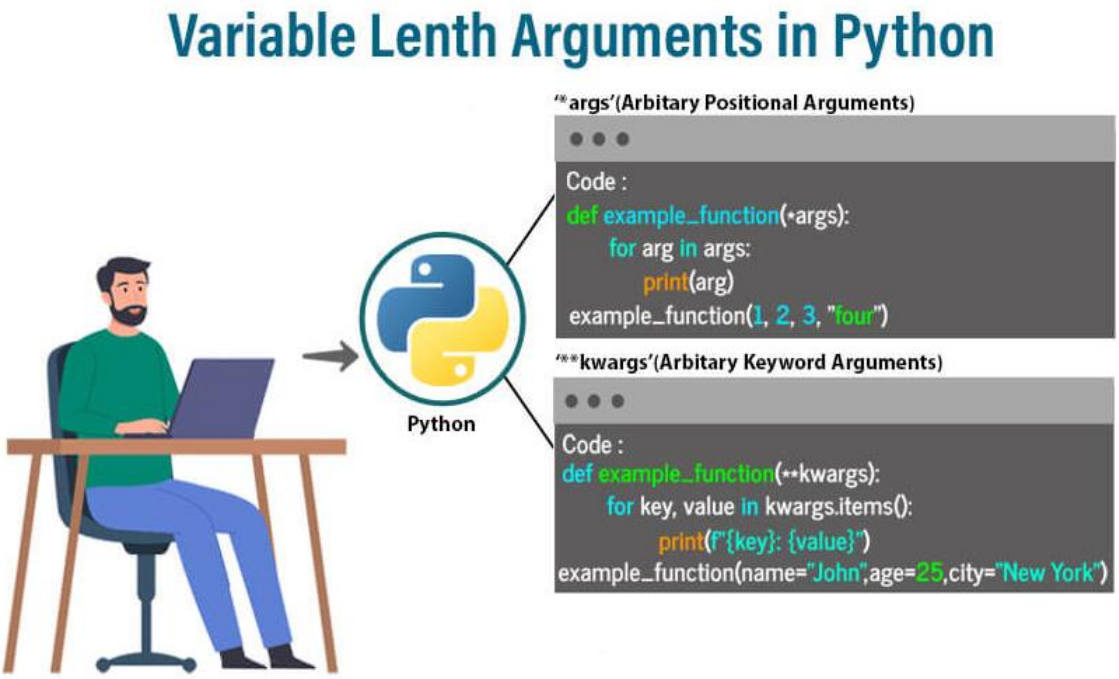
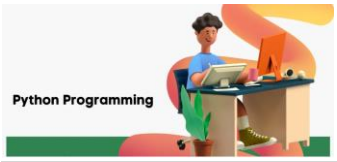
LECTURE

FOURTEEN

Variable Length Arguments in Python (*args and **kwargs)



- In Python, we can pass a variable number of arguments to a function using special symbols. There are two special symbols:
  1. *args (Non Keyword Arguments)
  2. **kwargs (Keyword Arguments)
- We use *args and **kwargs as an argument when we are unsure about the number of arguments to pass in the functions.
- *args passes variable number of non-keyworded arguments and on which operation of the tuple can be performed.
- **kwargs passes variable number of keyword arguments dictionary to function on which operation of a dictionary can be performed.
- *args and **kwargs make the function flexible.



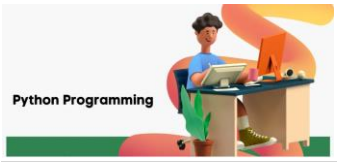
- using *args to pass the variable length arguments to the function, we should use an asterisk * before the parameter name to pass variable length arguments. The arguments are passed as a tuple and these passed arguments make tuple inside the function with same name as the parameter excluding asterisk *.

