# Python: Web Data Acquisition

*Pertemuan 8*

*MK Algoritma Pemrograman II*

**Ika Qutsiati Utami, S.Kom., M.Sc.**

**M. N. Fakhruzzaman, S.Kom., M.Sc.**

Program Studi S1 Teknologi Sains Data

Fakultas Teknologi Maju dan Multidisiplin

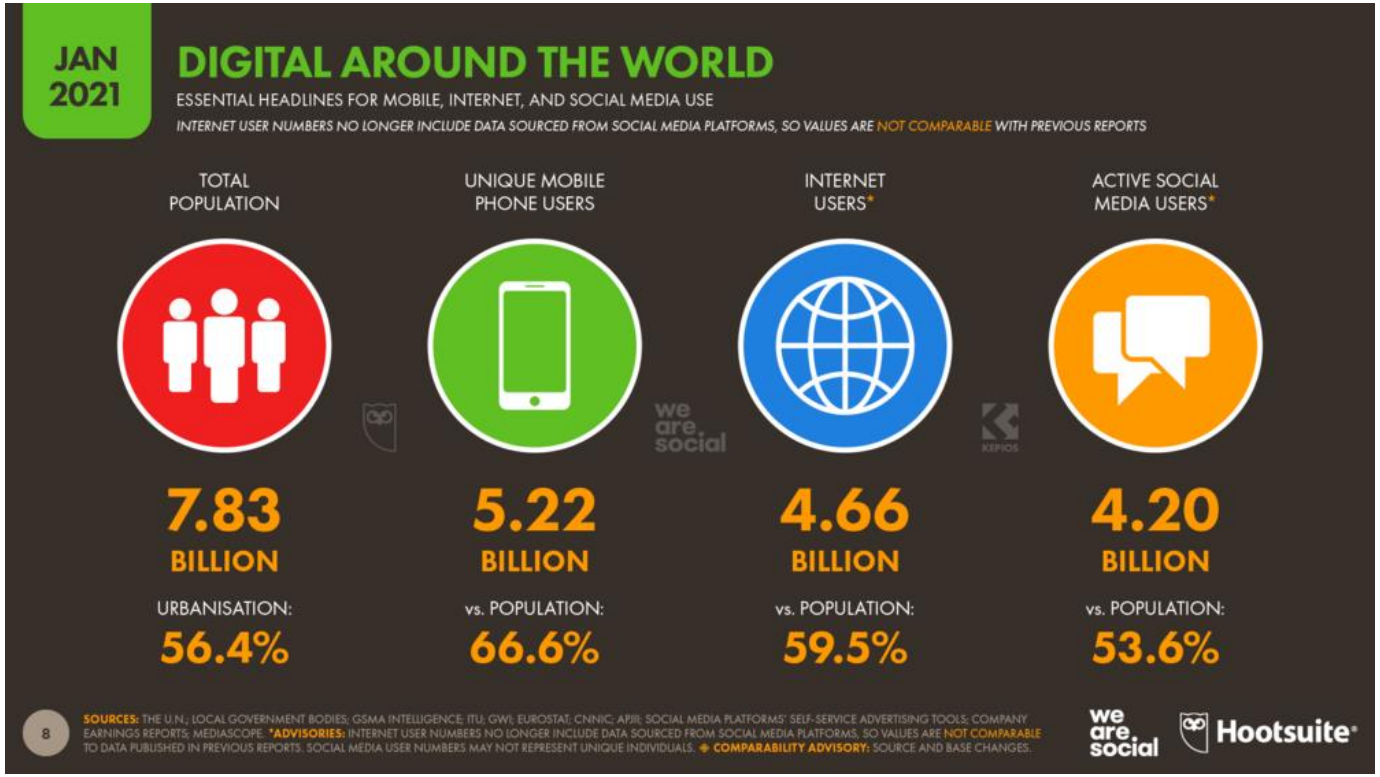
Universitas Airlangga Indonesia

# Outline

- Why web / internet?
- Python Requests & BeautifulSoup
- Kaggle data API and others..
- Web Scraping
- Web Crawling
- Selenium

# Why Web or Internet?

- BIG DATA (3V)
- Abundance of data (raw, unpolished, hidden insights)
- Main source of today's communication
- Public repositories, Open dataset, publicly accessible data
- Internet is (nearly) ubiquitous

