CSc 110, Autumn 2017

Lecture 21: Line-Based File Input

Adapted from slides by Marty Stepp and Stuart Reges

IMDb movies problem

- Consider the following Internet Movie Database (IMDb) data:

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

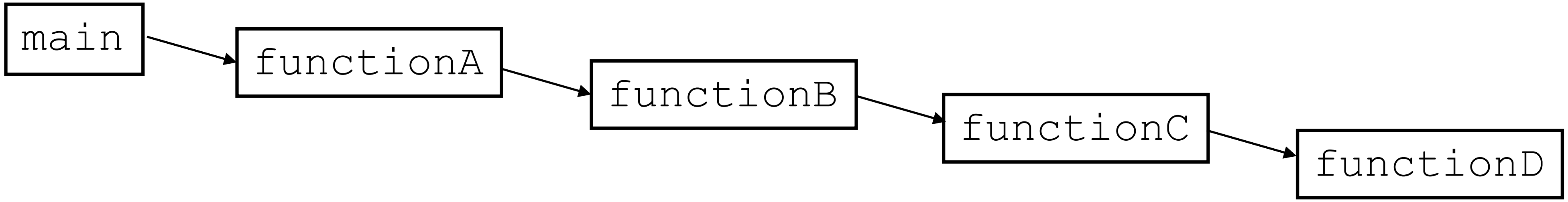
- Write a program that displays any movies containing a phrase:

```
Search word? part

Rank    Votes   Rating  Title
2       139085  9.0     The Godfather: Part II (1974)
40      129172  8.5     The Departed (2006)
95      20401   8.2     The Apartment (1960)
192     30587   8.0     Spartacus (1960)
4 matches.
```

  - Is this a token or line-based problem?

"Chaining"

- `main` should be a concise summary of your program.
  - It is bad if each function calls the next without ever returning (we call this *chaining*):



main

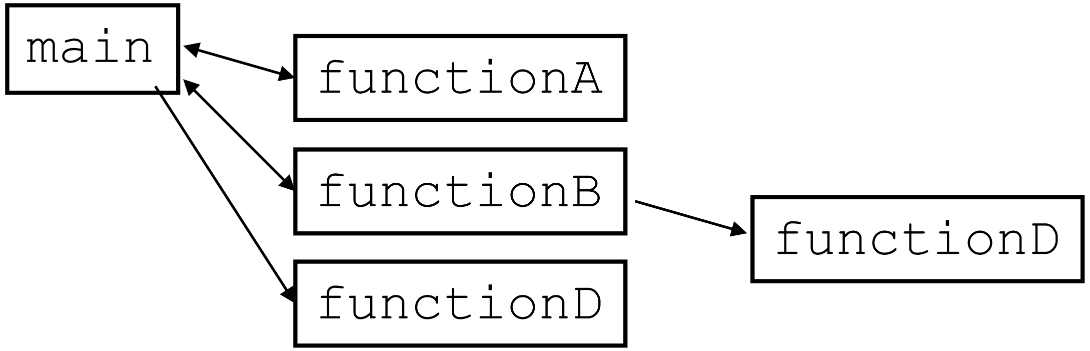
functionA

functionB

functionC

functionD

- A better structure has `main` make most of the calls.
  - Functions must return values to `main` to be passed on later.



main

functionA

functionB

functionD

functionD

Bad IMDb "chained" code 1

```
# Displays IMDB's Top 250 movies that match a search string.
def main():
    get_word()

# Asks the user for their search word and returns it.
def get_word():
    search_word = input("Search word: ")
    search_word = search_word.lower()
    print()
    file = open("imdb.txt")
    search(file, search_word)

# Breaks apart each line, looking for lines that match the search word.
def search(file, search_word):
    matches = 0
    for line in file:
        line_lower = line.lower()      # case-insensitive match
        if (search_word in line_lower):
            matches += 1
            print("Rank\tVotes\tRating\tTitle")
            display(line)
```

