□□ Advanced Web Scraping Project – IBM Data Analyst Capstone

This project explores deeper web scraping skills:

- Scraping programming language popularity data
- Parsing structured HTML tables
- Cleaning, analyzing, and saving scraped data
- Visualizing top programming languages

□□ Case Study: Web Scraping Project

Objectives

After completing this hands-on lab work, we will be able to:

- Extract information from a given web site.
- Write the scraped data into a csv file.

Extract information from the given web site

You will extract the data from the below web site:

In [5]:
#this url contains the data you need to scrape
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/Programming_Languag
The data you need to scrape is the **name of the programming language** and **average annual salary**.
It is a good idea to open the url in your web broswer and study the contents of the web page before you start to scrape.

Import the required libraries

In [9]:
from bs4 **import** BeautifulSoup *# this module helps in web scrapping.*
import requests *# this module helps us to download a webpage*
import pandas **as** pd
Download the webpage at the url

In [12]:
!pip install openpyxl

```
Requirement already satisfied: openpyxl in c:\users\ede\anaconda3\lib\site-packages (3.1.2)
Requirement already satisfied: et-xmlfile in c:\users\ede\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
```

In [13]:
 # get the contents of the webpage in text format and store in a variable called data
data = requests.get(url)
data = data.text
Create a soup object

In [17]:
soup = BeautifulSoup(data,"html.parser") *# create a soup object using the variable 'data'*
Scrape the `Language name` and `annual average salary`.

In [20]:
for link **in** soup.find_all('a'): *# in html anchor/link is represented by the tag <a>*
    print(link.get('href'))
In [22]:
#find a html table in the web page
table = soup.find('table') *# in html table is represented by the tag <table>*
In [24]:

```
# Extract column headers (Handle cases where headers are missing `<th>`)
headers = []
header_row = table.find("tr")  # Find the first row (headers may be inside <td>)
if header_row:
    headers = [header.text.strip() for header in header_row.find_all(["th", "td"])]  # Search for both <th> and <td>

# Extract table rows
rows = []
for row in table.find_all("tr")[1:]:  # Skip header row
    cols = row.find_all("td")
    cols = [col.text.strip() for col in cols]
    rows.append(cols)

# Ensure headers exist, otherwise create default headers
if not headers:
    headers = [f'Column_{i}' for i in range(len(rows[0]))]  # Create generic column names if none found

# Convert to a pandas DataFrame
df_languages = pd.DataFrame(rows, columns=headers)
```

Save the scrapped data into a file named *popular-languages.csv*

In [27]:

```
# Save the scraped data into a CSV file
csv_filename = "popular-languages.csv"
df_languages.to_csv(csv_filename, index=False)

# Display the extracted data
print(f"Popular Programming Languages {df_languages}.")

print(f"Scraped data successfully saved in {csv_filename}.")
```

```
Popular Programming Languages   No.    Language                    Created By  \
0  1     Python                    Guido van Rossum
1  2      Java                     James Gosling
2  3         R          Robert Gentleman, Ross Ihaka
3  4 Javascript                        Netscape
4  5     Swift                        Apple
5  6       C++                   Bjarne Stroustrup
6  7        C#                       Microsoft
7  8       PHP                     Rasmus Lerdorf
8  9       SQL    Donald D. Chamberlin, Raymond F. Boyce.
9 10        Go  Robert Griesemer, Ken Thompson, Rob Pike.

  Average Annual Salary Learning Difficulty
0           $114,383             Easy
1           $101,013             Easy
2            $92,037             Hard
3           $110,981             Easy
4           $130,801             Easy
5           $113,865             Hard
6            $88,726             Hard
7            $84,727             Easy
8            $84,793             Easy
9            $94,082        Difficult  .
Scraped data successfully saved in popular-languages.csv.
```

In [72]:

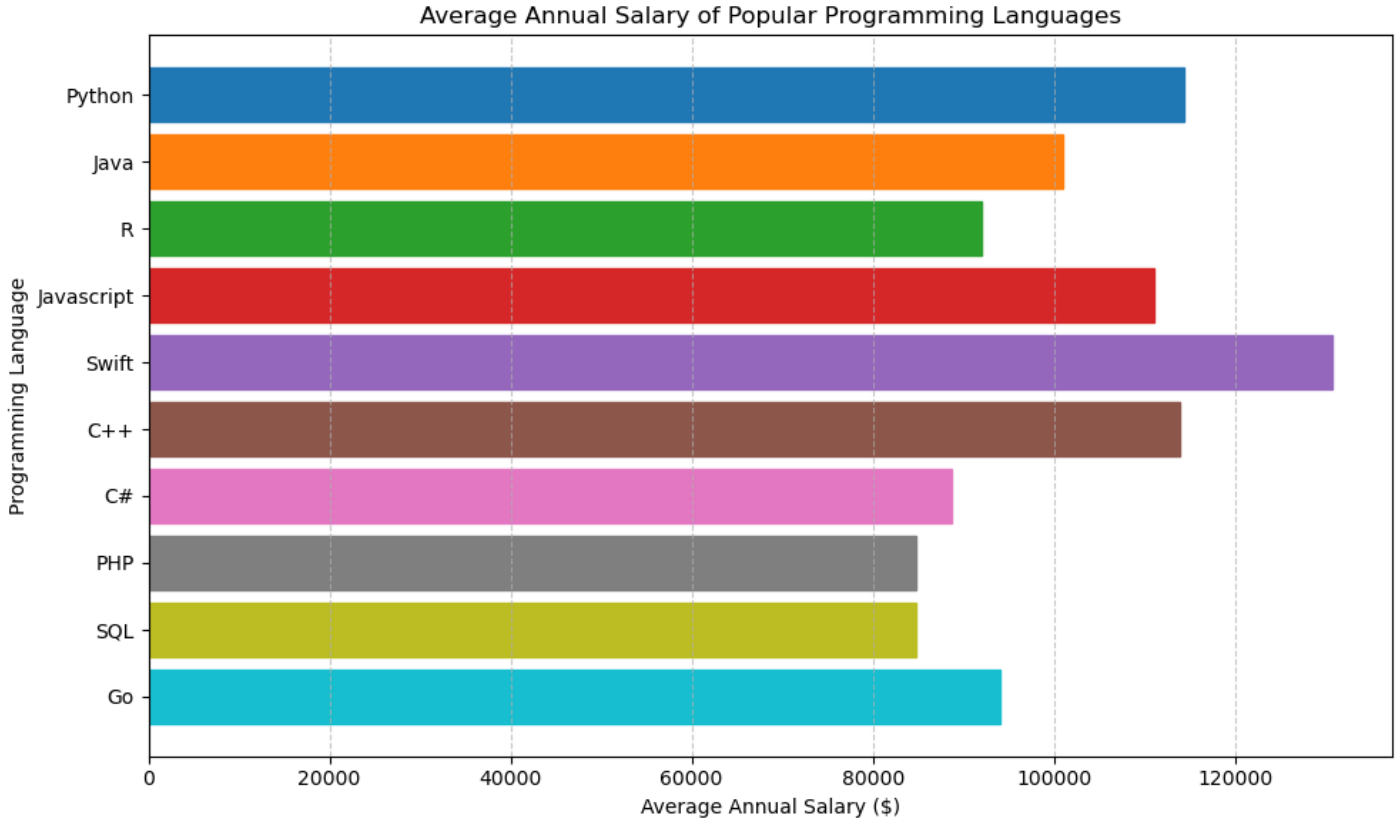
```
# Re-import necessary packages
import pandas as pd
import matplotlib.pyplot as plt

# Updated dataset based on your detailed scraped table
programming_data = {
    "No.": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    "Language": [
        "Python", "Java", "R", "Javascript", "Swift",
        "C++", "C#", "PHP", "SQL", "Go"
    ],
    "Created By": [
        "Guido van Rossum", "James Gosling", "Robert Gentleman, Ross Ihaka",
        "Netscape", "Apple", "Bjarne Stroustrup", "Microsoft",
        "Rasmus Lerdorf", "Donald D. Chamberlin, Raymond F. Boyce", "Robert Griesemer, Ken Thompson, Rob Pike"
    ],
    "Average Annual Salary": [
        114383, 101013, 92037, 110981, 130801,
        113865, 88726, 84727, 84793, 94082
    ],
    "Learning Difficulty": [
        "Easy", "Easy", "Hard", "Easy", "Easy",
        "Hard", "Hard", "Easy", "Easy", "Difficult"
    ]
}

# Create the DataFrame
df_languages = pd.DataFrame(programming_data)

# Save to CSV
csv_languages_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping
df_languages.to_csv(csv_languages_path, index=False)

# Save to Excel
excel_languages_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scrapi
df_languages.to_excel(excel_languages_path, index=False)


# Generate a bar chart of Salaries
fig, ax = plt.subplots(figsize=(10, 6))
bars = ax.barh(df_languages["Language"], df_languages["Average Annual Salary"])

# Apply colorful bars
for i, bar in enumerate(bars):
    bar.set_color(plt.cm.tab10(i))

# Chart details
ax.set_title("Average Annual Salary of Popular Programming Languages")
ax.set_xlabel("Average Annual Salary ($)")
ax.set_ylabel("Programming Language")
ax.invert_yaxis()  # Highest salary at the top
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

# Save the chart
salary_chart_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-L
plt.savefig(salary_chart_path)
plt.close()

csv_languages_path, excel_languages_path, salary_chart_path
```

