Github DATA visualisation / analysis

YouCode - Bentaleb SAAD

In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
import seaborn as sns
from bokeh.io import output_notebook
output_notebook()
from plot_utils import plot_barplot, draw_points_graph
from bokeh.plotting import figure, show
from bokeh.palettes import Category20c, HighContrast3
from bokeh.transform import cumsum
import ast
```

BokehJS 3.2.0 successfully loaded.

Loading Data

In [ ]:
```
# Loading data from the csv files
repositories_df = pd.read_csv('data/repos.csv')
commits_df = pd.read_csv('data/commits.csv')
languages_df = pd.read_csv('data/languages.csv')
```

In [ ]:
```
print('Repositories:')
print(repositories_df.info())
print('\nCommits:')
print(commits_df.info())
print('\nLanguages:')
print(languages_df.info())
```

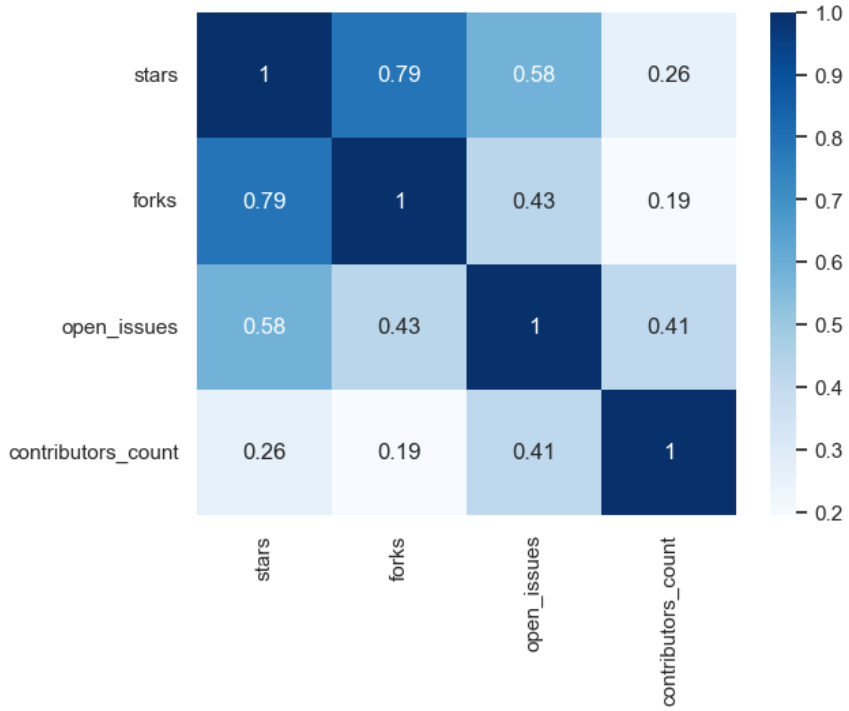
```
Repositories:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15490 entries, 0 to 15489
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   name                 15490 non-null  object
 1   url                  15490 non-null  object
 2   description          15490 non-null  object
 3   stars                15490 non-null  int64 
 4   created_at           15490 non-null  object
 5   language             15490 non-null  object
 6   forks                15490 non-null  int64 
 7   watchers             15490 non-null  int64 
 8   open_issues          15490 non-null  int64 
 9   owner                15490 non-null  object
 10  contributors_count   15490 non-null  int64 
 11  contributors         15490 non-null  object
 12  contributions_count  15490 non-null  object
 13  languages            15490 non-null  object
 14  bites_of_code        15490 non-null  object
dtypes: int64(5), object(10)
memory usage: 1.8+ MB
None

Commits:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 324635 entries, 0 to 324634
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   url             324635 non-null  object
 1   id              324635 non-null  object
 2   author_name     324635 non-null  object
 3   author_date     324635 non-null  object
 4   committer_name  324635 non-null  object
 5   committer_date  324635 non-null  object
 6   repo_name       324635 non-null  object
 7   languages       324635 non-null  object
dtypes: object(8)
memory usage: 19.8+ MB
None

Languages:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15490 entries, 0 to 15489
Columns: 289 entries, url to NSIS
dtypes: int64(288), object(1)
memory usage: 34.2+ MB
None
```

Data topography

i - Variables correlation (Is there a variable in the dataset that can be replaces by another ?)

In [ ]:
```
# correlation between variables
corr = repositories_df[['stars', 'forks', 'open_issues', 'contributors_count']].corr()
# plot the heatmap
sns.heatmap(corr, annot=True, cmap='Blues')
```

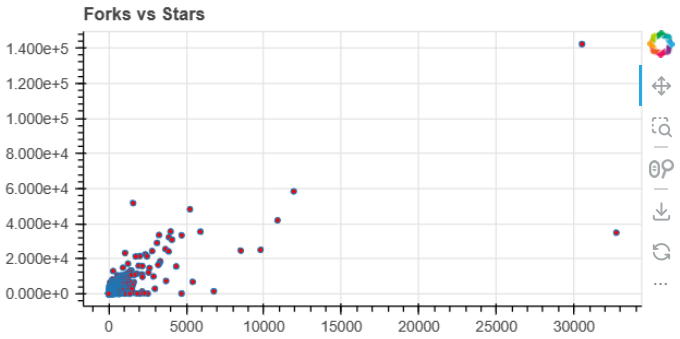
Out[ ]:
```
<Axes: >
```



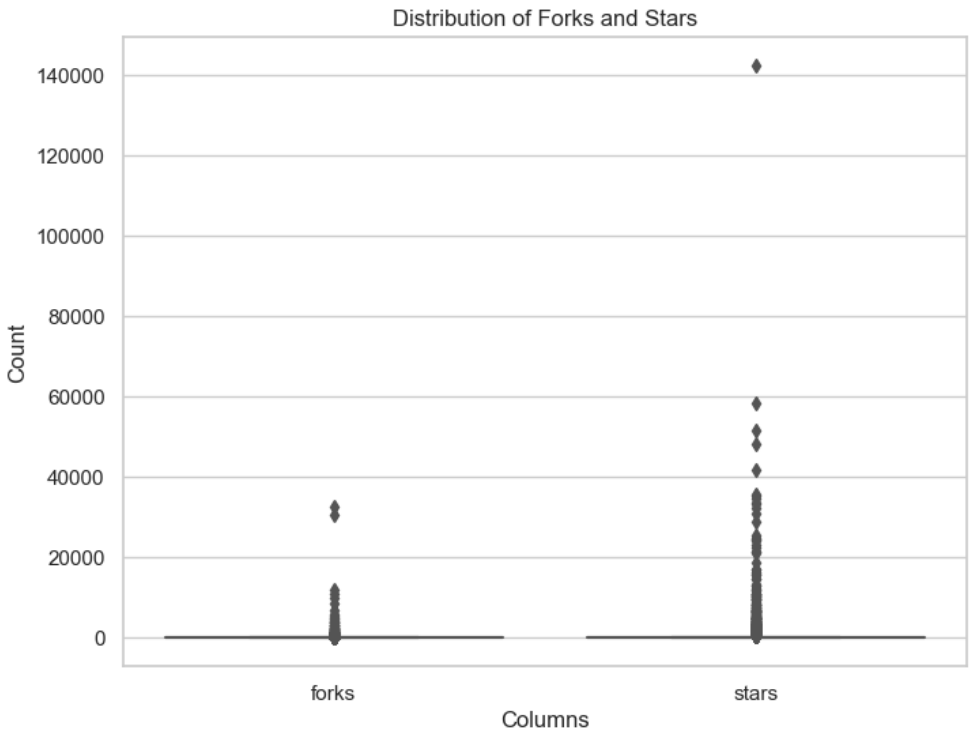
In [ ]:
```
# r squared value
print('R squared value between stars and forks: ', round(r2_score(repositories_df['stars'], repositories_df['forks']), 2))
```

```
R squared value between stars and forks:  0.3
```

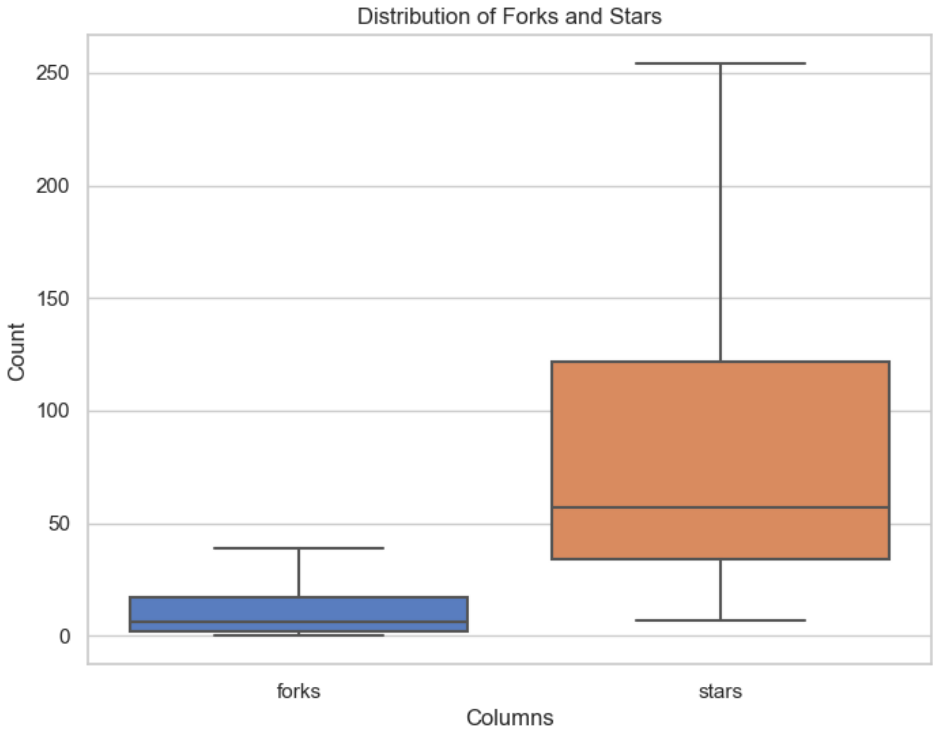
In [ ]:
```
draw_points_graph(repositories_df['forks'], repositories_df['stars'], average=False,title="Forks vs Stars")
```



In [ ]:
```
from matplotlib import pyplot as plt
# Create a figure and axes
fig, ax = plt.subplots(figsize=(8, 6))

# Create the boxplot using Seaborn
sns.boxplot(data=repositories_df[['forks', 'stars']], ax=ax)

# Set the labels and title
ax.set_xlabel('Columns')
ax.set_ylabel('Count')
ax.set_title('Distribution of Forks and Stars')

# Display the plot
plt.show()
```



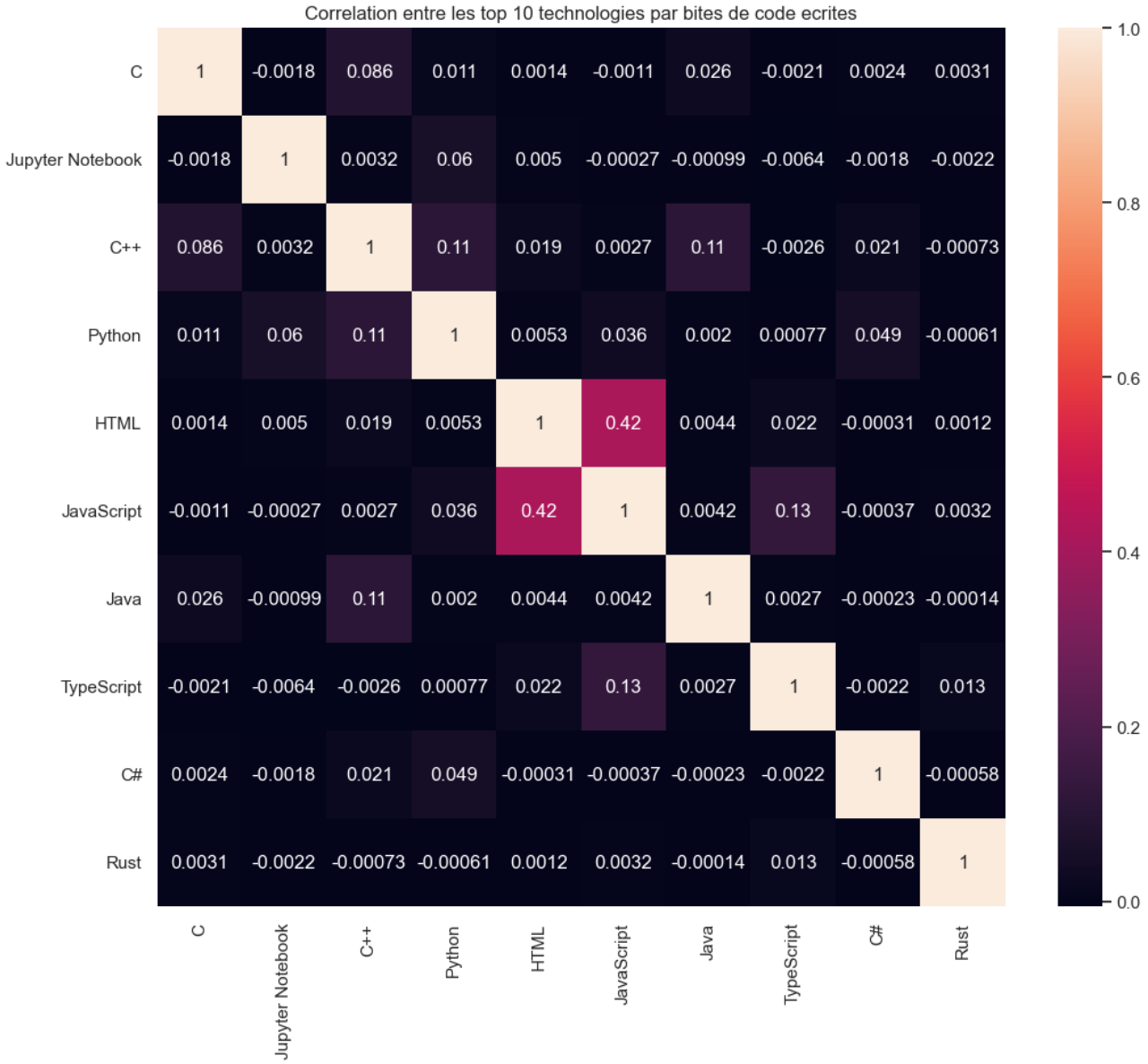
In [ ]:
```
# dont show the outliers
fig, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(data=repositories_df[['forks', 'stars']], ax=ax, showfliers=False)
ax.set_xlabel('Columns')
ax.set_ylabel('Count')
ax.set_title('Distribution of Forks and Stars')
plt.show()
```



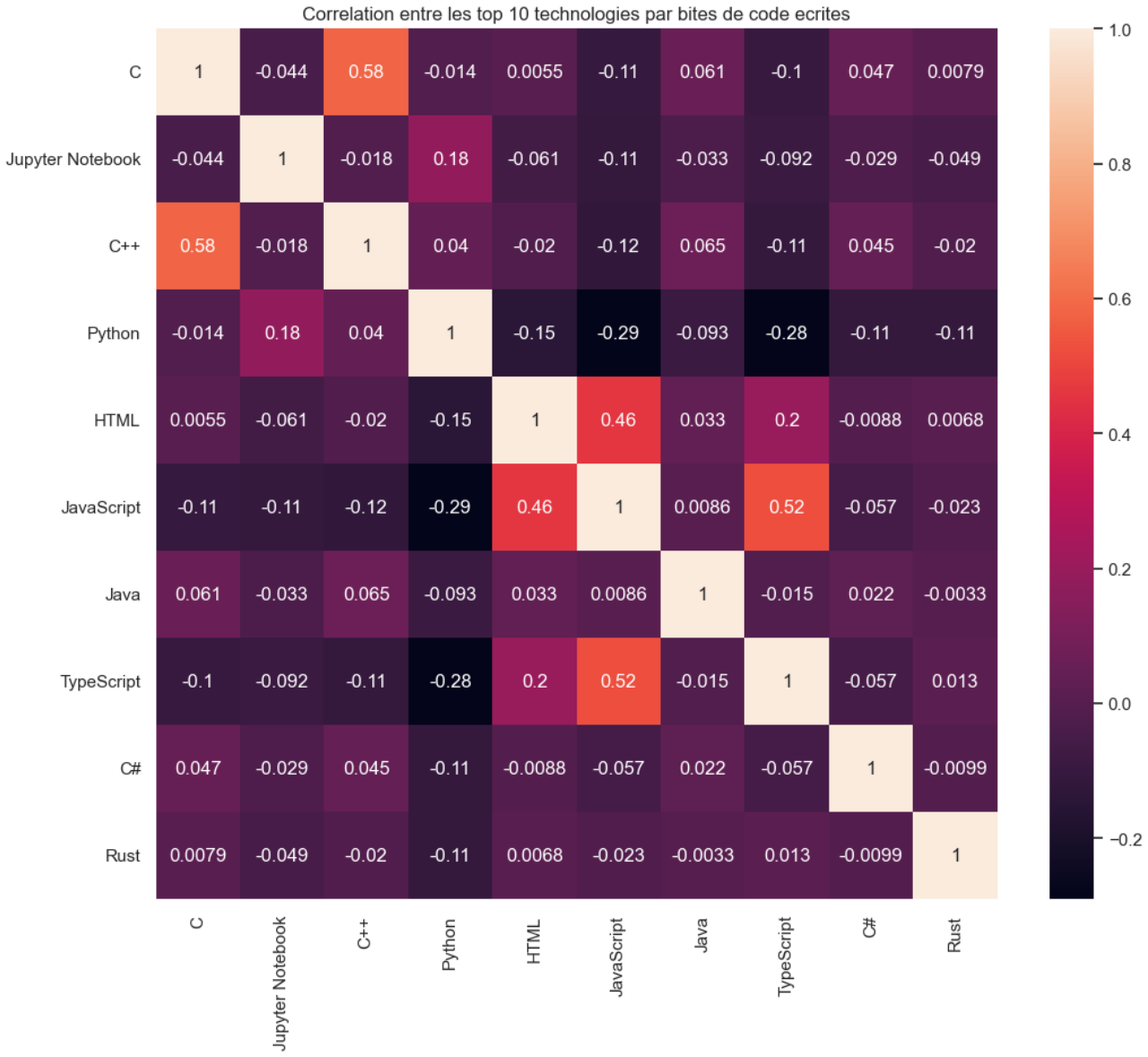
ii - Popular languages / technologies correlation (What are some commonly used together languages ?)

In [ ]:
```
# give the sum of the values for each language
languages_no_url = languages_df.drop('url', axis=1)
sum = languages_no_url.sum().sort_values(ascending=False) / 1000000
```

```
# top ten based on the sum columns namesof sum = languages_no_url.sum().sort_values(ascending=False)
top_ten = sum.iloc[:10].index
# correlation entre les technologies
corr_hm = sns.heatmap(languages_no_url[top_ten].corr(), annot=True)
corr_hm.set_title('Correlation entre les top 10 technologies par bites de code ecrites')
# increase the size of the heatmap
corr_hm.figure.set_size_inches(12, 10)
```



In [ ]:
```
top_ten_for_corr = languages_no_url[top_ten]
# instead of numbers give 1 for each language if the value is not 0
top_ten_for_corr = top_ten_for_corr.applymap(lambda x: 1 if x > 0 else 0)
# correlation entre les technologies
corr_hm = sns.heatmap(top_ten_for_corr.corr(), annot=True)
corr_hm.set_title('Correlation entre les top 10 technologies par bites de code ecrites')
# increase the size of the heatmap
corr_hm.figure.set_size_inches(12, 10)
```

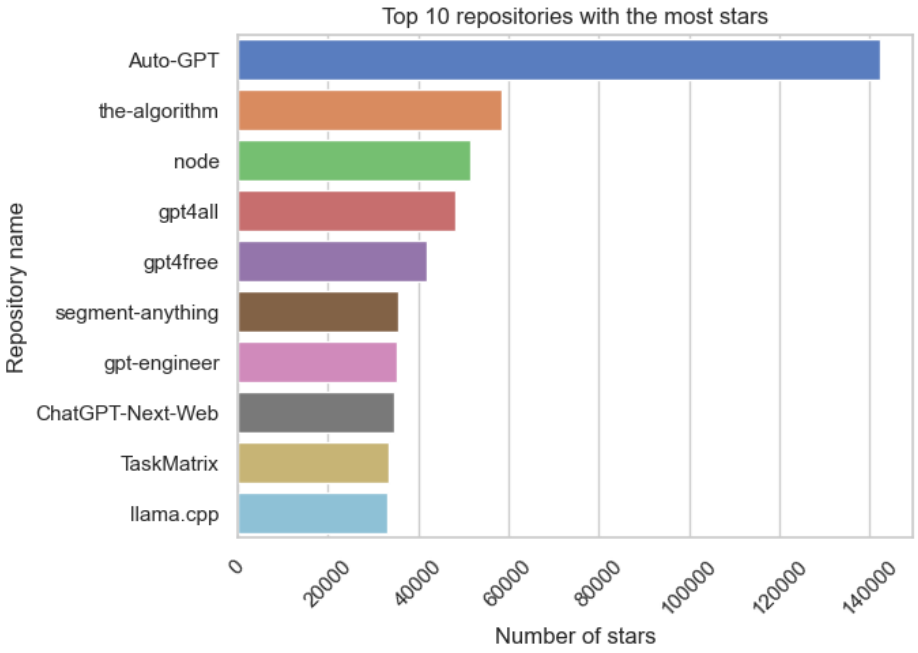


Trend analysis

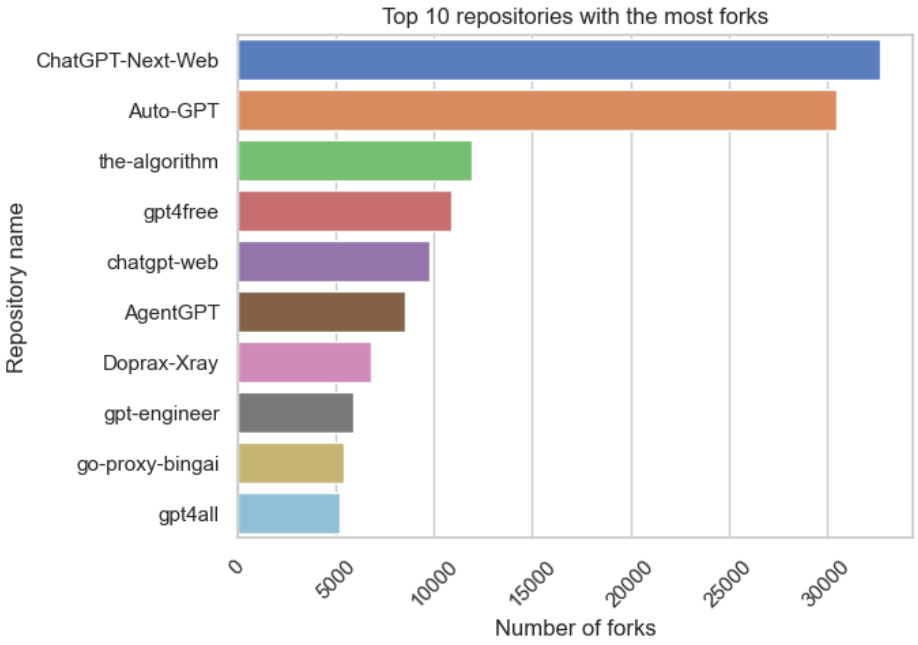
i - popular repositories (what are the popular repositories based on different factors ?)

a - By stars / forks

In [ ]:
```
# plot the barplot for the top 10 repositories with the most stars
repositories_df = repositories_df.sort_values(by='stars', ascending=False)
plot_barplot(data=repositories_df.head(10), x='stars', y='name', hue=None, title='Top 10 repositories with the most stars', xlabel
```

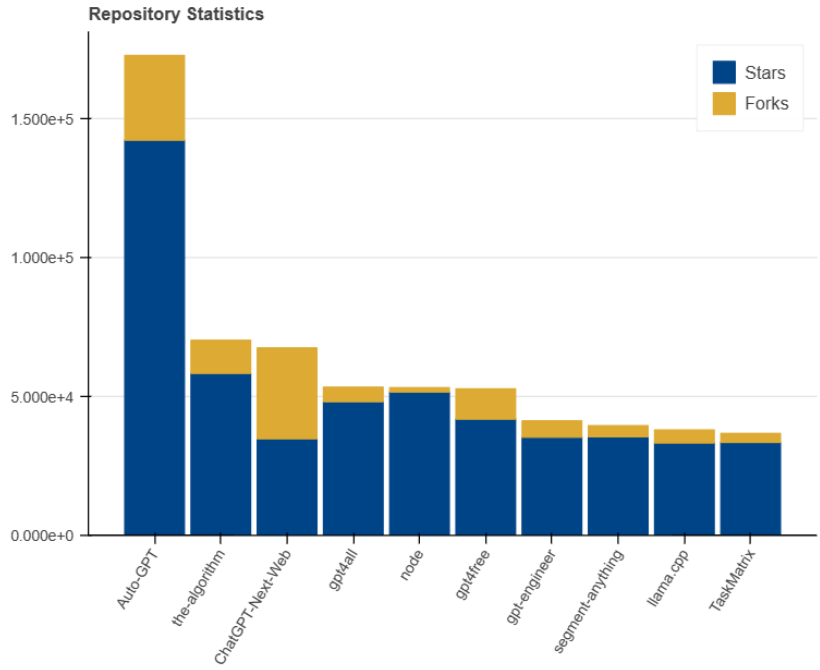


In [ ]:
```
# plot the barplot for the top 10 repositories with the most forks
repositories_df = repositories_df.sort_values(by='forks', ascending=False)
plot_barplot(data=repositories_df.head(10), x='forks', y='name', hue=None, title='Top 10 repositories with the most forks', xlabel=
```



In [ ]:
```
repositories_df['sum'] = repositories_df['stars'] + repositories_df['forks']
repositories_df = repositories_df.sort_values(by='sum', ascending=False)
repositories = repositories_df['name'].head(10)
data = {'repositories': repositories,
        'stars': repositories_df['stars'].head(10),
        'forks': repositories_df['forks'].head(10)}

p = figure(x_range=repositories, height=500, title="Repository Statistics",
           toolbar_location=None, tools="hover", tooltips="$name @repositories: @$name")

bars = p.vbar_stack(['stars',  'forks'], x='repositories', width=0.9, color=HighContrast3[0:2], source=data,
            legend_label=['Stars',  'Forks'])

p.y_range.start = 0
p.x_range.range_padding = 0.1
p.xgrid.grid_line_color = None
p.axis.minor_tick_line_color = None
p.outline_line_color = None
p.legend.location = "top_right"
```

```
p.legend.orientation = "vertical"

p.xaxis.major_label_orientation = 45

show(p)
```



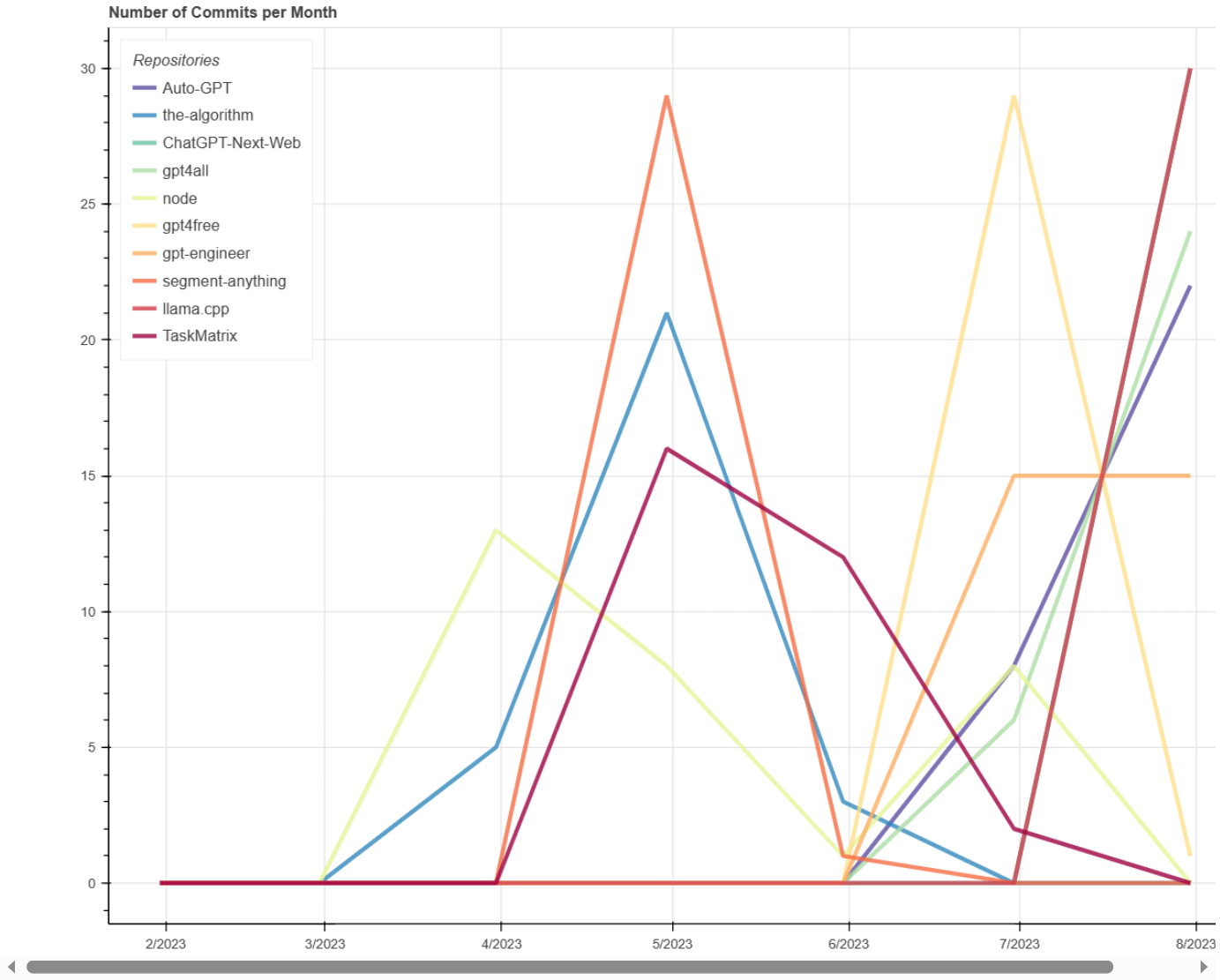
b - By most active

In [ ]:
```
months = pd.date_range(start='2023-01-31 00:00:00+00:00', end='2023-08-30 00:00:00+00:00', freq='M')
# add another column called count and set it to 0
months = pd.DataFrame(months, columns=['month'])
weeks = pd.date_range(start='2023-01-31 00:00:00+00:00', end='2023-08-30 00:00:00+00:00', freq='W')
weeks = pd.DataFrame(weeks, columns=['week'])
weeks['count'] = 0
months['count'] = 0
```

In [ ]:
```
# top 10 repositories
top_ten_repositories = repositories_df.sort_values(by='sum', ascending=False).head(10)
# from the commits_df get the commits for the top 10 repositories
top_ten_commits = commits_df[commits_df['url'].isin(top_ten_repositories['url'])]

# Convert 'author_date' column to datetime
top_ten_commits['author_date'] = pd.to_datetime(top_ten_commits['author_date'])


# Create a figure
p = figure(width=1000, height=800, x_axis_type="datetime")
p.title.text = 'Number of Commits per Month'

# Set the color palette for the lines
color_palette = Spectral10[:len(top_ten_repositories)]

# Iterate over the top 10 repositories and plot the number of commits per month
for i, (_, row) in enumerate(top_ten_repositories.iterrows()):
    repo_url = row['url']
    color = color_palette[i]

    # Filter commits for the current repository and count the number of commits per month
    monthly_counts = top_ten_commits[top_ten_commits['url'] == repo_url].groupby(pd.Grouper(key='author_date', freq='M')).size()
    df = pd.DataFrame({'author_date': monthly_counts.index, 'count': monthly_counts.values})

    # merge the two dataframes to have the months with 0 commits
    df = pd.merge(months, df, how='left', left_on='month', right_on='author_date')
    df['count'] = df['count_y'].fillna(0)
    df['author_date'] = df['month']

    p.line(df['author_date'], df['count'], line_width=3.5, color=color, alpha=0.8, legend_label=row['name'])

# Configure the legend
p.legend.location = "top_left"
p.legend.orientation = "vertical"
p.legend.click_policy = "hide"
```

```
p.legend.title = 'Repositories'

# Display the plot
show(p)
```

```
C:\Users\YouCode\AppData\Local\Temp\ipykernel_7760\2898847901.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  top_ten_commits['author_date'] = pd.to_datetime(top_ten_commits['author_date'])
```



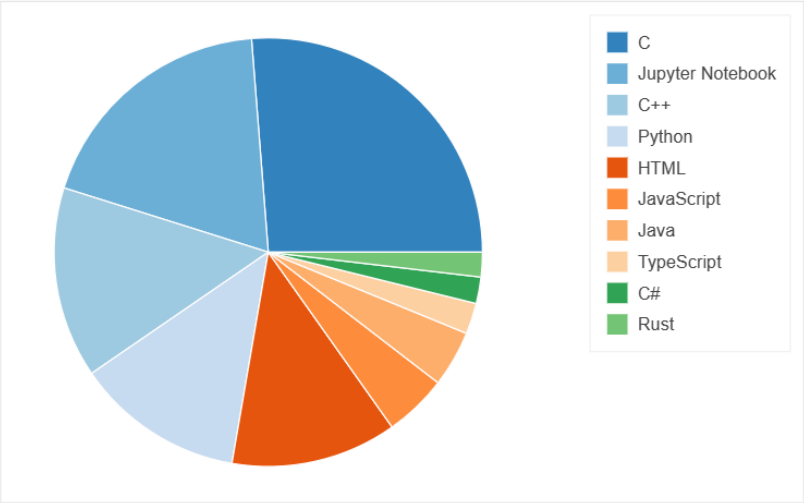
ii - Popular languages / technologies (What are the 2023 technologies ?)

a - Per code bytes written

In [ ]:
```
languages_no_url = languages_df.drop('url', axis=1)
sum = languages_no_url.sum().sort_values(ascending=False)

data = pd.Series(sum.head(10)).reset_index(name='value').rename(columns={'index':'language'})
data['angle'] = data['value']/data['value'].sum() * 2*np.pi
data['color'] = Category20c[10]


# calculate the percentage of each language then round it to 2 decimal places and convert it to string to be displayed in the leger
data['percentage'] = data['value']/data['value'].sum() * 100
data['percentage'] = data['percentage'].apply(lambda x: round(x, 2))
data['percentage'] = data['percentage'].apply(lambda x: str(x) + '%')

p = figure (title="Top 10 languages / technologies used in 2023", toolbar_location=None, tools="hover", tooltips="@language : @valu
```

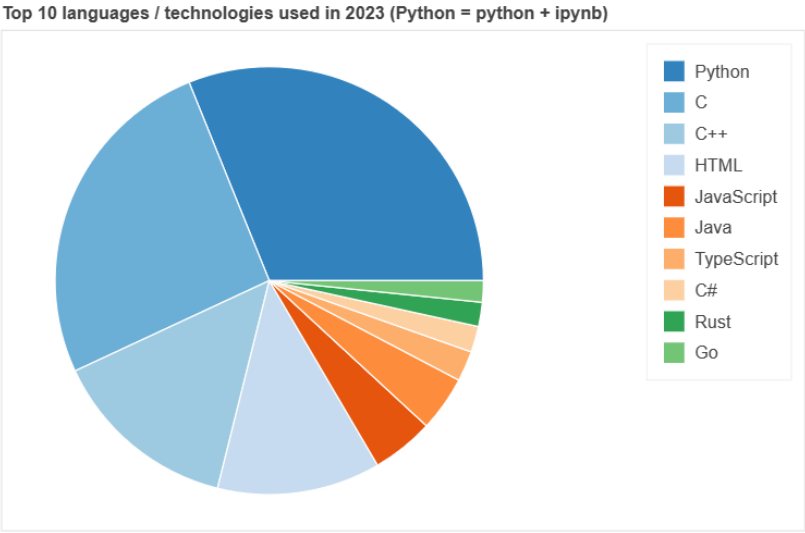
```
p.wedge(x=0, y=1, radius=0.4,
        start_angle=cumsum('angle', include_zero=True), end_angle=cumsum('angle'),
        line_color="white", fill_color='color', legend_field='language', source=data)

p.axis.axis_label=None
p.axis.visible=False
p.grid.grid_line_color = None


# add the percentage to the graph

show(p)
```

Top 10 languages / technologies used in 2023



NB : Consdering jupyter notebooks as python.

In [ ]:
```
languages_df1 = languages_df.copy()
languages_df1['Python'] = languages_df1['Python'] + languages_df1['Jupyter Notebook']
languages_df1 = languages_df1.drop('Jupyter Notebook', axis=1)
languages_no_url = languages_df1.drop('url', axis=1)
sum = languages_no_url.sum().sort_values(ascending=False)

data = pd.Series(sum.head(10)).reset_index(name='value').rename(columns={'index':'language'})
data['angle'] = data['value']/data['value'].sum() * 2*np.pi
data['color'] = Category20c[10]

# calculate the percentage of each language then round it to 2 decimal places and convert it to string to be displayed in the leger
data['percentage'] = data['value']/data['value'].sum() * 100
data['percentage'] = data['percentage'].apply(lambda x: round(x, 2))
data['percentage'] = data['percentage'].apply(lambda x: str(x) + '%')

p = figure (title="Top 10 languages / technologies used in 2023 (Python = python + ipynb)", toolbar_location=None, tools="hover",

p.wedge(x=0, y=1, radius=0.4,
        start_angle=cumsum('angle', include_zero=True), end_angle=cumsum('angle'),
        line_color="white", fill_color='color', legend_field='language', source=data)

p.axis.axis_label=None
p.axis.visible=False
p.grid.grid_line_color = None

show(p)
```



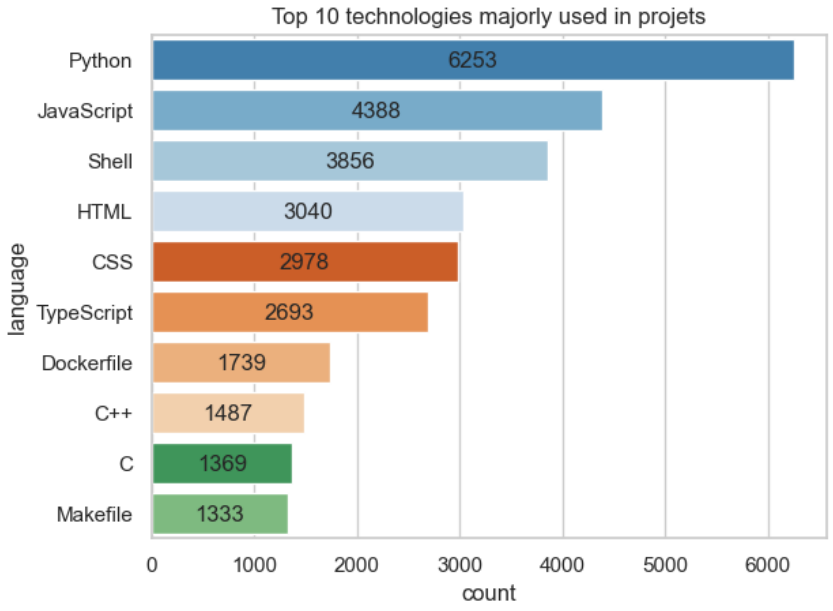
b - Per dominant language in the repositories

In [ ]:
```
languages = languages_df.columns[1:]

# create a new dataframe to store the number of repositories that use each language
languages_count = pd.DataFrame(columns=['language', 'count'])

# change the type of the languages column from object to list if it is not already
if type(repositories_df['languages'][0]) != list:
    repositories_df['languages'] = repositories_df['languages'].apply(lambda x: ast.literal_eval(x))

# iterate over the languages and count the number of repositories that use each language
counts = []
for language in languages:
    # check if language is in the repositories_df['languages'] column as an array
    count = len([x for x in repositories_df['languages'] if language in x])
    counts.append(count)

# add the languages and the counts to the dataframe
languages_count['language'] = languages
languages_count['count'] = counts