Bad IMDb "chained" code 2

```
# Displays the line in the proper format on the screen.
def display(line):
    parts = line.split()
    rank = parts[0]
    rating = parts[1]
    votes = parts[2]
    title = ""
    for i in range(3, len(parts)):
        title += parts[i] + " "    # the rest of the line
    print(rank + "\t" + votes + "\t" + rating + "\t" + title)
```

Better IMDb answer 1

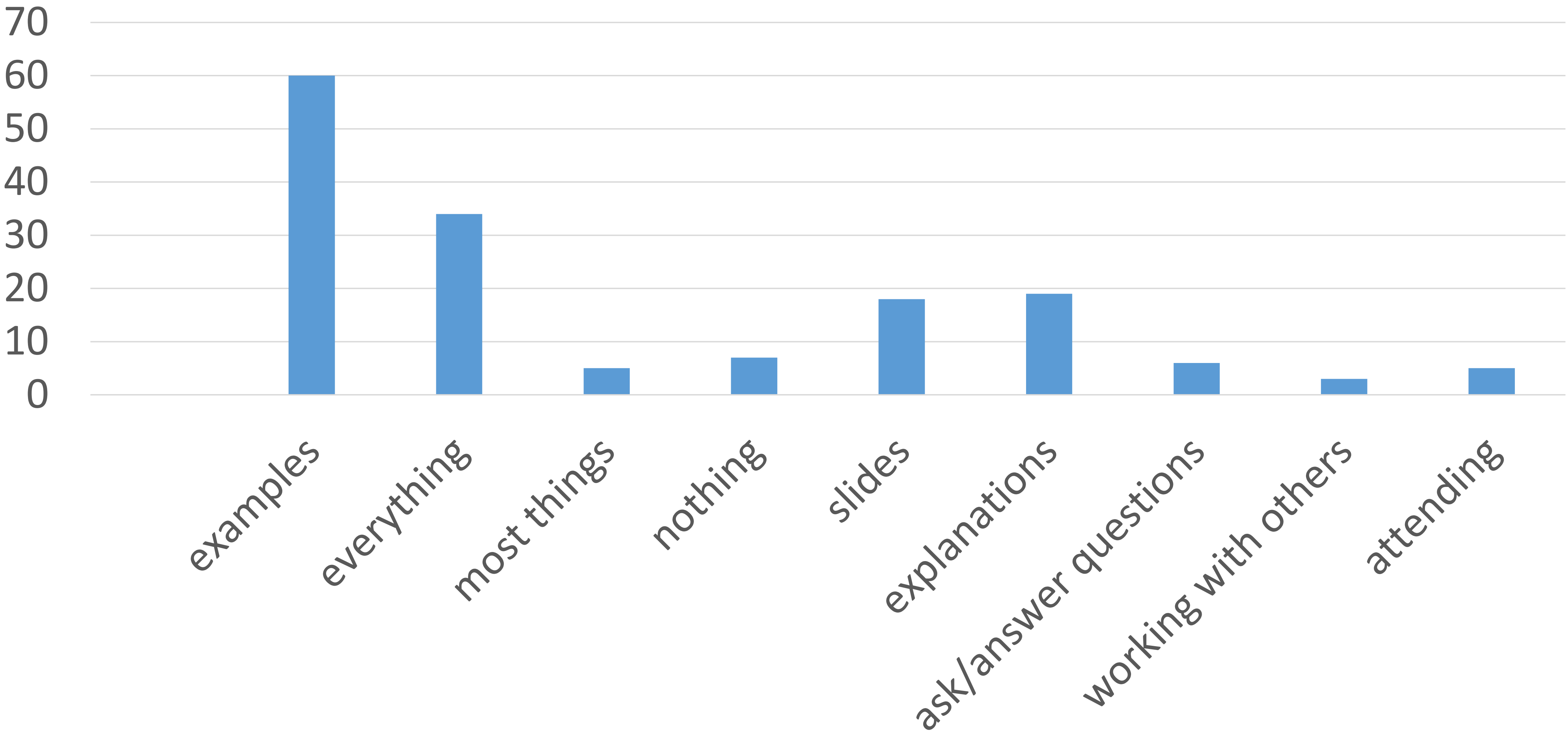
```
# Displays IMDB's Top 250 movies that match a search string.

def main():
    search_word = get_word()
    file = open("imdb.txt")
    line = search(file, search_word)

    if (len(line) > 0):
        print("Rank\tVotes\tRating\tTitle")
        matches = 0
        while (len(line) > 0):
            display(line)
            line = search(file, search_word)
            matches += 1
        print(str(matches) + " matches.")

# Asks the user for their search word and returns it.
def get_word():
    search_word = input("Search word: ")
    search_word = search_word.lower()
    print()
    return search_word
...
```