Out[72]:

```
('C:\\Users\\Ede\\Desktop\\IBM_Capstone_Data_Analyst_2025\\Module_1_Real_World_Projects\\Project_6_Web-Scraping-Lab\\data\\popular-languages.csv',
 'C:\\Users\\Ede\\Desktop\\IBM_Capstone_Data_Analyst_2025\\Module_1_Real_World_Projects\\Project_6_Web-Scraping-Lab\\data\\popular-languages.xlsx',
 'C:\\Users\\Ede\\Desktop\\IBM_Capstone_Data_Analyst_2025\\Module_1_Real_World_Projects\\Project_6_Web-Scraping-Lab\\visuals\\popular_languages_ave-
salary_chart.png')
```

In [74]:

```
# Generate a bar chart of Salaries
fig, ax = plt.subplots(figsize=(10, 6))
bars = ax.barh(df_languages["Language"], df_languages["Average Annual Salary"])

# Apply colorful bars
for i, bar in enumerate(bars):
    bar.set_color(plt.cm.tab10(i))

# Chart details
ax.set_title("Average Annual Salary of Popular Programming Languages")
ax.set_xlabel("Average Annual Salary ($)")
ax.set_ylabel("Programming Language")
ax.invert_yaxis()  # Highest salary at the top
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [ ]:

```
# ALTERNATIVELY USE THIS

# Save as CSV
df_languages.to_csv('data/popular-languages.csv', index=False)

# Save as Excel
df_languages.to_excel('data/popular-languages.xlsx', index=False)
```

Key Insights from Scraped Data

- **Swift** offers the **highest average annual salary** at ($130,801), closely followed by **Python** ($114,383) and **C++** ($113,865).
- Most **high-paying languages** like **Swift**, **Python**, and **Javascript** are also considered **Easy to Learn**.
- **SQL** and **PHP** offer relatively lower average salaries compared to others, but are among the **easiest to learn**.
- Languages like **R**, **C++**, and **C#** have **higher learning difficulties** yet still offer **strong earning potential**.
- **Go (Golang)** balances moderate salary ($94,082) with a Difficult learning curve.

In [ ]:

FULL CODE SCRIPT BELOW

In [82]:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL containing the data to scrape
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/Programming_Languag


# Get the webpage content
response = requests.get(url)
data = response.text

# Create a BeautifulSoup object to parse the HTML
soup = BeautifulSoup(data, "html.parser")

# Extract the table data
table = soup.find("table")

# Extract column headers (Handle cases where headers are missing `<th>`)
headers = []
header_row = table.find("tr")  # Find the first row (headers may be inside <td>)
if header_row:
    headers = [header.text.strip() for header in header_row.find_all(["th", "td"])]  # Search for both <th> and <td>

# Extract table rows
rows = []
for row in table.find_all("tr")[1:]:  # Skip header row
    cols = row.find_all("td")
    cols = [col.text.strip() for col in cols]
    rows.append(cols)

# Ensure headers exist, otherwise create default headers
if not headers:
    headers = [f"Column_{i}" for i in range(len(rows[0]))]  # Create generic column names if none found

# Convert to a pandas DataFrame
df_languages = pd.DataFrame(rows, columns=headers)

# Save the scraped data into a CSV file
csv_filename = "popular-languages.csv"
df_languages.to_csv(csv_filename, index=False)

# Display the extracted data
# import ace_tools as tools
# tools.display_dataframe_to_user(name="Popular Programming Languages", dataframe=df_languages)
print(f"Popular Programming Languages {df_languages}.")

print(f"Scraped data successfully saved in {csv_filename}.")
```