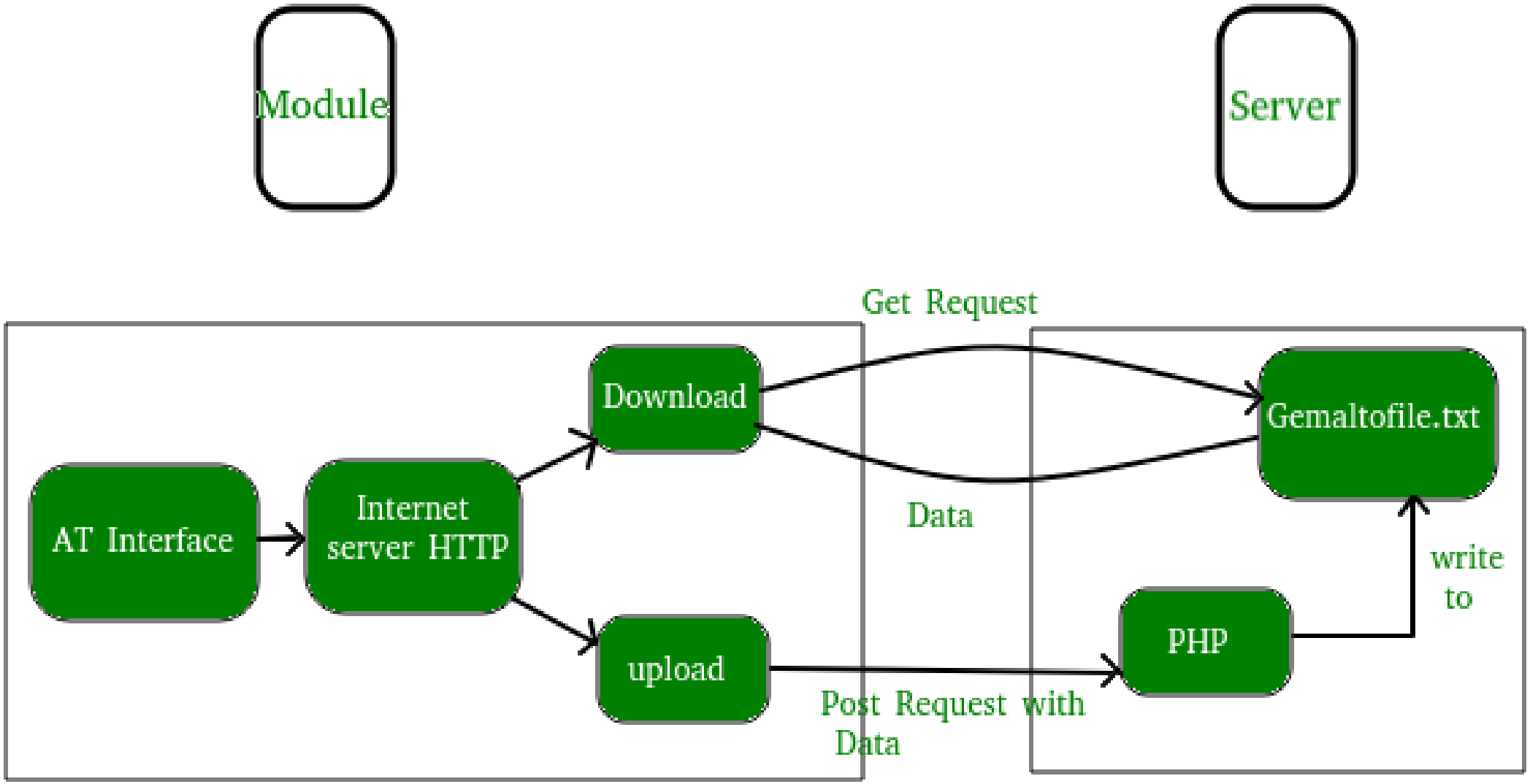


# Examples of Internet Acquired Data

- **Any website, pages, news sites, blogs**
- GIS / Map data (Openstreetmap has publicly accessible API for acq)
- **Kaggle Dataset** (Kaggle has data API)
- Github repos (sometimes datasets are available)
- Mendeley data repo (has accessible API)
- Twitter (main source of public discourse, trends, etc)
- Facebook Family (has limited API but useful)
- Gutenberg (text based books)
- Google open dataset (or its family) (has API)

