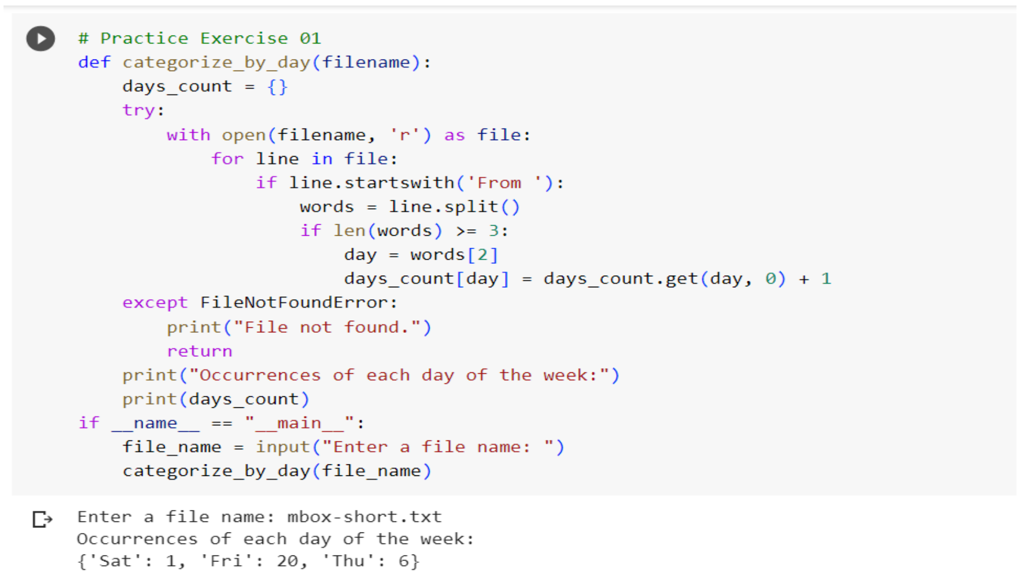
Practice Exercise 01: Write a program that categorizes each mail message by which day of the week the commit was done. To do this look for lines that start with “From”,then look for the third word and keep a running count of each of the days of the week. At the end of the program print out the contents of your dictionary (order does not matter).

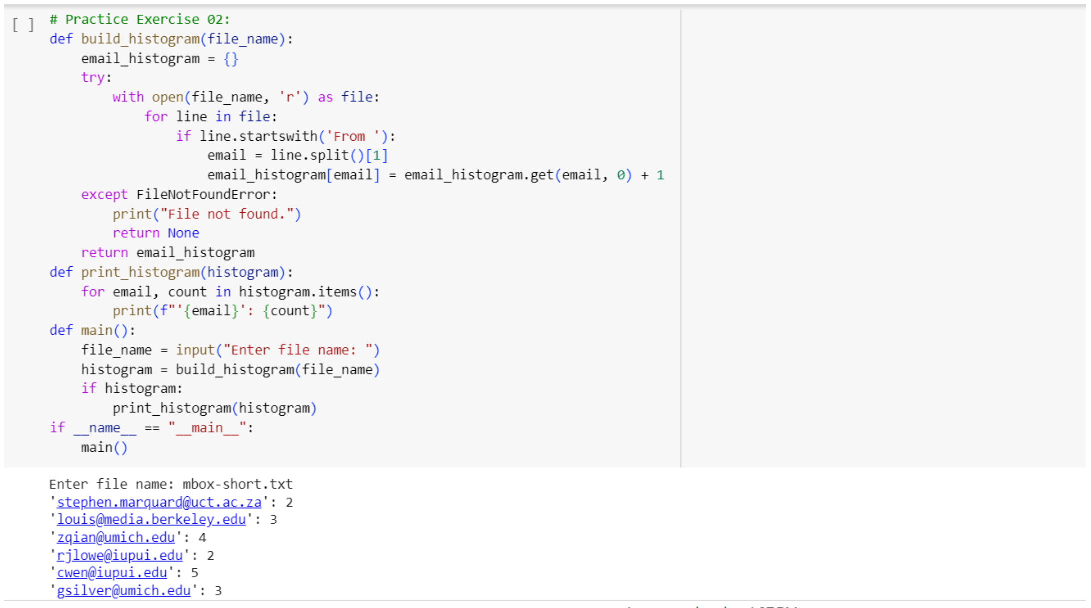


```python
# Practice Exercise 01
def categorize_by_day(filename):
    days_count = {}
    try:
        with open(filename, 'r') as file:
            for line in file:
                if line.startswith('From '):
                    words = line.split()
                    if len(words) >= 3:
                        day = words[2]
                        days_count[day] = days_count.get(day, 0) + 1
    except FileNotFoundError:
        print("File not found.")
        return
    print("Occurrences of each day of the week:")
    print(days_count)
if __name__ == "__main__":
    file_name = input("Enter a file name: ")
    categorize_by_day(file_name)
```

```
Enter a file name: mbox-short.txt
Occurrences of each day of the week:
{'Sat': 1, 'Fri': 20, 'Thu': 6}
```

