```python
print("hello world"); #printing hello world
```

```
hello world
```

```python
#concatination
a= 6;
b= 4;
d= "Anir";
e= "udh";
c= a+b; #concatination of number or addition of numbers
f= d+e; #concatination of strings
print(c)
print(f)
```

```
10
Vibhu
```

```python
#package installation
!pip install pygame
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: pygame in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (2.5.2)


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```python
#import the package
import pygame
```

```
pygame 2.5.2 (SDL 2.28.3, Python 3.10.7)
Hello from the pygame community.
https://www.pygame.org/contribute.html
```

```python
#initializing
pygame.mixer.init()
```

```python
#creation of internal instance of the audio stream
pygame.mixer.music.load(r"C:\Users\Vibhu Aadhav.M\Downloads\sample-
3s.wav")
```

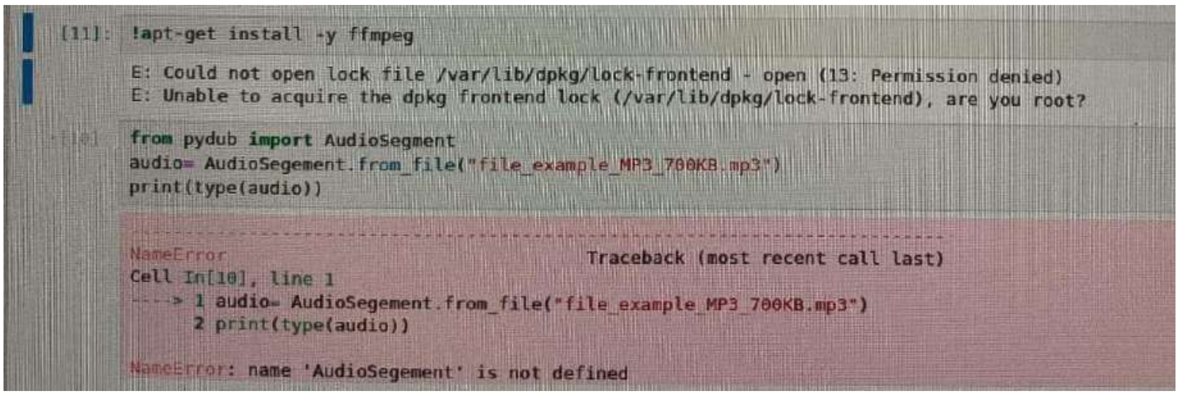
```python
print("Playing sound...")
pygame.mixer.music.play()
```

```
Playing sound...
```

```python
while pygame.mixer.music.get_busy():
    pygame.time.Clock().tick(10)
```

Lab-1

NAME:VIBHU AADHAV.M
REGNO:22MID0141

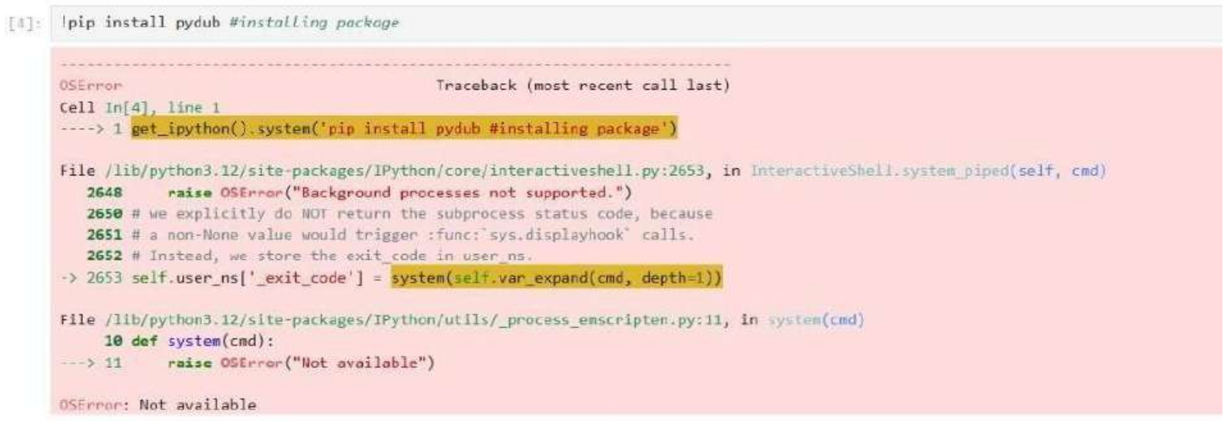


```
[11]: !apt-get install -y ffmpeg

      E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
      E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?

[10]: from pydub import AudioSegment
      audio= AudioSegement.from_file("file_example_MP3_700KB.mp3")
      print(type(audio))

      ---------------------------------------------------------------------------
      NameError                                 Traceback (most recent call last)
      Cell In[10], line 1
      ----> 1 audio= AudioSegement.from_file("file_example_MP3_700KB.mp3")
            2 print(type(audio))

      NameError: name 'AudioSegement' is not defined
```