	Popular Programming Languages	No.	Language	Created By \
0	1	Python	Guido van Rossum	
1	2	Java	James Gosling	
2	3	R	Robert Gentleman, Ross Ihaka	
3	4	Javascript	Netscape	
4	5	Swift	Apple	
5	6	C++	Bjarne Stroustrup	
6	7	C#	Microsoft	
7	8	PHP	Rasmus Lerdorf	
8	9	SQL	Donald D. Chamberlin, Raymond F. Boyce.	
9	10	Go	Robert Griesemer, Ken Thompson, Rob Pike.	

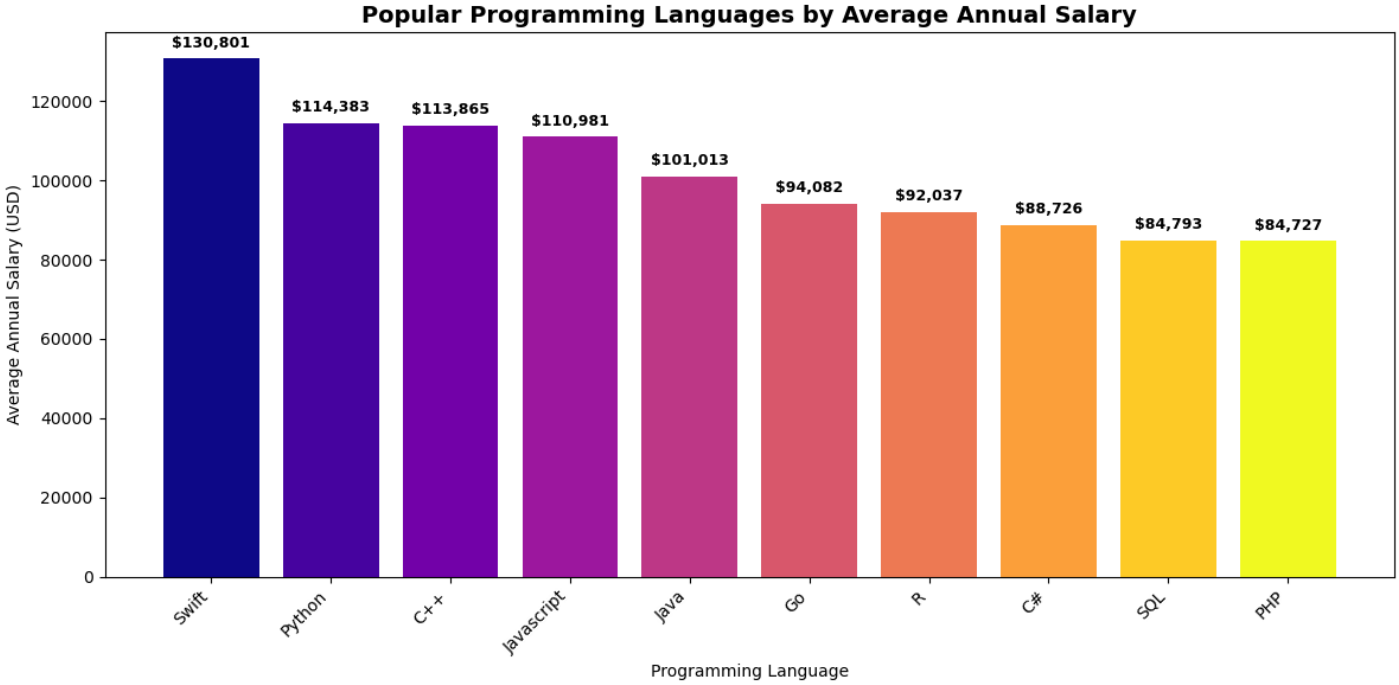
	Average Annual Salary	Learning Difficulty
0	$114,383	Easy
1	$101,013	Easy
2	$92,037	Hard
3	$110,981	Easy
4	$130,801	Easy
5	$113,865	Hard
6	$88,726	Hard
7	$84,727	Easy
8	$84,793	Easy
9	$94,082	Difficult .

