Assignment 1: Data Frame

SET A

1. Create a dictionary that stores the mobile names list as value for ‘Mobiles’ key and their price list as value for ‘Price’ key. Create DataFrame from this dictionary.

Answer

```
import pandas as pd

# Creating dictionary
data = {
    'Mobiles': ['iPhone 14', 'Samsung Galaxy S23', 'OnePlus 11', 'Redmi Note 12'],
    'Price': [79999, 74999, 56999, 18999]
}

# Creating DataFrame
df = pd.DataFrame(data)

print(df)
```

2. Create a data.csv file containing employee name, designation, salary of an employee. Display the file as DataFrame.

Answer

```
import pandas as pd

# Creating data
data = {
    'Employee_Name': ['John', 'Emma', 'David', 'Sophia'],
    'Designation': ['Manager', 'Analyst', 'Clerk', 'HR'],
    'Salary': [60000, 45000, 30000, 40000]
}

df = pd.DataFrame(data)

# Saving to CSV
df.to_csv('data.csv', index=False)

# Reading CSV
df2 = pd.read_csv('data.csv')

print(df2)
```

3. Write a python program to calculate sum, mean, median, mode, Standard deviation of following values:
CA:[87,89,98,94,78,77]

Answer

```
import numpy as np
import statistics as stats

CA = [87, 89, 98, 94, 78, 77]

# Calculations
print("Sum:", sum(CA))
print("Mean:", np.mean(CA))
print("Median:", np.median(CA))
print("Mode:", stats.mode(CA))
print("Standard Deviation:", np.std(CA))
```

4. Write a python program to fill missing values (NaN) in a DataFrame using fillna() with mean value and detect duplicates using duplicated() method.

Answer

```
import pandas as pd
import numpy as np

# Sample DataFrame with missing values and duplicates
data = {
   'A': [10, 20, np.nan, 40, 20],
   'B': [5, np.nan, 15, 20, 5]
}

df = pd.DataFrame(data)

# Fill missing values with mean
df_filled = df.fillna(df.mean())

print("After Filling NaN:")
print(df_filled)

# Detect duplicates
print("Duplicate Rows:")
print(df_filled.duplicated())
```

5. Create a DataFrame from a dictionary and display the first 5 rows.

Answer

```
import pandas as pd

data = {
   'Name': ['A', 'B', 'C', 'D', 'E', 'F', 'G'],
   'Marks': [85, 90, 78, 88, 76, 95, 89]
}

df = pd.DataFrame(data)
```

```
print(df.head())
```

6. Load sales.csv file into a DataFrame

Answer

Option 2: Create sales.csv using Python

You can also generate it programmatically:

```
import pandas as pd

# Create sample sales data
data = {
    'customer_name': ['John', 'Emma', 'David', 'Sophia', 'Alice'],
    'product': ['Laptop', 'Mobile', 'Tablet', 'Headphones', 'Keyboard'],
    'quantity': [2, 1, 3, 5, 2],
    'price': [50000, 30000, 20000, 2000, 1500],
    'region': ['North', 'South', 'North', 'East', 'North']
}

# Create DataFrame
df = pd.DataFrame(data)

# Save as CSV
df.to_csv('sales.csv', index=False)

print("sales.csv file created successfully!")
```

a) check its shape
b) Add a new column total_amount = price * quantity.
c) Select only the columns: customer_name, product, quantity, price.
d) Filter rows where the region is "North".

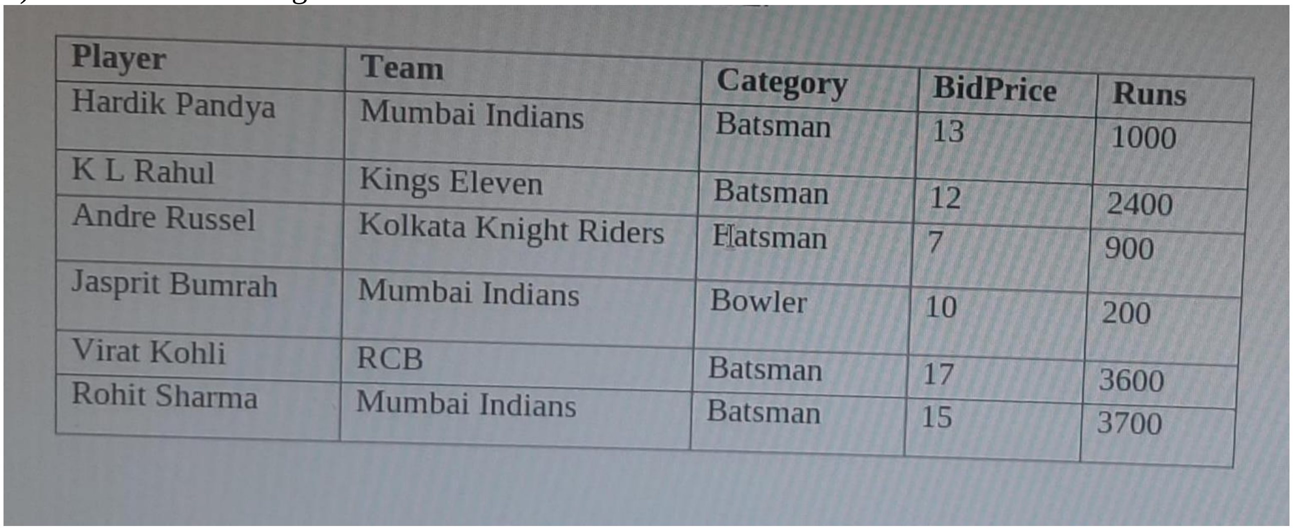
Answer

```
import pandas as pd

# Load CSV file
df = pd.read_csv('sales.csv')

# a) Check shape
print("Shape of DataFrame:", df.shape)

# b) Add new column total_amount
df['total_amount'] = df['price'] * df['quantity']

# c) Select specific columns
selected_columns = df[['customer_name', 'product', 'quantity', 'price']]
print(selected_columns)

# d) Filter rows where region is "North"
```

north_region = df[df['region'] == 'North']
print(north_region)

SET B

1) Consider a following record in DataFrame IPL.



Player	Team	Category	BidPrice	Runs
Hardik Pandya	Mumbai Indians	Batsman	13	1000
K L Rahul	Kings Eleven	Batsman	12	2400
Andre Russel	Kolkata Knight Riders	Batsman	7	900
Jasprit Bumrah	Mumbai Indians	Bowler	10	200
Virat Kohli	RCB	Batsman	17	3600
Rohit Sharma	Mumbai Indians	Batsman	15	3700

Create a above DataFrame in python write python code for following

a) Retrieve first 2 rows

b) Retrieve last 3 rows

c) Add null values in DataFrame.

d) Find most expensive player.

e) Print total players per team.