Figure 1: lab system

I tried to use the pydub library to play an audio file. Even though I installed the library correctly, I faced two main issues:

1. Spelling Mistake: I accidentally wrote AudioSegement instead of the correct spelling AudioSegment. This caused Python to say the name is not defined.
2. Permission Problem While Installing ffmpeg:
   pydub needs a tool called ffmpeg to play audio files. I tried to install it, but it failed because the Jupyter environment didn't have permission to do so (it needs admin access, which I didn't have).



```
[4]: !pip install pydub #installing package

     ---------------------------------------------------------------------------
     OSError                                   Traceback (most recent call last)
     Cell In[4], line 1
     ----> 1 get_ipython().system('pip install pydub #installing package')

     File /lib/python3.12/site-packages/IPython/core/interactiveshell.py:2653, in InteractiveShell.system_piped(self, cmd)
        2648         raise OSError("Background processes not supported.")
        2650 # we explicitly do NOT return the subprocess status code, because
        2651 # a non-None value would trigger :func:`sys.displayhook` calls.
        2652 # Instead, we store the exit_code in user_ns.
     -> 2653 self.user_ns['_exit_code'] = system(self.var_expand(cmd, depth=1))

     File /lib/python3.12/site-packages/IPython/utils/_process_emscripten.py:11, in system(cmd)
          10 def system(cmd):
     ---> 11     raise OSError("Not available")

     OSError: Not available
```

Figure 2: pc

Since I couldn't install pydub because the system didn't allow it, I looked for another way to play audio files. That's when I found an option called pygame.