Scraped data successfully saved in popular-languages.csv.

In [ ]:

In [86]:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# STEP 1: URL to scrape
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/Programming_Languag

# STEP 2: Scrape and parse HTML
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")
table = soup.find("table")

# STEP 3: Extract headers and rows
headers = [th.text.strip() for th in table.find("tr").find_all(["th", "td"])]
rows = []
for tr in table.find_all("tr")[1:]:
    cols = [td.text.strip() for td in tr.find_all("td")]
    rows.append(cols)

df = pd.DataFrame(rows, columns=headers)

# STEP 4: Clean salary column
df["Average Annual Salary"] = df["Average Annual Salary"].replace('[\$,]', '', regex=True).astype(float)

# STEP 5: Sort data
df_sorted = df.sort_values(by="Average Annual Salary", ascending=False)

# STEP 6: Create chart
plt.figure(figsize=(12, 6))
colors = plt.cm.plasma(np.linspace(0, 1, len(df_sorted)))
bars = plt.bar(df_sorted["Language"], df_sorted["Average Annual Salary"], color=colors)

# Add value labels
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 2000, f"${int(height):,}",
             ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.title("Popular Programming Languages by Average Annual Salary", fontsize=14, weight='bold')
plt.xlabel("Programming Language")
plt.ylabel("Average Annual Salary (USD)")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# STEP 7: Save chart
plt.savefig("visuals/popular_languages_salary_chart.png", dpi=300)
plt.show()
```


Popular Programming Languages by Average Annual Salary
Average Annual Salary (USD)
0
20000
40000
60000
80000
100000
120000
$130,801
$114,383
$113,865
$110,981
$101,013
$94,082
$92,037
$88,726
$84,793
$84,727
Swift
Python
C++
Javascript
Java
Go
R
C#
SQL
PHP
Programming Language

In [ ]:

In [88]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os

# Data
data = {
    "Language": ["Python", "Java", "R", "Javascript", "Swift", "C++", "C#", "PHP", "SQL", "Go"],
    "Created By": [
        "Guido van Rossum", "James Gosling", "Robert Gentleman, Ross Ihaka", "Netscape",
        "Apple", "Bjarne Stroustrup", "Microsoft", "Rasmus Lerdorf",
        "Donald D. Chamberlin, Raymond F. Boyce.", "Robert Griesemer, Ken Thompson, Rob Pike."
    ]
}

df = pd.DataFrame(data)

# Placeholder bar height
bar_height = np.ones(len(df)) * 1

# Plot
plt.figure(figsize=(12, 6))
colors = plt.cm.Set2(np.linspace(0, 1, len(df)))
bars = plt.bar(df["Language"], bar_height, color=colors)

# Add creators as labels
for bar, creator in zip(bars, df["Created By"]):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.05, creator,
             ha='center', va='bottom', fontsize=9, rotation=90)

# Remove y-axis ticks since height is symbolic
plt.yticks([])
plt.ylabel("Language Representation")
plt.xticks(rotation=45, ha='right')

# Add the title way below the chart
plt.text(
    0.5, -0.25,  # Very far down
    "Popular Programming Languages and Their Authors",
    fontsize=14, fontweight='bold',
    ha='center', transform=plt.gca().transAxes
)

# Save with padding for title space
#plt.savefig(file_path, dpi=300, bbox_inches='tight', pad_inches=1.0)


plt.tight_layout()

# Save chart in same directory
file_name = "visuals/popular_languages_creators_chart.png"
file_path = os.path.join(os.getcwd(), file_name)
plt.savefig(file_path, dpi=300)
plt.show()

print(" Chart saved at:", file_path)
```