# Python Request

- To use HTTP Request from Python
- HTTP Request is important:
  - **GET (start here) : to request data from the server.**
  - POST : to submit data to be processed to the server.
- HTTP GET method is client request to server via HTTP which instructs to get a specific item
- HTTP GET can be used to request data from APIs (sometimes need payload/queries), Websites, or other HTTP accessible resource
- To install, **simply pip install requests**


Module
Server
Get Request
Download
Gemaltofile.txt
AT Interface
Internet server HTTP
Data
write to
upload
PHP
Post Request with Data

# Python Requests Usage

- Request and Get specific web page

```
[16] response = requests.get('https://api.github.com')
     response.text
```

```
'{\n  "current_user_url": "https://api.github.com/user",\n  "current_user_authorizations_html_url": "https://github.com/settings/connections/applicat
ons{/client_id}",\n  "authorizations_url": "https://api.github.com/authorizations",\n  "code_search_url": "https://api.github.com/search/code?q={quer
y}{&page,per_page,sort,order}",\n  "commit_search_url": "https://api.github.com/search/commits?q={query}{&page,per_page,sort,order}",\n  "emails_url"
"https://api.github.com/user/emails",\n  "emojis_url": "https://api.github.com/emojis",\n  "events_url": "https://api.github.com/events",\n  "feeds_u
l": "https://api.github.com/feeds",\n  "followers_url": "https://api.github.com/user/followers",\n  "following_url": "https://api.github.com/user/fol
owing{/target}",\n  "gists_url": "https://api.github.com/gists{/gist_id}",\n  "hub_url": "https://api.github.com/hub",\n  "issue_search_url": "http
s://api.github.com/search/issues?q={query}{&page,per_page,sort,order}",\n  "issues_url": "https://api.gi...'
```

- Data from APIs (which accessible by httprequests)

```
>>> import requests
>>> url = 'https://api.github.com'
>>>
>>> requests.get(url)
<Response [200]>
>>>
```

```
>>> url_invalid = 'https://api.github.com/invalid'
>>> response = requests.get(url_invalid)
>>> response
<Response [404]>
>>> response.status_code
404
```

# Python Request Get Webpage

```
response = requests.get('https://ftmm.unair.ac.id')
response.text
```

```
'\n<!DOCTYPE html>\n<html class="html" lang="id-ID">\n<head>\n\t<meta charset="UTF-8">\n\t<link rel="profi
ta name=\'robots\' content=\'index, follow, max-image-preview:large, max-snippet:-1, max-video-preview:-1\
ef="https://ftmm.unair.ac.id" />\n<link rel="alternate" hreflang="en" href="https://ftmm.unair.ac.id/en/"
ice-width, initial-scale=1">\n\t<!-- This site is optimized with the Yoast SEO plugin v16.0.2 - https://yo
>Beranda - Fakultas Teknologi Maju dan Multidisiplin | Universitas Airlangga</title>\n\t<link rel="canonic
meta property="og:locale" content="id_ID" />\n\t<meta property="og:type" content="website" />\n\t<meta pro
Teknologi Maju dan Multidisiplin | Universitas Airlangga" />\n\t<meta property="og:description" cont...'
```

Then, how to parse it?

Let’s talk about accessible API first

# Kaggle API

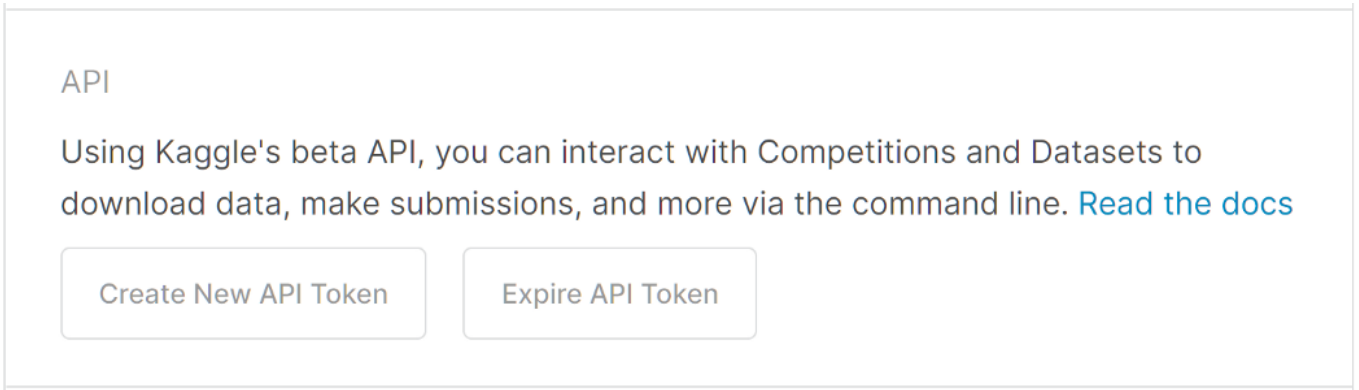
- Kaggle offers API service to access and download datasets they hosted
- Secured by API authentication
- **pip install kaggle**
- **import Kaggle**

```
import kaggle
```