With pygame, I was able to play sound without any problems

I didn't have to install anything extra or ask for special permissions for pygame.

```
my_string = "Hello, Python!"
print(my_string)
```

```
Hello, Python!
```

```
my_string = "Python"
print(my_string[0])  # Output: P
print(my_string[-1]) # Output: n (last character)
```

```
P
n
```

```
my_string = "Programming"
print(my_string[0:6])    # Output: Progra
print(my_string[3:])     # Output: gramming
```

```
Progra
gramming
```

```
first = "Data"
second = "Science"
combined = first + " " + second
print(combined)  # Output: Data Science
```

```
Data Science
```

```
print("Hi! " * 3)   # Output: Hi! Hi! Hi!
```

```
Hi! Hi! Hi!
```

```
name = "Alice"
age = 25
print(f"My name is {name} and I am {age} years old.")
```

```
My name is Alice and I am 25 years old.
```

```
quote = "She said, \"Python is fun!\""
print(quote)
```

```
She said, "Python is fun!"
```

```
multiline = """This is
a multiline
string."""
print(multiline)
```

```
This is
a multiline
string.
```

```
text = "   Machine Learning!   "
print(text.strip())         # Removes leading/trailing spaces
print(text.lower())         # Converts to lowercase
print(text.replace("!", "."))  # Replaces characters
```

```
Machine Learning!
   machine learning!
   Machine Learning.
```

```
text = "Artificial Intelligence"
print("Intel" in text)    # Output: True
```

```
True
```

```
name = "Vibhu" age
= 21
print(f"My name is {name} and I am {age} years old.")
```

```
My name is Vibhu and I am 21 years old.
```

```
name = "Harshini"
age = 21
print("My name is {} and I am {} years old.".format(name, age))
```

```
My name is Vibhu and I am 21 years old.
```

```
text = "python programming"
print(text.upper())  # Output: PYTHON PROGRAMMING
```

```
PYTHON PROGRAMMING
```

```
text = "PYTHON Programming"
print(text.lower())  # Output: python programming
```

```
python programming
```

```
text = "machine learning with python"
print(text.title())  # Output: Machine Learning With Python
```

```
Machine Learning With Python
```

```
text = "   Hello World"
print(text.lstrip())  # Output: 'Hello World' (without leading spaces)
```

```
Hello World
```

```
!pip install nltk
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: nltk in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages  (3.9.1)
Requirement already satisfied: click in c:\program files\python310\
lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (from nltk) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in c:\users\harshini
supriya\appdata\roaming\python\python310\site-packages (from nltk)
(2024.11.6)
Requirement already satisfied: tqdm in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (from nltk) (4.67.1)
Requirement already satisfied: colorama in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (from click->nltk)
(0.4.6)


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
import nltk
from urllib import request
from nltk.tokenize import word_tokenize

# Download the tokenizer data (do this only once)
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to C:\Users\Harshini
[nltk_data]     Supriya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
url = "https://www.gutenberg.org/files/98/98-0.txt"

response = request.urlopen(url)
raw_text = response.read().decode('utf-8')

tokens = word_tokenize(raw_text)
print(tokens[:100])  # Display first 100 tokens
```

```
['\ufeffThe', 'Project', 'Gutenberg', 'eBook', 'of', 'A', 'Tale',
'of', 'Two', 'Cities', ',', 'by', 'Charles', 'Dickens', 'This',
'eBook', 'is', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in',
'the', 'United', 'States', 'and', 'most', 'other', 'parts', 'of',
'the', 'world', 'at', 'no', 'cost', 'and', 'with', 'almost', 'no',
'restrictions', 'whatsoever', '.', 'You', 'may', 'copy', 'it', ',',
'give', 'it', 'away', 'or', 're-use', 'it', 'under', 'the', 'terms',
'of', 'the', 'Project', 'Gutenberg', 'License', 'included', 'with',
'this', 'eBook', 'or', 'online', 'at', 'www.gutenberg.org', '.', 'If',
'you', 'are', 'not', 'located', 'in', 'the', 'United', 'States', ',',
'you', 'will', 'have', 'to', 'check', 'the', 'laws', 'of', 'the',
'country', 'where', 'you', 'are', 'located', 'before', 'using',
'this', 'eBook']
```

```
!pip install nltk html2text
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: nltk in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (3.9.1)
Requirement already satisfied: html2text in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (2025.4.15)
Requirement already satisfied: click in c:\program files\python310\
lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (from nltk) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in c:\users\harshini
supriya\appdata\roaming\python\python310\site-packages (from nltk)
(2024.11.6)
Requirement already satisfied: tqdm in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (from nltk) (4.67.1)
Requirement already satisfied: colorama in c:\users\harshini supriya\
appdata\roaming\python\python310\site-packages (from click->nltk)
(0.4.6)


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
import nltk
import html2text
from urllib import request
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to C:\Users\Harshini
[nltk_data]     Supriya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
url = "https://www.gutenberg.org/cache/epub/1497/pg1497-images.html"
response = request.urlopen(url)
html_content = response.read().decode('utf-8')  # convert to UTF-8

text = html2text.html2text(html_content)

tokens = word_tokenize(text)

vocabulary = sorted(set(tokens))  # removes duplicates and sorts
print(vocabulary[:100])  # print first 100 unique words
print(f"\nTotal unique words (vocabulary size): {len(vocabulary)}")
```

```
['!', '#', '$', '%', '(', ')', '*', '+', ',', '-', '--', '-end', '.',
'...', '//www.gutenberg.org', '//www.gutenberg.org/donate/', '1',
'1.A', '1.B', '1.C', '1.D', '1.E', '1.E.1', '1.E.2', '1.E.3', '1.E.4',
'1.E.5', '1.E.6', '1.E.7', '1.E.8', '1.E.9', '1.F', '1.F.1', '1.F.2',
'1.F.3', '1.F.4', '1.F.5', '1.F.6', '1/1', '1/2', '1/3', '10',
'10,000', '100', '1000', '1000.', '11', '12', '13', '1497', '150',
'1500', '1514', '1568-1639', '16', '1688', '17,500', '18', '193',
'1998', '1\\', '1st', '2', '2/1', '2/2', '20', '200,000', '2001',
'2021', '21', '215', '216', '25', '27', '2700', '2\\', '2nd', '3',
'3/2', '30', '300,000', '33', '34', '35', '365', '398-381', '3:4',
'3\\', '3rd', '4', '4/3', '400', '400/9', '411', '435', '436', '456',
'47', '48', '49']

Total unique words (vocabulary size): 12353
```

```python
import string
from nltk.corpus import stopwords
nltk.download('stopwords')

words = [word.lower() for word in tokens if word.isalpha()]  # remove
punctuation/numbers
words = [word for word in words if word not in
stopwords.words('english')]

vocab_clean = sorted(set(words))
print(vocab_clean[:100])
print(f"\nClean vocabulary size: {len(vocab_clean)}")
```

```
[nltk_data] Downloading package stopwords to C:\Users\Harshini
[nltk_data]     Supriya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

['abandoned', 'abate', 'abated', 'abates', 'abdera', 'abhor',
'abhorrence', 'abide', 'abiding', 'ability', 'abject', 'able',
'abler', 'abnegation', 'abnormal', 'abode', 'abolished', 'abolishes',
'abolition', 'abominable', 'abominate', 'abominates', 'aborigines',
'abound', 'abounds', 'abroad', 'absence', 'absolute', 'absolutely',
'absolution', 'absorb', 'absorbed', 'absorbs', 'abstain', 'abstained',
'abstainer', 'abstains', 'abstinence', 'abstract', 'abstracted',
'abstraction', 'abstractions', 'abstracts', 'absurd', 'absurdities',
'absurdity', 'absurdum', 'abundance', 'abundantly', 'abuse',
'abusing', 'academic', 'accent', 'accept', 'acceptance', 'accepted',
'accepting', 'access', 'accessed', 'accessible', 'accessory',
'accident', 'accidental', 'accidents', 'accommodate', 'accommodated',
'accommodation', 'accommodations', 'accompanied', 'accompanies',
'accompaniment', 'accompaniments', 'accompany', 'accompanying',
'accomplish', 'accomplished', 'accomplishing', 'accomplishments',
'accord', 'accordance', 'according', 'accordingly', 'accords',
'account', 'accounted', 'accruing', 'accumulate', 'accumulating',
'accumulation', 'accumulations', 'accuracy', 'accurate', 'accursed',
'accusation', 'accusations', 'accuse', 'accused', 'accuser',
'accustomed', 'accustoming']

Clean vocabulary size: 10175
```