Figure 01: program 1

Practice Exercise 02: Write a program to read through a mail log, build a histogram using a dictionary to count how many messages have come from each email address and print the dictionary.

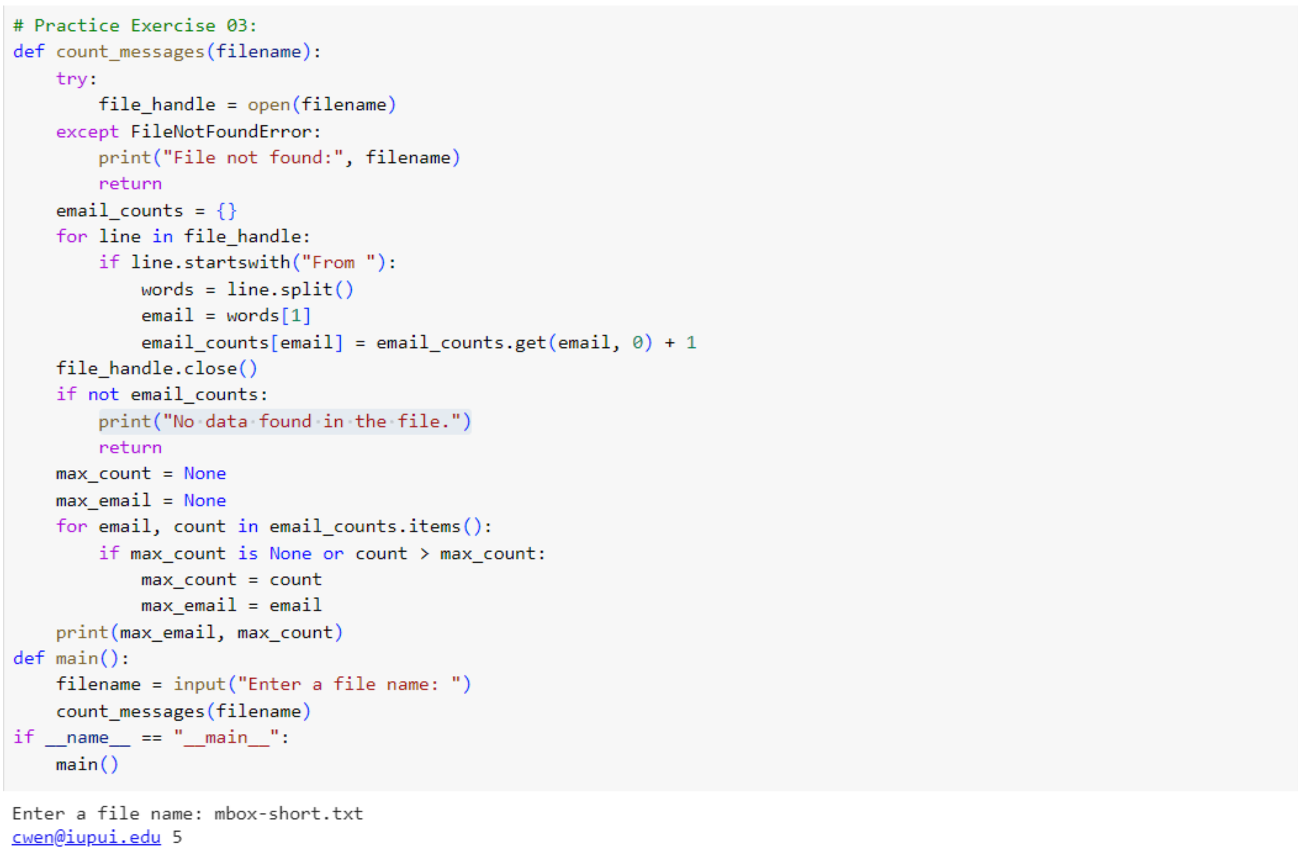


```python
# Practice Exercise 02:
def build_histogram(file_name):
    email_histogram = {}
    try:
        with open(file_name, 'r') as file:
            for line in file:
                if line.startswith('From '):
                    email = line.split()[1]
                    email_histogram[email] = email_histogram.get(email, 0) + 1
    except FileNotFoundError:
        print("File not found.")
        return None
    return email_histogram
def print_histogram(histogram):
    for email, count in histogram.items():
        print(f"'{email}': {count}")
def main():
    file_name = input("Enter file name: ")
    histogram = build_histogram(file_name)
    if histogram:
        print_histogram(histogram)
if __name__ == "__main__":
    main()
```

```
Enter file name: mbox-short.txt
'stephen.marquard@uct.ac.za': 2
'louis@media.berkeley.edu': 3
'zqian@umich.edu': 4
'rjlowe@iupui.edu': 2
'cwen@iupui.edu': 5
'gsilver@umich.edu': 3
```