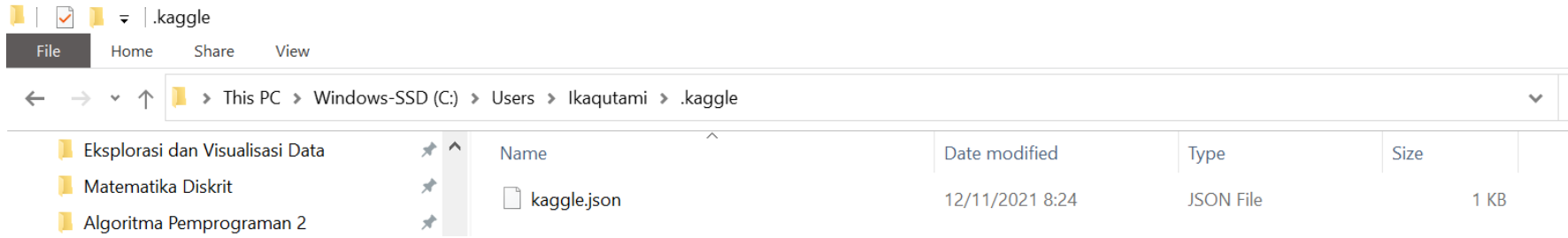
```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-5-2e5e3441a2d1> in <module>()
----> 1 import kaggle

C:\Users\James\anaconda3\envs\ml\lib\site-packages\kaggle\__init__.py in <module>()
     21
     22 api = KaggleApi(ApiClient())
---> 23 api.authenticate()

C:\Users\James\anaconda3\envs\ml\lib\site-packages\kaggle\api\kaggle_api_extended.py i
n authenticate(self)
    164                 raise IOError('Could not find {}. Make sure it\'s located in'
    165                               ' {}. Or use the environment method.'.format(
--> 166                                 self.config_file, self.config_dir))
    167
    168         # Step 3: load into configuration!

OSError: Could not find kaggle.json. Make sure it's located in C:\Users\James\.kaggle.
Or use the environment method.
```

# How to use Kaggle API

- **Log-in** to Kaggle (or sign up)
- **Navigate** to your Account page (click top-right profile picture)
- Click **Create new Token**




- Save **Kaggle.json** to path



# Kaggle CLI Tool

- For searching competitions, datasets, etc.

`kaggle competitions list` : list the currently active competitions

`kaggle competitions download -c [COMPETITION]` : download files associated with a competition

`kaggle competitions submit -c [COMPETITION] -f [FILE] -m [MESSAGE]` : make a competition submission

`kaggle datasets list -s [KEYWORD]` : list datasets matching a search term

`kaggle datasets download -d [DATASET]` : download files associated with a dataset

```
C:\Users\Ikaqutami>kaggle competitions list
```