Lab-2 (15/7/25)
NAME:VIBHU AADHAV.M
REGNO:22MID0141

1. Source of the Website:

For my scraping task, I used Project Gutenberg – https://www.gutenberg.org.

I specifically scraped the text from the classic book A Tale of Two Cities by Charles Dickens. (Link: https://www.gutenberg.org/files/98/98-0.txt)

2. What Data I Scraped and Why:

I scraped the full plain text of the book.

The main reason was to practice text processing techniques like tokenization, which are used in Natural Language Processing (NLP).

This was done purely for learning purposes, not for any commercial use.

3. Under What Clause It Is Allowed:

Project Gutenberg makes it very clear that their books are in the public domain.

This means the content is free to use, copy, and share legally, especially for personal or educational use like mine.They even mention:

"This eBook is for the use of anyone anywhere… with almost no restrictions whatsoever."

So scraping this text is completely allowed.

4. Copyright Clause :

Since the book is in the public domain, there is no copyright on it anymore.

That's why I didn't need any special permission to use or scrape the content.

It's open to everyone, which is perfect for students and researchers.

```python
# Travel Itinerary Planner

# Format: [ [date_string, [ [time_slot, destination], ... ] ], ... ]
itinerary = []

# Function to add a new day's plan
def add_day():
    date = input("Enter travel date (YYYY-MM-DD): ")

    time_slots = ["Morning", "Afternoon", "Evening"]
    daily_plan = []

    for slot in time_slots:
        dest = input(f"Where are you planning to go in the {slot}? ")
        daily_plan.append([slot, dest])  # append(): adds a time slot
entry to daily_plan

    itinerary.append([date, daily_plan])  # append(): adds a full day
to itinerary
    print(f"Plan for {date} added.")

# Function to display the entire itinerary
def view_itinerary():
    if not itinerary:
        print("Itinerary is empty.")
        return

    for i, (date, plans) in enumerate(itinerary, start=1):  #
enumerate(): to number days
        print(f"\nDay {i} - {date}")
        for slot, dest in plans:
            print(f"  {slot}: {dest}")

# Function to edit a specific slot in a day
def edit_day():
    date = input("Enter the date to edit: ")

    for day_index, (day, plans) in enumerate(itinerary):
        if day == date:
            print(f"Current plan for {date}:")
            for i, (slot, dest) in enumerate(plans, start=1):
                print(f"{i}. {slot}: {dest}")
            choice = int(input("Enter the slot number to edit (1-3):
"))
            new_place = input("Enter new destination: ")
            itinerary[day_index][1][choice - 1][1] = new_place  #
indexing: update nested value
            print("Updated successfully.")
            return
    print("Date not found.")
```

```python
# Function to remove an entire day's plan
def remove_day():
    date = input("Enter date to remove: ")
    for day in itinerary:
        if day[0] == date:
            itinerary.remove(day)  # remove(): deletes a matching item
            print(f"Plan for {date} removed.")
            return
    print("Date not found.")

# Function to show summary statistics using list operations
def summary():
    if not itinerary:
        print("No data to summarize.")
        return

    # Flatten the destinations into a single list
    all_places = [dest for _, plans in itinerary for _, dest in plans]
# list comprehension

    # Count total places and unique places
    total = len(all_places)  # len(): total number of places
    unique_places = list(set(all_places))  # set(): removes
duplicates; list(): converts back

    print(f"Total destinations planned: {total}")
    print(f"Unique destinations: {len(unique_places)}")

    # Find most and least frequent destination
    if all_places:
        most_visited = max(set(all_places), key=all_places.count)  #
max(): most frequent
        least_visited = min(set(all_places), key=all_places.count)  #
min(): least frequent
        print(f"Most visited place: {most_visited}")
        print(f"Least visited place: {least_visited}")

    # count(): check how many times a destination appears
    for place in unique_places:
        print(f"{place} - {all_places.count(place)} time(s)")

# Function to search for any place in the itinerary
def search_place():
    query = input("Enter destination to search for: ").lower()
    found = False
    for date, plans in itinerary:
        for slot, dest in plans:
            if query in dest.lower():  # in: membership test
                print(f"{dest} found on {date} during {slot}")
                found = True
    if not found:
        print("Destination not found.")

# Function to clear all entries in the itinerary
def clear_all():
    confirm = input("Are you sure you want to delete all plans?
(yes/no): ")
