

Занпок:

I have a python code with graphic that contains x y axes, title and labels - how can I add my custom colors to it? the colors are #F27500, #271200, #4D2900, #7C4B04

DeepSeek:

To customize the colors of your plot in Python using Matplotlib, you can specify your custom colors using their hexadecimal codes. Below is an example of how you can apply your custom colors ('#F27500', '#271200', '#4D2900', '#7C4B04') to the axes, title, labels, and other elements of your plot.

Here's a sample code:

```
```python
import matplotlib.pyplot as plt

Sample data
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]

Custom colors
color1 = "#F27500" # Orange
color2 = "#271200" # Dark Brown
color3 = "#4D2900" # Brown
color4 = "#7C4B04" # Light Brown

Create the plot
fig, ax = plt.subplots()

Plot the data with custom color
ax.plot(x, y, color=color1, label="Data Line")

Customize axes, title, and labels
ax.set_title("Custom Colored Plot", color=color2, fontsize=14) # Title color
ax.set_xlabel("X-axis", color=color3, fontsize=12) # X-axis label color
ax.set_ylabel("Y-axis", color=color3, fontsize=12) # Y-axis label color

Customize tick colors
ax.tick_params(axis='x', colors=color4) # X-axis tick color
ax.tick_params(axis='y', colors=color4) # Y-axis tick color

Customize spine colors (axes borders)
for spine in ax.spines.values():
 spine.set_color(color2)

Add a legend with custom text color
legend = ax.legend()
for text in legend.get_texts():
 text.set_color(color2)
```

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

#### ### Explanation:

1. **Line Color**: The `ax.plot()` function uses `color=color1` to set the line color to `#F27500`.
2. **Title Color**: The `ax.set_title()` function uses `color=color2` to set the title color to `#271200`.
3. **Axis Labels**: The `ax.set_xlabel()` and `ax.set_ylabel()` functions use `color=color3` to set the axis label colors to `#4D2900`.
4. **Tick Colors**: The `ax.tick_params()` function sets the tick colors to `#7C4B04` for both the x and y axes.
5. **Spine Colors**: The `ax.spines.values()` loop sets the color of the plot borders (spines) to `#271200`.
6. **Legend Text Color**: The legend text color is set to `#271200` using a loop over `legend.get_texts()`.

You can adjust these colors further or apply them to other elements of the plot as needed. Let me know if you need more help!

### Занпок:

I have a dataset with columns: Price, Restaurants, Station. I need to make a python code that shows graphically the gradation from the station with most expensive restaurants to station with cheapest restaurant prices. However some data in the price column is invalid or empty - how can I filter all the broken data and make station an index for grouping price data?

### DeepSeek:

To achieve your goal, you can use Python with libraries like `pandas` for data manipulation and `matplotlib` or `seaborn` for visualization. Below is a step-by-step guide and code to filter out invalid or empty data, group the data by station, and visualize the gradation of restaurant prices.

#### ### Step-by-Step Guide:

1. **Load the dataset**: Use `pandas` to load your dataset.
2. **Filter invalid or empty data**: Remove rows where the `Price` column is invalid or empty.
3. **Group by station**: Group the data by the `Station` column and calculate the average price for each station.
4. **Sort by average price**: Sort the stations from the most expensive to the cheapest.

5. **Visualize the data**: Use a bar plot or similar visualization to show the gradation.

### Python Code:

```
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Load the dataset
# Replace 'your_dataset.csv' with the path to your dataset
df = pd.read_csv('your_dataset.csv')

# Step 2: Filter invalid or empty data
# Remove rows where 'Price' is NaN or non-numeric
df = df[pd.to_numeric(df['Price'], errors='coerce').notna()]

# Step 3: Group by station and calculate average price
# Convert 'Price' to numeric (in case it's not already)
df['Price'] = pd.to_numeric(df['Price'])
station_prices = df.groupby('Station')['Price'].mean().reset_index()

# Step 4: Sort by average price (most expensive to cheapest)
station_prices_sorted = station_prices.sort_values(by='Price', ascending=False)

# Step 5: Visualize the data
plt.figure(figsize=(10, 6))
sns.barplot(x='Price', y='Station', data=station_prices_sorted, palette='viridis')
plt.title('Gradation of Restaurant Prices by Station')
plt.xlabel('Average Price')
plt.ylabel('Station')
plt.show()
```
```

### Explanation:

1. **Loading the dataset**: The `pd.read_csv()` function loads your dataset into a pandas DataFrame.
2. **Filtering invalid data**: `pd.to_numeric(df['Price'], errors='coerce')` converts the `'Price'` column to numeric, setting invalid entries to `'NaN'`. The `notna()` function filters out rows with `'NaN'` values.
3. **Grouping by station**: `groupby('Station')['Price'].mean()` calculates the average price for each station.
4. **Sorting**: `sort_values(by='Price', ascending=False)` sorts the stations by average price in descending order.
5. **Visualization**: `seaborn.barplot()` creates a bar plot showing the gradation of prices across stations.

### Notes:

- Replace `'your_dataset.csv'` with the actual path to your dataset.

- If your dataset is in a different format (e.g., Excel), use the appropriate pandas function (e.g., `pd.read_excel()`).
- Customize the plot (e.g., colors, labels) as needed.

This code will give you a clear visual representation of the gradation of restaurant prices by station, with invalid data filtered out.

### Занпок:

Nice! pls optimize this process so I could make it in one string

### DeepSeek:

Certainly! You can optimize the entire process into a single line of code using method chaining in pandas. Here's how you can do it:

### Optimized One-Liner Code:

```
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# One-liner to load, filter, group, sort, and plot
(pd.read_csv('your_dataset.csv')
 .assign(Price=lambda x: pd.to_numeric(x['Price'], errors='coerce')) # Convert 'Price' to
 numeric
 .dropna(subset=['Price']) # Drop rows with invalid 'Price'
 .groupby('Station', as_index=False)['Price'].mean() # Group by 'Station' and calculate
 mean price
 .sort_values('Price', ascending=False) # Sort by average price
 .pipe(lambda df: sns.barplot(x='Price', y='Station', data=df, palette='viridis')) # Plot
)

plt.title('Gradation of Restaurant Prices by Station')
plt.xlabel('Average Price')
plt.ylabel('Station')
plt.show()
```
```

### Explanation of the One-Liner:

1. `pd.read_csv('your_dataset.csv')`: Load the dataset.
2. `.assign(Price=lambda x: pd.to_numeric(x['Price'], errors='coerce'))`: Convert the `'Price'` column to numeric, coercing invalid entries to `'NaN'`.
3. `.dropna(subset=['Price'])`: Drop rows where `'Price'` is `'NaN'`.
4. `.groupby('Station', as_index=False)['Price'].mean()`: Group by `'Station'` and calculate the mean price for each station.
5. `.sort_values('Price', ascending=False)`: Sort the stations by average price in descending order.

6. `df.pipe(lambda df: sns.barplot(x='Price', y='Station', data=df, palette='viridis'))`: Pass the resulting DataFrame to `seaborn.barplot` for visualization.

### Notes:

- Replace `'your_dataset.csv'` with the path to your dataset.
- The `pipe()` method is used to pass the DataFrame to the plotting function (`sns.barplot`) in a clean, chainable way.
- The `plt` commands (e.g., `plt.title`, `plt.show`) are still separate for clarity, but the entire data processing and plotting logic is condensed into one line.

This one-liner is efficient and concise while maintaining readability!

### Заняе:

I've made a code on python for word cloud, can you show me how to set colours #F2DEC4 and #F27405 to be random for words?

### DeepSeek:

Certainly! To create a word cloud in Python with custom colors (in your case, `#F2DEC4` and `#F27405`), you can use the `WordCloud` library. To randomly assign these colors to words, you can define a custom color function.

Here's how you can do it:

### Step 1: Install the required libraries

If you haven't already installed the `wordcloud` and `matplotlib` libraries, you can install them using pip:

```
```bash
pip install wordcloud matplotlib
```
```

### Step 2: Create the word cloud with custom colors

Here's an example code that sets the colors `#F2DEC4` and `#F27405` randomly for words:

```
```python
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import random

# Define your custom colors
colors = ['#F2DEC4', '#F27405']

# Create a function to randomly assign colors
def random_color_func(word=None, font_size=None, position=None, orientation=None,
font_path=None, random_state=None):
```

```

return random.choice(colors)

# Your text data
text = "Python is an amazing programming language. It is great for data science, machine
learning, and web development."

# Generate the word cloud
wordcloud = WordCloud(
    width=800,
    height=400,
    background_color="white", # Set background color
    color_func=random_color_func # Use the custom color function
).generate(text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off") # Hide the axis
plt.show()
'''

```

Explanation:

1. **Custom Colors**: The `colors` list contains the two hex color codes you provided.
2. **Random Color Function**: The `random_color_func` function randomly selects one of the two colors for each word.
3. **WordCloud Configuration**: The `WordCloud` object is configured to use the `random_color_func` for coloring words.
4. **Display**: The word cloud is displayed using `matplotlib`.

Output:

The word cloud will display words with colors randomly chosen from `#F2DEC4` and `#F27405`.

Let me know if you need further assistance!

3anpoc:

How to add custom colour to the scatter graph? I need to colour the numbers along the axis

DeepSeek:

To add custom colors to the numbers along the axes in a scatter plot, you can use libraries like Matplotlib in Python. Here's how you can do it:

Example using Matplotlib:

```

'''python
import matplotlib.pyplot as plt
import numpy as np

```

```

# Sample data
x = np.random.rand(50)
y = np.random.rand(50)
colors = np.random.rand(50) # Random colors for the points

# Create the scatter plot
plt.scatter(x, y, c=colors, cmap='viridis')

# Customize the color of the numbers along the axes
ax = plt.gca() # Get the current axes
ax.tick_params(axis='x', colors='red') # Color for x-axis numbers
ax.tick_params(axis='y', colors='blue') # Color for y-axis numbers

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

```

Explanation:

1. `plt.scatter(x, y, c=colors, cmap='viridis')`: This creates a scatter plot where the points are colored based on the `colors` array using the `viridis` colormap.
2. `ax.tick_params(axis='x', colors='red')`: This sets the color of the numbers (ticks) on the x-axis to red.
3. `ax.tick_params(axis='y', colors='blue')`: This sets the color of the numbers (ticks) on the y-axis to blue.

Customizing Further:

- You can also change the font size, font style, and other properties of the tick labels using additional parameters in `tick_params`.
- If you want to customize the color of specific ticks, you can use `ax.set_xticks` and `ax.set_xticklabels` with a list of colors.

```

'''python
# Example of customizing specific ticks
ax.set_xticks([0, 0.5, 1])
ax.set_xticklabels(['0', '0.5', '1'], color=['red', 'green', 'blue'])
'''

```

This approach allows you to fully customize the appearance of the numbers along the axes in your scatter plot.