***args to pass the variable length arguments to the function**

```
def adder(*num):
    print("Data type of argument:",type(num))
    sum = 0
    for n in num:
        sum = sum + n
    print("Sum:",sum)
adder(3,5)
adder(4,5,6,7)
adder(1,2,3,5,6)
```

Output

```
<class 'tuple'>
Sum: 8
Sum: 22
Sum: 17
```



- Using **kwargs to pass the variable keyword arguments to the function, we use the double asterisk ** before the parameter name to denote this type of argument. The arguments are passed as a dictionary and these arguments make a dictionary inside function with name same as the parameter excluding double asterisk **.

kwargs **to pass the variable length arguments to the function

```
def intro(**data):
    print("\nData type of argument:",type(data))
    for key, value in data.items():
        print("{} is {}".format(key,value))
intro(Firstname="Sita", Lastname="Sharma", Age=22, Phone=1234567890)
intro(Firstname="John", Lastname="Wood", Email="johnwood@nomail.com")
```

Output

Data type of argument: <class 'dict'>

Firstname is Sita

Lastname is Sharma

Age is 22

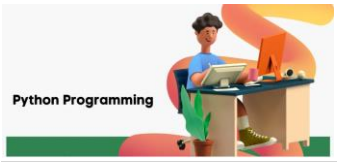
Phone is 1234567890

Data type of argument: <class 'dict'>

Firstname is John

Lastname is Wood

Email is johnwood@nomail.com



Python pip

- pip is the standard package manager for Python. We can use pip to install additional packages that are not available in the Python standard library.
- Suppose we want to install requests, a popular HTTP library for Python. The pip install command:

  pip install requests

  The pip uninstall command:

  pip uninstall requests
- The pip list command can be used to list all the available packages in the current Python environment.

  pip list
- The pip freeze command displays the packages and their version in the format of the requirements file.

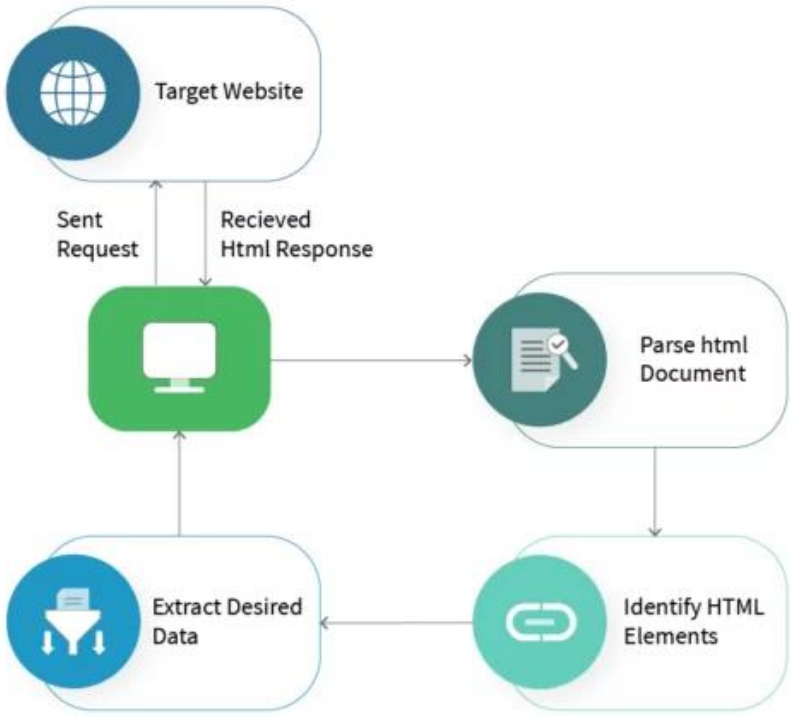
  pip freeze > requirements.txt
- We can install all these packages and their dependencies by using a single command in pip.

  pip install -r requirements.txt

Trend Projects Based on Python

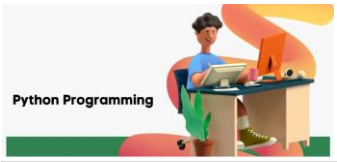
Case Study: Web Scraping

- The internet is an endless source of information, and for many data-driven tasks, accessing this information is critical. For this reason, **Web Scraping**, the practice of extracting data from websites, has become an increasingly important tool for machine learning developers, data analysts, researchers, and businesses alike.

