

```
In [1]: # TASK-1
# Threading significantly speed up the program, but it depends on the task that you are doing
import time
start = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
do_something()
finish = time.perf_counter() #using to find the time the entirer sequecnce takes from here
print(f'Finished in {round(finish-start,2)} seconds')
```

Sleeping 1 second...  
Done Sleeping  
Finished in 1.0 seconds

```
In [2]: # TASK-2
# Run the function twice
import time
start = time.perf_counter() #to find the time the entrirer sequecnce takes from here
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
do_something()
do_something()
finish = time.perf_counter() #to find the time the entrirer sequecnce till here
print(f'Finished in {round(finish-start,2)} seconds')
```

Sleeping 1 second...  
Done Sleeping  
Sleeping 1 second...  
Done Sleeping  
Finished in 2.0 seconds

```
In [3]: # TASK-3
import threading #its a existing python package, this is the traditional way of doing threading, more effe
import time
start = time.perf_counter()
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
#instead the running the two functions like this, threads are used both of these
#do_something()
#do_something()
t1 = threading.Thread(target = do_something) #do not pass the function with (), as we dont intend to run the f
t2 = threading.Thread(target = do_something)
finish = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
print(f'Finished in {round(finish-start,2)} seconds')
```

Finished in 0.0 seconds

```
In [4]: # TASK-4
import threading #its a existing python package, this is the traditional way of doing threading, more effe
import time
start = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
#instead the running the two functions like this, threads are used both of these
#do_something()
#do_something()
t1 = threading.Thread(target = do_something) #do not pass the function with (), as we dont intend to run the f
t2 = threading.Thread(target = do_something)
t1.start()
t2.start()
finish = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
print(f'Finished in {round(finish-start,2)} seconds')
```

Sleeping 1 second...Sleeping 1 second...

Finished in 0.01 seconds  
Done Sleeping  
Done Sleeping

```
In [6]: #TASK-5
import threading
import time
start = time.perf_counter()
```

```
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')

#instead the running the two functions like this, threads are used both of these
do_something()
do_something()

t1 = threading.Thread(target = do_something) #do not pass the function with (), as we dont intend to run the function
t2 = threading.Thread(target = do_something)

t1.start()
t2.start()
t1.join()
t2.join()

finish = time.perf_counter() #using to find the time the entire sequence takes from here
print(f'Finished in {round(finish-start,2)} seconds')
```

```
In [14]: import threading      #no need to install, its already a part of python package, this is the traditional way of
import time
start = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
for _ in range(10):          # underscore variable is a throwaway variable to simply loop for 10 times and we are i
    t = threading.Thread(target = do_something)
    t.start()                # We cant use join() within the loop as it will join on the thread before looping thro
                             # To do this we can create a list of threads and perform join()
finish = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
print(f'Finished in {round(finish-start,2)} seconds')
```

```
In [20]: #TASK-7
# Threading effect with 10 calls
import threading
import time
start = time.perf_counter()
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
threads = []

for _ in range(10):
    t = threading.Thread(target = do_something)
    t.start()
    threads.append(t)
    for thread in threads:
        thread.join()

finish = time.perf_counter()
print(f'Finished in {round(finish-start,2)} seconds')
```

In [23]:

In [40]:

```
import concurrent.futures
# import threading - Not required
import time
start = time.perf_counter()
def do_something(seconds):
    print(f'Sleeping {seconds} second(s)...')
    time.sleep(seconds)
    #print('Done Sleeping')
    return 'Done Sleeping...'
with concurrent.futures.ThreadPoolExecutor() as executor:
    f1 = executor.submit(do_something, 1) #submit function will schedule the execution of function and returns
    print(f1.result())

finish = time.perf_counter()

print(f'Finished in {round(finish-start,2)} seconds')
import threading
import time
```

```

import random
def print_names():
    for name in ('John', 'Mark', 'Elon', 'Callahan'):
        print (name)
        time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
    for _ in range(4):
        print(random.randint(20,50))

        time.sleep(random.uniform(0.5,1.5))
t1 = threading.Thread(target = print_names)
t2 = threading.Thread(target = print_ages)
# The above threads are not doing anything yet. To do that, use start()
t1.start()
t2.start()
# t1.join()
# t2.join()

```

Sleeping 1 second(s)...  
 Done Sleeping...  
 Finished in 1.01 seconds  
 John  
 26  
 Mark  
 24  
 Elon  
 42  
 Callahan  
 34

```

In [42]: import concurrent.futures
         # import threading - Not required
         import time
         start = time.perf_counter()
         def do_something(seconds):
             print(f'Sleeping {seconds} second(s)...')
             time.sleep(seconds)
             #print('Done Sleeping')
             return 'Done Sleeping...'
         with concurrent.futures.ThreadPoolExecutor() as executor:
             f1 = executor.submit(do_something, 1) #submit function will schedule the execution of function and returns
             f2 = executor.submit(do_something, 1)
             print(f1.result())
             print(f2.result())

         finish = time.perf_counter()
         print(f'Finished in {round(finish-start,2)} seconds')

```

Sleeping 1 second(s)...  
 Sleeping 1 second(s)...  
 Done Sleeping...  
 Done Sleeping...  
 Finished in 1.01 seconds

```

In [43]: #TASK-11
         import concurrent.futures
         import time
         start = time.perf_counter()
         def do_something(seconds):
             print(f'Sleeping {seconds} second(s)...')
             time.sleep(seconds)
             return 'Done Sleeping...'
         with concurrent.futures.ThreadPoolExecutor() as executor:
             results = [executor.submit(do_something, 1) for _ in range(10)] #list comprehension, alternative to loop
             for f in concurrent.futures.as_completed(results):
                 print(f.result())

         finish = time.perf_counter()
         print(f'Finished in {round(finish-start,2)} seconds')

```

```

Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Finished in 1.01 seconds

```

```

In [44]: #TASK-12
import concurrent.futures
import time
start = time.perf_counter()
def do_something(seconds):
    print(f'Sleeping {seconds} second(s)...')
    time.sleep(seconds)
    return f'Done Sleeping...{seconds}'
with concurrent.futures.ThreadPoolExecutor() as executor:
    s = [5,4,3,2,1] #different sleeping time for threads
    results = [executor.submit(do_something, s) for sec in s] #list comprehension, alternative to loop
    for f in concurrent.futures.as_completed(results):
        print(f.result())

finish = time.perf_counter()
print(f'Finished in {round(finish-start,2)} seconds')

```

```

Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...

```

```

-----
TypeError                                Traceback (most recent call last)
Cell In[44], line 13
     11 results = [executor.submit(do_something, s) for sec in s] #list comprehension, alternative to loop
     12 for f in concurrent.futures.as_completed(results):
--> 13     print(f.result())
     16 finish = time.perf_counter()
     17 print(f'Finished in {round(finish-start,2)} seconds')

File C:\ProgramData\anaconda3\Lib\concurrent\futures\_base.py:449, in Future.result(self, timeout)
     447     raise CanceledError()
     448 elif self._state == FINISHED:
--> 449     return self.__get_result()
     451 self._condition.wait(timeout)
     453 if self._state in [CANCELLED, CANCELLED_AND_NOTIFIED]:

File C:\ProgramData\anaconda3\Lib\concurrent\futures\_base.py:401, in Future.__get_result(self)
     399 if self._exception is not None:
     400     try:
--> 401         raise self._exception
     402     finally:
     403         # Break a reference cycle with the exception in self._exception
     404         self = None

File C:\ProgramData\anaconda3\Lib\concurrent\futures\thread.py:59, in _WorkItem.run(self)
     56     return
     58 try:
--> 59     result = self.fn(*self.args, **self.kwargs)
     60 except BaseException as exc:
     61     self.future.set_exception(exc)

Cell In[44], line 7, in do_something(seconds)
      5 def do_something(seconds):
      6     print(f'Sleeping {seconds} second(s)...')
--> 7     time.sleep(seconds)
      8     return f'Done Sleeping...{seconds}'

TypeError: 'list' object cannot be interpreted as an integer

```

In [45]:

```
#TASK-12 (CORRECTED CODE)
import concurrent.futures
import time
start = time.perf_counter()
def do_something(seconds):
    print(f'Sleeping {seconds} second(s)...')
    time.sleep(seconds)
    return f'Done Sleeping...{seconds}'
with concurrent.futures.ThreadPoolExecutor() as executor:
    s = [5, 4, 3, 2, 1] # different sleeping time for threads
    results = [executor.submit(do_something, sec) for sec in s]
    for f in concurrent.futures.as_completed(results):
        print(f.result())
finish = time.perf_counter()
print(f'Finished in {round(finish - start, 2)} seconds')
```

```
Sleeping 5 second(s)...
Sleeping 4 second(s)...
Sleeping 3 second(s)...
Sleeping 2 second(s)...
Sleeping 1 second(s)...
Done Sleeping...1
Done Sleeping...2
Done Sleeping...3
Done Sleeping...4
Done Sleeping...5
Finished in 5.01 seconds
```

In [50]:

```
#TASK-13
import threading
import time

import random
def print_names():
    for name in ('John', 'Mark', 'Elon', 'Callahan'):
        print (name)
        time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
    for _ in range(4):
        print(random.randint(20,50))
        time.sleep(random.uniform(0.5,1.5))
print_names()
print_ages()
# without using thread concept, simply calling the function
```

```
John
Mark
Elon
Callahan
26
30
43
23
```

In [52]:

```
#TASK-14
import threading
import time
import random
def print_names():
    for name in ('John', 'Mark', 'Elon', 'Callahan'):
        print (name)
        time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
    for _ in range(4):
        print(random.randint(20,50))
        time.sleep(random.uniform(0.5,1.5))

t1 = threading.Thread(target = print_names)
t2 = threading.Thread(target = print_ages)
# The above threads are not doing anything yet. To do that, use start()
t1.start()
t2.start()
# t1.join()
# t2.join()
# JOIN() NOT used - So, it will not make sure that the threads complete before moving on to the next part of t
```

```
John
49
Mark
42
23
Elon
Callahan
44
```

```
In [53]: #TASK-15
import threading
import time
import random
def print_names():
    for name in ('John', 'Mark', 'Elon', 'Callahan'):
        print (name)
        time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
    for _ in range(4):
        print(random.randint(20,50))
        time.sleep(random.uniform(0.5,1.5))
t1 = threading.Thread(target = print_names)
t2 = threading.Thread(target = print_ages)
# The above threads are not doing anything yet. To do that, use start()
t1.start()
t2.start()
t1.join()
t2.join()
# JOIN() is used - It makes sure that the threads complete before moving on to the next part of the code
```

```
John
47
Mark
48
Elon
27
Callahan
29
```

```
In [57]: #TASK-16
import threading
import requests
from pathlib import Path
# Create the Downloads directory if it doesn't exist
Path("Downloads").mkdir(exist_ok=True)
def download_file(url, filename):
    print(f'Downloading {url} to {filename}')
    try:
        response = requests.get(url)
        response.raise_for_status() # Raise an exception for bad status codes (4xx or 5xx)
        Path(filename).write_bytes(response.content)
        print(f'Finished Downloading {filename}')
    except requests.exceptions.RequestException as e:
        print(f'Error downloading {url}: {e}')
# Replace these URLs with your GitHub raw file URLs
urls = [
    'https://github.com/DheviSri/python-lab/blob/main/Lab_Activity_10_team_devishree.pdf',

    'https://github.com/DheviSri/python-lab/blob/main/lab7_activity.ipynb',
]
threads = []
for url in urls:
    # Use the last part of the URL as the filename
    filename = Path("Downloads") / url.split("/")[-1]
    t = threading.Thread(target=download_file, args=(url, filename))
    t.start()
    threads.append(t)
# Wait for all threads to complete
[t.join() for t in threads]
print("All downloads complete.")
```

```
Downloading https://github.com/DheviSri/python-lab/blob/main/lab7_activity.ipynb to Downloads\lab7_activity.ipyn
b
Finished Downloading Downloads\lab7_activity.ipynb
All downloads complete.
```

```
In [60]: pip install nbconvert
```

Defaulting to user installation because normal site-packages is not writeable  
Requirement already satisfied: nbconvert in c:\programdata\anaconda3\lib\site-packages (7.16.6)  
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (4.12.3)  
Requirement already satisfied: bleach!=5.0.0 in c:\programdata\anaconda3\lib\site-packages (from bleach[css]!=5.0.0->nbconvert) (6.2.0)  
Requirement already satisfied: defusedxml in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0.7.1)  
Requirement already satisfied: Jinja2>=3.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (3.1.6)  
Requirement already satisfied: jupyter-core>=4.7 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.7.2)  
Requirement already satisfied: jupyterlab-pygments in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0.3.0)  
Requirement already satisfied: markupsafe>=2.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (3.0.2)  
Requirement already satisfied: mistune<4,>=2.0.3 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (3.1.2)  
Requirement already satisfied: nbclient>=0.5.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0.10.2)  
Requirement already satisfied: nbformat>=5.7 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.10.4)  
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (24.2)  
Requirement already satisfied: pandocfilters>=1.4.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (1.5.0)  
Requirement already satisfied: pygments>=2.4.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (2.19.1)  
Requirement already satisfied: traitlets>=5.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.14.3)  
Requirement already satisfied: webencodings in c:\programdata\anaconda3\lib\site-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nbconvert) (0.5.1)  
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from bleach[css]!=5.0.0->nbconvert) (1.4.0)  
Requirement already satisfied: platformdirs>=2.5 in c:\programdata\anaconda3\lib\site-packages (from jupyter-core>=4.7->nbconvert) (4.3.7)  
Requirement already satisfied: pywin32>=300 in c:\programdata\anaconda3\lib\site-packages (from jupyter-core>=4.7->nbconvert) (308)  
Requirement already satisfied: jupyter-client>=6.1.12 in c:\programdata\anaconda3\lib\site-packages (from nbclient>=0.5.0->nbconvert) (8.6.3)  
Requirement already satisfied: python-dateutil>=2.8.2 in c:\programdata\anaconda3\lib\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.9.0.post0)  
Requirement already satisfied: pyzmq>=23.0 in c:\programdata\anaconda3\lib\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (26.2.0)  
Requirement already satisfied: tornado>=6.2 in c:\programdata\anaconda3\lib\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.5.1)  
Requirement already satisfied: fastjsonschema>=2.15 in c:\programdata\anaconda3\lib\site-packages (from nbformat>=5.7->nbconvert) (2.20.0)  
Requirement already satisfied: jsonschema>=2.6 in c:\programdata\anaconda3\lib\site-packages (from nbformat>=5.7->nbconvert) (4.23.0)  
Requirement already satisfied: attrs>=22.2.0 in c:\programdata\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (24.3.0)  
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\programdata\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (2023.7.1)  
Requirement already satisfied: referencing>=0.28.4 in c:\programdata\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.30.2)  
Requirement already satisfied: rpds-py>=0.7.1 in c:\programdata\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.22.3)  
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (1.17.0)  
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3\lib\site-packages (from beautifulsoup4->nbconvert) (2.5)  
Note: you may need to restart the kernel to use updated packages.

In [61]: conda install nbconvert

Note: you may need to restart the kernel to use updated packages.Jupyter detected...  
3 channel Terms of Service accepted  
Retrieving notices: done

EnvironmentNotWritableError: The current user does not have write permissions to the target environment.  
environment location: C:\ProgramData\anaconda3

In [ ]:



```
In [2]: import pandas as pd
import numpy as np
import io
# Create a CSV file
csv_data = """Name,Age,City,Salary,Education
Alice,25,New York,70000,Bachelors
Bob,30,Los Angeles,85000,Masters
Charlie,35,Chicago,92000,PhD
David,28,Houston,75000,Bachelors
Eve,32,Phoenix,88000,Masters
"""
```

```
In [7]: # String as a File
df = pd.read_csv(io.StringIO(csv_data)) # Initialize a StringIO object with an existing string, and then read from it
df.to_csv(r"C:\Users\dhevi\Downloads\employee_data.csv", index=False) #writes the contents of a DataFrame to a file
```

```
In [8]: data = pd.read_csv(r"C:\Users\dhevi\Downloads\Housing.csv") # Reads the CSV file into a DataFrame
print(data.head()) # Displays the first 5 rows of the DataFrame
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

  

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	no	yes	2	yes	furnished
1	no	yes	3	no	furnished
2	no	no	2	yes	semi-furnished
3	no	yes	3	yes	furnished
4	no	yes	2	no	furnished

```
In [9]: print(data.info()) # Provides a summary of the DataFrame, including data types and non-null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 546 entries, 0 to 545
Data columns (total 13 columns):
#   Column              Non-Null Count  Dtype
---  -
0   price               546 non-null   int64
1   area                546 non-null   int64
2   bedrooms            546 non-null   int64
3   bathrooms            546 non-null   int64
4   stories              546 non-null   int64
5   mainroad             546 non-null   object
6   guestroom            546 non-null   object
7   basement             546 non-null   object
8   hotwaterheating      546 non-null   object
9   airconditioning      546 non-null   object
10  parking              546 non-null   int64
11  prefarea             546 non-null   object
12  furnishingstatus     546 non-null   object
dtypes: int64(6), object(7)
memory usage: 55.6+ KB
None
```

```
In [10]: print(data.shape) # Displays Rows and columns

(546, 13)
```

```
In [11]: df = pd.read_csv("employee_data.csv") # reading data from a CSV file. It automatically parses the file, recognizes the data types, and loads it into a DataFrame.
print(df)
```

	Name	Age	City	Salary	Education
0	Alice	25	New York	70000	Bachelors
1	Bob	30	Los Angeles	85000	Masters
2	Charlie	35	Chicago	92000	PhD
3	David	28	Houston	75000	Bachelors
4	Eve	32	Phoenix	88000	Masters

```
In [17]: chunk_size = 100
count = 1
#sets chunksize to 100, so instead of loading the entire file at once, it reads and processes the data in small chunks
for chunk in pd.read_csv(r"C:\Users\dhevi\Downloads\Housing.csv", chunksize=chunk_size):
    #chunks of 100 rows each
    print(f"DataFrame Chunk {count}:")
    print(chunk)
    count += 1
```

DataFrame Chunk 1:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	
..	...	...	...	...	...	...	...	...	
95	6300000	4100	3	2	3	yes	no	no	
96	6300000	9000	3	1	1	yes	no	yes	
97	6300000	6400	3	1	1	yes	yes	yes	
98	6293000	6600	3	2	3	yes	no	no	
99	6265000	6000	4	1	3	yes	yes	yes	

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	no	yes	2	yes	furnished
1	no	yes	3	no	furnished
2	no	no	2	yes	semi-furnished
3	no	yes	3	yes	furnished
4	no	yes	2	no	furnished
..	...	...	...	...	...
95	no	yes	2	no	semi-furnished
96	no	no	1	yes	furnished
97	no	yes	1	yes	semi-furnished
98	no	yes	0	yes	unfurnished
99	no	no	0	yes	unfurnished

[100 rows x 13 columns]

DataFrame Chunk 2:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
100	6230000	6600	3	2	1	yes	no	yes	
101	6230000	5500	3	1	3	yes	no	no	
102	6195000	5500	3	2	4	yes	yes	no	
103	6195000	6350	3	2	3	yes	yes	no	
104	6195000	5500	3	2	1	yes	yes	yes	
..	...	...	...	...	...	...	...	...	
195	4970000	4410	4	3	2	yes	no	yes	
196	4970000	7686	3	1	1	yes	yes	yes	
197	4956000	2800	3	2	2	no	no	yes	
198	4935000	5948	3	1	2	yes	no	no	
199	4907000	4200	3	1	2	yes	no	no	

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
100	no	yes	0	yes	unfurnished
101	no	no	1	yes	unfurnished
102	no	yes	1	no	semi-furnished
103	no	yes	0	no	furnished
104	no	no	2	yes	furnished
..	...	...	...	...	...
195	no	no	2	no	semi-furnished
196	yes	no	0	no	semi-furnished
197	no	yes	1	no	semi-furnished
198	no	yes	0	no	semi-furnished
199	no	no	1	no	furnished

[100 rows x 13 columns]

DataFrame Chunk 3:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
200	4900000	4520	3	1	2	yes	no	yes	
201	4900000	4095	3	1	2	no	yes	yes	
202	4900000	4120	2	1	1	yes	no	yes	
203	4900000	5400	4	1	2	yes	no	no	
204	4900000	4770	3	1	1	yes	yes	yes	
..	...	...	...	...	...	...	...	...	
295	4200000	2325	3	1	2	no	no	no	
296	4200000	4600	3	2	2	yes	no	no	
297	4200000	3640	3	2	2	yes	no	yes	
298	4200000	5800	3	1	1	yes	no	no	
299	4200000	7000	3	1	1	yes	no	no	

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
200	no	yes	0	no	semi-furnished
201	no	yes	0	no	semi-furnished
202	no	no	1	no	semi-furnished
203	no	no	0	no	semi-furnished
204	no	no	0	no	semi-furnished
..	...	...	...	...	...
295	no	no	0	no	semi-furnished
296	no	yes	1	no	semi-furnished
297	no	no	0	no	unfurnished
298	yes	no	2	no	semi-furnished
299	no	no	3	no	furnished

[100 rows x 13 columns]

DataFrame Chunk 4:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
300	4200000	4079	3	1	3	yes	no	no	
301	4200000	3520	3	1	2	yes	no	no	
302	4200000	2145	3	1	3	yes	no	no	
303	4200000	4500	3	1	1	yes	no	yes	
304	4193000	8250	3	1	1	yes	no	yes	
..	...	...	...	...	...	...	...	...	
395	3500000	3600	6	1	2	yes	no	no	
396	3500000	3640	2	1	1	yes	no	no	
397	3500000	5900	2	1	1	yes	no	no	
398	3500000	3120	3	1	2	yes	no	no	
399	3500000	7350	2	1	1	yes	no	no	

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
300	no	no	0	no	semi-furnished
301	no	no	0	yes	semi-furnished
302	no	no	1	yes	unfurnished
303	no	no	0	no	furnished
304	no	no	3	no	semi-furnished
..	...	...	...	...	...
395	no	no	1	no	unfurnished
396	no	no	1	no	semi-furnished
397	no	no	1	no	furnished
398	no	no	1	no	unfurnished
399	no	no	1	no	semi-furnished

[100 rows x 13 columns]

DataFrame Chunk 5:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
400	3500000	3512	2	1	1	yes	no	no	
401	3500000	9500	3	1	2	yes	no	no	
402	3500000	5880	2	1	1	yes	no	no	
403	3500000	12944	3	1	1	yes	no	no	
404	3493000	4900	3	1	2	no	no	no	
..	...	...	...	...	...	...	...	...	
495	2730000	4000	3	1	2	yes	no	no	
496	2695000	4000	2	1	1	yes	no	no	
497	2660000	3934	2	1	1	yes	no	no	
498	2660000	2000	2	1	2	yes	no	no	
499	2660000	3630	3	3	2	no	yes	no	

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
400	no	no	1	yes	unfurnished
401	no	no	3	yes	unfurnished
402	no	no	0	no	unfurnished
403	no	no	0	no	unfurnished
404	no	no	0	no	unfurnished
..	...	...	...	...	...
495	no	no	1	no	unfurnished
496	no	no	0	no	unfurnished
497	no	no	0	no	unfurnished
498	no	no	0	no	semi-furnished
499	no	no	0	no	unfurnished

[100 rows x 13 columns]

DataFrame Chunk 6:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
500	2660000	2800	3	1	1	yes	no	no	
501	2660000	2430	3	1	1	no	no	no	
502	2660000	3480	2	1	1	yes	no	no	
503	2660000	4000	3	1	1	yes	no	no	
504	2653000	3185	2	1	1	yes	no	no	
505	2653000	4000	3	1	2	yes	no	no	
506	2604000	2910	2	1	1	no	no	no	
507	2590000	3600	2	1	1	yes	no	no	
508	2590000	4400	2	1	1	yes	no	no	
509	2590000	3600	2	2	2	yes	no	yes	
510	2520000	2880	3	1	1	no	no	no	
511	2520000	3180	3	1	1	no	no	no	
512	2520000	3000	2	1	2	yes	no	no	
513	2485000	4400	3	1	2	yes	no	no	
514	2485000	3000	3	1	2	no	no	no	
515	2450000	3210	3	1	2	yes	no	yes	
516	2450000	3240	2	1	1	no	yes	no	
517	2450000	3000	2	1	1	yes	no	no	
518	2450000	3500	2	1	1	yes	yes	no	
519	2450000	4840	2	1	2	yes	no	no	
520	2450000	7700	2	1	1	yes	no	no	
521	2408000	3635	2	1	1	no	no	no	
522	2380000	2475	3	1	2	yes	no	no	
523	2380000	2787	4	2	2	yes	no	no	
524	2380000	3264	2	1	1	yes	no	no	

525	2345000	3640	2	1	1	yes	no	no
526	2310000	3180	2	1	1	yes	no	no
527	2275000	1836	2	1	1	no	no	yes
528	2275000	3970	1	1	1	no	no	no
529	2275000	3970	3	1	2	yes	no	yes
530	2240000	1950	3	1	1	no	no	no
531	2233000	5300	3	1	1	no	no	no
532	2135000	3000	2	1	1	no	no	no
533	2100000	2400	3	1	2	yes	no	no
534	2100000	3000	4	1	2	yes	no	no
535	2100000	3360	2	1	1	yes	no	no
536	1960000	3420	5	1	2	no	no	no
537	1890000	1700	3	1	2	yes	no	no
538	1890000	3649	2	1	1	yes	no	no
539	1855000	2990	2	1	1	no	no	no
540	1820000	3000	2	1	1	yes	no	yes
541	1767150	2400	3	1	1	no	no	no
542	1750000	3620	2	1	1	yes	no	no
543	1750000	2910	3	1	1	no	no	no
544	1750000	3850	3	1	2	yes	no	no
545	9240000	7800	3	2	2	yes	no	no

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
500	no	no	0	no	unfurnished
501	no	no	0	no	unfurnished
502	no	no	1	no	semi-furnished
503	no	no	0	no	semi-furnished
504	no	yes	0	no	unfurnished
505	no	yes	0	no	unfurnished
506	no	no	0	no	unfurnished
507	no	no	0	no	unfurnished
508	no	no	0	no	unfurnished
509	no	no	1	no	furnished
510	no	no	0	no	unfurnished
511	no	no	0	no	unfurnished
512	no	no	0	no	furnished
513	no	no	0	no	unfurnished
514	no	no	0	no	semi-furnished
515	no	no	0	no	unfurnished
516	no	no	1	no	unfurnished
517	no	no	1	no	unfurnished
518	no	no	0	no	unfurnished
519	no	no	0	no	unfurnished
520	no	no	0	no	unfurnished
521	no	no	0	no	unfurnished
522	no	no	0	no	furnished
523	no	no	0	no	furnished
524	no	no	0	no	unfurnished
525	no	no	0	no	unfurnished
526	no	no	0	no	unfurnished
527	no	no	0	no	semi-furnished
528	no	no	0	no	unfurnished
529	no	no	0	no	unfurnished
530	yes	no	0	no	unfurnished
531	no	yes	0	yes	unfurnished
532	no	no	0	no	unfurnished
533	no	no	0	no	unfurnished
534	no	no	0	no	unfurnished
535	no	no	1	no	unfurnished
536	no	no	0	no	unfurnished
537	no	no	0	no	unfurnished
538	no	no	0	no	unfurnished
539	no	no	1	no	unfurnished
540	no	no	2	no	unfurnished
541	no	no	0	no	semi-furnished
542	no	no	0	no	unfurnished
543	no	no	0	no	furnished
544	no	no	0	no	unfurnished
545	no	no	1	yes	semi-furnished

```
In [26]: new_data = pd.DataFrame([[9240000,7800,3,2,2,'yes','no','no','no','no',1,'yes','semi-furnished']], columns=['price', 'sqft', 'bathrooms', 'bedrooms', 'parking', 'prefarea', 'hotwaterheating', 'airconditioning', 'furnishingstatus'])
#Appending new data to an existing CSV file is done by using the mode='a' and header=False parameters in the to_csv method
new_data.to_csv(r"C:\Users\dhevi\Downloads\Housing.csv", mode='a', header=False, index=False)
df_appended = pd.read_csv(r"C:\Users\dhevi\Downloads\Housing.csv")
print(df_appended)
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	
..	...	...	...	...	...	...	...	...	
542	1750000	3620	2	1	1	yes	no	no	
543	1750000	2910	3	1	1	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	
545	9240000	7800	3	2	2	yes	no	no	
546	9240000	7800	3	2	2	yes	no	no	

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	no	yes	2	yes	furnished
1	no	yes	3	no	furnished
2	no	no	2	yes	semi-furnished
3	no	yes	3	yes	furnished
4	no	yes	2	no	furnished
..	...	...	...	...	...
542	no	no	0	no	unfurnished
543	no	no	0	no	furnished
544	no	no	0	no	unfurnished
545	no	no	1	yes	semi-furnished
546	no	no	1	yes	semi-furnished

[547 rows x 13 columns]

```
In [27]: numeric_df = pd.DataFrame(np.random.randint(10, 100, size=(50, 3)), columns=['A', 'B', 'C'])
numeric_df.to_csv(r"C:\Users\dhevi\Downloads\numeric_data.csv", index=False) # writes the contents of a DataFrame to a CSV file
numeric_df_read = pd.read_csv(r"C:\Users\dhevi\Downloads\numeric_data.csv")
print(numeric_df_read)
```

	A	B	C
0	96	88	69
1	44	28	28
2	79	94	61
3	47	87	31
4	21	46	70
5	42	36	42
6	23	73	22
7	59	52	63
8	16	46	94
9	61	97	86
10	59	94	64
11	26	14	26
12	77	40	78
13	30	72	29
14	52	78	56
15	71	90	91
16	68	96	80
17	46	54	65
18	79	20	58
19	17	73	16
20	64	68	73
21	47	83	93
22	98	70	84
23	51	43	79
24	90	97	72
25	65	59	11
26	36	34	19
27	57	62	18
28	48	76	14
29	83	19	47
30	41	89	99
31	37	25	95
32	20	12	40
33	11	89	72
34	41	40	32
35	40	80	61
36	11	29	50
37	92	65	89
38	15	74	68
39	27	52	22
40	45	65	76
41	78	99	51
42	66	47	52
43	35	40	77
44	25	60	65
45	97	48	71
46	94	82	95
47	42	67	72
48	73	74	56
49	40	27	53

```
In [30]: text_data = {
        'Product': ['Laptop', 'Mouse', 'Keyboard', 'Monitor'],
        'Description': ['High-performance laptop', 'Wireless ergonomic mouse', 'Mechanical keyboard', '4K ultra-wide']
    }
text_df = pd.DataFrame(text_data)
text_df.to_csv(r"C:\Users\dhevi\Downloads\text_data.csv", index=False) # writes the contents of a DataFrame to a CSV file
text_df_read = pd.read_csv(r"C:\Users\dhevi\Downloads\text_data.csv")
print(text_df_read)
```

	Product	Description
0	Laptop	High-performance laptop
1	Mouse	Wireless ergonomic mouse
2	Keyboard	Mechanical keyboard
3	Monitor	4K ultra-wide monitor

```
In [ ]:
```

NAME : DHEVISHREE GN

```
In [ ]: REG NUM : 22MID0136
```

```
In [ ]: LAB 13
```

```
In [2]: import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]]) # Creating a 2-D Array
print(arr)
```

```
[[1 2 3]
 [4 5 6]]
```

```
In [3]: print(arr.size) # Displays the number of elements in a array

6
```

```
In [4]: print(arr.shape) # Display the rows and columns

(2, 3)
```

```
In [5]: print(arr.ndim) # Displays the dimension of array

2
```

```
In [6]: # Create a 1D array
arr = np.array([1, 2, 3, 4, 5, 6])
print("Original array:", arr)
print("Original shape:", arr.shape)
```

```
Original array: [1 2 3 4 5 6]
Original shape: (6,)
```

```
In [8]: # Reshape to a 2D array (2 rows, 3 columns)
reshaped_arr = arr.reshape(2, 3) # Reshape a 1-D array to 2-D array
print(reshaped_arr)
print("New shape:", reshaped_arr.shape)
```

```
[[1 2 3]
 [4 5 6]]
New shape: (2, 3)
```

```
In [9]: import numpy as np
def numpysum(n):
    a = np.arange(n) ** 2 #Creates an array [0,1,..n-1] and element-wise squaring
    b = np.arange(n) ** 3
    c = a + b #element wise adding - NO LOOP Overhead, hence Memory Efficient
    return c
numpysum(10) # function call
```

```
Out[9]: array([ 0,  2, 12, 36, 80, 150, 252, 392, 576, 810])
```

```
In [10]: #ARRAY CREATION FUNCTIONS
import numpy as np
np.arange(10) # 0 TO BEFORE 10, Step Value 1
```

```
Out[10]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [11]: np.arange(2, 10, dtype=float) # 2. to Before 10., step value 1
```

```
Out[11]: array([2., 3., 4., 5., 6., 7., 8., 9.])
```

```
In [12]: np.arange(2, 3, 0.1) # 2 to Before 3, step value 0.1
```

```
Out[12]: array([2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9])
```

```
In [13]: #Creating Arrays from Sequences
#a list of numbers will create a 1D array,
a1D = np.array([1, 2, 3, 4])
print(a1D)
```

```
[1 2 3 4]
```

```
In [18]: #a list of lists will create a 2D array
a2D = np.array([[1, 2], [3, 4]])
print(a2D)
```

```
[[1 2]
 [3 4]]
```

```
In [19]: #further nested lists will create higher-dimensional arrays.
#In general, any array object is called an ndarray in NumPy.
a3D = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
```

```
print(a3D)
```

```
[[[1 2]
   [3 4]]
```

```
[[5 6]
 [7 8]]]
```

```
In [20]: import numpy as np
np.array([127, 128, 129], dtype=np.int32)
```

```
Out[20]: array([127, 128, 129], dtype=int32)
```

```
In [22]: import numpy as np
np.array([127, 128, 129], dtype=np.int8)
# Error: OverflowError because the np.int8 data type cannot hold the values 128 and 129
```

```
-----
OverflowError                                Traceback (most recent call last)
Cell In[22], line 2
      1 import numpy as np
----> 2 np.array([127, 128, 129], dtype=np.int8)

OverflowError: Python integer 128 out of bounds for int8
```

```
In [23]: #2D Array Creation functions
#properties of special matrices represented as 2D arrays.
np.eye(3) # Identity matrix of 3X3
```

```
Out[23]: array([[1., 0., 0.],
               [0., 1., 0.],
               [0., 0., 1.]])
```

```
In [24]: np.eye(3, 5) # Identity matrix of 3X5
```

```
Out[24]: array([[1., 0., 0., 0., 0.],
               [0., 1., 0., 0., 0.],
               [0., 0., 1., 0., 0.]])
```

```
In [25]: np.eye(3, 5, k=-2) #optional k argument
# When k is negative (e.g., k=-2), the diagonal shifts below the main diagonal
# when k is positive (e.g., k=1), the diagonal shifts above the main diagonal.
```

```
Out[25]: array([[0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.],
               [1., 0., 0., 0., 0.]])
```

```
In [26]: # It returns an array containing the elements of the specified diagonal.
np.diag([1, 2, 3])
```

```
Out[26]: array([[1, 0, 0],
               [0, 2, 0],
               [0, 0, 3]])
```

```
In [27]: # To hold this upper diagonal, NumPy creates a (n + k) × (n + k) square matrix, where n is the length of the i
# So for [1, 2, 3]: Length = 3, k = 1, Matrix shape = 4 × 4
# Diagonal values are placed at positions: (0,1), (1,2), (2,3)
np.diag([1, 2, 3], 1)
```

```
Out[27]: array([[0, 1, 0, 0],
               [0, 0, 2, 0],
               [0, 0, 0, 3],
               [0, 0, 0, 0]])
```

```
In [28]: # numpy.diag can define either a square 2D array with given values along the diagonal or if given a 2D array
# It returns a 1D array that is only the diagonal elements.
a = np.array([[1, 2], [3, 4]])
np.diag(a)
```

```
Out[28]: array([1, 4])
```

```
In [29]: #np.linspace(0, 2, 5) → [0. , 0.5, 1. , 1.5, 2. ]
# N = 2 → two columns: x**1, x**0
np.vander(np.linspace(0, 2, 5), 2)
```

```
Out[29]: array([[0. , 1. ],
               [0.5, 1. ],
               [1. , 1. ],
               [1.5, 1. ],
               [2. , 1. ]])
```

```
In [30]: #Creating a 1D array of size 5 filled with ones
arr1 = np.ones(5) # shape=5 (1D), default dtype=float
print("1D Array of Ones:", arr1)
```



1D Array of Ones: [1. 1. 1. 1. 1.]

```
In [31]: #Creating a 2D array (3 rows, 4 columns) filled with ones
arr2 = np.ones((3, 4)) # shape=(3,4) means 3x4 matrix
print("\n2D Array (3x4) of Ones:\n", arr2)
```

2D Array (3x4) of Ones:

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

```
In [32]: #Specifying data type as integer
arr3 = np.ones((2, 3), dtype=int) # shape=(2,3), dtype=int
print("\n2D Integer Array (2x3) of Ones:\n", arr3)
```

2D Integer Array (2x3) of Ones:

```
[[1 1 1]
 [1 1 1]]
```

```
In [33]: #Creating a 3D array filled with ones
arr4 = np.ones((2, 2, 3)) # shape=(2,2,3) means 3D array (2 blocks, 2 rows, 3 columns)
print("\n3D Array of Ones:\n", arr4)
```

3D Array of Ones:

```
[[[1. 1. 1.]
  [1. 1. 1.]]
 [[1. 1. 1.]
  [1. 1. 1.]]]
```

```
In [34]: #Creating a boolean array (True represents 1)
arr5 = np.ones((3, 3), dtype=bool)
print("\nBoolean Array (3x3) of Ones (True values):\n", arr5)
```

Boolean Array (3x3) of Ones (True values):

```
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]
```

```
In [36]: #Creating a 1D array of size 5 filled with zeros
arr1 = np.zeros(5) # shape=5 (1D), default dtype=float
print("1D Array of Zeros:", arr1)
```

1D Array of Zeros: [0. 0. 0. 0. 0.]

```
In [37]: #Creating a 2D array (3 rows, 4 columns) filled with zeros
arr2 = np.zeros((3, 4)) # shape=(3,4) means 3x4 matrix
print("\n2D Array (3x4) of Zeros:\n", arr2)
```

2D Array (3x4) of Zeros:

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

```
In [38]: #Specifying data type as integer
arr3 = np.zeros((2, 3), dtype=int) # shape=(2,3), dtype=int
print("\n2D Integer Array (2x3) of Zeros:\n", arr3)
```

2D Integer Array (2x3) of Zeros:

```
[[0 0 0]
 [0 0 0]]
```

```
In [39]: #Creating a 3D array filled with zeros
arr4 = np.zeros((2, 2, 3)) # shape=(2,2,3) means 3D array (2 blocks, 2 rows, 3 columns)
print("\n3D Array of Zeros:\n", arr4)
```

3D Array of Zeros:

```
[[[0. 0. 0.]
  [0. 0. 0.]]
 [[0. 0. 0.]
  [0. 0. 0.]]]
```

```
In [40]: #Creating a boolean array (False represents 0)
arr5 = np.zeros((3, 3), dtype=bool)
print("\nBoolean Array (3x3) of Zeros (False values):\n", arr5)
```

Boolean Array (3x3) of Zeros (False values):

```
[[False False False]
 [False False False]
 [False False False]]
```

```
In [41]: #Creating indices for a 2D array of shape (3, 3)
indices_2d = np.indices((3, 3)) # shape=(3,3) means 3 rows and 3 columns
print("Indices for a 3x3 grid:\n", indices_2d)
```

Indices for a 3x3 grid:

```
[[[0 0 0]
   [1 1 1]
   [2 2 2]]
```

```
[[0 1 2]
 [0 1 2]
 [0 1 2]]]
```

```
In [42]: # indices_2d is actually a 3D array of shape (2, 3, 3):
# - The first 3x3 array contains row indices
# - The second 3x3 array contains column indices
```

```
In [43]: print("\nRow indices (first array):\n", indices_2d[0]) # Row positions
```

Row indices (first array):

```
[[0 0 0]
 [1 1 1]
 [2 2 2]]
```

```
In [44]: print("\nColumn indices (second array):\n", indices_2d[1]) # Column positions
```

Column indices (second array):

```
[[0 1 2]
 [0 1 2]
 [0 1 2]]
```

```
In [46]: #Creating indices for a 2x4 array
indices_2x4 = np.indices((2, 4)) # shape=(2,4) means 2 rows, 4 columns
```

```
In [47]: print("\nIndices for a 2x4 grid:\n", indices_2x4)
```

Indices for a 2x4 grid:

```
[[[0 0 0 0]
   [1 1 1 1]]
```

```
[[0 1 2 3]
 [0 1 2 3]]]
```

```
In [48]: print("\nRow indices:\n", indices_2x4[0])
```

Row indices:

```
[[0 0 0 0]
 [1 1 1 1]]
```

```
In [49]: print("\nColumn indices:\n", indices_2x4[1])
```

Column indices:

```
[[0 1 2 3]
 [0 1 2 3]]
```

```
In [50]: #Generate a single random float between 0 and 1
random_float = np.random.random()
print("Random Float (0 to 1):", random_float)
```

Random Float (0 to 1): 0.34084275159525823

```
In [51]: #Generate a 1D array of 5 random floats between 0 and 1
random_array = np.random.random(5)
print("\n1D Array of Random Floats:", random_array)
```

1D Array of Random Floats: [0.73009679 0.46059969 0.12029021 0.03861604 0.69150241]

```
In [52]: #Generate a 2D array (3x3) of random floats between 0 and 1
random_matrix = np.random.random((3, 3))

print("\n2D Array (3x3) of Random Floats:\n", random_matrix)
```

2D Array (3x3) of Random Floats:

```
[[0.78038619 0.45346296 0.92939059]
 [0.37551905 0.40383647 0.47406553]
 [0.73208494 0.89440378 0.73634312]]
```

```
In [53]: #Generate random integers between 10 and 50 (5 numbers)
random_integers = np.random.randint(10, 50, size=5)
print("\nRandom Integers between 10 and 50:", random_integers)
```

Random Integers between 10 and 50: [46 24 21 25 45]

```
In [54]: #Generate random numbers from a standard normal distribution (mean=0, std=1)
random_normal = np.random.randn(4)
print("\nRandom Numbers from Normal Distribution:", random_normal)
```

Random Numbers from Normal Distribution: [ 0.70827001 -0.72915831 -0.66503172 0.26473817]

```
In [55]: #Reproducibility - Set a random seed
np.random.seed(42)
```

```
print("\nRandom Numbers with Seed=42 (Reproducible):", np.random.random(3))
```

Random Numbers with Seed=42 (Reproducible): [0.37454012 0.95071431 0.73199394]

```
In [57]: #Create a 1D array
arr1d = np.array([10, 20, 30, 40, 50, 60])
print("1D Array:", arr1d)
```

1D Array: [10 20 30 40 50 60]

```
In [58]: print("\nIndexing on 1D Array:")
print("Element at index 0:", arr1d[0]) # First element
print("Element at index -1:", arr1d[-1]) # Last element (negative index)
```

Indexing on 1D Array:  
Element at index 0: 10  
Element at index -1: 60

```
In [60]: print("\nSlicing on 1D Array:")
print("Elements from index 1 to 4:", arr1d[1:5]) # From index 1 up to 4
print("Every second element:", arr1d[::2]) # Step of 2
print("Reverse array:", arr1d[::-1])
```

Slicing on 1D Array:  
Elements from index 1 to 4: [20 30 40 50]  
Every second element: [10 30 50]  
Reverse array: [60 50 40 30 20 10]

```
In [61]: #Create a 2D array (3x4 matrix)
arr2d = np.array([[1, 2, 3, 4],
                  [5, 6, 7, 8],
                  [9, 10, 11, 12]])
print("\n2D Array:\n", arr2d)
```

2D Array:  
[[ 1 2 3 4]  
 [ 5 6 7 8]  
 [ 9 10 11 12]]

```
In [62]: print("\nIndexing on 2D Array:")
print("Element at row 1, column 2:", arr2d[1, 2]) # 7 (row=1, col=2)
print("First row:", arr2d[0]) # Whole first row
print("First column:", arr2d[:, 0])
```

Indexing on 2D Array:  
Element at row 1, column 2: 7  
First row: [1 2 3 4]  
First column: [1 5 9]

```
In [63]: print("\nSlicing on 2D Array:")
print("Subarray (first 2 rows, columns 1 to 3):\n", arr2d[0:2, 1:3])
print("Last two rows:\n", arr2d[-2:, :]) # Last 2 rows
print("Every second column:\n", arr2d[:, ::2]) # All rows, step of 2 columns
```

Slicing on 2D Array:  
Subarray (first 2 rows, columns 1 to 3):  
[[2 3]  
 [6 7]]  
Last two rows:  
[[ 5 6 7 8]  
 [ 9 10 11 12]]  
Every second column:  
[[ 1 3]  
 [ 5 7]  
 [ 9 11]]

In [ ]:

## LAB ACTIVITY 14

NAME : DHEVISHREE GN; REG NO : 22MID0136; COURSE CODE: CSI3007;  
LAB : L7+L8;

### TASK-2: Handling EXCEL

#### 1. Read from an EXCEL file

```
In [56]: import pandas as pd

#Read the Excel file into a DataFrame
df = pd.read_excel('Call Test Measure.xlsx', sheet_name = "Sheet1") #Use the read_excel() function to import your
```

```
In [5]: print(df.head()) # Displays first 5 rows
```

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-01 00:00:27	-61.0	68.800003	1048.60	
1	2017-07-01 00:02:57	-61.0	68.769997	1855.54	
2	2017-07-01 00:05:29	-71.0	69.169998	1685.62	
3	2017-07-01 00:08:02	-65.0	69.279999	1770.92	
4	2017-07-01 00:10:30	-103.0	0.820000	256.07	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	UMTS	
1	90.0	SUCCESS	UMTS	
2	90.0	SUCCESS	UMTS	
3	90.0	SUCCESS	UMTS	
4	60.0	SUCCESS	UMTS	

	Call Test Setup Time (s)	MOS
0	0.56	2.1
1	0.45	3.2
2	0.51	2.1
3	0.00	1.0
4	3.35	3.6

```
In [7]: print(df.describe()) # Displays the statistics
```

	Date Of Test	Signal (dBm)	Speed (m/s)	\
count	105828	105821.000000	105828.000000	
mean	2017-09-01 11:26:42.371858176	-78.653623	8.629296	
min	2017-07-01 00:00:27	-140.000000	-1.000000	
25%	2017-08-03 11:11:47	-92.000000	0.000000	
50%	2017-09-03 18:23:42	-79.000000	0.000000	
75%	2017-09-30 07:30:44.750000128	-63.000000	7.770000	
max	2017-10-31 23:50:19	-51.000000	86.310516	
std	NaN	18.631699	18.008427	

	Distance from site (m)	Call Test Duration (s)	\
count	95469.000000	105828.000000	
mean	7797.172461	84.202264	
min	1.410000	12.900000	
25%	236.580000	60.000000	
50%	430.350000	90.000000	
75%	789.960000	90.000000	
max	745483.680000	900.000000	
std	49584.213355	66.250741	

	Call Test Setup Time (s)	MOS
count	105828.000000	105828.000000
mean	2.662776	3.105864
min	0.000000	1.000000
25%	0.640000	2.100000
50%	3.510000	3.100000
75%	4.080000	4.400000
max	45.330000	4.400000
std	2.057087	1.252348

```
In [9]: print(df.shape) # Displays the rows and columns
```

(105828, 9)

```
In [11]: print(df.info()) # Statistical Information about Excel file
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105828 entries, 0 to 105827
Data columns (total 9 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Date Of Test                          105828 non-null  datetime64[ns]
1   Signal (dBm)                          105821 non-null  float64
2   Speed (m/s)                           105828 non-null  float64
3   Distance from site (m)                 95469 non-null   float64
4   Call Test Duration (s)                 105828 non-null  float64
5   Call Test Result                       105828 non-null  object
6   Call Test Technology                   105828 non-null  object
7   Call Test Setup Time (s)               105828 non-null  float64
8   MOS                                    105828 non-null  float64
dtypes: datetime64[ns](1), float64(6), object(2)
memory usage: 7.3+ MB
None
```

## 2.Extract the contents

```
In [20]: # Extract the Success call test result from Call Test Measure.xlsx
call_test_df = df[df['Call Test Result'] == 'SUCCESS']
print("Calls Test Results:\n", call_test_df)
```

Calls Test Results:

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-01 00:00:27	-61.0	68.800003	1048.60	
1	2017-07-01 00:02:57	-61.0	68.769997	1855.54	
2	2017-07-01 00:05:29	-71.0	69.169998	1685.62	
3	2017-07-01 00:08:02	-65.0	69.279999	1770.92	
4	2017-07-01 00:10:30	-103.0	0.820000	256.07	
...	...	...	...	...	...
105823	2017-10-31 23:34:11	-114.0	-1.000000	1149.34	
105824	2017-10-31 23:38:56	-111.0	-1.000000	1177.42	
105825	2017-10-31 23:43:47	-111.0	0.800000	1159.05	
105826	2017-10-31 23:48:33	-111.0	-1.000000	1153.14	
105827	2017-10-31 23:50:19	-111.0	-1.000000	1153.14	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	UMTS	
1	90.0	SUCCESS	UMTS	
2	90.0	SUCCESS	UMTS	
3	90.0	SUCCESS	UMTS	
4	60.0	SUCCESS	UMTS	
...	...	...	...	...
105823	90.0	SUCCESS	LTE	
105824	90.0	SUCCESS	LTE	
105825	90.0	SUCCESS	LTE	
105826	90.0	SUCCESS	LTE	
105827	90.0	SUCCESS	LTE	

	Call Test Setup Time (s)	MOS
0	0.56	2.1
1	0.45	3.2
2	0.51	2.1
3	0.00	1.0
4	3.35	3.6
...	...	...
105823	0.53	4.3
105824	0.84	4.4
105825	0.43	4.4
105826	0.52	4.3
105827	0.80	4.4

[105148 rows x 9 columns]

## 3. Write the output of a Query into an EXCEL File

```
In [23]: # Write the query results to a new Excel file
output_file = 'call_test.xlsx'
call_test_df.to_excel(output_file, index=False)
```

```
In [25]: data = pd.read_excel('call_test.xlsx')
```

```
In [26]: # Display the first 5 rows of Output file extracted from the query
print(data.head())
```

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-01 00:00:27	-61.0	68.800003	1048.60	
1	2017-07-01 00:02:57	-61.0	68.769997	1855.54	
2	2017-07-01 00:05:29	-71.0	69.169998	1685.62	
3	2017-07-01 00:08:02	-65.0	69.279999	1770.92	
4	2017-07-01 00:10:30	-103.0	0.820000	256.07	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	UMTS	
1	90.0	SUCCESS	UMTS	
2	90.0	SUCCESS	UMTS	
3	90.0	SUCCESS	UMTS	
4	60.0	SUCCESS	UMTS	

	Call Test Setup Time (s)	MOS
0	0.56	2.1
1	0.45	3.2
2	0.51	2.1
3	0.00	1.0
4	3.35	3.6

## 4. Append to an Excel file

In [31]: *# Create a small DataFrame with new order data to append*

```
new_record = {
    "Date Of Test": "02-07-2017 10:30",
    "Signal (dBm)": -75,
    "Speed (m/s)": 55.2,
    "Distance from site (m)": 850.4,
    "Call Test Duration (s)": 120.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "LTE",
    "Call Test Setup Time (s)": 0.45,
    "MOS": 3.5
}

new_orders_df = pd.DataFrame([new_record])
```

In [35]: *# Define the file path*

```
file_path = 'Call Test Measure.xlsx'

existing_df = pd.read_excel(file_path)

#Concatenate the existing DataFrame and the new DataFrame
combined_df = pd.concat([existing_df, new_orders_df], ignore_index=True)

#Write the combined DataFrame back to the same Excel file, overwriting it
combined_df.to_excel(file_path, index=False)
print(f"Successfully appended new data and updated '{file_path}'.")
```

Successfully appended new data and updated 'Call Test Measure.xlsx'.

In [37]: *print(pd.read\_excel('Call Test Measure.xlsx', sheet\_name = 'Sheet1').iloc[105828]) # Display the new record ins*

```
Date Of Test          02-07-2017 10:30
Signal (dBm)         -75.0
Speed (m/s)          55.2
Distance from site (m) 850.4
Call Test Duration (s) 120.0
Call Test Result      SUCCESS
Call Test Technology   LTE
Call Test Setup Time (s) 0.45
MOS                  3.5
Name: 105828, dtype: object
```

## 5. Read a Excel Chunk-by-chunk

In [39]: *# File path*

```
file_path = "Call Test Measure.xlsx"

# Get total number of rows (without loading entire file)
total_rows = pd.read_excel(file_path, engine="openpyxl").shape[0]

# Define chunk size
chunk_size = 5000 # number of rows per chunk

# Read in chunks
for start_row in range(0, total_rows, chunk_size):
    # Read a chunk
    chunk = pd.read_excel(
        file_path,
```

```

engine="openpyxl",
skiprows=range(1, start_row + 1), # skip rows except header
nrows=chunk_size
)

print(f"Processing rows {start_row + 1} to {start_row + len(chunk)}...")
print(chunk.head())

```

Processing rows 1 to 5000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-01 00:00:27	-61.0	68.800003	1048.60	
1	2017-07-01 00:02:57	-61.0	68.769997	1855.54	
2	2017-07-01 00:05:29	-71.0	69.169998	1685.62	
3	2017-07-01 00:08:02	-65.0	69.279999	1770.92	
4	2017-07-01 00:10:30	-103.0	0.820000	256.07	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	UMTS	
1	90.0	SUCCESS	UMTS	
2	90.0	SUCCESS	UMTS	
3	90.0	SUCCESS	UMTS	
4	60.0	SUCCESS	UMTS	

	Call Test Setup Time (s)	MOS
0	0.56	2.1
1	0.45	3.2
2	0.51	2.1
3	0.00	1.0
4	3.35	3.6

Processing rows 5001 to 10000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-10 05:57:34	-114	0.0	400.43	
1	2017-07-10 06:00:07	-114	0.0	405.54	
2	2017-07-10 06:02:39	-114	0.0	397.18	
3	2017-07-10 06:05:12	-114	0.6	379.87	
4	2017-07-10 06:07:44	-114	0.6	379.87	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	LTE	
1	90.0	SUCCESS	LTE	
2	90.0	SUCCESS	LTE	
3	90.0	SUCCESS	LTE	
4	90.0	SUCCESS	LTE	

	Call Test Setup Time (s)	MOS
0	0.54	4.3
1	0.48	4.3
2	0.54	4.3
3	0.69	4.3
4	0.52	4.3

Processing rows 10001 to 15000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-14 22:34:48	-90.0	0.00	242.93	
1	2017-07-14 22:36:25	-103.0	0.00	532.08	
2	2017-07-14 22:37:20	-90.0	0.00	242.93	
3	2017-07-14 22:38:57	-92.0	7.14	62.33	
4	2017-07-14 22:39:52	-90.0	0.00	242.94	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	LTE	
1	90.0	SUCCESS	LTE	
2	90.0	SUCCESS	LTE	
3	90.0	SUCCESS	LTE	
4	90.0	SUCCESS	LTE	

	Call Test Setup Time (s)	MOS
0	0.51	4.4
1	0.55	4.3
2	0.59	4.4
3	0.50	4.3
4	0.48	4.4

Processing rows 15001 to 20000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-20 22:11:51	-71.0	-1.000000	5793.02	
1	2017-07-20 22:13:54	-105.0	1.140000	243.75	
2	2017-07-20 22:14:49	-63.0	69.279999	168.69	
3	2017-07-20 22:16:27	-105.0	1.140000	243.75	
4	2017-07-20 22:17:06	-59.0	-1.000000	2915.59	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.00	SUCCESS	UMTS	
1	60.00	SUCCESS	UMTS	
2	90.00	SUCCESS	UMTS	
3	60.00	SUCCESS	UMTS	

Call Test Setup Time (s) MOS

0	0.00	2.9
1	4.63	3.8
2	0.00	2.0
3	3.54	2.4
4	0.51	4.3

Processing rows 20001 to 25000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-07-26 21:24:30	-77	11.37		NaN
1	2017-07-26 21:24:50	-86	0.00		170.26
2	2017-07-26 21:26:15	-73	0.55		643.63
3	2017-07-26 21:26:47	-85	2.92		288.77
4	2017-07-26 21:27:06	-86	0.00		167.77

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	60.00	SUCCESS	UMTS	
1	59.93	SUCCESS	LTE	
2	60.00	SUCCESS	UMTS	
3	900.00	SUCCESS	UMTS	
4	60.00	SUCCESS	LTE	

Call Test Setup Time (s) MOS

0	3.81	2.7
1	0.63	4.3
2	4.04	2.7
3	3.79	1.6
4	0.74	4.4

Processing rows 25001 to 30000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-08-01 13:51:53	-77	6.34		1018.07
1	2017-08-01 13:52:01	-69	0.00		414.89
2	2017-08-01 13:55:01	-77	9.32		377.64
3	2017-08-01 13:56:35	-92	0.00		234.58
4	2017-08-01 13:58:27	-73	0.00		112.70

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	60.0	SUCCESS	GSM	
1	900.0	SUCCESS	UMTS	
2	60.0	SUCCESS	UMTS	
3	60.0	SUCCESS	LTE	
4	60.0	SUCCESS	UMTS	

Call Test Setup Time (s) MOS

0	3.58	4.4
1	3.76	2.1
2	3.81	2.7
3	1.00	4.3
4	3.60	2.7

Processing rows 30001 to 35000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-08-08 01:18:04	-79	0.0		126.23
1	2017-08-08 01:20:26	-81	0.0		125.99
2	2017-08-08 01:22:49	-79	0.0		126.09
3	2017-08-08 01:25:10	-81	0.0		126.20
4	2017-08-08 01:27:31	-77	0.0		126.19

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	UMTS	
1	90.0	SUCCESS	UMTS	
2	90.0	SUCCESS	UMTS	
3	90.0	SUCCESS	UMTS	
4	90.0	SUCCESS	UMTS	

Call Test Setup Time (s) MOS

0	4.09	2.1
1	4.01	2.1
2	3.78	2.1
3	3.99	1.0
4	5.08	2.1

Processing rows 35001 to 40000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-08-13 16:59:20	-83	13.92		1066.02
1	2017-08-13 17:01:11	-89	0.00		1178.43
2	2017-08-13 17:03:34	-91	0.00		1180.03
3	2017-08-13 17:05:56	-91	0.00		1180.52
4	2017-08-13 17:08:18	-91	0.00		1181.48

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	60.00	SUCCESS	UMTS	
1	89.86	SUCCESS	UMTS	
2	89.81	SUCCESS	UMTS	



3	89.83	SUCCESS	UMTS
4	90.00	SUCCESS	UMTS

Call Test Setup Time (s) MOS

0	0.00	2.7
1	3.82	2.1
2	3.46	4.4
3	3.81	2.1
4	3.69	1.0

Processing rows 40001 to 45000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0	2017-08-19 14:13:22	-87	-1.00	658.25
1	2017-08-19 14:14:32	-75	9.33	1432.96
2	2017-08-19 14:15:43	-85	-1.00	667.94
3	2017-08-19 14:17:18	-83	0.78	134.60
4	2017-08-19 14:18:04	-87	-1.00	681.29

	Call Test Duration (s)	Call Test Result	Call Test Technology \
0	90.0	SUCCESS	UMTS
1	60.0	SUCCESS	UMTS
2	90.0	SUCCESS	UMTS
3	60.0	SUCCESS	UMTS
4	90.0	SUCCESS	UMTS

Call Test Setup Time (s) MOS

0	4.42	2.7
1	4.21	2.7
2	4.21	1.0
3	0.00	2.7
4	3.98	1.0

Processing rows 45001 to 50000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0	2017-08-26 11:50:50	-69	1.182709	231.82
1	2017-08-26 11:53:12	-61	0.000000	114.06
2	2017-08-26 11:53:51	-81	2.270000	476.57
3	2017-08-26 11:55:34	-61	0.000000	112.28
4	2017-08-26 11:56:21	-77	3.450000	63.14

	Call Test Duration (s)	Call Test Result	Call Test Technology \
0	90.0	SUCCESS	UMTS
1	90.0	SUCCESS	UMTS
2	60.0	SUCCESS	UMTS
3	90.0	SUCCESS	UMTS
4	60.0	SUCCESS	UMTS

Call Test Setup Time (s) MOS

0	3.94	1.0
1	3.76	2.7
2	0.00	2.7
3	3.30	3.3
4	0.00	1.0

Processing rows 50001 to 55000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0	2017-08-30 19:12:24	-51	11.200005	275.11
1	2017-08-30 19:12:25	-71	3.470000	21.28
2	2017-08-30 19:13:18	-63	0.000000	551.37
3	2017-08-30 19:13:51	-71	0.000000	102.61
4	2017-08-30 19:14:47	-55	0.000000	435.66

	Call Test Duration (s)	Call Test Result	Call Test Technology \
0	90.0	SUCCESS	UMTS
1	60.0	SUCCESS	UMTS
2	60.0	SUCCESS	UMTS
3	60.0	SUCCESS	UMTS
4	90.0	SUCCESS	UMTS

Call Test Setup Time (s) MOS

0	4.08	4.2
1	2.10	1.0
2	3.31	1.0
3	8.76	4.4
4	4.55	3.4

Processing rows 55001 to 60000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0	2017-09-06 09:52:46	-67	6.520000	27.53
1	2017-09-06 09:52:48	-53	29.358074	17526.55
2	2017-09-06 09:53:49	-67	5.340000	NaN
3	2017-09-06 09:54:51	-89	7.590000	295.71
4	2017-09-06 09:54:53	-71	5.880000	365.43

	Call Test Duration (s)	Call Test Result	Call Test Technology \
0	60.0	SUCCESS	UMTS
1	90.0	SUCCESS	UMTS

2	60.0	SUCCESS	UMTS
3	60.0	SUCCESS	UMTS
4	60.0	SUCCESS	UMTS

Call Test Setup Time (s) MOS

0	0.00	2.7
1	4.24	2.0
2	4.34	2.7
3	3.96	3.3
4	0.00	2.7

Processing rows 60001 to 65000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-09-12 11:22:27	-107	0.0	282.24	
1	2017-09-12 11:23:14	-85	0.0	741.43	
2	2017-09-12 11:24:43	-107	0.0	282.65	
3	2017-09-12 11:25:35	-85	0.0	741.74	
4	2017-09-12 11:26:58	-107	0.0	282.51	

Call Test Duration (s) Call Test Result Call Test Technology \

0	60.0	SUCCESS	LTE
1	90.0	SUCCESS	UMTS
2	60.0	SUCCESS	LTE
3	90.0	SUCCESS	UMTS
4	60.0	SUCCESS	LTE

Call Test Setup Time (s) MOS

0	0.68	4.3
1	3.66	2.1
2	0.86	4.3
3	5.24	4.4
4	0.77	4.3

Processing rows 65001 to 70000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-09-15 18:27:40	-79	-1.0	241397.58	
1	2017-09-15 18:29:14	-65	0.0	424.43	
2	2017-09-15 18:30:10	-84	-1.0	241397.58	
3	2017-09-15 18:31:36	-65	0.0	424.45	
4	2017-09-15 18:32:41	-83	-1.0	241397.58	

Call Test Duration (s) Call Test Result Call Test Technology \

0	90.00	SUCCESS	LTE
1	90.00	SUCCESS	UMTS
2	90.00	SUCCESS	LTE
3	89.88	SUCCESS	UMTS
4	90.00	SUCCESS	LTE

Call Test Setup Time (s) MOS

0	0.40	4.4
1	3.50	4.4
2	0.51	4.4
3	3.22	2.7
4	0.79	4.4

Processing rows 70001 to 75000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-09-19 08:44:58	-102	9.020000	3592.35	
1	2017-09-19 08:45:00	-77	0.000000	1073.72	
2	2017-09-19 08:45:07	-75	47.340107	NaN	
3	2017-09-19 08:46:36	-75	0.000000	1074.20	
4	2017-09-19 08:47:18	-93	6.860000	676.84	

Call Test Duration (s) Call Test Result Call Test Technology \

0	90.0	SUCCESS	LTE
1	60.0	SUCCESS	UMTS
2	90.0	SUCCESS	UMTS
3	60.0	SUCCESS	UMTS
4	60.0	SUCCESS	UMTS

Call Test Setup Time (s) MOS

0	0.60	4.4
1	0.00	2.2
2	9.09	1.0
3	0.00	1.0
4	4.03	2.5

Processing rows 75001 to 80000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-09-28 04:35:27	-102	0.0	322.27	
1	2017-09-28 04:36:07	-73	0.0	496.59	
2	2017-09-28 04:38:00	-102	0.0	320.88	
3	2017-09-28 04:38:29	-71	0.0	496.58	
4	2017-09-28 04:40:31	-101	0.0	320.62	

Call Test Duration (s) Call Test Result Call Test Technology \

0	90.0	SUCCESS	LTE
---	------	---------	-----

1	90.0	SUCCESS	UMTS
2	90.0	SUCCESS	LTE
3	90.0	SUCCESS	UMTS
4	90.0	SUCCESS	LTE

Call Test Setup Time (s) MOS		
0	0.93	4.4
1	3.50	1.0
2	0.57	4.4
3	3.72	2.1
4	1.27	4.4

Processing rows 80001 to 85000...

Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0 2017-09-30 16:04:19	-79	0.0	518.03
1 2017-09-30 16:04:43	-89	-1.0	5536.40
2 2017-09-30 16:06:41	-79	0.0	518.20
3 2017-09-30 16:07:13	-81	-1.0	5536.40
4 2017-09-30 16:07:20	-71	13.5	261.44

Call Test Duration (s) Call Test Result Call Test Technology \			
0	90.0	SUCCESS	UMTS
1	90.0	SUCCESS	UMTS
2	90.0	SUCCESS	UMTS
3	90.0	SUCCESS	UMTS
4	60.0	SUCCESS	UMTS

Call Test Setup Time (s) MOS		
0	4.20	2.1
1	0.44	2.9
2	4.23	2.1
3	0.00	1.8
4	0.00	4.3

Processing rows 85001 to 90000...

Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0 2017-10-04 15:58:54	-51.0	1.310038	519.76
1 2017-10-04 15:59:31	-51.0	67.809998	38648.53
2 2017-10-04 16:01:15	-51.0	0.190263	523.74
3 2017-10-04 16:03:37	-51.0	8.630000	456.77
4 2017-10-04 16:04:17	-51.0	29.680000	640.54

Call Test Duration (s) Call Test Result Call Test Technology \			
0	90.0	SUCCESS	UMTS
1	90.0	SUCCESS	UMTS
2	90.0	SUCCESS	UMTS
3	90.0	SUCCESS	UMTS
4	90.0	SUCCESS	UMTS

Call Test Setup Time (s) MOS		
0	4.61	4.4
1	0.00	1.0
2	4.43	3.9
3	4.04	4.4
4	0.63	2.1

Processing rows 90001 to 95000...

Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0 2017-10-09 05:04:19	-108	-1.0	NaN
1 2017-10-09 05:06:51	-108	-1.0	NaN
2 2017-10-09 05:09:22	-108	-1.0	NaN
3 2017-10-09 05:11:53	-108	-1.0	NaN
4 2017-10-09 05:14:25	-108	-1.0	NaN

Call Test Duration (s) Call Test Result Call Test Technology \			
0	90.0	SUCCESS	LTE
1	90.0	SUCCESS	LTE
2	90.0	SUCCESS	LTE
3	90.0	SUCCESS	LTE
4	90.0	SUCCESS	LTE

Call Test Setup Time (s) MOS		
0	0.83	4.4
1	0.69	4.4
2	1.03	4.4
3	1.09	4.4
4	0.67	4.4

Processing rows 95001 to 100000...

Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m) \
0 2017-10-13 19:38:11	-81	0.0	512.15
1 2017-10-13 19:39:54	-51	-1.0	301678.25
2 2017-10-13 19:40:34	-81	0.0	512.12
3 2017-10-13 19:42:25	-51	-1.0	301678.25
4 2017-10-13 19:42:56	-81	0.0	512.14

Call Test Duration (s) Call Test Result Call Test Technology \			
--	--	--	--

0	90.0	SUCCESS	UMTS
1	90.0	SUCCESS	UMTS
2	90.0	SUCCESS	UMTS
3	90.0	SUCCESS	UMTS
4	90.0	SUCCESS	UMTS

	Call Test Setup Time (s)	MOS
0	5.14	2.7
1	0.00	2.8
2	4.04	4.4
3	0.00	3.4
4	3.66	2.1

Processing rows 100001 to 105000...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-10-20 17:32:23	-91	-1.0	566.47	
1	2017-10-20 17:34:46	-89	-1.0	566.47	
2	2017-10-20 17:37:08	-91	-1.0	566.47	
3	2017-10-20 17:39:29	-91	-1.0	1231.97	
4	2017-10-20 17:41:50	-77	-1.0	487.13	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	UMTS	
1	90.0	SUCCESS	UMTS	
2	90.0	SUCCESS	UMTS	
3	90.0	SUCCESS	UMTS	
4	90.0	SUCCESS	UMTS	

	Call Test Setup Time (s)	MOS
0	4.54	2.1
1	4.46	2.1
2	4.67	1.0
3	4.88	2.7
4	4.36	4.2

Processing rows 105001 to 105829...

	Date Of Test	Signal (dBm)	Speed (m/s)	Distance from site (m)	\
0	2017-10-30 22:47:13	-51	1.059292	234.28	
1	2017-10-30 22:49:37	-51	18.520002	198.93	
2	2017-10-30 22:49:57	-110	-1.000000	759.69	
3	2017-10-30 22:52:00	-51	41.260040	927.24	
4	2017-10-30 22:54:22	-57	43.760113	2306.65	

	Call Test Duration (s)	Call Test Result	Call Test Technology	\
0	90.0	SUCCESS	UMTS	
1	90.0	SUCCESS	UMTS	
2	90.0	SUCCESS	LTE	
3	90.0	SUCCESS	UMTS	
4	90.0	SUCCESS	UMTS	

	Call Test Setup Time (s)	MOS
0	4.69	4.4
1	4.06	3.8
2	0.56	4.4
3	4.79	3.2
4	5.33	4.4

## 6. Writing numeric data to a new Excel file

```
In [40]: # Create numeric data
data = {
    "Product_ID": [101, 102, 103, 104],
    "Quantity_Sold": [20, 35, 15, 50],
    "Price_Per_Unit": [100, 150, 120, 80],
    "Total_Sales": [20*100, 35*150, 15*120, 50*80] # Calculated column
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Write to Excel
output_file = "numeric_data_Excel_Sheet.xlsx"
df.to_excel(output_file, index=False)

print(f"Numeric data written successfully to {output_file}")
```

Numeric data written successfully to numeric\_data\_Excel\_Sheet.xlsx

```
In [43]: data = pd.read_excel('numeric_data_Excel_Sheet.xlsx')
print(data) # print the numeric data from excel
```

	Product_ID	Quantity_Sold	Price_Per_Unit	Total_Sales
0	101	20	100	2000
1	102	35	150	5250
2	103	15	120	1800
3	104	50	80	4000

## 7. Write text data into a Excel File

```
In [46]: import pandas as pd

# Create text data
data = {
    "Name": ["Alice", "Bob", "Charlie", "David"],
    "City": ["New York", "Los Angeles", "Chicago", "Houston"],
    "Department": ["HR", "IT", "Finance", "Marketing"]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Write to Excel
output_file = "text_data_excel_sheet.xlsx"
df.to_excel(output_file, index=False)

print(f"Text data written successfully to {output_file}")
```

Text data written successfully to text\_data\_excel\_sheet.xlsx

```
In [48]: data = pd.read_excel('text_data_excel_sheet.xlsx')
print(data) # print the text data from excel
```

	Name	City	Department
0	Alice	New York	HR
1	Bob	Los Angeles	IT
2	Charlie	Chicago	Finance
3	David	Houston	Marketing