| ref | deadline | category | reward | teamCount | userHasEntered |
|---|---|---|---|---|---|
| contradictory-my-dear-watson | 2030-07-01 23:59:00 | Getting Started | Prizes | 74 | False |
| gan-getting-started | 2030-07-01 23:59:00 | Getting Started | Prizes | 75 | False |
| store-sales-time-series-forecasting | 2030-06-30 23:59:00 | Getting Started | Knowledge | 592 | False |
| tpu-getting-started | 2030-06-03 23:59:00 | Getting Started | Knowledge | 140 | False |
| digit-recognizer | 2030-01-01 00:00:00 | Getting Started | Knowledge | 1480 | False |
| titanic | 2030-01-01 00:00:00 | Getting Started | Knowledge | 14627 | True |
| house-prices-advanced-regression-techniques | 2030-01-01 00:00:00 | Getting Started | Knowledge | 4463 | False |
| connectx | 2030-01-01 00:00:00 | Getting Started | Knowledge | 257 | False |
| nlp-getting-started | 2030-01-01 00:00:00 | Getting Started | Knowledge | 1218 | False |
| competitive-data-science-predict-future-sales | 2022-12-31 23:59:00 | Playground | Kudos | 12967 | False |
| jigsaw-toxic-severity-rating | 2022-02-07 23:59:00 | Featured | $50,000 | 183 | False |
| g-research-crypto-forecasting | 2022-02-01 23:59:00 | Featured | $125,000 | 453 | False |
| petfinder-pawpularity-score | 2022-01-13 23:59:00 | Research | $25,000 | 1941 | False |
| optiver-realized-volatility-prediction | 2022-01-10 23:59:00 | Featured | $100,000 | 3852 | False |
| nfl-big-data-bowl-2022 | 2022-01-06 23:59:00 | Analytics | $100,000 | 0 | False |
| sartorius-cell-instance-segmentation | 2021-12-30 23:59:00 | Featured | $75,000 | 706 | False |
| wikipedia-image-caption | 2021-12-09 11:59:00 | Playground | Swag | 76 | False |
| lux-ai-2021 | 2021-12-06 23:59:00 | Featured | $10,000 | 989 | False |
| tabular-playground-series-nov-2021 | 2021-11-30 23:59:00 | Playground | Swag | 732 | False |
| kaggle-survey-2021 | 2021-11-28 23:59:00 | Analytics | $30,000 | 0 | False |

# Initialize Kaggle API

- **from kaggle.api.kaggle_api_extended import KaggleApi**
- **api = KaggleApi()**
- **api.authenticate()**

There’s 2 type of datasets:

1. **Competition datasets**
2. **Standalone dataset**

# How to use Kaggle competition datasets

```
from kaggle.api.kaggle_api_extended import KaggleApi

api = KaggleApi()
api.authenticate()

# downloading from kaggle.com/c/sentiment-analysis-on-movie-reviews
# there are two files, train.tsv.zip and test.tsv.zip
# we write to the current directory with './'
api.competition_download_file('sentiment-analysis-on-movie-reviews',
                              'train.tsv.zip', path='./')
api.competition_download_file('sentiment-analysis-on-movie-reviews',
                              'test.tsv.zip', path='./')
```

```
api.competition_download_files('sentiment-analysis-on-movie-reviews',
path='./')
```

# How to use Standalone Datasets

```
from kaggle.api.kaggle_api_extended import KaggleApi


api = KaggleApi()
api.authenticate()


# downloading from kaggle.com/kazanova/sentiment140
# we write to the current directory path with './'
api.dataset_download_file('kazanova/sentiment140',
                          file_name='training.1600000.processed.noemoticon.csv',
                          path='./')
```

```
api.dataset_download_files('kazanova/sentiment140', path='./')
```

```
import zipfile


with zipfile.ZipFile('path/to/data.zip', 'r') as zipref:
    zipref.extractall('target/path')
```

# Web Scraping

- Web Scraping is the process of **using program/bots to extract / scrape data from the web / internet.**
- Scraping results is usually raw (internet / code is text-based)
- Some websites put **bot restriction**
- Crawling is a form of scraping (with additional feature of following urls)
- Simple web scraping we did -> **Python Requests**

- **BEFORE SCRAPING: Check website policy!!**

# Ethics of Scraping and Crawling

- Scraping or Crawling can be Illegal (in some cases...)
- Always check robots.txt on all websites before scraping
- Sometimes websites uses javascript / other which can’t be scraped (because it fetches data from other place)
- Don’t overscrape, it can be seen as an DDoS attempt
- Never include names (especially your own)

# HTML Parsing

- Using Python Request, HTML of a website is retrieved
- Parsing is needed to map content from raw html to become readable
- BeautifulSoup is an HTML/XML parsing library
- **Pip install beautifulsoup4**

