# Python Programs By Takkulu

**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 NUMBERS, LIST

**Python Numbers:**

```
# Python program to demonstrate different types of numbers

# Integer (int) example
integer_number = 10
print("Integer:", integer_number)

# Floating-point (float) example
```

```
float_number = 10.5
print("Float:", float_number)

# Complex number example
complex_number = 3 + 4j
print("Complex Number:", complex_number)

# Arithmetic operations with numbers
sum_result = integer_number + float_number
print("Sum of integer and float:", sum_result)

product_result = integer_number * complex_number
print("Product of integer and complex number:", product_result)

# Type checking
print("Type of integer_number:", type(integer_number))
print("Type of float_number:", type(float_number))
print("Type of complex_number:", type(complex_number))
```

**List:**

```
rivers = ["Missouri", "Fox", "Mississippi"]
print("Rivers:", rivers)

x = ["apple", 3, [4.0, 5.0]]
print("Multi-type list:", x)

fileExtension = ["jpg", "txt", "doc", "bmp", "tif"]
print("File Extensions:", fileExtension)
```

## 2. TUPLE, STRINGS, 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:**

```
#String handlling
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("SPLIT THE STRING BY WHITE-SPACE")
split_name = name.split()
split_name2 = name2.split()
```

```
print("Split Name:", split_name)
print("Split Name2:", split_name2)

print("Capitalize:")
print(name.capitalize())

print("In lower Case:")
print(name.lower())

print("In upper Case:")
print(name.upper())

print("Spliting name:")
print(name.split())
```

**Set:**

```
# SET CREATION AND OPERATIONS/FUNCTIONS

print("SET CREATION AND OPERATIONS/FUNCTIONS")
fruits=set(["Apple","Mango","Banana"])
print("Fruit-SET",fruits)
print("")
print("Add 'Orange' in the set")
fruits.add("Orange")
print(fruits)
print("")
print("Remove 'Banana' from set")
fruits.remove("Banana")
print(fruits)
print("COUNT ITEM IN THE SET")
print(len(fruits))
```

## 3. LAMBDA & FILTER IN PYTHON EXAMPLES

**Lambda:**

```
# Using lambda to square a number
square = lambda x: x * x
print("Square of 5:", square(5))

# Using lambda to add two numbers
add = lambda x, y: x + y
print("Sum of 3 and 7:", add(3, 7))

# Using lambda to find the maximum of two numbers
maximum = lambda x, y: x if x > y else y
print("Maximum of 10 and 20:", maximum(10, 20))

# Using lambda with map to double each number in a list
numbers = [1, 2, 3, 4, 5]
doubled_numbers = list(map(lambda x: x * 2, numbers))
print("Doubled numbers:", doubled_numbers)
```

**Filter In Python**

```
# List of numbers
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Using filter with lambda to get even numbers
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print("Even numbers:", even_numbers)

# Using filter with lambda to get odd numbers
odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
```

```
print("Odd numbers:", odd_numbers)

# Using filter with lambda to get numbers greater than 5
greater_than_five = list(filter(lambda x: x > 5, numbers))
print("Numbers greater than 5:", greater_than_five)
```

---

**4. Class Demo**

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

# Create a list to store student objects
students = []

# Add data for more than 5 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

students.append(Student())
```

```
students[2].name = "ALBIN"
students[2].regno = "RA240003"
students[2].year = "I-MCA"
students[2].CGPA = 9.7

students.append(Student())
students[3].name = "ALBERT"
students[3].regno = "RA240004"
students[3].year = "I-MCA"
students[3].CGPA = 9.6

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

---

### 5. Object

```
# define a class
class Student:
    name = ""
    regno = ""
    year = ""
    GCPA = 0

# create object of class
s1 = Student()

# access attributes and assign new values
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}")
```

---

## 6. CREATING METHODS IN PYTHON

```
# Python program to demonstrate Creating Methods
# Defining a simple method to add two numbers
def add_numbers(a, b):
    return a + b

# Defining a method to check if a number is even
def is_even(number):
    return number % 2 == 0

# Defining a method to find the maximum of three numbers
def find_max(a, b, c):
    return max(a, b, c)

# Calling the methods
print("Sum of 5 and 10:", add_numbers(5, 10))
print("Is 4 even?:", is_even(4))
print("Maximum of 3, 7, and 5:", find_max(3, 7, 5))
```