# sort the dataframe by the count of each language
languages_count = languages_count.sort_values(by='count', ascending=False)
```

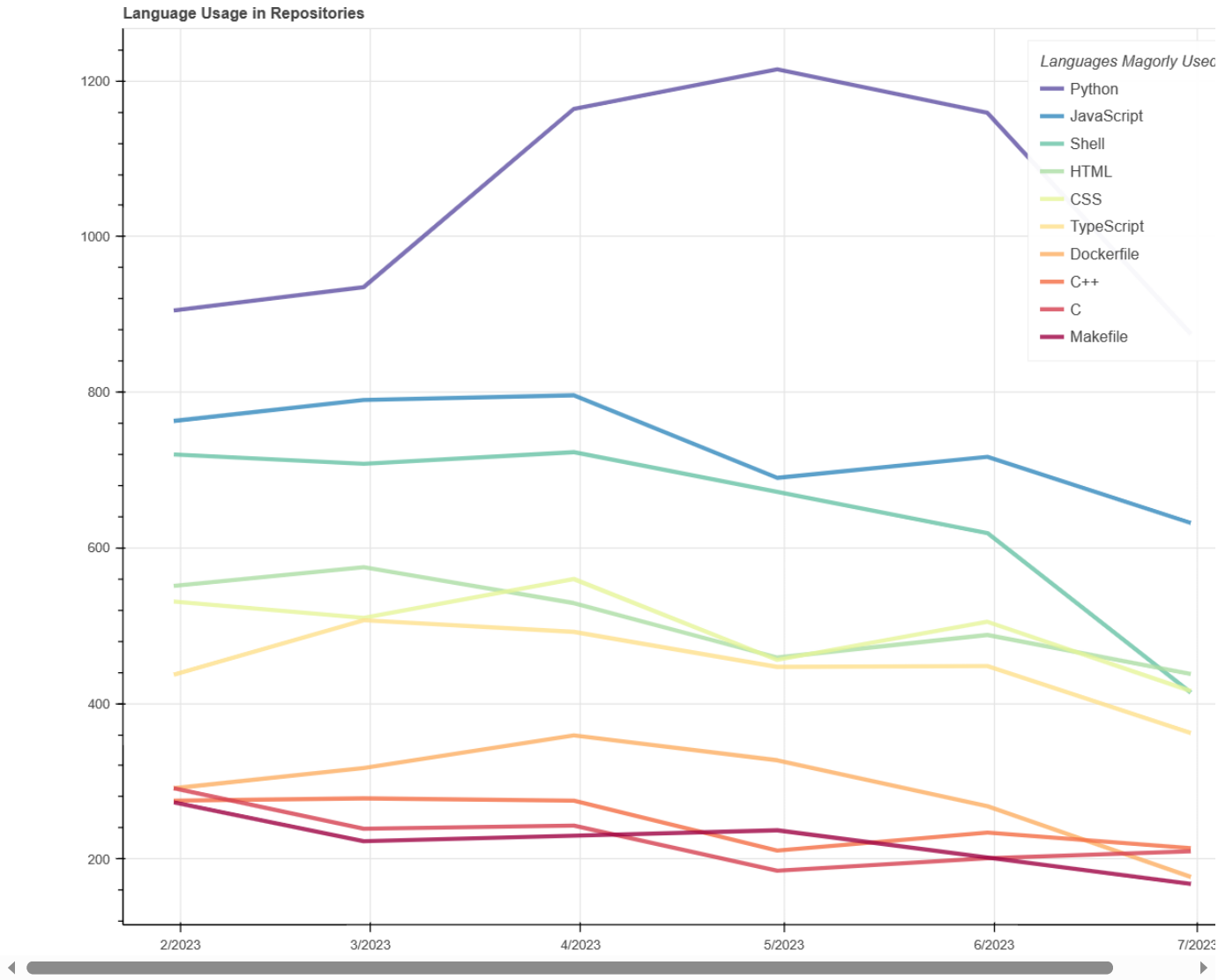
In [ ]:
```
import pandoc
```

In [ ]:
```
# top 10 most occuring languages
plt = sns.barplot(x = 'count',y='language', data=languages_count.head(10), palette='tab20c')

# add the value counts to the plot in the middle of the bars
for p in plt.patches:
    plt.annotate(format(p.get_width(), '.0f'),
                 (p.get_width() /2, p.get_y() + p.get_height()),
                 ha = 'center', va = 'center',
                    xytext = (0, 10),
                    textcoords = 'offset points')
plt.set_title('Top 10 technologies majorly used in projets')
```

Out[ ]:
```
Text(0.5, 1.0, 'Top 10 technologies majorly used in projets')
```



In [ ]:
```
from bokeh.palettes import Spectral10

# Define the list of languages you want to analyze
languages = languages_count['language'].head(10).tolist()
# date column is in string format, convert it to datetime
repositories_df['created_at'] = pd.to_datetime(repositories_df['created_at'])

# Filter the repositories_df based on the specified time period
start_date = '2023-01-01T00:03:06Z'
end_date = '2023-08-30T22:02:38Z'
filtered_df = repositories_df[
    (repositories_df['created_at'] >= start_date) &
    (repositories_df['created_at'] <= end_date)
]

# Create a figure
p = figure(width=1000, height=800, x_axis_type="datetime")
p.title.text = 'Language Usage in Repositories'

# Set the color palette for the lines
color_palette = Spectral10[:len(languages)]

# Iterate over the languages and plot the usage
for language, color in zip(languages, color_palette):
    # Count the occurrences of the language per month
    monthly_counts = filtered_df[filtered_df['languages'].apply(lambda langs: language in langs)].groupby(pd.Grouper(key='created_
    df = pd.DataFrame({'created_at': monthly_counts.index, 'count': monthly_counts.values})
    p.line(df['created_at'], df['count'], line_width=3.5, color=color, alpha=0.8, legend_label=language)


# Configure the legend
p.legend.location = "top_right"
p.legend.orientation = "vertical"
p.legend.click_policy = "hide"
p.legend.title = 'Languages Magorly Used'

# Display the plot
show(p)
```



iii - Popular Repos analysis

a - Languages in popular repositories

In [ ]:
```
repositories_df['sum'] = repositories_df['stars'] + repositories_df['forks']
repositories_df = repositories_df.sort_values(by='sum', ascending=False)
popular_repos = repositories_df.head(10)
popular_repos
```

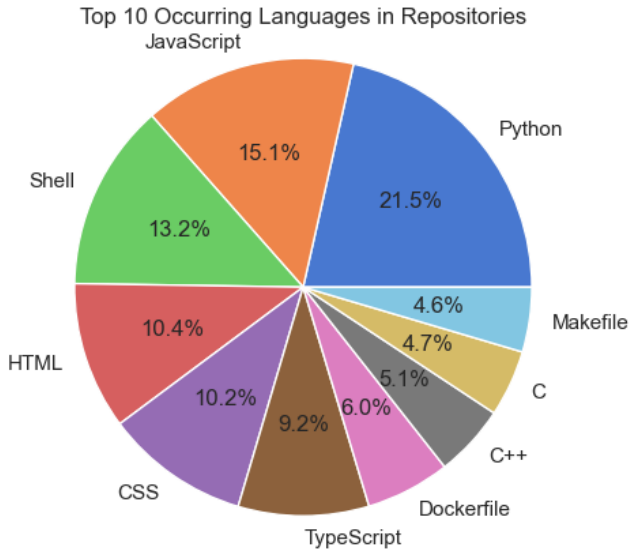
Out[ ]:

	name	url	description	stars	created_at	language	forks	watchers	open_issue
6547	Auto-GPT	https://github.com/Significant-Gravitas/Auto-GPT	An experimental open-source attempt to make GP...	142390	2023-03-16 09:21:07+00:00	Python	30523	142390	76
7493	the-algorithm	https://github.com/twitter/the-algorithm	Source code for Twitter's Recommendation Algor...	58374	2023-03-27 14:57:57+00:00	Scala	11928	58374	34
6028	ChatGPT-Next-Web	https://github.com/Yidadaa/ChatGPT-Next-Web	A well-designed cross-platform ChatGPT UI (Web...	34814	2023-03-10 18:27:54+00:00	TypeScript	32737	34814	2
7494	gpt4all	https://github.com/nomic-ai/gpt4all	gpt4all: an ecosystem of open-source chatbots ...	48173	2023-03-27 18:49:32+00:00	C++	5229	48173	41
2733	node	https://github.com/base-org/node	Everything required to run your own Base node	51652	2023-02-01 13:55:02+00:00	Shell	1571	51652	
7671	gpt4free	https://github.com/xtekky/gpt4free	The official gpt4free repository | various col...	41894	2023-03-29 17:00:43+00:00	Python	10888	41894	15
10355	gpt-engineer	https://github.com/AntonOsika/gpt-engineer	Specify what you want it to build, the AI asks...	35368	2023-04-29 12:52:15+00:00	Python	5903	35368	6
7152	segment-anything	https://github.com/facebookresearch/segment-an...	The repository provides code for running infer...	35542	2023-03-23 17:03:03+00:00	Jupyter Notebook	3989	35542	30
6029	llama.cpp	https://github.com/ggerganov/llama.cpp	Port of Facebook's LLaMA model in C/C++	33293	2023-03-10 18:58:00+00:00	C	4698	33293	40
5295	TaskMatrix	https://github.com/microsoft/TaskMatrix	repository of Task matrix	33500	2023-03-02 09:04:28+00:00	Python	3241	33500	22

In [ ]:

```
import matplotlib.pyplot as plt

# Extract the "Languages" column from the DataFrame
languages_column = repositories_df['languages']

# Get the unique languages
unique_languages = list(set([language for languages in languages_column for language in languages]))

# Count the occurrence of each language
language_counts = {}

for languages in languages_column:
    for language in languages:
        language_counts[language] = language_counts.get(language, 0) + 1

# Sort the languages based on their counts
sorted_languages = sorted(language_counts.items(), key=lambda x: x[1], reverse=True)

# Take the top 10 languages
top_languages = sorted_languages[:10]

# Extract the language labels and counts
labels = [language[0] for language in top_languages]
counts = [language[1] for language in top_languages]
```

```
# Create a pie plot
plt.pie(counts, labels=labels, autopct='%1.1f%%')
plt.axis('equal')  # Equal aspect ratio ensures that the pie is drawn as a circle.
plt.title('Top 10 Occurring Languages in Repositories')
plt.show()
```

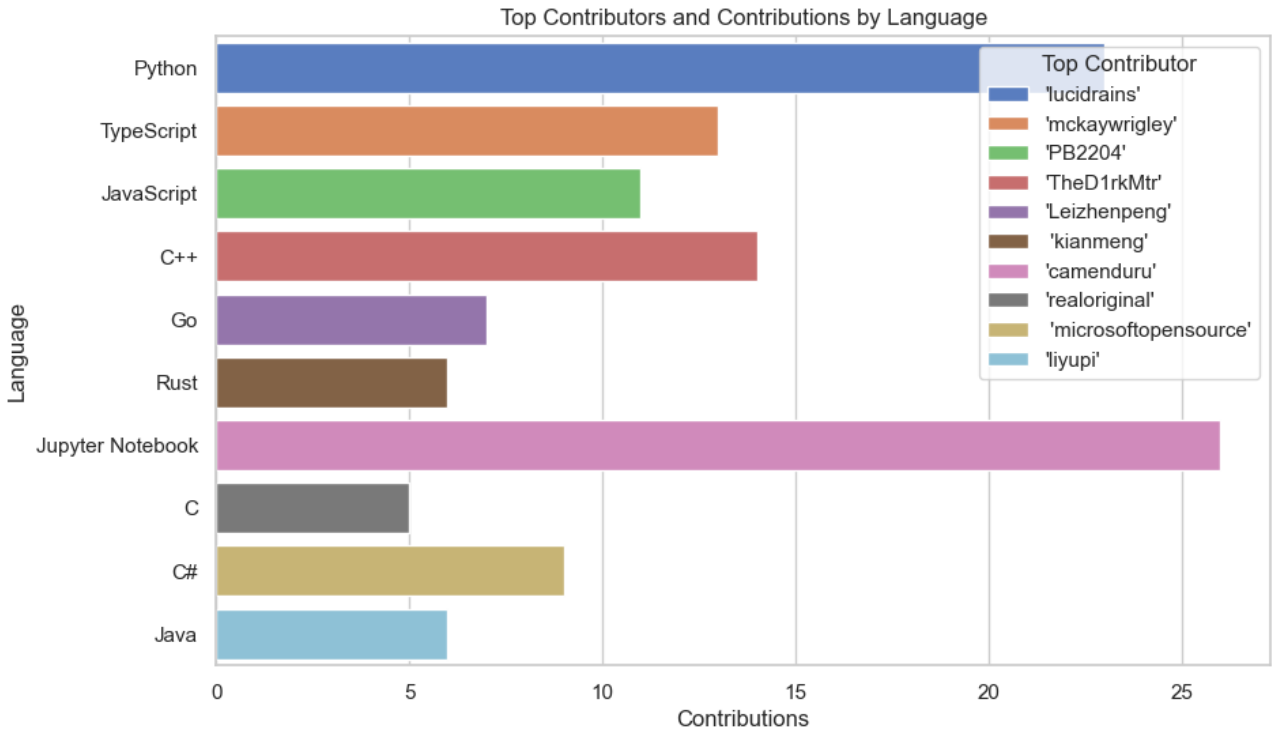


Active contributors and possible collaborations

In [ ]:
```
# Change the type of the contributors column from object to list if it is not already
if repositories_df['contributors'].dtype != list:
    repositories_df['contributors'] = repositories_df['contributors'].apply(lambda x: ast.literal_eval(x))

# List of the top languages
top_languages = repositories_df['language'].value_counts()[:10]
languages_list = top_languages.index.tolist()

# Create an empty DataFrame to store the results
result_df = pd.DataFrame(columns=['Language', 'Top Contributor', 'Contributions'])

for target_language in languages_list:
    # Filter the DataFrame based on the target language
    filtered_df = repositories_df[repositories_df['language'] == target_language]

    # Get the top contributors for the target language
    top_contributors = []
    for contributors_list in filtered_df['contributors']:
        top_contributors.extend(contributors_list)

    # Count the occurrences of each contributor
    contributor_counts = pd.Series(top_contributors).value_counts()

    # Filter out contributors with 'bot' in their names
    filtered_contributors = contributor_counts[~contributor_counts.index.str.contains('bot')]
    filtered_contributors = filtered_contributors[~filtered_contributors.index.str.contains('eltociear')]
    # Filter out empty contributors
    filtered_contributors = filtered_contributors[filtered_contributors.index != '']


    if len(filtered_contributors) > 0:
        # Get the top duplicate contributor
        top_duplicate_contributor = filtered_contributors.idxmax()

        # Get the number of contributions by the top duplicate contributor
        top_duplicate_contributions = filtered_contributors[top_duplicate_contributor]

        # Create a temporary DataFrame for the current language
        temp_df = pd.DataFrame({'Language': target_language, 'Top Contributor': top_duplicate_contributor, 'Contributions': top_du

        # Concatenate the temporary DataFrame with the result DataFrame
        result_df = pd.concat([result_df, temp_df], ignore_index=True)

# Print the resulting DataFrame
print(result_df)
```

	Language	Top Contributor	Contributions
0	Python	'lucidrains'	23
1	TypeScript	'mckaywrigley'	13
2	JavaScript	'PB2204'	11
3	C++	'TheD1rkMtr'	14
4	Go	'Leizhenpeng'	7
5	Rust	'kianmeng'	6
6	Jupyter Notebook	'camenduru'	26
7	C	'realoriginal'	5
8	C#	'microsoftopensource'	9
9	Java	'liyupi'	6

In [ ]:
```
import matplotlib.pyplot as plt
# Create the horizontal bar chart
plt.figure(figsize=(10, 6))
sns.barplot(data=result_df, y='Language', x='Contributions', hue='Top Contributor', dodge=False)
plt.xlabel('Contributions')
plt.ylabel('Language')
plt.title('Top Contributors and Contributions by Language')
plt.legend(title='Top Contributor', loc='best')

# Show the plot
plt.show()
```



In [ ]:
```
# From the commits_df DataFrame, select the list of unique authors
languages = languages_count['language'].head(10).tolist()
```