PYTHON LAB

Lab 1: Python Numbers, List

Lab 2: Tuple, Strings, Set

Lab 3: Lambda & Filter in Python Examples

Lab 4: Creating Class in Python

Lab 5: Creating Object in Python

Lab 6: Creating Methods in Python

Lab 7: Process standard streams.

Lab 8: Command-line arguments, shell variables

Lab 9: Python scripts here perform real tasks.

Lab 10: Client Socket Methods

Lab 11: General Socket Methods

Lab 12: Creating Thread Using Threading Module

Lab 13: Represent compound data using Python

Lab 14: Lists, tuples, dictionaries.

Lab 15: Read and write data from / to files in Python Programs

---

1. Python Number , List

```
seller_name = "RAJA"
age = 28
salary = 20000.500
c = 5j

print(type(seller_name))
print(type(age))
print(type(salary))
print(type(c))

fruits = ["Apple", "Banana", "Orange", "Grapes"]
print("Fruits List:", fruits)
```

```
l = len(fruits)
print("Length of the List:", l)

fruits.append("Cherry")
print("Updated Fruits List:", fruits)

l = len(fruits)
print("Length of the List:", l)

fruits.sort()
print("Sorted Fruits List:", fruits)

print("Search pineapple in fruits list")
s = "Pineapple" in fruits
print(s)

print("Search apple in fruits list")
s = "Apple" in fruits
print(s)

print("Fruits List after deleting", fruits[3], ":")
del fruits[3]
print(fruits)
```

---

2. Tuple, String, Set

```
#tuple

fruits = ("Apple", "Banana", "Orange", "Grapes")
print("Fruits List:", fruits)
l = len(fruits)
print("Length of the List:", l)
l = len(fruits)
```

```
print("Length of the List:", l)

#string
name = "SRM University"
name2 = "Trichy"
print("First Name:", name)
print("Second Name:", name2)
print("The first letter of a given Name")
print(name[0])
print("The first letter of a given Name2")
print(name2[0])
print(" ")

print("CONCATENATION OF TWO STRINGS")
print(name + name2)
print("Length of name-string:", name, len(name))
print("UPPER CASE: ", name.upper())
print("Lower Case: ", name.lower())

#set

print("SET CREATION AND OPERATIONS/ FUNCTIONS")

fruits = set(["Apple", "Mango", "Banana"])
print("Fruit-SET", fruits)

print("Add 'Orange' in the set")
fruits.add("Orange")
print(fruits)

print("Remove 'Banana' from set")
fruits.remove("Banana")
```

```
print(fruits)

print("COUNT ITEMS IN THE SET")
print("No. of Items in the set:", len(fruits))

print("Search 'Papaya' in the set")
print("Papaya" in fruits)

print("Search 'Apple' in the Set")
print("Apple" in fruits)
```

---

3. Lambda & Filter

```
n=[10,15,20,25,30,35,40,45,50]
Even=list(filter(lambda x: x%2==0 , n))
print("Filtered Even Numbers are:",Even)
```

---

4. Creating Class in Python

```
# define a class
class Student:
    name = ""
    regno = ""
    year = ""
    CGPA = 0.0

s1 = Student()
s1.name = "JANANE"
s1.regno = "RA240001"
```

```
s1.year = "I-MCA"
s1.CGPA = 9.8

print(f"Name: {s1.name}, Register no.: {s1.regno}, Year: {s1.year}, CGPA: {s1.CGPA}")
```

---

5. Creating Object in Python

```
class Student:
    name = ""
    regno = ""
    year = ""
    CGPA = 0.0

students = []

students.append(Student())
students[0].name = "JANANE"
students[0].regno = "RA240001"
students[0].year = "I-MCA"
students[0].CGPA = 9.8

students.append(Student())
students[1].name = "buvaneshwari"
students[1].regno = "RA240002"
students[1].year = "I-MCA"
students[1].CGPA = 9.8

for student in students:
    print(f"Name: {student.name}, Register no.: {student.regno}, Year: {student.year}, CGPA: {student.CGPA}")
```

---

6. Method in python

```
def AreaRect(length,width):
 Area= length * width

 return Area
# function call with two values
A= AreaRect(15, 5)
print("Area of the Rectangle: ", A)
```

---

7. Process standard streams

```
def write_file(filename, content):
    """Write data to a file (Output Stream - Writing Mode)."""
    with open(filename, "w") as file:
        file.write(content)
    print(f"Data written to {filename}")



def read_file(filename):
    """Read data from a file (Input Stream - Reading Mode)."""
    try:
        with open(filename, "r") as file:
            data = file.read()
        print(f"Data read from {filename}:\n{data}")
    except FileNotFoundError:
        print(f"Error: {filename} not found.")
```

```
def append_file(filename, content):
    """Append data to an existing file (Output Stream - Append Mode)."""
    with open(filename, "a") as file:
        file.write("\n" + content)
    print(f"Data appended to {filename}")



# Streams Example: File I/O Operations
filename = r"C:\Users\r15\Documents\Python R6\Python Exercise\r15.txt"


# Writing to a file (Output Stream)
write_file(filename, "HI MCA STUDENTS, WELCOME TO SRM IST TO LEARN
PYTHON")


# Reading from the file (Input Stream)
read_file(filename)


# Appending new data (Output Stream - Append Mode)
append_file(filename, "LEARN PYTHON AND BECOME A DATA SCIENTIST")


# Reading updated file (Input Stream)
read_file(filename)’
```