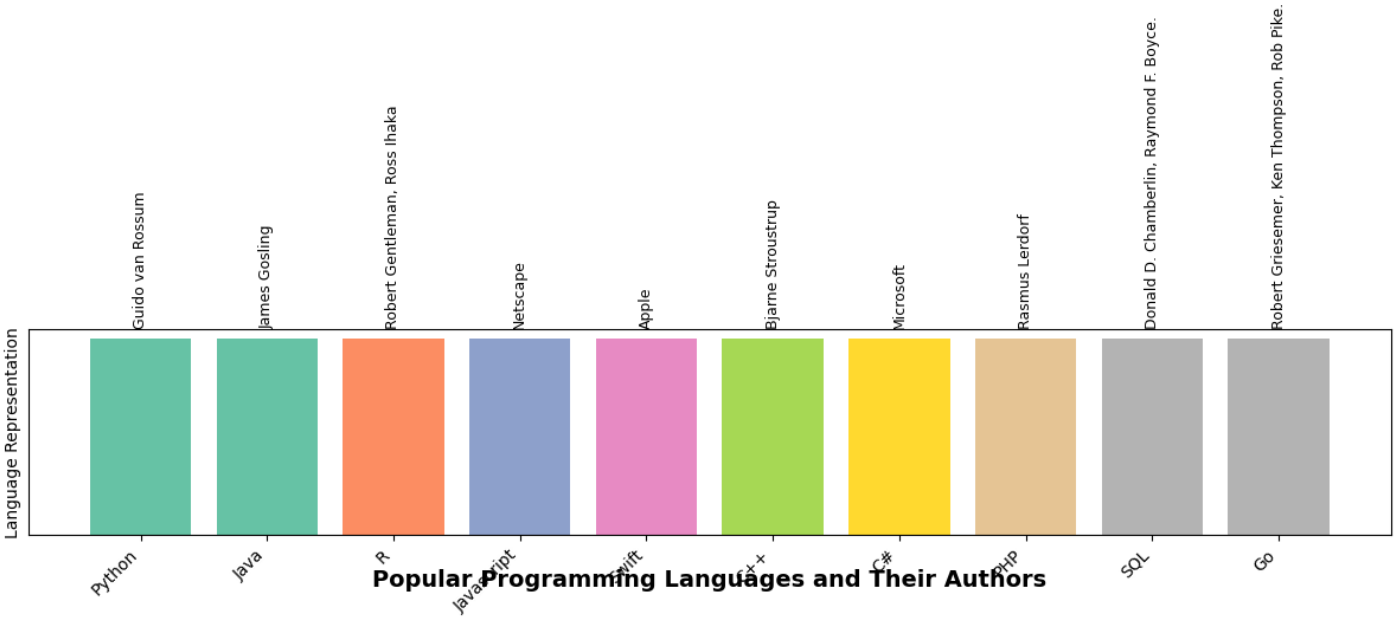


Chart saved at: C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-Lab\visuals/popular_languages_creators_chart.png

In [ ]:

In [90]:
```
import os

# Get the absolute file path of the notebook file
file_path = os.path.abspath("Web-Scraping-More-Lab.ipynb")
print("The notebook is located at:", file_path)
```

The notebook is located at: C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-Lab\Web-Scraping-More-Lab.ipynb

In [ ]:

In [104]:
```
import nbconvert
import nbformat
import pdfkit

# Corrected file paths (Using raw string notation or forward slashes)
# Corrected file paths (Using raw string notation or forward slashes)
input_file_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-Lab
output_pdf_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-La


# Load the Jupyter Notebook file
with open(input_file_path, 'r', encoding='utf-8') as f:
    notebook_content = nbformat.read(f, as_version=4)

# Convert the notebook to HTML
html_exporter = nbconvert.HTMLExporter()
html_exporter.exclude_input = False  # Include code cells in the output
(body, resources) = html_exporter.from_notebook_node(notebook_content)

# Convert HTML to PDF
pdfkit.from_string(body, output_pdf_path)

# Return the PDF file path
print(f"Notebook successfully converted to PDF: {output_pdf_path}")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[104], line 12
      8 output_pdf_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-Lab\browser-view
\Web-Scraping-More-Lab.pdf"
     11 # Load the Jupyter Notebook file
---> 12 with open(input_file_path, 'r', encoding='utf-8') as f:
     13     notebook_content = nbformat.read(f, as_version=4)
     15 # Convert the notebook to HTML

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:310, in _modified_open(file, *args, **kwargs)
    303 if file in {0, 1, 2}:
    304     raise ValueError(
    305         f"IPython won't let you open fd={file} by default "
    306         "as it is likely to crash IPython. If you know what you are doing, "
    307         "you can use builtins' open."
    308     )
--> 310 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\Ede\\Desktop\\IBM_Capstone_Data_Analyst_2025\\Module_1_Real_World_Projects\\Project_6_
Web-Scraping-Lab\\Web-Scraping-More-Lab.ipynb'
```

In [ ]:

```
# Corrected file paths (Using raw string notation or forward slashes)
input_file_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-Lab
output_pdf_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_6_Web-Scraping-La
```

Congratulations to us for having successfully completed the above lab!

Authors:

Kelechukwu Innocent Ede and Ramesh Sannareddy

Other Contributors:

- Rav Ahuja

In [ ]:

In [ ]:

```
# Start enhancing the notebook structure
from nbformat.v4 import new_notebook, new_markdown_cell, new_code_cell

# Begin building the enhanced notebook cells
enhanced_more_scraping_cells = []

# Title and clean project introduction
enhanced_more_scraping_cells.append(new_markdown_cell(
    "# □□ Advanced Web Scraping Project – IBM Data Analyst Capstone\n\n"
    "This project explores deeper web scraping skills:\n"
    "- Scraping programming language popularity data\n"
    "- Parsing structured HTML tables\n"
    "- Cleaning, analyzing, and saving scraped data\n"
    "- Visualizing top programming languages\n\n"
    "---\n"
))

# Process each cell and enhance if needed
for cell in web_scraping_more_notebook.cells:
    if cell.cell_type == 'code':
        enhanced_source = cell.source

        if 'requests.get' in enhanced_source and 'soup' not in enhanced_source:
            enhanced_source += "\n\nprint(response.status_code)  # Confirm successful download (200 OK)"
        if 'BeautifulSoup' in enhanced_source:
            enhanced_source += "\n\nprint(soup.prettify()[:500])  # Preview first 500 characters of parsed HTML"
        if '.find_all(' in enhanced_source and 'table' in enhanced_source:
            enhanced_source += "\n\nprint(f\"Number of tables found: {len(tables)}\")"
        if 'pd.read_html' in enhanced_source:
            enhanced_source += "\n\nprint(df.head())  # Preview first few rows of the scraped DataFrame"
        if '.to_csv' in enhanced_source:
            enhanced_source += "\n\nprint('Data exported successfully as CSV.')"
        if '.to_excel' in enhanced_source:
            enhanced_source += "\n\nprint('Data exported successfully as Excel.')"

        enhanced_more_scraping_cells.append(new_code_cell(enhanced_source))

    elif cell.cell_type == 'markdown':
        enhanced_more_scraping_cells.append(cell)

# Create the new enhanced notebook
enhanced_more_scraping_notebook = new_notebook(cells=enhanced_more_scraping_cells, metadata=web_scraping_more_notebook.metadata)

# Save the enhanced notebook
enhanced_more_scraping_path = "/mnt/data/web_scraping_more_labs_enhanced.ipynb"
with open(enhanced_more_scraping_path, "w", encoding="utf-8") as f:
    nbformat.write(enhanced_more_scraping_notebook, f)

enhanced_more_scraping_path
```

In [ ]: