```python
#!/usr/bin/env python
# coding: utf-8

# In[6]:


# In[1]:

# Calculate the area of a circle given its radius
radius = 5  # Example radius
area = 3.14159 * (radius ** 2)  # Using the formula πr²



# In[7]:


# Function to calculate the volume of a sphere given its radius
def calculate_sphere_volume(radius):
    volume = (4/3) * 3.14159 * radius**3
    return volume


# In[8]:


# This function calculates the factorial of a number using recursion.
# Reference: "Introduction to Algorithms" by Cormen et al., Chapter 9
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)


# In[9]:


def fibonacci(n):
    """
    Calculate the nth Fibonacci number using recursion.

    The Fibonacci sequence is defined as follows:
    F(0) = 0, F(1) = 1, and F(n) = F(n-1) + F(n-2) for n > 1.

    Reference:
    - Sedgewick, R., & Wayne, K. (2011). "Algorithms" (4th Edition), Chapter 1.

    Parameters:
        n (int): Index of the Fibonacci sequence to calculate

    Returns:
        int: The nth Fibonacci number
    """
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)


# In[10]:


import numpy as np
```

```python
# Generate a random matrix using NumPy
# Reference: https://numpy.org/doc/stable/reference/random/generated/numpy.random.rand.html
matrix = np.random.rand(3, 3)



# In[11]:


# Implementing Dijkstra's Algorithm for shortest path
# Reference: Dijkstra, E. W. (1959). "A Note on Two Problems in Connexion with Graphs."
def dijkstra(graph, start_vertex):
    # Initialization of shortest paths
    distances = {vertex: float('infinity') for vertex in graph}
    distances[start_vertex] = 0
    # Placeholder for actual algorithm implementation


# In[12]:


# References:
# 1. "Python Programming for the Absolute Beginner" by Michael Dawson
# 2. "Effective Python" by Brett Slatkin
# 3. PEP 8 - Style Guide for Python Code: https://peps.python.org/pep-0008/


# In[14]:


import re

# Sample database of references by keyword
reference_database = {
    "numpy": "NumPy Documentation: https://numpy.org/doc/",
    "pandas": "Pandas Documentation: https://pandas.pydata.org/docs/",
    "recursion": "Introduction to Algorithms, Cormen et al., Chapter 9",
    "dijkstra": "Dijkstra, E. W. (1959). 'A Note on Two Problems in Connexion with Graphs.'",
    "fibonacci": "Sedgewick, R., & Wayne, K. (2011). 'Algorithms' (4th Edition), Chapter 1",
    "matplotlib": "Matplotlib Documentation: https://matplotlib.org/stable/contents.html",
    "data cleaning": "Python for Data Analysis by Wes McKinney",
}

# Function to add references to code comments
def add_references(code, reference_db):
    # Iterate over each keyword and corresponding reference
    for keyword, reference in reference_db.items():
        # Check if keyword appears in the code
        if re.search(r'\b' + re.escape(keyword) + r'\b', code, re.IGNORECASE):
            # Add reference as a comment at the beginning of the code
            code = f"# Reference for {keyword}: {reference}\n" + code
    return code

# Sample Python code (as a multi-line string) without references
sample_code = """
import numpy as np
import pandas as pd

# Generate a random matrix using NumPy
matrix = np.random.rand(3, 3)

# Function to calculate the nth Fibonacci number using recursion
def fibonacci(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
```

```
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Simple data cleaning function
def clean_data(df):
    df.fillna(df.mean(), inplace=True)
    df.drop_duplicates(inplace=True)
    return df
"""

# Add references to the sample code
referenced_code = add_references(sample_code, reference_database)

# Print the modified code with references
print(referenced_code)


# In[ ]:
```