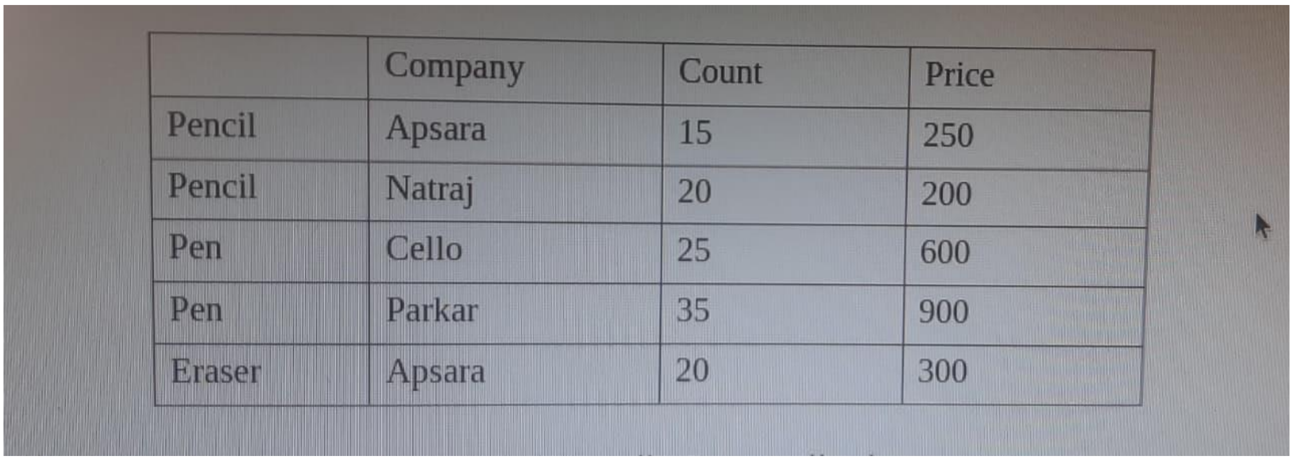
f) Find average runs of each player.

g) Drop rows with missing data.

Answer

```
import pandas as pd
import numpy as np

# 1 Create IPL DataFrame
data = {
    'Player': ['Hardik Pandya', 'K L Rahul', 'Andre Russel', 'Jasprit Bumrah', 'Virat Kohli',
'Rohit Sharma'],
    'Team': ['Mumbai Indians', 'Kings Eleven', 'Kolkata Knight Riders', 'Mumbai Indians',
'RCB', 'Mumbai Indians'],
    'Category': ['Batsman', 'Batsman', 'Batsman', 'Bowler', 'Batsman', 'Batsman'],
    'BidPrice': [13, 12, 7, 10, 17, 15],
    'Runs': [1000, 2400, 900, 200, 3600, 3700]
}

ipl_df = pd.DataFrame(data)

print("Original IPL DataFrame:")
print(ipl_df)

# ------------------------------------------------------
# a) Retrieve first 2 rows
print("\nFirst 2 rows:")
print(ipl_df.head(2))

# b) Retrieve last 3 rows
print("\nLast 3 rows:")
print(ipl_df.tail(3))

# c) Add null values (example: set 'Runs' of 3rd player as NaN)
ipl_df.loc[2, 'Runs'] = np.nan
print("\nDataFrame with null value added:")
print(ipl_df)

# d) Find most expensive player (highest BidPrice)
most_expensive = ipl_df.loc[ipl_df['BidPrice'].idxmax()]
print("\nMost expensive player:")
print(most_expensive)

# e) Print total players per team
players_per_team = ipl_df['Team'].value_counts()
print("\nTotal players per team:")
print(players_per_team)

# f) Find average runs of each player
# Since each player has only one 'Runs' value, we'll just show it as is
# But if there were multiple matches, we could group by player
average_runs = ipl_df[['Player', 'Runs']]
print("\nAverage runs per player:")
print(average_runs)

# g) Drop rows with missing data
ipl_no_nan = ipl_df.dropna()
print("\nDataFrame after dropping rows with missing data:")
print(ipl_no_nan)
```

2) Create a following DataFrame named as “data”. Write the python code for the following commands.



	Company	Count	Price
Pencil	Apsara	15	250
Pencil	Natraj	20	200
Pen	Cello	25	600
Pen	Parkar	35	900
Eraser	Apsara	20	300

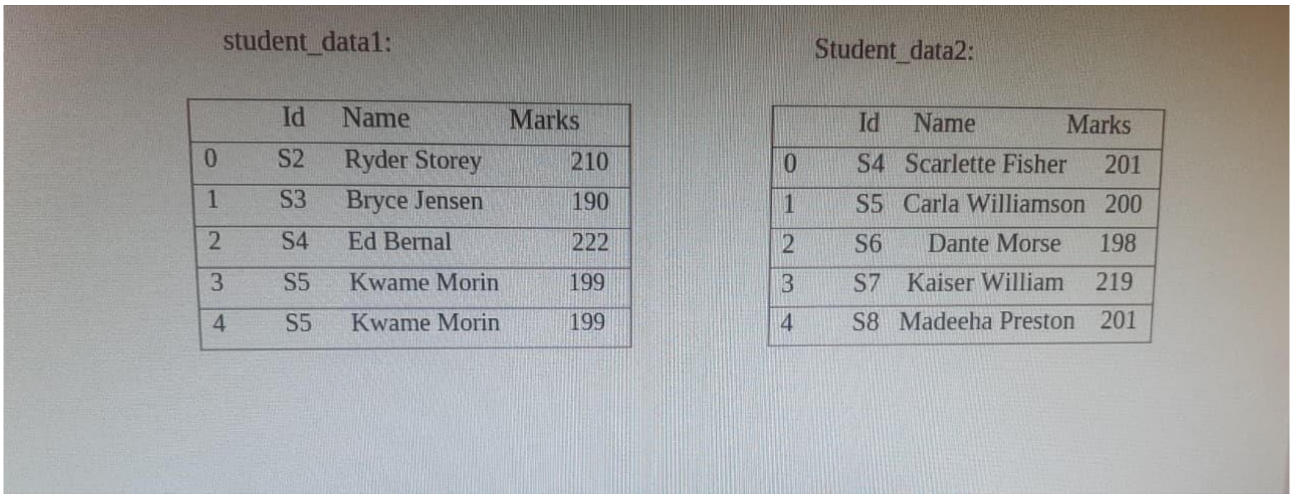
a) Find all rows with the label “Pencil”. Extract all columns
b) Change the Eraser count as 25 instead of 20.
c) List only the columns Company and Price.
d) List only rows with labels ‘Pencil’ and ‘Pen’
e) Delete column Count from the above DataFrame.

Answer

```
import pandas as pd

# 1□ Create the DataFrame
data_dict = {
   'Company': ['Apsara', 'Natraj', 'Cello', 'Parkar', 'Apsara'],
   'Product': ['Pencil', 'Pencil', 'Pen', 'Pen', 'Eraser'],
   'Count': [15, 20, 25, 35, 20],
   'Price': [250, 200, 600, 900, 300]
}

data = pd.DataFrame(data_dict)
print("Original DataFrame:")
print(data)

# --------------------------------------------------------
# a) Find all rows with the label “Pencil”. Extract all columns
pencil_rows = data[data['Product'] == 'Pencil']
print("\nRows with Product 'Pencil':")
print(pencil_rows)

# b) Change the Eraser count as 25 instead of 20
data.loc[data['Product'] == 'Eraser', 'Count'] = 25
print("\nDataFrame after changing Eraser count to 25:")
print(data)

# c) List only the columns Company and Price
company_price = data[['Company', 'Price']]
print("\nColumns Company and Price:")
print(company_price)
```

```
# d) List only rows with labels ‘Pencil’ and ‘Pen’
pencil_pen_rows = data[data['Product'].isin(['Pencil', 'Pen'])]
print("\nRows with Product 'Pencil' or 'Pen':")
print(pencil_pen_rows)

# e) Delete column Count from the DataFrame
data_without_count = data.drop(columns=['Count'])
print("\nDataFrame after deleting 'Count' column:")
print(data_without_count)
```

3)Write a python program to join the two DataFrames with matching records from both sides where available.



student_data1:

	Id	Name	Marks
0	S2	Ryder Storey	210
1	S3	Bryce Jensen	190
2	S4	Ed Bernal	222
3	S5	Kwame Morin	199
4	S5	Kwame Morin	199

Student_data2:

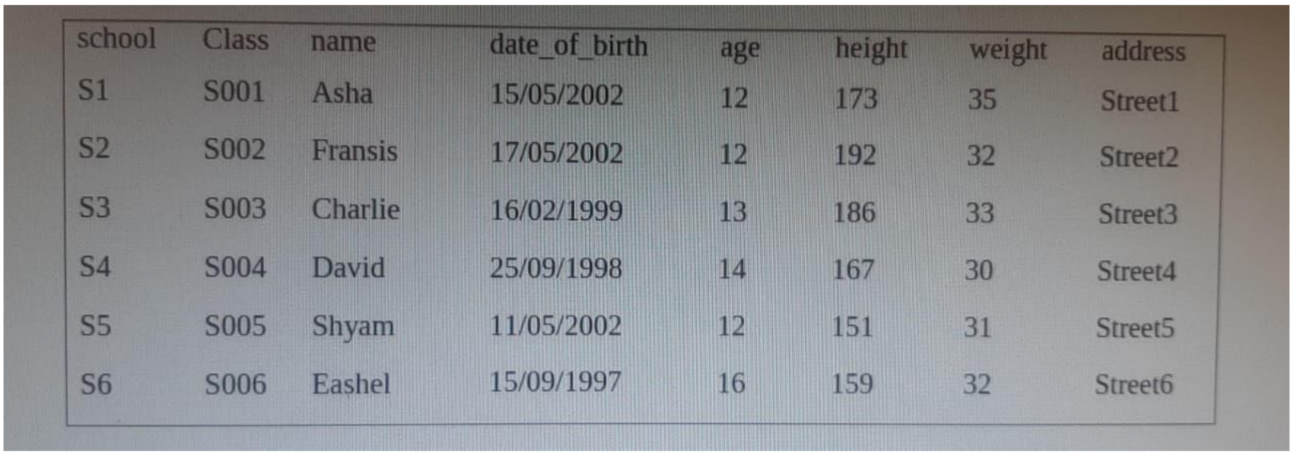
	Id	Name	Marks
0	S4	Scarlette Fisher	201
1	S5	Carla Williamson	200
2	S6	Dante Morse	198
3	S7	Kaiser William	219
4	S8	Madeeha Preston	201

Answer

```
import pandas as pd

# 1 Create student_data1
student_data1 = pd.DataFrame({
    'Id': ['S2', 'S3', 'S4', 'S5', 'S5'],
    'Name': ['Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin', 'Kwame Morin'],
    'Marks': [210, 190, 222, 199, 199]
})

# 2 Create student_data2
student_data2 = pd.DataFrame({
    'Id': ['S4', 'S5', 'S6', 'S7', 'S8'],
    'Name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha
Preston'],
    'Marks': [201, 200, 198, 219, 201]
})

# 3 Join the two DataFrames on 'Id' (matching records only)
merged_df = pd.merge(student_data1, student_data2, on='Id', how='inner', suffixes=('_1',
'_2'))

print("Merged DataFrame (only matching Ids):")
print(merged_df)
```

SET C

Dataframe



school	Class	name	date_of_birth	age	height	weight	address
S1	S001	Asha	15/05/2002	12	173	35	Street1
S2	S002	Fransis	17/05/2002	12	192	32	Street2
S3	S003	Charlie	16/02/1999	13	186	33	Street3
S4	S004	David	25/09/1998	14	167	30	Street4
S5	S005	Shyam	11/05/2002	12	151	31	Street5
S6	S006	Eashel	15/09/1997	16	159	32	Street6

Create above dataframe and split the dataframe into groups based on school code.
1) Check the type of GroupBy object.
2) Calculate mean, min, and max value of age for each school.

Answer

```
import pandas as pd

# 1□ Create the DataFrame
data = {
    'school': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6'],
    'Class': ['S001', 'S002', 'S003', 'S004', 'S005', 'S006'],
    'name': ['Asha', 'Fransis', 'Charlie', 'David', 'Shyam', 'Eashel'],
    'date_of_birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002',
'15/09/1997'],
    'age': [12, 12, 13, 14, 12, 16],
    'height': [173, 192, 186, 167, 151, 159],
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['Street1', 'Street2', 'Street3', 'Street4', 'Street5', 'Street6']
}

df = pd.DataFrame(data)

print("Original DataFrame:")
print(df)

# ---------------------------------------------------------
# 2□ Split the DataFrame into groups based on 'school'
grouped = df.groupby('school')

# 3□ Check the type of GroupBy object
print("\nType of grouped object:", type(grouped))
```

```
# 4⃣ Calculate mean, min, and max of 'age' for each school
age_stats = grouped['age'].agg(['mean', 'min', 'max'])
print("\nAge statistics for each school:")
print(age_stats)
```

Assignment 2: Functions, Iterators and Generators

SET A

1. Write a function area_of_circle(radius) that calculates and returns the area of a circle (Use 3.14 as the value of π).

Answer

```
def area_of_circle(radius):
    pi = 3.14
    area = pi * radius ** 2
    return area

# Example usage
print("Area of circle with radius 5:", area_of_circle(5))
```

2. Create a lambda function to calculate the sum, difference, product and cube of numbers.

Answer

```
# Sum of two numbers
sum_lambda = lambda x, y: x + y

# Difference of two numbers
diff_lambda = lambda x, y: x - y

# Product of two numbers
prod_lambda = lambda x, y: x * y

# Cube of a number
cube_lambda = lambda x: x ** 3

# Example usage
print("Sum:", sum_lambda(5, 3))
print("Difference:", diff_lambda(5, 3))
print("Product:", prod_lambda(5, 3))
print("Cube of 4:", cube_lambda(4))
```

3. Write a recursive function to calculate sum of digits of number till single digit. Example 457=16=7

Answer

```
def sum_of_digits(n):
    if n < 10:
        return n
    else:
        total = sum(int(d) for d in str(n))
        return sum_of_digits(total)
```

```
# Example usage
num = 457
print(f"Recursive sum of digits for {num}:", sum_of_digits(num))
```

4. Create a function productInfo() that takes a product name and price and display the names of products whose price is less than Rs.100. If price is not given, assume the price is 100.

Answer

```
def productInfo(name, price=100):
    if price < 100:
        print(f"Product: {name}, Price: {price} (less than Rs.100)")

# Example usage
productInfo("Notebook", 50)
productInfo("Pen")  # price defaults to 100
productInfo("Eraser", 30)
```

5. Write a function that uses duck typing to process different data types.

Answer

```
def process_data(data):
    try:
        # Try to iterate (for lists, strings, tuples)
        for item in data:
            print(item, end=' ')
        print()
    except TypeError:
        # If not iterable, just print the value
        print(data)

# Example usage
process_data([1, 2, 3])
process_data("Hello")
process_data(42)
```

6. Write a generator that yields even numbers up to 10

Answer

```
def even_numbers():
    for i in range(2, 11, 2):
        yield i

# Example usage
print("Even numbers up to 10:")
for num in even_numbers():
    print(num, end=' ')
```

SET B

1. Create a function check_pass_fail(marks) that returns “Pass” if marks ≥ 40, else “Fail”.

Answer

```
def check_pass_fail(marks):
    return "Pass" if marks >= 40 else "Fail"

# Example usage
print("Marks 35:", check_pass_fail(35))
print("Marks 75:", check_pass_fail(75))
```

2. Define a function min_max(numbers) that returns the minimum and maximum values from a list.

Answer

```
def min_max(numbers):
    return min(numbers), max(numbers)

# Example usage
nums = [12, 5, 78, 34, 90, 2]
minimum, maximum = min_max(nums)
print("Min:", minimum, "Max:", maximum)
```

3. Create a class that acts as an iterator returning the Fibonacci series up to 100.

Answer

```
class Fibonacci:
    def __init__(self, limit=100):
        self.limit = limit
        self.a = 0
        self.b = 1

    def __iter__(self):
        return self

    def __next__(self):
        fib = self.a
        if fib > self.limit:
            raise StopIteration
        self.a, self.b = self.b, self.a + self.b
        return fib

# Example usage
print("Fibonacci series up to 100:")
for num in Fibonacci():
    print(num, end=' ')
```

4. Use a lambda function with map() to square all elements in the list [1, 2, 3, 4, 5]

Answer

```
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x ** 2, numbers))
print("\nSquared elements:", squared)
```

5. Use list comprehension to create a list of prime numbers between 1 and 50.

Answer

```
primes = [n for n in range(2, 51) if all(n % i != 0 for i in range(2, int(n**0.5)+1))]
print("Prime numbers between 1 and 50:", primes)
```

6. Write a function that accepts variable keyword arguments using **kwargs.

Answer

```
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

# Example usage
print("\nUser Info:")
print_info(name="Alice", age=25, city="Delhi")
```

SET C

1. Create a function employee_details(name, **info) that accepts an employee name and any number of keyword arguments (e.g., department, salary) and prints them.