    if confirm.lower() == "yes":
        itinerary.clear()  # clear(): removes all elements
        print("Itinerary cleared.")

# Function to demonstrate more list functions
def advanced_features():
    if not itinerary:
        print("Itinerary is empty.")
        return

    # Demonstrating copy()
    copied_itinerary = itinerary.copy()
    print("Copied itinerary (using copy()):")
    for date, plans in copied_itinerary:
        print(f"{date}: {[place for _, place in plans]}")

    # Demonstrating extend(): adding extra destinations to a specific
day
    extra = [["Night", "Street Food Tour"], ["Late Night", "City
Lights"]]
    itinerary[0][1].extend(extra)  # extend(): adds multiple sublists
    print("Extra slots added to the first day using extend().")

    # Demonstrating insert()
    itinerary[0][1].insert(1, ["Mid-Morning", "City Park"])  #
insert(): insert at index
    print("Inserted 'Mid-Morning' slot.")

    # Demonstrating pop()
    removed_item = itinerary[0][1].pop()  # pop(): removes and returns
last item
    print(f"Removed the last slot: {removed_item}")

    # Demonstrating slicing
    first_two_days = itinerary[:2]  # slicing: get first 2 days
    print(f"First 2 days (using slicing): {first_two_days}")

    # Demonstrating del
    if len(itinerary[0][1]) >= 2:
        del itinerary[0][1][0]  # del: remove by index
        print("Deleted first slot using del.")
```

```python
# Main menu loop
while True:
    print("\nMenu")
    print("1. Add Day")
    print("2. View Itinerary")
    print("3. Edit a Day")
    print("4. Remove a Day")
    print("5. Summary Statistics")
    print("6. Search a Place")
    print("7. Clear All")
    print("8. Advanced List Features")
    print("9. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':
        add_day()
    elif choice == '2':
        view_itinerary()
    elif choice == '3':
        edit_day()
    elif choice == '4':
        remove_day()
    elif choice == '5':
        summary()
    elif choice == '6':
        search_place()
    elif choice == '7':
        clear_all()
    elif choice == '8':
        advanced_features()
    elif choice == '9':
        print("Goodbye!")
        break
    else:
        print("Invalid input.")
```

```
Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit
```

```
Enter your choice:  1
Enter travel date (YYYY-MM-DD):  2026-01-22
Where are you planning to go in the Morning?  Eiffel Tower
Where are you planning to go in the Afternoon? Louvre Museum
Where are you planning to go in the Evening?  Seine River Cruise

Plan for 2026-01-22 added.

Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit

Enter your choice:  1
Enter travel date (YYYY-MM-DD):  2026-02-27
Where are you planning to go in the Morning?   Notre-Dame Cathedral
Where are you planning to go in the Afternoon?   Montmartre
Where are you planning to go in the Evening? Moulin Rouge

Plan for 2026-02-27 added.

Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit

Enter your choice:  2


Day 1 - 2026-01-22
  Morning: Eiffel Tower
  Afternoon: Louvre Museum
  Evening: Seine River Cruise

Day 2 - 2026-02-27
  Morning:  Notre-Dame Cathedral
  Afternoon:  Montmartre
  Evening: Moulin Rouge
```

```
Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit

Enter your choice:  3
Enter the date to edit: 2025-08-02

Date not found.

Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit

Enter your choice:  6
Enter destination to search for:  louvre

Louvre Museum found on 2026-01-22 during Afternoon

Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit

Enter your choice:  5

Total destinations planned: 6
Unique destinations: 6
Most visited place: Notre-Dame Cathedral
Least visited place: Notre-Dame Cathedral
 Notre-Dame Cathedral - 1 time(s)
Louvre Museum - 1 time(s)
```

```
Seine River Cruise - 1 time(s)
 Montmartre - 1 time(s)
Moulin Rouge - 1 time(s)
Eiffel Tower - 1 time(s)

Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit

Enter your choice:  5

Total destinations planned: 6
Unique destinations: 6
Most visited place: Notre-Dame Cathedral
Least visited place: Notre-Dame Cathedral
 Notre-Dame Cathedral - 1 time(s)
Louvre Museum - 1 time(s)
Seine River Cruise - 1 time(s)
 Montmartre - 1 time(s)
Moulin Rouge - 1 time(s)
Eiffel Tower - 1 time(s)

Menu
1. Add Day
2. View Itinerary
3. Edit a Day
4. Remove a Day
5. Summary Statistics
6. Search a Place
7. Clear All
8. Advanced List Features
9. Exit
```

```python
import math

print("100! =", math.factorial(100))
```

```
100! =
9332621544394415268169923885626670049071596826438162146859296389521759
9993229915608941463976156518286253697920827223758251185210916864000000
000000000000000000
```

```python
big_number = 9999999999999999999999999999999999
squared = big_number ** 2
print("Big number squared:", squared)
```

```
Big number squared:
99999999999999999999999999999999980000000000000000000000000000000001
```

```python
s = "hello"
print("Original string:", s)
print("ID of original string:", id(s))

# Try to modify the string (actually creates a new string)
s = s + " world"
print("\nModified string:", s)
print("ID after modification:", id(s))
```

```
Original string: hello
ID of original string: 4401955776

Modified string: hello world
ID after modification: 4405756528
```

```python
# Typing special characters (like emojis) using Unicode
print("Smiling Face:", "\U0001F600")
print("Rocket:", "\U0001F680")
print("Laptop:", "\U0001F4BB")
print("Fire:", "\U0001F525")
print("Glowing Star:", "\U0001F31F")
```

```
Smiling Face: 😀
Rocket: 🚀
Laptop: 🚀
Fire: 🚀
Glowing Star:
🚀
```

```python
char = 'A'
print("ASCII of", char, "is", ord(char))  # Output: 65

# Unicode of a non-ASCII character
char2 = '₹'
print("Unicode of", char2, "is", ord(char2))  # Output: 8377
```

```
ASCII of A is 65
Unicode of ₹ is 8377
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize
import nltk

nltk.download('punkt')
nltk.download('punkt_tab')

url = "https://www.gutenberg.org/files/1342/1342-h/1342-h.htm"  #
Pride and Prejudice
response = requests.get(url)
html = response.text

soup = BeautifulSoup(html, 'html.parser')
text = soup.get_text()

words = word_tokenize(text)

print("First 500 characters:\n", text[:500])
print("\nTotal words:", len(words))
print("First 20 words:", words[:20])
# same thing runs on google colab but fails to run on jupyter for me
```

```
[nltk_data] Error loading punkt: <urlopen error [SSL:
[nltk_data]     CERTIFICATE_VERIFY_FAILED] certificate verify failed:
[nltk_data]     unable to get local issuer certificate (_ssl.c:1000)>
[nltk_data] Error loading punkt_tab: <urlopen error [SSL:
[nltk_data]     CERTIFICATE_VERIFY_FAILED] certificate verify failed:
[nltk_data]     unable to get local issuer certificate (_ssl.c:1000)>
```

```
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call
last)
Cell In[9], line 20
     16 soup = BeautifulSoup(html, 'html.parser')
     17 text = soup.get_text()
---> 20 words = word_tokenize(text)
     23 print("First 500 characters:\n", text[:500])
     24 print("\nTotal words:", len(words))

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/nltk/tokenize/__init__.py:142, in word_tokenize(text,
language, preserve_line)
    127 def word_tokenize(text, language="english",
preserve_line=False):
    128     """
    129     Return a tokenized copy of *text*,
    130     using NLTK's recommended word tokenizer
   (...)    140     :type preserve_line: bool
    141     """
--> 142     sentences = [text] if preserve_line else
sent_tokenize(text, language)
    143     return [
    144         token for sent in sentences for token in
_treebank_word_tokenizer.tokenize(sent)
    145     ]

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/nltk/tokenize/__init__.py:119, in sent_tokenize(text,
language)
    109 def sent_tokenize(text, language="english"):
    110     """
    111     Return a sentence-tokenized copy of *text*,
    112     using NLTK's recommended sentence tokenizer
  (...)     117     :param language: the model name in the Punkt
corpus
    118     """
--> 119     tokenizer = _get_punkt_tokenizer(language)
    120     return tokenizer.tokenize(text)

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/nltk/tokenize/__init__.py:105, in
_get_punkt_tokenizer(language)
     96 @functools.lru_cache
     97 def _get_punkt_tokenizer(language="english"):
     98     """
     99     A constructor for the PunktTokenizer that utilizes
    100     a lru cache for performance.
   (...)    103     :type language: str
    104     """
--> 105     return PunktTokenizer(language)

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/nltk/tokenize/punkt.py:1744, in
PunktTokenizer.__init__(self, lang)
   1742 def __init__(self, lang="english"):
   1743     PunktSentenceTokenizer.__init__(self)
-> 1744     self.load_lang(lang)

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/nltk/tokenize/punkt.py:1749, in
PunktTokenizer.load_lang(self, lang)
```

```
   1746 def load_lang(self, lang="english"):
   1747     from nltk.data import find
-> 1749     lang_dir = find(f"tokenizers/punkt_tab/{lang}/")
   1750     self._params = load_punkt_params(lang_dir)
   1751     self._lang = lang

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/nltk/data.py:579, in find(resource_name, paths)
    577 sep = "*" * 70
    578 resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579 raise LookupError(resource_not_found)

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/Users/sujith10x/nltk_data'
    - '/Library/Frameworks/Python.framework/Versions/3.12/nltk_data'
    -
'/Library/Frameworks/Python.framework/Versions/3.12/share/nltk_data'
    -
'/Library/Frameworks/Python.framework/Versions/3.12/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

```
# --- List slicing example ---
m = [0, 1, 2, 3, 4, 5]

print(m[1:3])     # elements from index 1 to 2
print(m[2:])      # elements from index 2 to end
print(m[-2:])     # last two elements
print(m[2:1])     # empty list because start > end
```

```
[1, 2]
[2, 3, 4, 5]
[4, 5]
[]
```

```
# --- Shallow and deep copy example ---
l1 = m[:]    # deep copy using slicing
l2 = m       # shallow copy (just a reference)

print(m, id(m), id(l1), id(l2), l1, l2)
```

```
[0, 1, 2, 3, 4, 5] 2435043323648 2435026475840 2435043323648 [0, 1, 2,
3, 4, 5] [0, 1, 2, 3, 4, 5]
```

```
# --- Append example ---
m.append(4)
print(m, l1, l2)
```

```
[0, 1, 2, 3, 4, 5, 4] [0, 1, 2, 3, 4, 5] [0, 1, 2, 3, 4, 5, 4]
```

```
# --- Looping through a slice ---
w = m[:3]    # first three elements
for i in w:
    print(i)  # prints the first three elements
```

```
0
1
2
```

```
# --- Tuple slicing example ---
t = (50, 100, 150)
print(t[1:2])    # slicing a tuple, gives (100,)
```

```
(100,)
```

```
# 1. Off-by-One Behavior
# Off-by-one errors are super common when looping.
# Happens when I loop one time extra or one time less than I should.
# Remember: Python indexing starts at 0, but len() counts from 1
(length).

print("=== 1. Off-by-One Behavior ===")
numbers = [10, 20, 30, 40, 50]
```

```
# Wrong way (extra loop) → This will crash with IndexError
# for i in range(len(numbers) + 1):  # goes from 0 to 5, but index 5
doesn't exist
#     print(numbers[i])

# Correct way → only go till len(numbers)-1
for i in range(len(numbers)):  # range(5) → 0 to 4
    print(f"Index {i}, Value {numbers[i]}")
print()
```

```
=== 1. Off-by-One Behavior ===
Index 0, Value 10
Index 1, Value 20
Index 2, Value 30
Index 3, Value 40
Index 4, Value 50
```

```
# 2. Slicing a List
# Slicing syntax → list[start:end:step]
# start = inclusive, end = exclusive
# step = how much to jump

print("=== 2. Slicing a List ===")
fruits = ["apple", "banana", "cherry", "date", "elderberry", "fig"]