---

8. COMMAND-LINE ARGUMENTS AND SHELL VARIABLES

```
import sys
import os


def command_line_args():
    print("Command-line arguments:", sys.argv)
    if len(sys.argv) > 1:
```

```
        print("First argument:", sys.argv[1])
    else:
        print("No additional arguments provided.")

def shell_variables():
    user = os.getenv("USER") or os.getenv("USERNAME")  # For Windows compatibility
    path = os.getenv("PATH")
    print("User:", user)
    print("System PATH:", path)

command_line_args()
shell_variables()
```

---

9. Python script to perform a real task

```
import os
import shutil

# Define file categories and their corresponding extensions
FILE_CATEGORIES = {
    "Images": [".jpg", ".jpeg", ".png", ".gif", ".bmp", ".svg"],
    "Documents": [".pdf", ".doc", ".docx", ".txt", ".xlsx", ".pptx"],
    "Videos": [".mp4", ".avi", ".mov", ".mkv"],
    "Music": [".mp3", ".wav", ".aac", ".flac"],
    "Archives": [".zip", ".rar", ".tar", ".gz"],
    "Programs": [".exe", ".msi", ".sh", ".bat"],
    "Others": []
}

def organize_files(directory):
    """Organizes files in the specified directory based on their extensions."""
    if not os.path.exists(directory):
        print("Error: Directory does not exist!")
        return

    for file in os.listdir(directory):
        file_path = os.path.join(directory, file)

        if os.path.isfile(file_path):
            file_extension = os.path.splitext(file)[1].lower()
            folder_name = "Others"

            for category, extensions in FILE_CATEGORIES.items():
                if file_extension in extensions:
                    folder_name = category
```

```
            break

        target_folder = os.path.join(directory, folder_name)
        os.makedirs(target_folder, exist_ok=True)

        shutil.move(file_path, os.path.join(target_folder, file))
        print(f"Moved: {file} → {folder_name}")

if __name__ == "__main__":
    folder_path = input("Enter the folder path to organize: ").strip()
    organize_files(folder_path)
    print("✅ File organization completed!")
```

10.Client Socket Method

```
#server.py
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 8080))
server_socket.listen(5)
print("Server is listening on port 8080...")

while True:
    conn, addr = server_socket.accept()
    print(f"Connected by {addr}")

    data = conn.recv(1024)
    if data:
        print("Received:", data.decode('utf-8'))
        conn.sendall(b"Message received!")
```

```
    conn.close()

#client.py
import socket

def start_client():
    """Creates a client socket, connects to the server, and sends a message."""
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server_address = ('127.0.0.1', 8080)
    try:
        client_socket.connect(server_address)
        print(f"Connected to {server_address}")

        message = "Hello, Server!"
        client_socket.sendall(message.encode('utf-8'))

        response = client_socket.recv(1024)
        print("Server response:", response.decode('utf-8'))

    except ConnectionRefusedError:
        print("Error: Server is not running or unreachable.")

    finally:
        client_socket.close()
        print("Client socket closed.")

if __name__ == "__main__":
    start_client()
```

11.General Socket Methods

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print("Socket created successfully")

sock_type = sock.type
print("Socket Type:", sock_type)

sock_family = sock.family
print("Socket Family:", sock_family)

timeout = sock.gettimeout()
print("Default Timeout:", timeout)

sock.settimeout(10)
print("New Timeout:", sock.gettimeout())

sock.bind(('localhost', 0))
print("Socket bound to:", sock.getsockname())

sock.close()
print("Socket closed")
```

12.Creating Thread Using Threading Module

```
import threading
import time

def display():
    for i in range(5):
```

```
        print(f"Thread is running: {i}")
        time.sleep(1)

# Create a thread
my_thread = threading.Thread(target=display)

# Start the thread
my_thread.start()

# Main thread continues
print("Main thread is continuing...")

# Wait for the thread to finish
my_thread.join()
print("Thread has finished.")
```

13.Compound Data using Python

```
students = ["Alice", "Bob", "Charlie"]

student_details = {
    "name": "Alice",
    "age": 21,
    "course": "MCA"
}

birth_date = (15, "April", 2003)

subjects = {"Python", "Math", "Data Science"}

print("Student List:", students)
```

```
print("Details of a student:", student_details)
print("Birth Date:", birth_date)
print("Subjects:", subjects)
```

---

14. Dictionaries

```
d = {0: "Air", 1: "Brilliant", 2: "Character", 3: "Doctor"}

print("Elements of dictionary D")
print(d)

print("Length of dictionary")
print(len(d))

print("Min of dictionary:", min(d))

print("Search Air in dictionary:")
print("Air" in d.values())

print("Search Doctor in dictionary:")
print("Doctor" in d.values())
```

15.Read and write data from / to files in python programs

```
file_name = 'example.txt'

with open(file_name, 'w') as file:
    file.write("Welcome to SRM / MCA students!\n")
    file.write("LEARN, LEAP AND LEAD")
```

```
file_name = 'example.txt'

with open(file_name, 'r') as file:
    content = file.read()
    print(content)

file_name = 'example.txt'

with open(file_name, 'a') as file:
    file.write("\nAll the Students are active learners.")
```