Figure 02: program 2

Practice Exercise 03: Add code to the above program to figure out who has the most messages in the file. After all the data has been read and the dictionary has been created, look through the dictionary using a maximum loop to find who has the most messages and print how many messages the person has.

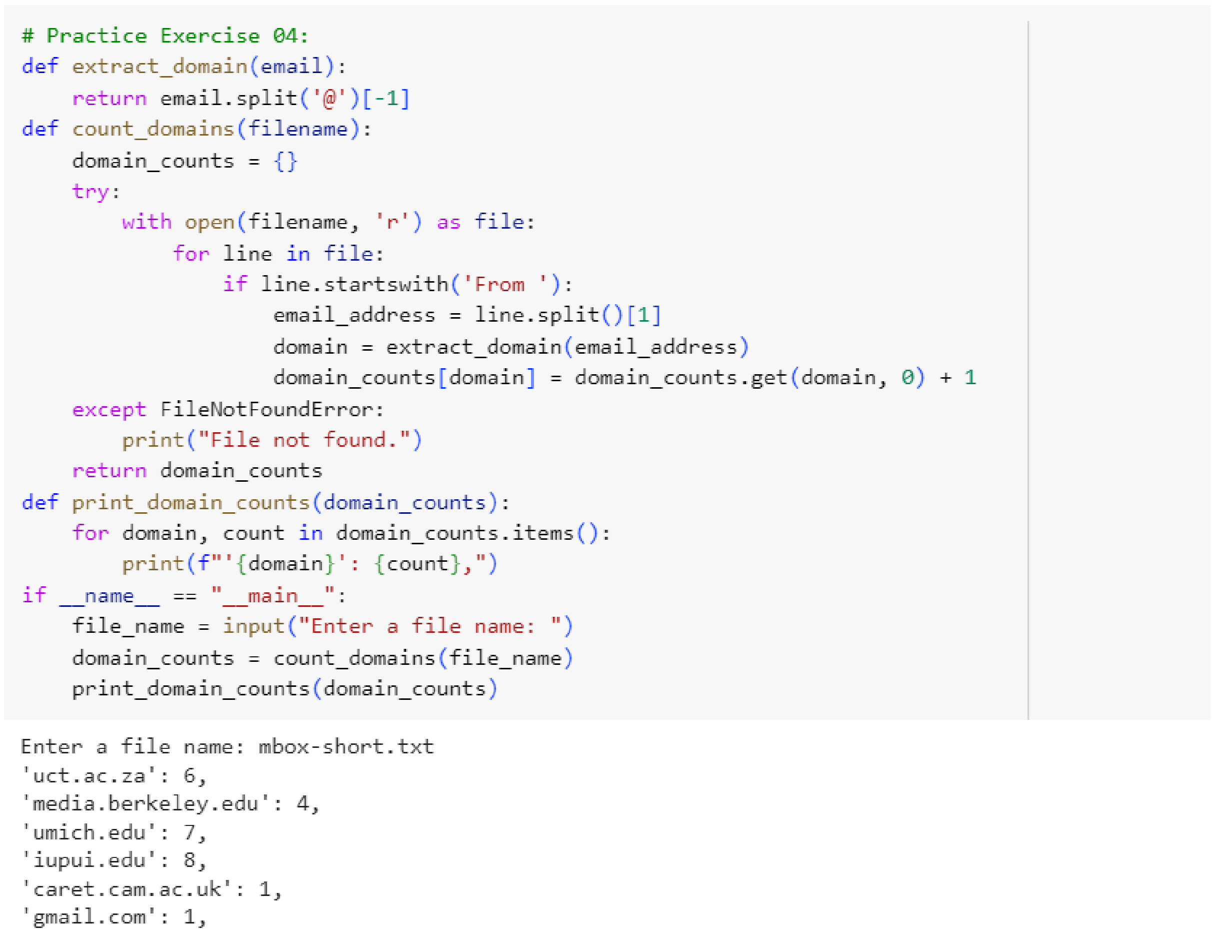


```python
# Practice Exercise 03:
def count_messages(filename):
    try:
        file_handle = open(filename)
    except FileNotFoundError:
        print("File not found:", filename)
        return
    email_counts = {}
    for line in file_handle:
        if line.startswith("From "):
            words = line.split()
            email = words[1]
            email_counts[email] = email_counts.get(email, 0) + 1
    file_handle.close()
    if not email_counts:
        print("No data found in the file.")
        return
    max_count = None
    max_email = None
    for email, count in email_counts.items():
        if max_count is None or count > max_count:
            max_count = count
            max_email = email
    print(max_email, max_count)
def main():
    filename = input("Enter a file name: ")
    count_messages(filename)
if __name__ == "__main__":
    main()
```

```
Enter a file name: mbox-short.txt
cwen@iupui.edu 5
```