# Parsing requests response

```
response = requests.get('https://ftmm.unair.ac.id')
response.text
```

```
'\n<!DOCTYPE html>\n<html class="html" lang="id-ID">\n<head>\n\t<meta charset="UTF-8">\n\t<link rel="prof:
ta name=\'robots\' content=\'index, follow, max-image-preview:large, max-snippet:-1, max-video-preview:-1\
ef="https://ftmm.unair.ac.id" />\n<link rel="alternate" hreflang="en" href="https://ftmm.unair.ac.id/en/"
ice-width, initial-scale=1">\n\t<!-- This site is optimized with the Yoast SEO plugin v16.0.2 - https://yo
>Beranda - Fakultas Teknologi Maju dan Multidisiplin | Universitas Airlangga</title>\n\t<link rel="canonic
meta property="og:locale" content="id_ID" />\n\t<meta property="og:type" content="website" />\n\t<meta pro
Teknologi Maju dan Multidisiplin | Universitas Airlangga" />\n\t<meta property="og:description" cont...'
```

```
from bs4 import BeautifulSoup

response = requests.get('https://ftmm.unair.ac.id')
rawhtml = response.text
soup = BeautifulSoup(rawhtml, 'html.parser')
```

# BeautifulSoup Methods

- **find()**
- **find_all()**
- All with search parameter, you can also use RegEx

```
for i in soup.find_all('a'):  #mencari semua text dengan tag <a>
    print(i.get('href'))
```

```
https://ftmm.unair.ac.id/teknik-elektro-program-studi/
https://ftmm.unair.ac.id/rekayasa-nanoteknologi-program-studi/
https://ftmm.unair.ac.id/category/stmm/
https://ftmm.unair.ac.id/dosen-teknik-industri-ciptakan-aplikasi-medicaltourism-id-untuk/
https://ftmm.unair.ac.id/dosen-teknik-industri-ciptakan-aplikasi-medicaltourism-id-untuk/
https://ftmm.unair.ac.id/dosen-teknik-industri-ciptakan-aplikasi-medicaltourism-id-untuk/
https://ftmm.unair.ac.id/summer-programs-2021-migrant-workers-empowerment/
https://ftmm.unair.ac.id/summer-programs-2021-migrant-workers-empowerment/
https://ftmm.unair.ac.id/pengumuman-kegiatan-pengenalan-kehidupan-kampus-bagi-mahasiswa-baru-pkkmb-2021/
https://ftmm.unair.ac.id/pengumuman-kegiatan-pengenalan-kehidupan-kampus-bagi-mahasiswa-baru-pkkmb-2021/
```

# Another example

```
for i in soup.find_all('h1'):  #mencari semua text dengan tag <h1> biasa untuk judul
    print(i.get_text())
```

```
Fakultas Teknologi Maju dan Multidisiplin
Program Studi Kami
```

```
[2] from bs4 import BeautifulSoup

    response = requests.get('https://detik.com')
    rawhtml = response.text
    soup = BeautifulSoup(rawhtml, 'html.parser')
```

```
    #print(soup.find_all('a'))
    for i in soup.find_all('h2'):  #mencari semua text dengan tag <h1> biasa untuk judul
        print(i.get_text())
```

```
Viral Video Pemuda Palak-Cegat Mobil di Tengah Tol, Lakukan Ini Saat Berhadapan


Rayuan Cepat Untung Investasi Bodong


Hasil Liga Champions Tadi Malam: Juve Hajar Chelsea, Barca Dibantai
```

# Another example

```
import requests
from bs4 import BeautifulSoup

response = requests.get('https://www.bmkg.go.id/cuaca/prakiraan-cuaca.bmkg?Kota=Surabaya&AreaID=501306&Prov=35')
rawhtml = response.text
soup = BeautifulSoup(rawhtml, 'html.parser')
```

```
#print(soup.find_all('a'))
for i in soup.find_all('h2'):  #mencari semua text dengan tag <h1> biasa untuk judul
    print(i.get_text())
```

```
28°C
22:00 WIB
27°C
01:00 WIB
25°C
04:00 WIB
26°C
07:00 WIB
28°C
10:00 WIB
33°C
```

# Web Crawling

- Improvement of scraping
- Includes **crawler bots or Spider** (Internet bot that systematically browses the World Wide Web)
- Bots can follow links on starting urls, enabling to scrape all website content (not just 1 page)
- Scrapers + loops and ifs
- Famous library: **ScraPy**

# Scrapy Basics

- Pip install scrapy
- **Scrapy startproject <projectname> -** will create a project folder containing code templates.

```
scrapy genspider [-t template] <name> <domain>
```

```
$ scrapy genspider -l
Available templates:
  basic
  crawl
  csvfeed
  xmlfeed

$ scrapy genspider example example.com
Created spider 'example' using template 'basic'

$ scrapy genspider -t crawl scrapyorg scrapy.org
Created spider 'scrapyorg' using template 'crawl'
```

# Scrapy Basic Usage

- Scrapy crawl <spider name>

```
> scrapy crawl example -t json -o output.json
```

- Spiders need configuration….

