“I'm a student doing a Python project in a DH course. Act as a good tutor and give me advice without giving me the full answers. im currently working in a project where im working with a dataframe, counting the lengths of the articles, with separate tables grouping lengths by year and month, here i want to create a new data frame, df\_titles\_filtered where i want to include articles from 2017 t0 2023

**ChatGPT said:**

Awesome — you’re working on a solid DH project there, and your problem breakdown is nice and clear. Let’s walk through this like a good tutor would:

**What you're trying to do:**

👉 Filter your existing dataframe so that it only includes articles from **2017 to 2023** — and store that in a new dataframe called df\_titles\_filtered.

**Key things you should double-check first:**

1. ✅ Make sure your date column is in datetime format — this will make filtering by year much easier.
   * If it's not, you can use pd.to\_datetime() to convert it.
2. ✅ Once your dates are proper datetime, you can access the .dt.year attribute to filter by year ranges.

**What you should be thinking about:**

* Use a boolean mask to filter the rows where the year is **between 2017 and 2023 (inclusive)**.
* Apply this mask to your dataframe to get df\_titles\_filtered.

**Questions for you to check:**

* What is the exact name of the column where your publication date is stored?  
  (Is it 'date', 'published\_on', 'timestamp'?)
* Is your date column already in datetime64 format? If not — how would you convert it?

**Small hint without full code:**

The kind of filter you’ll likely want uses something like:

python

CopyEdit

df[(df['year'] >= 2017) & (df['year'] <= 2023)]

But of course, you'd replace 'year' with the actual column or its .dt.year value if it's a datetime.

If you like, tell me:

* What’s the name of your date column?
* Is it already a datetime type?
* And do you need a refresher on how to check or convert it?

I can nudge you from there 👌

**You said:**

got it

**ChatGPT said:**

Excellent — sounds like you’re on the right track! 🔥

Once you’ve got df\_titles\_filtered working, you can easily plug it into your article length calculations and groupings.

If you run into anything weird or unexpected, like NaT values, odd year ranges, or strange filtering behavior — just holler. Always happy to help you debug cleanly. Keep going! 🚀

**You said:**

“I'm a student doing a Python project in a DH course. Act as a good tutor and give me advice without giving me the full answers. im currently working in a project where im working with a dataframe, counting the lengths of the articles, with separate tables grouping lengths by year and month. im writing script to generate graph and i want to get the indes of highest value in coulumn. can you exapain how to get

3/3

**ChatGPT said:**

Love the focus you’re keeping — nice incremental build-up of your workflow 🔥. Let’s walk through this together.

**What you want to do:**

👉 From a dataframe (or a Series) — you want to find the **index of the highest value in a specific column**.

**Key things you should ask yourself:**

1. **Are you working with a Series or a DataFrame?**
   * If it’s a Series (like one column you’ve pulled out), you can directly use a method to get the index of the highest value.
   * If it’s a DataFrame, you’ll need to specify which column to check.
2. **What’s the name of the column you’re targeting?**

**Concepts to think about:**

* In pandas, there’s a method called idxmax()  
  👉 It gives you the **index label where the maximum value occurs** in a Series.

**For example (but not your full code!):**

If you had:

python

CopyEdit

df['article\_length']

as a Series, you could use:

python

CopyEdit

df['article\_length'].idxmax()

This would return the **index** of the row with the highest article length.

**If you’re working inside a grouped dataframe** or something multi-level — you might need to reset the index or specify the axis.

**Questions you might check:**

* Are there any NaN or missing values in your column?  
  (Could mess with the result — think about whether you need .dropna())
* Do you need the **index** or the **row itself** with the highest value?  
  (Depending on that, you might either use idxmax() or loc[] with it)

**Bonus Thought:**

Once you have the index of the highest value — you can use .loc[] to retrieve the corresponding row, which might be handy for labeling your graph or annotating the peak.

**If you’re stuck, feel free to share:**

* The name of your column
* Whether your dataframe has any grouped/multi-index
* And what you plan to do with that index after you get it

I can nudge you the rest of the way 📊⚙️

**You said:**

in pandas if I have a dataframe of article titles and i wanna check if any word from a list exists inside each title what would be a good way to write a small function for this and apply it to each title? would it be possible to do it without defining a full function separately

3/3

**ChatGPT said:**

Excellent question — you’re thinking like a data scientist now ⚡️.

Yes — **both ways are possible in pandas**:

**🔍 Concept 1: With a small custom function**

You could write a small function that takes a title, checks if any word from your list is in that title, and returns True/False (or the matched word, or whatever you need).

But you asked if you could avoid defining a full function separately — and yes, you can, using a **lambda function** inside .apply().

**🔍 Concept 2: Using a lambda + list comprehension / any()**

This is the cleanest, most Pythonic way for this situation.

**Rough pattern (without full code):**

python

CopyEdit

df['title'].apply(lambda x: any(word in x for word in word\_list))

**Key notes:**

* x is each title in your column.
* word\_list is your list of words you’re checking for.
* any() returns True if at least one word from word\_list is found in the title.

**📌 Things to consider:**

* Do you care about **case sensitivity**?  
  If not, maybe convert both title and word to lower before checking.

python

CopyEdit

any(word.lower() in x.lower() for word in word\_list)

* Do you need to match **whole words only** or can partial matches count?  
  (e.g., does "man" matching "human" count?)  
  → If you want whole words only, regex might be safer.

**🔍 Concept 3: Regex with str.contains()**

If you’re comfortable with regex patterns:

python

CopyEdit

pattern = '|'.join(word\_list)

df['title'].str.contains(pattern, case=False, regex=True)

This checks if **any word from your list exists in each title**.

**📌 Tutor Check-in:**

* Which of these would fit your need better?
* Are your titles clean, or do they have NaNs/missing values to handle?
* Would you prefer to just get a boolean (True/False) or the matching word(s)?

Happy to nudge you whichever way you prefer ⚡