## TASK-3: Handling JSON

### 1. Write the output of a Python Code into a JSON Format/File

```
In [52]: import json
```

```
In [58]: # Extract the Success call test result from Call Test Measure.xlsx
call_test_df = df[df['Call Test Result'] == 'SUCCESS']
print("Calls Test Results:\n", call_test_df)
```

Calls Test Results:

	Date Of Test	Signal (dBm)	Speed (m/s)	\
0	2017-07-01 00:00:27	-61.0	68.800003	
1	2017-07-01 00:02:57	-61.0	68.769997	
2	2017-07-01 00:05:29	-71.0	69.169998	
3	2017-07-01 00:08:02	-65.0	69.279999	
4	2017-07-01 00:10:30	-103.0	0.820000	
...	...	...	...	
105824	2017-10-31 23:38:56	-111.0	-1.000000	
105825	2017-10-31 23:43:47	-111.0	0.800000	
105826	2017-10-31 23:48:33	-111.0	-1.000000	
105827	2017-10-31 23:50:19	-111.0	-1.000000	
105828	02-07-2017 10:30	-75.0	55.200000	

	Distance from site (m)	Call Test Duration (s)	Call Test Result	\
0	1048.60	90.0	SUCCESS	
1	1855.54	90.0	SUCCESS	
2	1685.62	90.0	SUCCESS	
3	1770.92	90.0	SUCCESS	
4	256.07	60.0	SUCCESS	
...	...	...	...	
105824	1177.42	90.0	SUCCESS	
105825	1159.05	90.0	SUCCESS	
105826	1153.14	90.0	SUCCESS	
105827	1153.14	90.0	SUCCESS	
105828	850.40	120.0	SUCCESS	

	Call Test Technology	Call Test Setup Time (s)	MOS
0	UMTS	0.56	2.1
1	UMTS	0.45	3.2
2	UMTS	0.51	2.1
3	UMTS	0.00	1.0
4	UMTS	3.35	3.6
...	...	...	...
105824	LTE	0.84	4.4
105825	LTE	0.43	4.4
105826	LTE	0.52	4.3
105827	LTE	0.80	4.4
105828	LTE	0.45	3.5

[105149 rows x 9 columns]

```
In [60]: # The 'orient' parameter controls the JSON format.
# 'records' creates a list of JSON objects, with each object being a row.
# The 'indent' parameter makes the file human-readable.
call_test_df.to_json(
    'call_test.json',
    orient='records',
    indent=4
)

print("Call test results have been written to call_test.json")
```

Call test results have been written to call\_test.json

```
In [62]: output_filename = 'call_test.json' #Write the filtered DataFrame to a JSON file.
call_test_df.to_json(
    output_filename,
    orient='records',
    indent=4
)
```

```
In [66]: # Open the JSON file safely
try:
    with open(output_filename, 'r') as json_file:
        # Load the data into a Python object (list/dict)
        data = json.load(json_file)

    # Pretty-print with indentation
    print(f"Displaying the content of {output_filename} (first 10 records only):")
    if isinstance(data, list):
        # Print only first 10 items to avoid flooding console
        print(json.dumps(data[:10], indent=4))
    else:
        print(json.dumps(data, indent=4))

    print(f"\nCall test results have been written to and displayed from {output_filename}")

except FileNotFoundError:
    print(f"Error: File '{output_filename}' not found.")
except json.JSONDecodeError:
    print(f"Error: File '{output_filename}' is not a valid JSON file.")
except Exception as e:
    print("An unexpected error occurred:", e)
```

Displaying the content of call\_test.json (first 10 records only):

```
[
  {
    "Date Of Test": 1498867227000,
    "Signal (dBm)": -61.0,
    "Speed (m/s)": 68.8000030518,
    "Distance from site (m)": 1048.6,
    "Call Test Duration (s)": 90.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "UMTS",
    "Call Test Setup Time (s)": 0.56,
    "MOS": 2.1
  },
  {
    "Date Of Test": 1498867377000,
    "Signal (dBm)": -61.0,
    "Speed (m/s)": 68.7699966431,
    "Distance from site (m)": 1855.54,
    "Call Test Duration (s)": 90.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "UMTS",
    "Call Test Setup Time (s)": 0.45,
    "MOS": 3.2
  },
  {
    "Date Of Test": 1498867529000,
    "Signal (dBm)": -71.0,
    "Speed (m/s)": 69.1699981689,
    "Distance from site (m)": 1685.62,
    "Call Test Duration (s)": 90.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "UMTS",
    "Call Test Setup Time (s)": 0.51,
    "MOS": 2.1
  },
  {
    "Date Of Test": 1498867682000,
    "Signal (dBm)": -65.0,
    "Speed (m/s)": 69.2799987793,
    "Distance from site (m)": 1770.92,
    "Call Test Duration (s)": 90.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "UMTS",
    "Call Test Setup Time (s)": 0.0,
    "MOS": 1.0
  },
  {
    "Date Of Test": 1498867830000,
    "Signal (dBm)": -103.0,
    "Speed (m/s)": 0.8199999928,
    "Distance from site (m)": 256.07,
    "Call Test Duration (s)": 60.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "UMTS",
    "Call Test Setup Time (s)": 3.35,
    "MOS": 3.6
  },
  {
    "Date Of Test": 1498867837000,
    "Signal (dBm)": -61.0,
    "Speed (m/s)": 68.8600006104,
    "Distance from site (m)": 452.5,
    "Call Test Duration (s)": 90.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "UMTS",
    "Call Test Setup Time (s)": 0.0,
    "MOS": 1.0
  },
  {
    "Date Of Test": 1498867988000,
    "Signal (dBm)": -63.0,
    "Speed (m/s)": 68.7600021362,
    "Distance from site (m)": 899.88,
    "Call Test Duration (s)": 90.0,
    "Call Test Result": "SUCCESS",
    "Call Test Technology": "UMTS",
    "Call Test Setup Time (s)": 0.49,
    "MOS": 2.1
  },
  {
    "Date Of Test": 1498868277000,
    "Signal (dBm)": -73.0,
    "Speed (m/s)": 70.0199966431,
```

```

        "Distance from site (m)": 296.19,
        "Call Test Duration (s)": 90.0,
        "Call Test Result": "SUCCESS",
        "Call Test Technology": "UMTS",
        "Call Test Setup Time (s)": 0.0,
        "MOS": 1.0
    },
    {
        "Date Of Test": 1498868429000,
        "Signal (dBm)": -78.0,
        "Speed (m/s)": 27.7900009155,
        "Distance from site (m)": null,
        "Call Test Duration (s)": 90.0,
        "Call Test Result": "SUCCESS",
        "Call Test Technology": "LTE",
        "Call Test Setup Time (s)": 0.42,
        "MOS": 4.4
    },
    {
        "Date Of Test": 1498868581000,
        "Signal (dBm)": -61.0,
        "Speed (m/s)": 22.1200008392,
        "Distance from site (m)": 21532.25,
        "Call Test Duration (s)": 90.0,
        "Call Test Result": "SUCCESS",
        "Call Test Technology": "UMTS",
        "Call Test Setup Time (s)": 0.52,
        "MOS": 2.1
    }
]

```

Call test results have been written to and displayed from call\_test.json

## 2.Parse a JSON file and search for a value

```

In [68]: import json

# Define the filename and the value you want to find.
filename = 'call_test.json'
search_value = 'UMTS'

# Read and parse the JSON file.
try:
    with open(filename, 'r') as file:
        # data is a list of dictionaries because of `orient='records'`
        data = json.load(file)
except FileNotFoundError:
    print(f"Error: The file '{filename}' was not found.")
    data = None
except json.JSONDecodeError:
    print(f"Error: The file '{filename}' is not a valid JSON file.")
    data = None

# Proceed only if the file was loaded successfully.
if data and isinstance(data, list): # Check if data is a list
    # The data variable is already the list of records.
    call_test_tech = []
    for tech in data:
        if tech.get('Call Test Technology') == search_value:
            call_test_tech.append(tech)

    # Print the results.
    if call_test_tech:
        print(f"Found {len(call_test_tech)} call test technologies in {search_value}:")
        for tech in call_test_tech:
            # Adjust the print statement based on what keys are in your data.
            print(f" - Call Test Duration (s): {tech.get('Call Test Duration (s)')}, Call Test Result: {tech.get('Call Test Result')}")
    else:
        print(f"No call test technologies found in {search_value}.")
else:
    print("Failed to load or parse JSON file correctly.")

```

Found 72233 call test technologies in UMTS:

```

- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 60.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS
- Call Test Duration (s): 90.0, Call Test Result: SUCCESS

```



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]



[illegible]



[illegible]

[illegible]



[illegible]

[illegible]





[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]







































[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]











[illegible]



[illegible]









[illegible]























[illegible]







[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]































[illegible]

[illegible]































































[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

















































[illegible]























[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]





[illegible]













[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]







[illegible]



[illegible]











[illegible]

[illegible]













[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]



[illegible]

[illegible]

[illegible]



[illegible]







































[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]



[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]



[illegible]





[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]



[illegible]



[illegible]

[illegible]



[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]



[illegible]

[illegible]



[illegible]

[illegible]



[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]



[illegible]

[illegible]

[illegible]



















[illegible]











[illegible]

[illegible]























[illegible]























[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]







[illegible]







[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]



[illegible]

[illegible]



[illegible]







[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]











[illegible]

IOPub data rate exceeded.

The Jupyter server will temporarily stop sending output to the client in order to avoid crashing it.

To change this limit, set the config variable  
`--ServerApp.iopub\_data\_rate\_limit`.

Current values:

ServerApp.iopub\_data\_rate\_limit=1000000.0 (bytes/sec)

ServerApp.rate\_limit\_window=3.0 (secs)

In [ ]:

## CSI3007 – Advanced Python Programming

NAME : DHEVISHREE GN

REG NUM : 22MID0136

### LAB ACTIVITY – 12

Datasets Used: Input Dataset :

Housing.csv Output Dataset :

employee\_data.csv (Created CSV file using io.StringIO())

numeric\_data.csv (Writing numeric data to a new CSV file)

text\_data.csv (Write text data into a CSV File)

<https://drive.google.com/drive/u/1/folders/1XeLwtloIK7P8ydsy0suyjwGJAD3kQ0I0>

### LAB ACTIVITY – 14

Datasets Used:

Input Dataset : call\_test.xlsx

<https://drive.google.com/drive/u/1/folders/1l54wREO5O3aBKQFSJgCzW3aUVH7LnvWz>