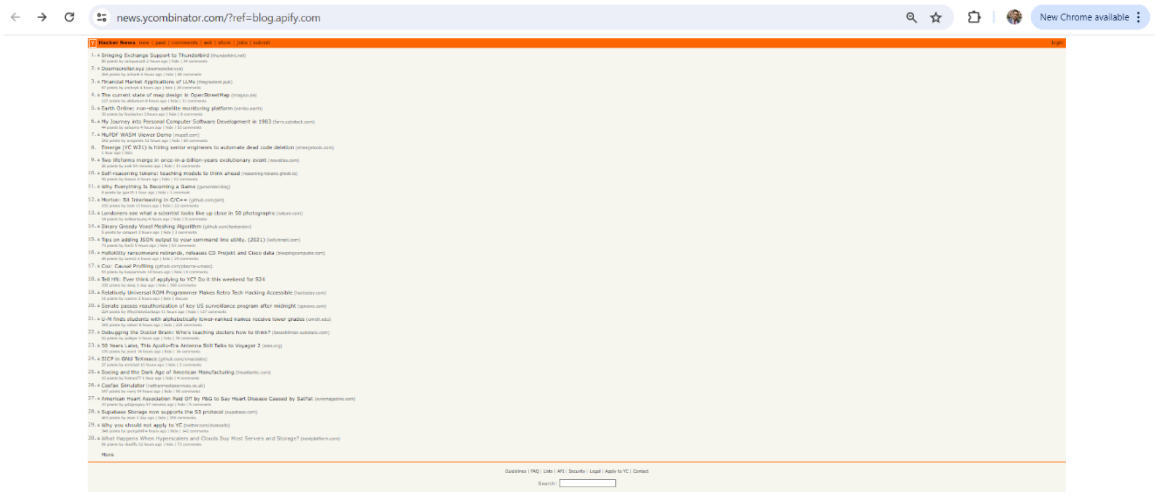


- One of the most popular Python web scraping tools are **requests**, **Beautiful Soup**, etc.
- requests is a library that manages HTTP communication and the retrieval of HTML documents.
- Beautiful Soup is a Python library that allows you to parse HTML and XML documents. Beautiful Soup makes it easy to extract specific pieces of information from web pages.
- Let's use pip to install requests and Beautiful Soup.

```
pip install requests beautifulsoup4
```



- So, for starters, we need an HTML document. For instance, **Hacker News** ("https://news.ycombinator.com/")



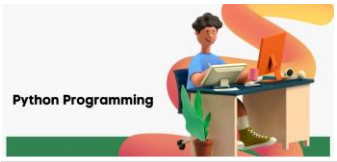
- For that purpose, we will be using Python's Requests package and fetch the main page of HackerNews.
- We now need to parse the HTML and load it into a BS4 structure.
- Now, it's very handy and allows us to easily access many useful pieces of information.
- let's say I want to extract all links in this page.