Answer

```
def employee_details(name, **info):
    print(f"Employee Name: {name}")
    for key, value in info.items():
        print(f"{key}: {value}")

# Example usage
employee_details("Alice", department="HR", salary=50000, city="Delhi")
```

2. Write a function analyze_string(s) that returns the number of vowels, consonants, and digits in the string.

Answer

```
def analyze_string(s):
    vowels = 'aeiouAEIOU'
    num_vowels = sum(1 for c in s if c in vowels)
    num_digits = sum(1 for c in s if c.isdigit())
    num_consonants = sum(1 for c in s if c.isalpha() and c not in vowels)
    return num_vowels, num_consonants, num_digits

# Example usage
v, c, d = analyze_string("Hello World 123")
print(f"Vowels: {v}, Consonants: {c}, Digits: {d}")
```

3. Combine *args and **kwargs in a single function and call it with mixed arguments.

Answer

```
def mixed_args(*args, **kwargs):
    print("Positional arguments:", args)
    print("Keyword arguments:", kwargs)

# Example usage
mixed_args(1, 2, 3, name="Alice", age=25)
```

4. Implement a custom iterator class for generating permutations of a string.

Answer

```
from itertools import permutations

class StringPermutations:
    def __init__(self, s):
        self.perms = permutations(s)

    def __iter__(self):
        return self

    def __next__(self):
        return ''.join(next(self.perms))

# Example usage
print("Permutations of 'ABC':")
perm_iter = StringPermutations("ABC")
for _ in range(6):
    print(next(perm_iter))
```

5. Create a generator pipeline to process and filter a stream of sensor data.

Answer

```
def sensor_data():
    """Simulate sensor readings"""
    for i in range(20):
        yield i  # yield readings 0 to 19

def filter_even(data):
    """Filter even numbers"""
    for x in data:
        if x % 2 == 0:
            yield x

def multiply_by_10(data):
    """Multiply each reading by 10"""
    for x in data:
        yield x * 10

# Pipeline usage
pipeline = multiply_by_10(filter_even(sensor_data()))
print("Processed sensor data:")
for reading in pipeline:
    print(reading, end=' ')
```

6. Analyze the memory efficiency of generators versus lists using sys.getsizeof()

Answer

```
import sys

# List of numbers
numbers_list = [i for i in range(10000)]
numbers_gen = (i for i in range(10000))

print("Memory used by list:", sys.getsizeof(numbers_list), "bytes")
print("Memory used by generator:", sys.getsizeof(numbers_gen), "bytes")
```

Assignment 3: Regular Expression

SET A

1) Write a python program to extract number from a given string.

Answer

```
import re

def extract_numbers(s):
    # Find all sequences of digits
    numbers = re.findall(r'\d+', s)
    return numbers

# Example usage
string1 = "My order number is 457 and total is 3000"
print("Numbers in string:", extract_numbers(string1))
```

2) Write a python program to remove all whitespaces from a given string.

Answer

```
def remove_whitespace(s):
    return s.replace(" ", "")

# Example usage
string2 = "  Python   Programming  "
print("String without spaces:", remove_whitespace(string2))
```

3) Write a python program to check if a particular substring is present at the beginning of the given string.

Answer

```
def starts_with_substring(s, sub):
    return s.startswith(sub)

# Example usage
string3 = "Hello World"
substring = "Hello"
print(f"Does '{string3}' start with '{substring}'?", starts_with_substring(string3, substring))
```

4) Write a python program to validate strings like bit, but, bat, hit, hut, hat.

Answer

```
import re
```

```
def validate_string(s):
    # Pattern matches exactly the allowed words
    pattern = r'^(bit|but|bat|hit|hut|hat)$'
    if re.match(pattern, s):
        return True
    else:
        return False

# Example usage
test_words = ['bit', 'bat', 'batman', 'hut', 'hat', 'hello']
for word in test_words:
    print(word, "->", validate_string(word))
```

5) Write a Python program to replace all occurrences of a space, comma, or dot with a colon.
[Example: Input: 'Python Exercises, Data Structure exercises.'
Output: Python:Exercises::Data Structure:exercises:]

Answer

```
import re

def replace_with_colon(s):
    # Replace space, comma, or dot with colon
    return re.sub(r'[ ,.]', ':', s)

# Example usage
string5 = "Python Exercises, Data Structure exercises."
print(replace_with_colon(string5))
```

SET B

1) Write a python program to check weather given url is valid or not?
(for Ex: https://www.yahoo.com)

Answer

```
import re

def is_valid_url(url):
    # Simple regex pattern for http(s) URLs
    pattern = r'^(https?://)?(www\.)?([a-zA-Z0-9-]+)\.([a-zA-Z]{2,})(/[^\s]*)?$'
    return bool(re.match(pattern, url))

# Example usage
urls = ["https://www.yahoo.com", "http://example.com/path", "invalid_url"]
for url in urls:
    print(f"{url} ->", is_valid_url(url))
```

2) Write a python program to match word and single letter separated by a comma and single

space as in first and last name.

Answer

```
import re

def match_word_letter(s):
    pattern = r'^[A-Za-z]+, [A-Za-z]$'
    return bool(re.match(pattern, s))

# Example usage
examples = ["Smith, J", "John, D", "Doe,AB"]
for ex in examples:
    print(f"{ex} ->", match_word_letter(ex))
```

3) Write a python program to accept a string, Count total number of vowels ,consonents and special symbols from it.

Answer

```
def count_vowels_consonants_symbols(s):
    vowels = 'aeiouAEIOU'
    num_vowels = sum(1 for c in s if c in vowels)
    num_consonants = sum(1 for c in s if c.isalpha() and c not in vowels)
    num_symbols = sum(1 for c in s if not c.isalnum() and not c.isspace())
    return num_vowels, num_consonants, num_symbols

# Example usage
string3 = "Hello World! 123"
v, c, s = count_vowels_consonants_symbols(string3)
print(f"Vowels: {v}, Consonants: {c}, Symbols: {s}")
```

4) Write a python program to accept a string, Find all words that do not contain vowels.

Answer

```
import re

def words_without_vowels(s):
    words = re.findall(r'\b\w+\b', s)
    return [word for word in words if not re.search(r'[aeiouAEIOU]', word)]

# Example usage
string4 = "This is my gym plan by Dr. Smith"
print("Words without vowels:", words_without_vowels(string4))
```

5) Write a python program to accept a string, Extract all words that start with a capital letter and remove all special characters from a string.

Answer

```
import re
```

```
def extract_capital_words(s):
    # Remove special characters first
    clean_s = re.sub(r'[^A-Za-z0-9\s]', '', s)
    # Find words starting with capital letter
    return re.findall(r'\b[A-Z][a-zA-Z0-9]*\b', clean_s)

# Example usage
string5 = "Hello, World! Python3 is fun. Dr. Smith"
print("Capitalized words:", extract_capital_words(string5))
```

SET C

1) Write a python program to check the validity of a password given by the user. The password should satisfy the following criteria:
i) Contain at least 1 letter between a and z
ii) Contain at least 1 number between 0 and 9
iii) Contain at least 1 letter between A and Z
iv) Contain at least 1 character from $, #, @
v) Minimum length of password: 6
vi) Maximum length of password: 12

Answer

```
import re

def is_valid_password(password):
    if len(password) < 6 or len(password) > 12:
        return False
    # Check for at least one lowercase letter
    if not re.search(r'[a-z]', password):
        return False
    # Check for at least one uppercase letter
    if not re.search(r'[A-Z]', password):
        return False
    # Check for at least one digit
    if not re.search(r'\d', password):
        return False
    # Check for at least one special character ($, #, @)
    if not re.search(r'[$#@]', password):
        return False
    return True

# Example usage
passwords = ["Abc@123", "abc123", "ABC#123", "Abc123", "A1$bcdefgHI"]
for pw in passwords:
    print(f"{pw} ->", "Valid" if is_valid_password(pw) else "Invalid")
```

2) Write a Python program to validate mobile number based on following criteria
i) The first digit should contain numbers between 6 to 9.
ii) The rest 9 digit can contain any number between 0 to 9.
iii) The mobile number can have 11 digits also by including 0 at the starting.
iv)The mobile number can be of 12 digits also by including 91 at the starting

Answer

```
import re

def is_valid_mobile(number):
    pattern = r'^(?:0|91)?[6-9]\d{9}$'
    return bool(re.match(pattern, number))

# Example usage
numbers = ["9876543210", "09876543210", "919876543210", "1234567890", "987654321"]
for num in numbers:
    print(f"{num} ->", "Valid" if is_valid_mobile(num) else "Invalid")
```

Assignment 4: Modules and Packages

SET A

1) Write a Python program to create a user defined module that will ask your college name and will display the name of the college.

Answer

```
# User-defined module and main program in one code
def college_name():
 name = input("Enter your college name: ")
 print("College Name:", name)
college_name()
```

2) Write a Python program to shuffle the elements of a given list. [Hint: Use random.shuffle()]

Answer

```
import random
lst = [1, 2, 3, 4, 5]
random.shuffle(lst)
print("Shuffled list:", lst)
```

3) Write a python program to generate a random radius and computes the area (use math module).

Answer

```
import random
import math
radius = random.randintset A (1, 10)
area = math.pi * radius * radius
print("Radius:", radius)
print("Area of Circle:", area)
```

4) Write a Python program to get current time and convert to local time. (use time module)

Answer

```
import time
current_time = time.time()
local_time = time.ctime(current_time)
print("Current Time (seconds since epoch):", current_time)
print("Local Time:", local_time)
```

5) Create a package vehicle with a sub package types, and modules cars and bikes. Each module should have a function available_models() returning a list of models. Write a program to display them.

Answer

```
# Simulating package: vehicle
# Sub-package: types
# Modules: cars and bikes
# cars module
def car_available_models():
 return ["Sedan", "SUV", "Hatchback"]
# bikes module
def bike_available_models():
 return ["Sports Bike", "Cruiser", "Scooter"]
# Main program
print("Car Models:", car_available_models())
print("Bike Models:", bike_available_models())
```

6) Build a package education with a sub package courses, containing modules science.py and arts.py. Each module should have a function get_subjects(). Write a program to display all subjects.

Answer

```
# Simulating package: education
# Sub-package: courses
# Modules: science and arts
# science module
def science_subjects():
 return ["Physics", "Chemistry", "Biology"]
# arts module
def arts_subjects():
 return ["History", "Literature", "Philosophy"]
# Main program
print("Science Subjects:", science_subjects())
print("Arts Subjects:", arts_subjects())
```

7) Write a Python program to shuffle a deck of card. Display the cards obtained by a user in 5 attempts.

Answer

```
import random
# Create a deck of cards
suits = ['Hearts', 'Diamonds', 'Clubs', 'Spades']
ranks = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'Jack', 'Queen', 'King', 'Ace']
deck = [rank + " of " + suit for suit in suits for rank in ranks]
# Shuffle the deck
random.shuffle(deck)
# Give user 5 cards
print("Your 5 cards are:")
for i in range(5):
 print(deck[i])
```

SET B

1) Write a python program to create a module containing factorial function, power dictionary and list of vowels. Save this as variables.py. Import variables module to print the factorial of a number, print the power of 6, and second alphabet from the vowels list.

Answer

```
# Factorial function
def factorial(n):
 fact = 1
 for i in range(1, n + 1):
 fact = fact * i
 return fact
# Power dictionary
power = {
 2: 4,
 3: 9,
 4: 16,
 5: 25,
 6: 36
}
# List of vowels
vowels = ['a', 'e', 'i', 'o', 'u']
# Using the function and variables
print("Factorial of 5:", factorial(5))
print("Power of 6:", power[6])
print("Second vowel:", vowels[1])
```

2) Write a python program to create a list, string and tuple in python. Apply random module to display unique multiple random numbers or values from above created sequence type.

Answer

**Python program to create a list, string, and tuple and display unique random values using random
module**

```
import random
# Create list, string, and tuple
my_list = [10, 20, 30, 40, 50]
my_string = "PYTHON"
my_tuple = (1, 2, 3, 4, 5)
# Display unique random values
print("Random values from list:", random.sample(my_list, 3))
print("Random values from string:", random.sample(my_string, 3))
print("Random values from tuple:", random.sample(my_tuple, 3))
```

3) Create a package banking with modules account.py and loan.py. account.py should contain a function open_account(name), and loan.py should contain apply_loan(amount). Write a program to demonstrate both functions.

Answer

```
# Simulating banking/account.py
class account:
 @staticmethod
 def open_account(name):
 print(f"Account opened successfully for {name}")
# Simulating banking/loan.py
class loan:
 @staticmethod
 def apply_loan(amount):
 print(f"Loan of amount {amount} applied successfully")
# Using both "modules"
account.open_account("Alice")
loan.apply_loan(5000)
```

4) Write a Python program to get current time and apply following functions:
a) ctime() b) gmtime() c) strftime() d) strptime()

Answer

```
import time
from datetime import datetime
# a) ctime() - Current time as a readable string
current_time = time.ctime()
print("ctime():", current_time)
# b) gmtime() - Current time in UTC as a struct_time
utc_time = time.gmtime()
print("gmtime():", utc_time)
# c) strftime() - Format time into a readable string
formatted_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
print("strftime():", formatted_time)
# d) strptime() - Convert a string into datetime object
time_string = "2026-02-09 14:30:00"
parsed_time = datetime.strptime(time_string, "%Y-%m-%d %H:%M:%S")
print("strptime():", parsed_time)
```

SET C:

1) Create a package named library with a sub package books, containing modules fiction and nonfiction. Each module should have a function list_books() returning a list of book names. Write a main program to display the lists from both.

Answer

```
# Simulating library.books.fiction module
class fiction:
 @staticmethod
 def list_books():
 return ["The Alchemist", "Harry Potter", "Pride and Prejudice"]
# Simulating library.books.nonfiction module
class nonfiction:
 @staticmethod
 def list_books():
```

```
 return ["Sapiens", "Educated", "The Diary of a Young Girl"]
# Using both modules
print("Fiction Books:", fiction.list_books())
print("Non-Fiction Books:", nonfiction.list_books())
```

2.) Create a package finance_tools with modules interest.py and currency.py. Add functions to calculate interest and convert currency. Import both in a main script and perform chained calculations.

Answer

```
# Simulating finance_tools.interest module
class interest:
 @staticmethod
 def calculate_simple_interest(principal, rate, time):
 """
 Calculate simple interest
 SI = (P * R * T) / 100
 """
 return (principal * rate * time) / 100
# Simulating finance_tools.currency module
class currency:
 @staticmethod
 def convert_usd_to_inr(usd_amount, rate=82.5):
 """
 Convert USD to INR using given exchange rate
 """
 return usd_amount * rate
# Using both "modules" in chained calculations
principal_amount = 1000
rate = 5 # percent
time = 2 # years
# Step 1: Calculate interest
interest_amount = interest.calculate_simple_interest(principal_amount, rate, time)
print("Simple Interest:", interest_amount)
# Step 2: Convert interest to INR
interest_in_inr = currency.convert_usd_to_inr(interest_amount)
print("Interest in INR:", interest_in_inr)
# Step 3: Total amount in INR
total_inr = currency.convert_usd_to_inr(principal_amount) + interest_in_inr
print("Total Amount in INR:", total_inr)
```

3.) Case Study: Food Delivery App
Scenario: A food delivery service needs to handle restaurants, orders, and delivery agents.
Task: Create a Package: food_delivery
Subpackage: system
Modules: restaurants.py, orders.py, agents.py
Each module includes functions like:
list_restaurants()
place_order()

assign_agent()
Main script should place an order and assign it to a delivery agent

Answer

```
# ================== SET A ==================
print("\n--- SET A ---")
# 1) Extract numbers from string
import re
string = "Hello123World45"
numbers = re.findall(r'\d+', string)
print("Extracted numbers:", numbers)
# 2) Remove all whitespaces
s = "Python is fun"
print("Without whitespaces:", s.replace(" ", ""))
# 3) Check substring at beginning
substring = "Python"
if s.startswith(substring):
 print("Substring found at beginning")
else:
 print("Substring not found at beginning")
# 4) Validate strings like bit, but, bat, hit, hut, hat
pattern = r'^[bh][iua]t$'
test_str = "bat"
if re.match(pattern, test_str):
 print(f"{test_str} is valid")
else:
 print(f"{test_str} is invalid")
# 5) Replace space, comma, or dot with colon
text = "Python Exercises, Data Structure exercises."
replaced = re.sub(r'[ ,.]', ':', text)
print("Replaced text:", replaced)
# ================== SET B ==================
print("\n--- SET B ---")
# 1) Create module with factorial, power dictionary, vowels
class variables:
 @staticmethod
 def factorial(n):
 fact = 1
 for i in range(1, n+1):
 fact *= i
 return fact
 power = {1:1,2:4,3:9,4:16,5:25,6:36}
 vowels = ['a','e','i','o','u']
print("Factorial of 5:", variables.factorial(5))
print("Power of 6:", variables.power[6])
print("Second vowel:", variables.vowels[1])
# 2) Random values from list, string, tuple
import random
my_list = [10,20,30,40,50]
my_string = "PYTHON"
my_tuple = (1,2,3,4,5)
```

```
print("Random list:", random.sample(my_list,3))
print("Random string:", random.sample(my_string,3))
print("Random tuple:", random.sample(my_tuple,3))
# 3) Finance Tools: interest and currency
class interest:
 @staticmethod
 def calculate_simple_interest(P,R,T):
 return (P*R*T)/100
class currency:
 @staticmethod
 def convert_usd_to_inr(usd, rate=82.5):
 return usd*rate
principal = 1000
rate = 5
time_years = 2
SI = interest.calculate_simple_interest(principal, rate, time_years)
SI_inr = currency.convert_usd_to_inr(SI)
total_inr = currency.convert_usd_to_inr(principal)+SI_inr
print("Simple Interest:", SI)
print("Interest in INR:", SI_inr)
print("Total Amount in INR:", total_inr)
# 4) Current time functions
import time
from datetime import datetime
print("ctime():", time.ctime())
print("gmtime():", time.gmtime())
print("strftime():", time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
time_string = "2026-02-09 14:30:00"
parsed_time = datetime.strptime(time_string, "%Y-%m-%d %H:%M:%S")
print("strptime():", parsed_time)
# ================= SET C ==================
print("\n--- SET C ---")
# Library Package simulation
class fiction:
 @staticmethod
 def list_books():
 return ["The Alchemist", "Harry Potter", "Pride and Prejudice"]
class nonfiction:
 @staticmethod
 def list_books():
 return ["Sapiens", "Educated", "The Diary of a Young Girl"]
print("Fiction Books:", fiction.list_books())
print("Non-Fiction Books:", nonfiction.list_books())
# ================= Food Delivery App ==================
print("\n--- Food Delivery App ---")
class restaurants:
 @staticmethod
 def list_restaurants():
 return ["Pizza Palace", "Sushi World", "Burger Hub"]
class orders:
 @staticmethod
 def place_order(customer_name, restaurant_name, items):
```

```
 order_id = 101
 print(f"Order placed! ID:{order_id}, Customer:{customer_name},
Restaurant:{restaurant_name}, Items:{items}")
 return order_id
class agents:
 @staticmethod
 def assign_agent(order_id):
 agent_name = "John Doe"
 print(f"Order {order_id} assigned to agent: {agent_name}")
 return agent_name
# Place an order and assign agent
print("Available Restaurants:", restaurants.list_restaurants())
order_id = orders.place_order("Alice", "Pizza Palace", ["Margherita Pizza","Coke"])
assigned_agent = agents.assign_agent(order_id)
```

4.) Case Study: Travel Agency System
Scenario: A travel agency offers domestic and international travel packages.
Task: Create a Package travel_agency.
Subpackage: packages with modules domestic.py and international.py.
Functions:
get_packages()
book_package (destination)
Create a main script to display all packages and book one.

Answer

```
# ================== SET A ==================
print("\n--- SET A ---")
import re
import random
import time
from datetime import datetime
# 1) Extract numbers from string
string = "Hello123World45"
numbers = re.findall(r'\d+', string)
print("Extracted numbers:", numbers)
# 2) Remove all whitespaces
s = "Python is fun"
print("Without whitespaces:", s.replace(" ", ""))
# 3) Check substring at beginning
substring = "Python"
if s.startswith(substring):
 print("Substring found at beginning")
else:
 print("Substring not found at beginning")
# 4) Validate strings like bit, but, bat, hit, hut, hat
pattern = r'^[bh][iua]t$'
test_str = "bat"
if re.match(pattern, test_str):
 print(f"{test_str} is valid")
```

```
else:
 print(f"{test_str} is invalid")
# 5) Replace space, comma, or dot with colon
text = "Python Exercises, Data Structure exercises."
replaced = re.sub(r'[ ,.]', ':', text)
print("Replaced text:", replaced)
# ================== SET B ===================
print("\n--- SET B ---")
# Module simulation: variables
class variables:
 @staticmethod
 def factorial(n):
 fact = 1
 for i in range(1, n+1):
 fact *= i
 return fact
 power = {1:1,2:4,3:9,4:16,5:25,6:36}
 vowels = ['a','e','i','o','u']
print("Factorial of 5:", variables.factorial(5))
print("Power of 6:", variables.power[6])
print("Second vowel:", variables.vowels[1])
# Random values from list, string, tuple
my_list = [10,20,30,40,50]
my_string = "PYTHON"
my_tuple = (1,2,3,4,5)
print("Random list:", random.sample(my_list,3))
print("Random string:", random.sample(my_string,3))
print("Random tuple:", random.sample(my_tuple,3))
# Finance Tools: interest and currency
class interest:
 @staticmethod
 def calculate_simple_interest(P,R,T):
 return (P*R*T)/100
class currency:
 @staticmethod
 def convert_usd_to_inr(usd, rate=82.5):
 return usd*rate
principal = 1000
rate_val = 5
time_years = 2
SI = interest.calculate_simple_interest(principal, rate_val, time_years)
SI_inr = currency.convert_usd_to_inr(SI)
total_inr = currency.convert_usd_to_inr(principal)+SI_inr
print("Simple Interest:", SI)
print("Interest in INR:", SI_inr)
print("Total Amount in INR:", total_inr)
# Current time functions
print("ctime():", time.ctime())
print("gmtime():", time.gmtime())
print("strftime():", time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
time_string = "2026-02-09 14:30:00"
parsed_time = datetime.strptime(time_string, "%Y-%m-%d %H:%M:%S")
```

```
print("strptime():", parsed_time)
# ================== SET C ==================
print("\n--- SET C ---")
# Library package simulation
class fiction:
 @staticmethod
 def list_books():
 return ["The Alchemist", "Harry Potter", "Pride and Prejudice"]
class nonfiction:
 @staticmethod
 def list_books():
 return ["Sapiens", "Educated", "The Diary of a Young Girl"]
print("Fiction Books:", fiction.list_books())
print("Non-Fiction Books:", nonfiction.list_books())
# ================== Food Delivery App ==================
print("\n--- Food Delivery App ---")
class restaurants:
 @staticmethod
 def list_restaurants():
 return ["Pizza Palace", "Sushi World", "Burger Hub"]
class orders:
 @staticmethod
 def place_order(customer_name, restaurant_name, items):
 order_id = 101
 print(f"Order placed! ID:{order_id}, Customer:{customer_name}, 
Restaurant:{restaurant_name}, Items:{items}")
 return order_id
class agents:
 @staticmethod
 def assign_agent(order_id):
 agent_name = "John Doe"
 print(f"Order {order_id} assigned to agent: {agent_name}")
 return agent_name
print("Available Restaurants:", restaurants.list_restaurants())
order_id = orders.place_order("Alice", "Pizza Palace", ["Margherita Pizza","Coke"])
assigned_agent = agents.assign_agent(order_id)
# ================== Travel Agency System ==================
print("\n--- Travel Agency System ---")
class domestic:
 @staticmethod
 def get_packages():
 return ["Goa Tour", "Kerala Backwaters", "Rajasthan Forts"]
 @staticmethod
 def book_package(destination):
 print(f"Domestic package booked: {destination}")
 return destination
class international:
 @staticmethod
 def get_packages():
 return ["Paris Sightseeing", "Thailand Beaches", "Dubai Desert Safari"]
 @staticmethod
 def book_package(destination):
```

```
 print(f"International package booked: {destination}")
 return destination
print("Available Domestic Packages:", domestic.get_packages())
print("Available International Packages:", international.get_packages())
booked_domestic = domestic.book_package("Goa Tour")
booked_international = international.book_package("Paris Sightseeing")
```

Assignment 6: Working with Files

SET A:

1) Write a python program to open a file in read mode and display its content line by line.
2) Write a python program to create a file which stores the first name, middle name and last name of your 5 friends.
3) Write a python program to append a string to an existing file.
4) Open a file data.txt in read mode and print the file name and mode.
5) Open a file, read 6 characters, and print the pointer location before and after using .tell Use .seek(0) to reset pointer to start and re-read the first few characters.
6) Write a Python program to list all files and folders in the current directory using os.listdir().

Answer

```
# ==============================
# SET A - File Handling Programs
# ==============================

import os

# ---------------------------------------------
# 1) Open a file in read mode and display content line by line
# ---------------------------------------------

print("\n1) Reading file line by line:")
try:
    with open("sample.txt", "r") as file:
        for line in file:
            print(line.strip())
except FileNotFoundError:
    print("sample.txt not found. Please create the file first.")


# ---------------------------------------------
# 2) Create a file and store names of 5 friends
# ---------------------------------------------

print("\n2) Creating friends.txt and writing names:")
friends = [
    "John Michael Smith",
    "Emily Rose Johnson",
    "David Lee Brown",
    "Sophia Grace Wilson",
    "Daniel James Taylor"
]

with open("friends.txt", "w") as file:
    for friend in friends:
        file.write(friend + "\n")

print("friends.txt created successfully.")


# -----------------------------------------------
# 3) Append a string to an existing file
# -----------------------------------------------

print("\n3) Appending data to sample.txt:")
with open("sample.txt", "a") as file:
    file.write("\nThis line is appended to the file.")

print("Data appended successfully.")


# -----------------------------------------------
# 4) Open data.txt in read mode and print file name and mode
# -----------------------------------------------

print("\n4) Printing file name and mode:")
try:
    with open("data.txt", "r") as file:
        print("File Name:", file.name)
        print("File Mode:", file.mode)
except FileNotFoundError:
    print("data.txt not found. Please create the file first.")


# -----------------------------------------------
# 5) Read 6 characters and use tell() and seek()
# -----------------------------------------------

print("\n5) Demonstrating tell() and seek():")
try:
    with open("sample.txt", "r") as file:
        print("Pointer before reading:", file.tell())

        data = file.read(6)
        print("Read data:", data)

        print("Pointer after reading:", file.tell())

        file.seek(0)
        print("Pointer after seek(0):", file.tell())

        data_again = file.read(6)
        print("Re-read first 6 characters:", data_again)
except FileNotFoundError:
    print("sample.txt not found.")
```

```
# ----------------------------------------------
# 6) List all files and folders in current directory
# ----------------------------------------------

print("\n6) Listing all files and folders:")
items = os.listdir()

for item in items:
    print(item)
```

SET B

1. Write a python program to compute the number of characters, words and lines in a file
2. Write a python program to copy contents of one file into another file.
3. Write a function to list all files in a directory and count the number of .py files.
4. Print all entries in a directory and identify each as file or folder.
5. Create a function that takes a position as input, seeks to it, and prints the content from that byte onward.

Answer

```
# ===================================
# SET B - File Handling Programs
# ===================================

import os

# ----------------------------------------------
# 1) Count number of characters, words and lines
# ----------------------------------------------

print("\n1) Counting characters, words, and lines:")

try:
    with open("sample.txt", "r") as file:
        content = file.read()

        characters = len(content)
        words = len(content.split())
        lines = len(content.splitlines())

        print("Number of Characters:", characters)
        print("Number of Words:", words)
        print("Number of Lines:", lines)

except FileNotFoundError:
    print("sample.txt not found.")


# ----------------------------------------------
# 2) Copy contents of one file into another file
# ------------------------------------------------

print("\n2) Copying contents from sample.txt to copy.txt:")

try:
    with open("sample.txt", "r") as source:
        data = source.read()

    with open("copy.txt", "w") as destination:
        destination.write(data)

    print("Content copied successfully.")

except FileNotFoundError:
    print("sample.txt not found.")


# ------------------------------------------------
# 3) Function to list all files and count .py files
# ------------------------------------------------

def count_py_files(directory):
    files = os.listdir(directory)
    py_count = 0

    for file in files:
        if file.endswith(".py"):
            py_count += 1

    print("\n3) Python files in directory:")
    print("Total .py files:", py_count)


count_py_files(".")


# ------------------------------------------------
# 4) Print all entries and identify file or folder
# ------------------------------------------------

print("\n4) Identifying files and folders:")

entries = os.listdir()

for entry in entries:
    if os.path.isfile(entry):
        print(entry, "-> File")
    elif os.path.isdir(entry):
        print(entry, "-> Folder")


# ------------------------------------------------
# 5) Function to seek to a position and print content
# -----------------------------------------------

def read_from_position(filename, position):
    try:
        with open(filename, "r") as file:
            file.seek(position)
            data = file.read()
            print("\n5) Content from position", position, ":")
            print(data)
    except FileNotFoundError:
        print(filename, "not found.")


# Example call
read_from_position("sample.txt", 5)
```

SET C

1. Use with open() to inspect file attributes inside and outside the block. Explain the change in .closed.
2. Recursively list all .txt files in all subdirectories using os.walk().
3. Create a report that lists files along with their sizes and modification dates.
4. Recursively scan all directories and print each item's type (file, folder, symlink).
5. Rename all files in a folder by appending today's date to their names (e.g., log.txt → log_2025-06-26.txt).
6. Create a complete folder backup (all files + subfolders) using shutil.copytree().

Answer

```
# ===================================
# SET C - File Handling & OS Operations
# ===================================

import os
import shutil
from datetime import date
import time

# -----------------------------------------------
# 1) Using with open() to inspect file attributes
# -----------------------------------------------

print("\n1) Inspecting file attributes with 'with open()':")

with open("sample.txt", "r") as file:
    print("Inside 'with' block:")
    print("File name:", file.name)
    print("File mode:", file.mode)
    print("Is file closed?:", file.closed)

print("Outside 'with' block:")
print("Is file closed?:", file.closed)
# Explanation: Inside the block, the file is open. Outside, it is automatically closed.

# -----------------------------------------------
# 2) Recursively list all .txt files using os.walk()
# -----------------------------------------------

print("\n2) Recursively listing all .txt files:")

for root, dirs, files in os.walk("."):
    for file in files:
        if file.endswith(".txt"):
            print(os.path.join(root, file))

# -----------------------------------------------
# 3) Report of files with sizes and modification dates
# -----------------------------------------------

print("\n3) File report (size and modification date):")

for root, dirs, files in os.walk("."):
    for file in files:
        filepath = os.path.join(root, file)
        size = os.path.getsize(filepath)
        mod_time = time.ctime(os.path.getmtime(filepath))
        print(f"{filepath} -> Size: {size} bytes, Modified: {mod_time}")

# -----------------------------------------------
# 4) Recursively scan directories and print item type
# -----------------------------------------------

print("\n4) Item types in directories:")

for root, dirs, files in os.walk("."):
    for name in dirs + files:
        path = os.path.join(root, name)
        if os.path.isfile(path):
            type_str = "File"
        elif os.path.isdir(path):
            type_str = "Folder"
        elif os.path.islink(path):
            type_str = "Symlink"
        else:
            type_str = "Other"
        print(f"{path} -> {type_str}")

# -----------------------------------------------
# 5) Rename all files in a folder by appending today's date
# -----------------------------------------------

print("\n5) Renaming all files in current folder by appending today's date:")
```

```
today_str = date.today().isoformat()  # e.g., '2026-02-18'
folder = "."  # current folder

for filename in os.listdir(folder):
    path = os.path.join(folder, filename)
    if os.path.isfile(path):
        name, ext = os.path.splitext(filename)
        new_name = f"{name}_{today_str}{ext}"
        new_path = os.path.join(folder, new_name)
        os.rename(path, new_path)
        print(f"{filename} -> {new_name}")

# -----------------------------------------------
# 6) Complete folder backup using shutil.copytree()
# -----------------------------------------------

print("\n6) Creating folder backup using shutil.copytree():")

source_folder = "."            # current folder
backup_folder = "backup_folder"

# Remove previous backup if exists
if os.path.exists(backup_folder):
    shutil.rmtree(backup_folder)

shutil.copytree(source_folder, backup_folder)
print(f"Backup created successfully in '{backup_folder}'")
```

Assignment 7: Exception Handling

Set A:

1. Write a Python program that prompts the user to input a number to handle a ZeroDivisionError exception when dividing a number by zero.
2. Write a Python program that prompts the user to input an integer and raises a ValueError exception if the input is not a valid integer.
3. Write a Python program that prompts the user to input two numbers and raises a TypeError exception if the inputs are not numerical.
4. Write a Python program that executes an operation on a list and handles an IndexError exception if the index is out of range.

Answer

```
# ====================================
# SET A - Exception Handling Programs
# ====================================

# -----------------------------------------------
# 1) Handle ZeroDivisionError
# -----------------------------------------------
print("\n1) Handling ZeroDivisionError:")

try:
    num = float(input("Enter a number to divide 100 by: "))
    result = 100 / num
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Cannot divide by zero!")
except ValueError:
    print("Error: Invalid input, please enter a number.")

# -----------------------------------------------
# 2) Raise ValueError if input is not an integer
# -----------------------------------------------
print("\n2) Raising ValueError for non-integer input:")

try:
    user_input = input("Enter an integer: ")
    number = int(user_input)  # will raise ValueError if invalid
    print("You entered:", number)
except ValueError:
    print("Error: This is not a valid integer!")

# -----------------------------------------------
# 3) Raise TypeError if inputs are not numerical
# -----------------------------------------------
print("\n3) Raising TypeError for non-numerical input:")

def check_numbers(a, b):
    if not (isinstance(a, (int, float)) and isinstance(b, (int, float))):
        raise TypeError("Both inputs must be numbers")
    return a + b  # example operation

try:
    a = input("Enter first number: ")
    b = input("Enter second number: ")

    # Convert to float if possible
    try:
        a = float(a)
        b = float(b)
    except ValueError:
        raise TypeError("Both inputs must be numbers")

    result = check_numbers(a, b)
    print("Sum of numbers:", result)

except TypeError as te:
    print("Error:", te)

# ----------------------------------------------
# 4) Handle IndexError
# ----------------------------------------------
print("\n4) Handling IndexError:")

my_list = [10, 20, 30, 40, 50]

try:
    index = int(input("Enter the index to access in list [0-4]: "))
    print("Element at index", index, "is", my_list[index])
except IndexError:
    print("Error: Index out of range!")
except ValueError:
    print("Error: Invalid input, please enter an integer.")
```

Set B:

1. Write a Python program that prompts the user to input a number and handles a KeyboardInterrupt exception if the user cancels the input.
2. Write a Python program that executes a list operation and handles an AttributeError exception if the attribute does not exist.
3. Write a python program to handle multiple exception types in a single block.
4. Write a Python program to demonstrate a user-defined exception that checks whether the entered marks are between 0 and 100. Display an appropriate message for invalid input and out-of-range values.

Answer

```
# ====================================
# SET B - Exception Handling Programs
# ================================

# -----------------------------------------------
# 1) Handle KeyboardInterrupt
# -----------------------------------------------
print("\n1) Handling KeyboardInterrupt:")

try:
    num = input("Enter a number (press Ctrl+C to cancel): ")
    print("You entered:", num)
except KeyboardInterrupt:
    print("\nInput cancelled by user!")

# -----------------------------------------------
# 2) Handle AttributeError
# -----------------------------------------------
print("\n2) Handling AttributeError:")

my_list = [1, 2, 3, 4, 5]

try:
    # Trying to use a non-existent attribute 'sort_desc'
    my_list.sort_desc()
except AttributeError:
    print("Error: Attribute does not exist!")

# -----------------------------------------------
# 3) Handle multiple exceptions in a single block
# -----------------------------------------------
print("\n3) Handling multiple exceptions:")

try:
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    result = a / b
    print("Division result:", result)
except (ValueError, ZeroDivisionError) as e:
    print("Error occurred:", e)

# -----------------------------------------------
# 4) User-defined exception to check marks range
# -----------------------------------------------
print("\n4) User-defined exception for marks range:")

class MarksError(Exception):
    pass

try:
    marks = float(input("Enter marks (0-100): "))
    if not (0 <= marks <= 100):
        raise MarksError("Marks should be between 0 and 100")
    print("Marks entered:", marks)
```

```
except ValueError:
    print("Invalid input! Please enter a numeric value.")
except MarksError as me:
    print("Error:", me)
```