=4)`
Read JSON from a File	`python with open('filename.json', 'r') as f:  data = json.load(f)`
Write JSON to a File	`python with open('filename.json', 'w') as f:  json.dump(data, f, indent=4)`