fetch data from Hacker News

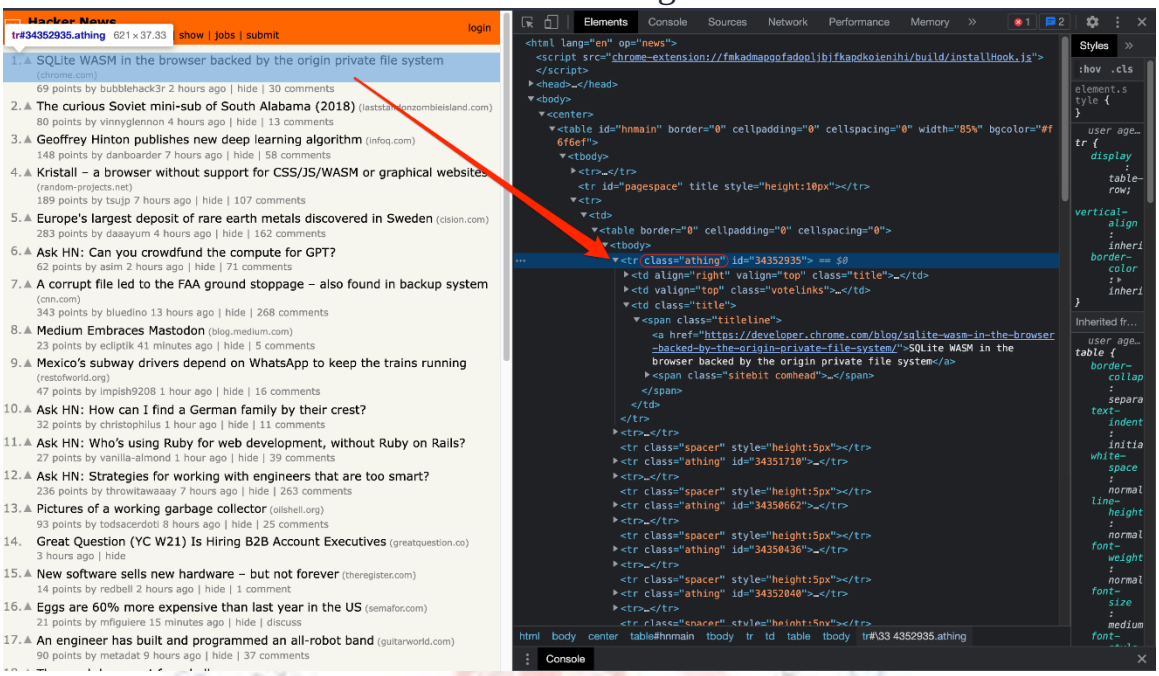
```
import requests
from bs4 import BeautifulSoup
response = requests.get("https://news.ycombinator.com/")
content = response.content
soup = BeautifulSoup(content, 'html.parser')
print(soup.title)  # The title tag of the page
nb_links = len(soup.find_all('a'))        # All links in the page

print(f"There are {nb_links} links in this page")
print(soup.get_text())  # Text from the page

all_hrefs = [a.get('href') for a in soup.find_all('a')]
print(all_hrefs)
```



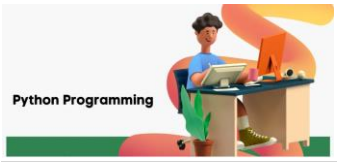
- Before we select an element, let's use the developer tools to inspect the page and find what selectors we need to use to target the data we want to extract.



- select all elements containing the athing class and save them to a variable named articles. Let's loop through each article and print its text contents.

fetch specific class from Hacker News

```
import requests
from bs4 import BeautifulSoup
response = requests.get("https://news.ycombinator.com/")
content = response.content
soup = BeautifulSoup(response.content, 'html.parser')
articles = soup.find_all(class_='athing')
for article in articles:
    print(article.text)
```



♠ Next, let's create a while loop that continues scraping until the scraper reaches the last page. Within the loop, we will send a GET request to the current page of Hacker News, so we can execute the rest of our script to extract the **URL, title,** and **rank** of each article and store the data in a dictionary with keys **"URL"**, **"title"**, and **"rank"**. We will then append the dictionary to the output list.

Continues scraping until the scraper reaches the last page.

```
import requests
from bs4 import BeautifulSoup

scraping_hn = True
page = 1
output = []

while scraping_hn:
    response = requests.get(f'https://news.ycombinator.com/?p={page}')
    content = response.content
    soup = BeautifulSoup(content, 'html.parser')
    articles = soup.find_all(class_='athing')
    for article in articles:
        data = {
            'URL': article.find(class_='titleline').find('a').get('href'),
            'title': article.find(class_='titleline').getText(),
            'rank': article.find(class_='rank').getText().replace('.', ''),
        }
        output.append(data)
    next_page = soup.find(class_='morelink')

    if next_page is not None:
        page += 1
    else:
        scraping_hn = False
        print(f'Finished scraping! Scraped a total of {len(output)} items.')
print(output)
```