# Parsing Scrapy Response and Configure Crawler

- Scrapy use **Xpath (XML) and CSS selector**, BeautifulSoup uses **HTML selector**

```
import scrapy


class BasicftmmSpider(scrapy.Spider):
    name = 'basicftmm'
    allowed_domains = ['ftmm.unair.ac.id']
    start_urls = ['http://ftmm.unair.ac.id/']

    def parse(self, response):
        for text in response.css('a'):
            yield{
                    'title':text.css('a::text').get(),
                    'link':text.css('a::attr(href)').get()
                }
        pass
```

# Cont'd

- But we only scraped 1 page ☹
- Let's configure the spider
- Ps: every websites has different CSS structures. **So, inspect them all!**

- **Try them first in scrapy shell before code!**

```
import scrapy


class BasicftmmcatSpider(scrapy.Spider):
    name = 'basicftmmcat'

    start_urls = ['https://ftmm.unair.ac.id/category/stmm/']

    def parse(self, response):
        for text in response.css('a'):
            yield{
                    'title':text.css('a::text').get(),
                    'link':text.css('a::attr(href)').get()

                }
        NEXT_PAGE_SELECTOR = '.page-numbers.current + a::attr(href)'
        next_page = response.css(NEXT_PAGE_SELECTOR).extract_first()
        if next_page:
            yield scrapy.Request(
            response.urljoin(next_page),
            callback=self.parse)
```

# Try in Scrapy Shell first ☺

```
C:\Users\Ikaqutami>scrapy shell https://ftmm.unair.ac.id/stmm/page/3/
2021-11-12 09:18:37 [scrapy.utils.log] INFO: Scrapy 2.5.1 started (bot: scrapybot)
2021-11-12 09:18:37 [scrapy.utils.log] INFO: Versions: lxml 4.6.4.0, libxml2 2.9.5, cssselect 1.1.0, parsel 1.6.0, w3lib 1.22.0, Twisted 21.7.0, Python 3.9.1 (tags/v3.9.1
e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)], pyOpenSSL 21.0.0 (OpenSSL 1.1.1l  24 Aug 2021), cryptography 35.0.0, Platform Windows-10-10.0.19041-SP0
2021-11-12 09:18:37 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.SelectReactor
2021-11-12 09:18:37 [scrapy.crawler] INFO: Overridden settings:
{'DUPEFILTER_CLASS': 'scrapy.dupefilters.BaseDupeFilter',
 'LOGSTATS_INTERVAL': 0}
2021-11-12 09:18:37 [scrapy.extensions.telnet] INFO: Telnet Password: da90d29c430e243d
2021-11-12 09:18:37 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole']
2021-11-12 09:18:38 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2021-11-12 09:18:38 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2021-11-12 09:18:38 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2021-11-12 09:18:38 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2021-11-12 09:18:38 [scrapy.core.engine] INFO: Spider opened
2021-11-12 09:18:40 [scrapy.downloadermiddlewares.redirect] DEBUG: Redirecting (301) to <GET http://ftmm.unair.ac.id> from <GET https://ftmm.unair.ac.id/stmm/page/3/>
2021-11-12 09:18:40 [scrapy.downloadermiddlewares.redirect] DEBUG: Redirecting (301) to <GET https://ftmm.unair.ac.id/> from <GET http://ftmm.unair.ac.id>
2021-11-12 09:18:42 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://ftmm.unair.ac.id/> (referer: None)
```

```
In [1]: response.css('.page-numbers.current + a::attr(href)').get()
Out[1]: 'https://ftmm.unair.ac.id/category/stmm/page/4/'

In [2]:
```

# Other method: Selenium

- In some cases that robots are not allowed, or specific user behavior is needed, we need browser emulation
- A package called Selenium can be used to emulate browser and do things human do!
- Selenium is an open-source web automation tool
- Primarily for automated web app test
- Advantage: able to scrap images, videos, etc.
- Try it yourself ☺

# Class Activity

- Try scraping and crawling!
- Choose your own website (which interests you, e.g., cookpad, Tokopedia, ps store, news etc.)
- Show what you've got ☺