Figure 03: program 3

Practice Exercise 04: This program records the domain name (instead of the address) where the message was sent from instead of who the mail came from (i.e., the whole email address). At the end of the program, print out the contents of your dictionary

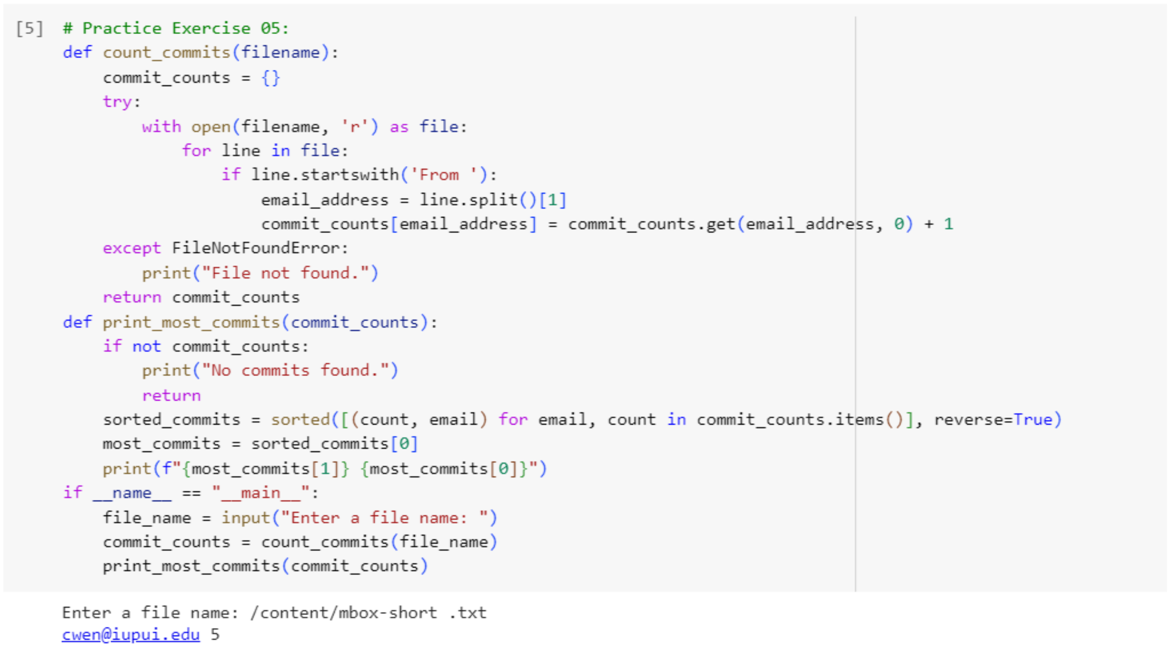


```python
# Practice Exercise 04:
def extract_domain(email):
    return email.split('@')[-1]
def count_domains(filename):
    domain_counts = {}
    try:
        with open(filename, 'r') as file:
            for line in file:
                if line.startswith('From '):
                    email_address = line.split()[1]
                    domain = extract_domain(email_address)
                    domain_counts[domain] = domain_counts.get(domain, 0) + 1
    except FileNotFoundError:
        print("File not found.")
    return domain_counts
def print_domain_counts(domain_counts):
    for domain, count in domain_counts.items():
        print(f"'{domain}': {count},")
if __name__ == "__main__":
    file_name = input("Enter a file name: ")
    domain_counts = count_domains(file_name)
    print_domain_counts(domain_counts)
```

```
Enter a file name: mbox-short.txt
'uct.ac.za': 6,
'media.berkeley.edu': 4,
'umich.edu': 7,
'iupui.edu': 8,
'caret.cam.ac.uk': 1,
'gmail.com': 1,
```

Figure 04: program 4

Practice Exercise 05: Revise a previous program as follows: Read and parse the “From” lines and pull out the addresses from the line. Count the number of messages from each person using a dictionary. After all the data has been read, print the person with the most commits by creating a list of (count, email) tuples from the dictionary. Then sort the list in reverse order and print out the person who has the most commits.