Better IMDb answer 2

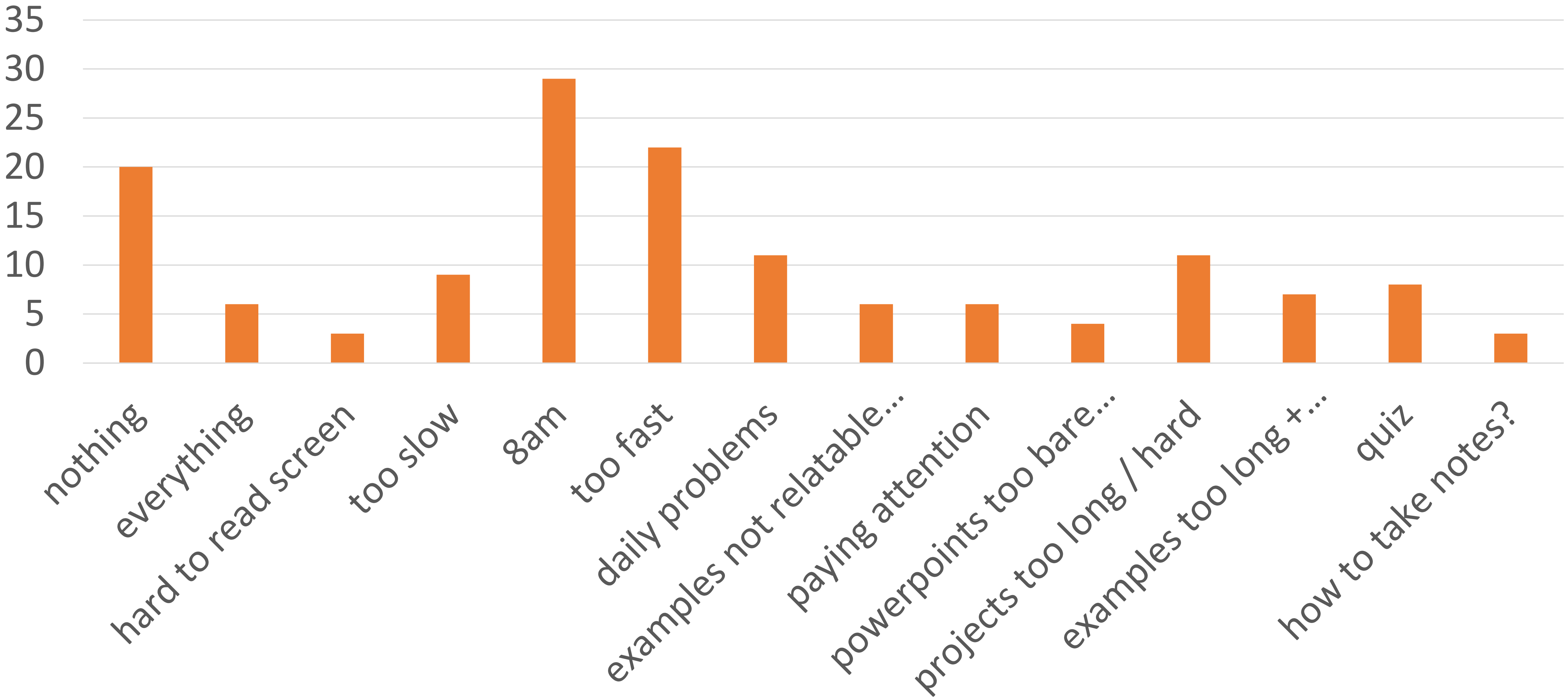
```
    ...

# Breaks apart each line, looking for lines that match the search word.
def search(file, search_word):
    for line in file:
        line_lower = line.lower()     # case-insensitive match
        if (search_word in line):
            return line
    return ""   # not found

# displays the line in the proper format on the screen.
def display(line):
    parts = line.split()
    rank = parts[0]
    rating = parts[1]
    votes = parts[2]
    title = ""
    for i in range(3, len(parts)):
        title += parts[i] + " "    # the rest of the line
    print(rank + "\t" + votes + "\t" + rating + "\t" + title)
```