# Some slicing practice
print(fruits[1:4])    # from banana to date
print(fruits[:3])     # first 3 items
print(fruits[2:])     # from cherry till the end
print(fruits[::2])    # every 2nd fruit
print()
```

```
=== 2. Slicing a List ===
['banana', 'cherry', 'date']
['apple', 'banana', 'cherry']
['cherry', 'date', 'elderberry', 'fig']
['apple', 'cherry', 'elderberry']
```

```
# 3. Looping Through a List
print("=== 3. Looping Through a List ===")

# Easiest way: loop directly through elements
for fruit in fruits:
    print(fruit)

# Index-based loop (if I need positions)
for i in range(len(fruits)):
    print(f"Index {i} → {fruits[i]}")
print()
```

```
=== 3. Looping Through a List ===
apple
banana
cherry
date
elderberry
fig
Index 0 → apple
Index 1 → banana
Index 2 → cherry
Index 3 → date
Index 4 → elderberry
Index 5 → fig
```

```python
# 4. Looping Through a Slice
print("=== 4. Looping Through a Slice ===")

# Loop only over banana to elderberry
for fruit in fruits[1:5]:
    print(fruit)

# Loop over every alternate fruit
for fruit in fruits[::2]:
    print(fruit)
print()
```

```
=== 4. Looping Through a Slice ===
banana
cherry
date
elderberry
apple
cherry
elderberry
```

```python
# 5. Understanding Tuples
print("=== 5. Understanding Tuples ===")

# Tuples = like lists but can't be changed (immutable)
person = ("Alice", 25, "Engineer")

# Access normally
print("Name:", person[0])
print("Age:", person[1])
print("Profession:", person[2])

# Tuple unpacking
name, age, profession = person
print(f"Unpacked → Name: {name}, Age: {age}, Profession:
```

```
{profession}")

# Tuples can hold lists, other tuples, etc.
nested_tuple = (1, (2, 3), [4, 5])
print("Nested tuple:", nested_tuple)
```

```
=== 5. Understanding Tuples ===
Name: Alice
Age: 25
Profession: Engineer
Unpacked → Name: Alice, Age: 25, Profession: Engineer
Nested tuple: (1, (2, 3), [4, 5])
```

```
# --- Reverse a list using slicing ---
nums = [10, 20, 30, 40, 50]
print(nums[::-1])  # reverse the list
```

```
[50, 40, 30, 20, 10]
```

```
# --- Slice with step ---
print(nums[0:5:2])  # every 2nd element
```

```
[10, 30, 50]
```

```
# --- Copy list using slicing ---
copy_nums = nums[:]
print(copy_nums)
```

```
[10, 20, 30, 40, 50]
```

```
# --- Modify original list and see effect ---
nums.append(60)
print("Original:", nums)
print("Copy:", copy_nums)  # remains unchanged because of deep copy
```

```
Original: [10, 20, 30, 40, 50, 60]
Copy: [10, 20, 30, 40, 50]
```

```
# --- Nested slicing ---
nested = [[1, 2], [3, 4], [5, 6]]
print(nested[1:3])      # slice of sublists
print(nested[1][0:2])   # slice inside a nested list
```

```
[[3, 4], [5, 6]]
[3, 4]
```

```
# --- Loop through a tuple ---
my_tuple = ('red', 'green', 'blue')
for color in my_tuple:
    print(color.upper())
```

```
RED
GREEN
BLUE
```

```python
# --- Tuple unpacking ---
a, b, c = my_tuple
print(a, b, c)
```

```
red green blue
```

```python
# --- Converting tuple to list and back ---
temp_list = list(my_tuple)
temp_list.append('yellow')
my_tuple = tuple(temp_list)
print(my_tuple)
```

```
('red', 'green', 'blue', 'yellow')
```

```python
# --- Using negative step in slicing ---
letters = ['a', 'b', 'c', 'd', 'e']
print(letters[4:1:-1])  # ['e', 'd', 'c']
```

```
['e', 'd', 'c']
```

```python
# --- Slice assignment (only works on lists) ---
letters[1:3] = ['x', 'y']
print(letters)
```

```
['a', 'x', 'y', 'd', 'e']
```

```python
# --- Using range with slices ---
r = list(range(10))
print(r[::3])  # every 3rd number from 0 to 9
```

```
[0, 3, 6, 9]
```

```python
# --- Checking shallow copy behavior ---
list1 = [[1, 2], [3, 4]]
list2 = list1[:]  # shallow copy
list1[0][0] = 99  # modifies both lists because inner lists are shared
print(list1)
print(list2)
```

```
[[99, 2], [3, 4]]
[[99, 2], [3, 4]]
```