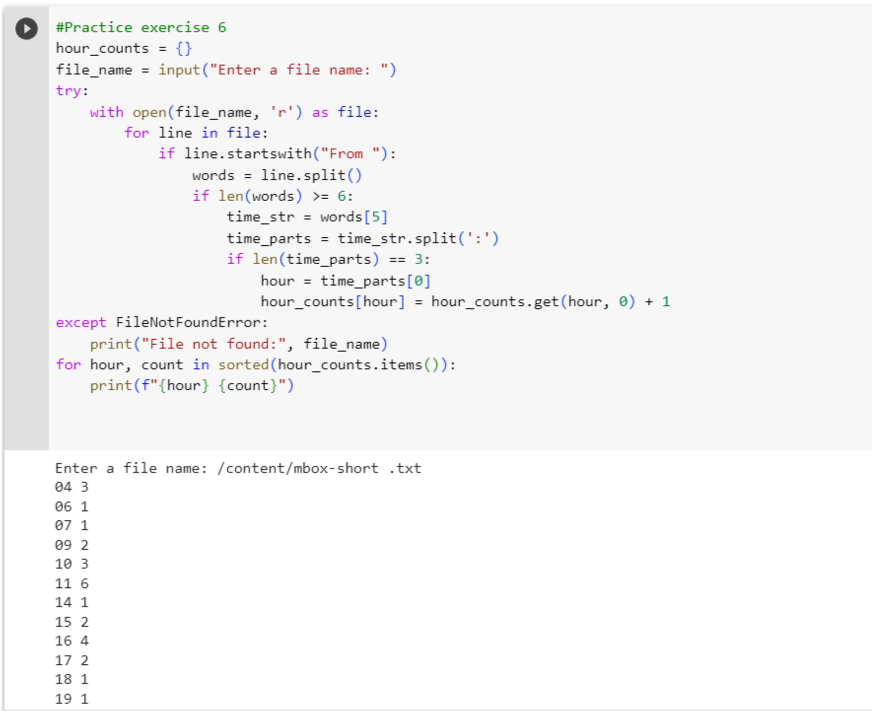


```
[5] # Practice Exercise 05:
    def count_commits(filename):
        commit_counts = {}
        try:
            with open(filename, 'r') as file:
                for line in file:
                    if line.startswith('From '):
                        email_address = line.split()[1]
                        commit_counts[email_address] = commit_counts.get(email_address, 0) + 1
        except FileNotFoundError:
            print("File not found.")
        return commit_counts
    def print_most_commits(commit_counts):
        if not commit_counts:
            print("No commits found.")
            return
        sorted_commits = sorted([(count, email) for email, count in commit_counts.items()], reverse=True)
        most_commits = sorted_commits[0]
        print(f"{most_commits[1]} {most_commits[0]}")
    if __name__ == "__main__":
        file_name = input("Enter a file name: ")
        commit_counts = count_commits(file_name)
        print_most_commits(commit_counts)

    Enter a file name: /content/mbox-short .txt
    cwen@iupui.edu 5
```

Figure 05: program 5

Practice Exercise 06: This program counts the distribution of the hour of the day for each of the messages. You can pull the hour from the “From” line by finding the time string and then splitting that string into parts using the colon character. Once you have accumulated the counts for each hour, print out the counts, one per line, sorted by hour as shown below



```
#Practice exercise 6
hour_counts = {}
file_name = input("Enter a file name: ")
try:
    with open(file_name, 'r') as file:
        for line in file:
            if line.startswith("From "):
                words = line.split()
                if len(words) >= 6:
                    time_str = words[5]
                    time_parts = time_str.split(':')
                    if len(time_parts) == 3:
                        hour = time_parts[0]
                        hour_counts[hour] = hour_counts.get(hour, 0) + 1
except FileNotFoundError:
    print("File not found:", file_name)
for hour, count in sorted(hour_counts.items()):
    print(f"{hour} {count}")

Enter a file name: /content/mbox-short .txt
04 3
06 1
07 1
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

Figure 06: program 6