Survey Results

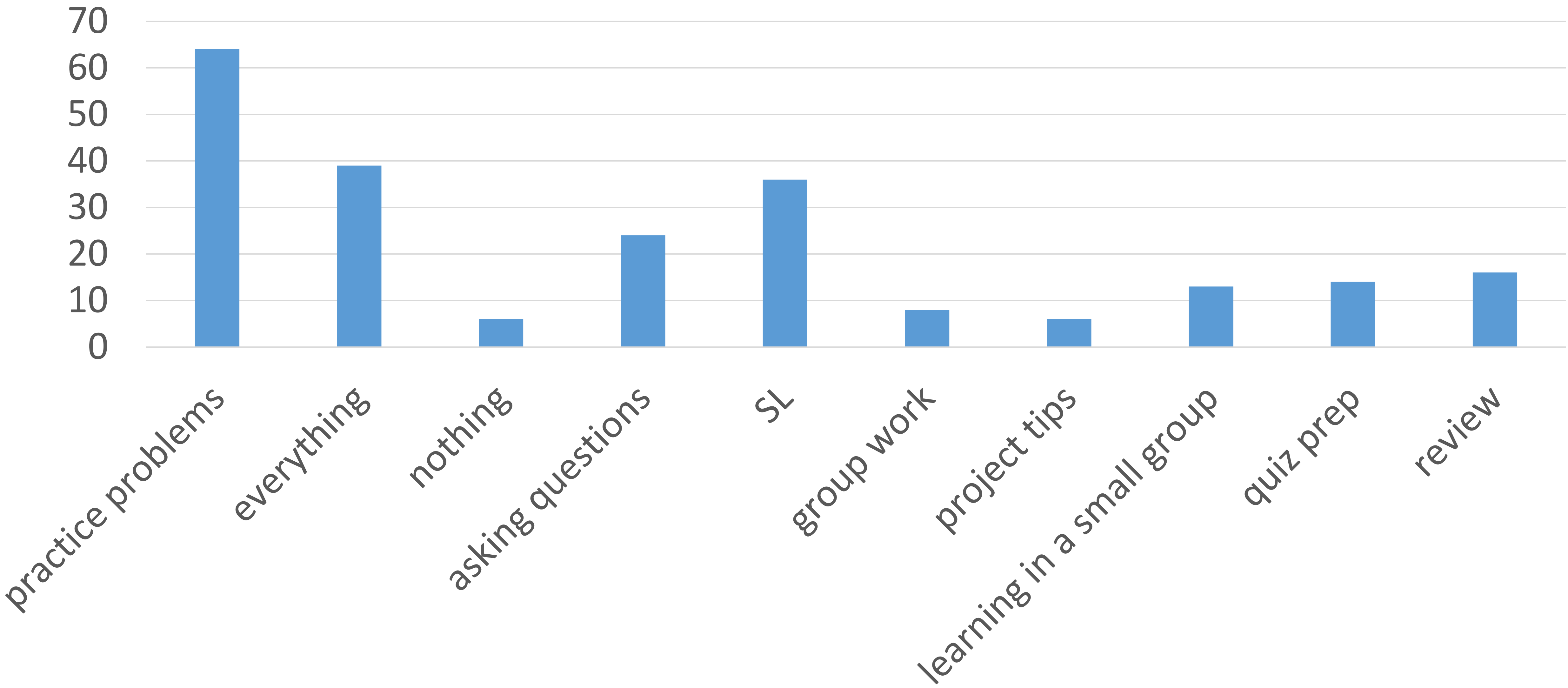
What you like about lecture



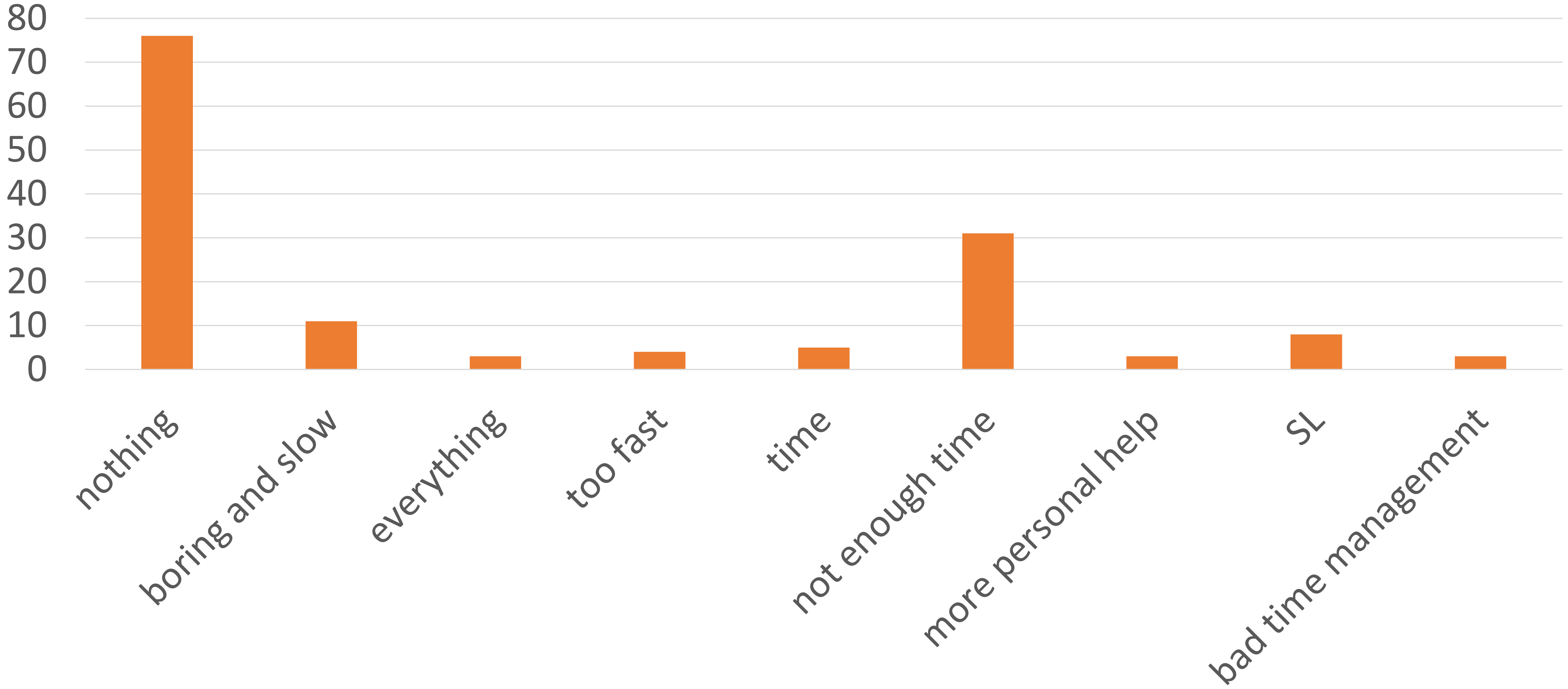
What you dislike about lecture



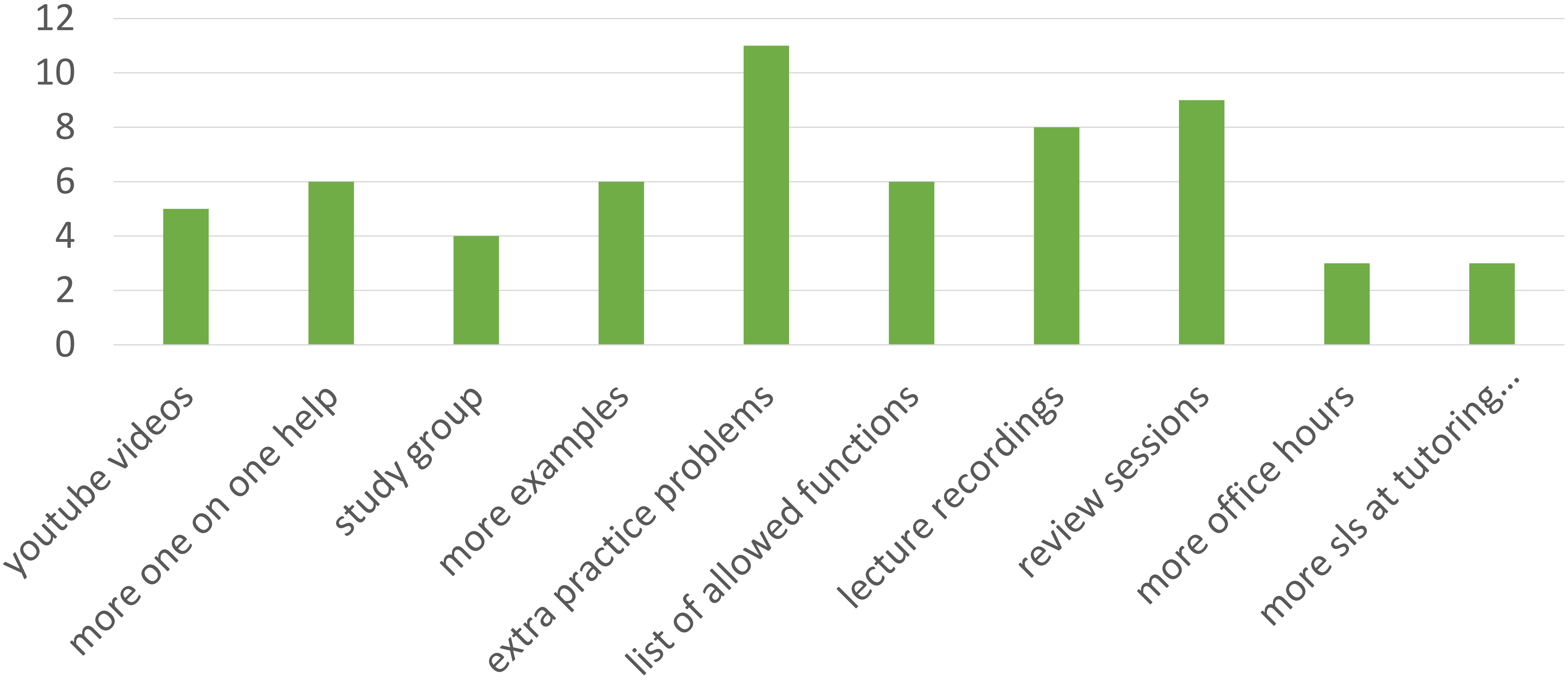
What you like about section



What you dislike about section



Resources you would like



Least favorite

- Handwritten exams
- Quizzes
- Time of deadlines
- Daily problems
- Collaboration
- Projects

Favorite

- Problem solving
- Joy of getting a problem right
- Programming
- Resources
- Cartoons

"The projects are probably one of the most satisfying pieces of schoolwork I've ever done. I just feel a real sense of accomplishment when I complete one."