---

## 7. IO Stream

```
# Standard: PEP 8 (Proper indentation, naming conventions, and docstrings)

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\swami\Documents\Python R6\Python Exercise\r6.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

```
# Python program to demonstrate command-line arguments and shell variables

import sys
import os

# Command-line arguments
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.")

# Shell environment variables
def shell_variables():
    user = os.getenv("USER")
    path = os.getenv("PATH")
    print("User:", user)
    print("System PATH:", path)

# Calling the functions
command_line_args()
```

```
shell_variables()
```

---

### 9. Python script to perform a real task

```
# Python script to perform a real task: Fetching and displaying weather information

import requests

def get_weather(city):
    api_key = "your_api_key_here"  # Replace with a valid API key
    base_url = "http://api.openweathermap.org/data/2.5/weather"
    params = {"q": city, "appid": api_key, "units": "metric"}

    response = requests.get(base_url, params=params)
    if response.status_code == 200:
        data = response.json()
        print(f"Weather in {city}:")
        print(f"Temperature: {data['main']['temp']}°C")
        print(f"Humidity: {data['main']['humidity']}%")
        print(f"Condition: {data['weather'][0]['description']}")
    else:
        print("Error fetching weather data")

# Example usage
city_name = input("Enter city name: ")
get_weather(city_name)
```

---

### 10. CLIENT SOCKET METHODS

```
# Python program to demonstrate a simple server socket
```

```
import socket

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

while True:
    conn, addr = serv.accept()
    print("Connected by", addr)
    from_client = ''
    while True:
        data = conn.recv(4096)
        if not data:
            break
        from_client += data.decode('utf-8')
        print("Received:", from_client)
        conn.sendall("Message received".encode('utf-8'))
    conn.close()
    print("Client disconnected")
```

---

**11. General Socket Methods**

```
# Python program to demonstrate General Socket Methods

import socket
```

```
# Creating a socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print("Socket created successfully")

# Getting the socket type
sock_type = sock.type
print("Socket Type:", sock_type)

# Getting the socket family
sock_family = sock.family
print("Socket Family:", sock_family)

# Getting the default timeout
timeout = sock.gettimeout()
print("Default Timeout:", timeout)

# Setting a timeout
sock.settimeout(10)
print("New Timeout:", sock.gettimeout())

# Getting the socket's own address
sock.bind(('localhost', 0))  # Bind to an available port
print("Socket bound to:", sock.getsockname())

# Closing the socket
sock.close()
print("Socket closed")
```

---

## 12. Creating Thread Using Threading Module

```
# Python program to demonstrate Creating Thread Using Threading Module

import threading
import time

def print_numbers():
    for i in range(1, 6):
        print(f"Number: {i}")
        time.sleep(1)

def print_letters():
    for letter in 'ABCDE':
        print(f"Letter: {letter}")
        time.sleep(1)

# Creating threads
thread1 = threading.Thread(target=print_numbers)
thread2 = threading.Thread(target=print_letters)

# Starting threads
thread1.start()
thread2.start()

# Waiting for threads to complete
thread1.join()
thread2.join()
```

```
print("Threads execution completed")
```

---

## 13. COMPOUND DATA USING PYTHON

```
# Python program to represent compound data using Python

# Using a dictionary to represent a student's information
student = {
    "name": "John Doe",
    "age": 20,
    "grades": [85, 90, 78],
    "address": {
        "street": "123 Main St",
        "city": "New York",
        "zip": "10001"
    }
}

# Printing student information
print("Student Information:")
print("Name:", student["name"])
print("Age:", student["age"])
print("Grades:", student["grades"])
print("Address:", student["address"]["street"], ",", student["address"]["city"],
student["address"]["zip"])

# Calculating the average grade
average_grade = sum(student["grades"]) / len(student["grades"])
print("Average Grade:", average_grade)
```

---

## 14. LISTS, TUPLES, DICTIONARIES

**Tuples:-**

```
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)
```

**Dictionaries:-**

```
# Python program to demonstrate dictionary operations

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())
```

**Lists:-**

```
rivers = ["Missouri", "Fox", "Mississippi"]
print("Rivers:", rivers)

x = ["apple", 3, [4.0, 5.0]]
print("Multi-type list:", x)

fileExtension = ["jpg", "txt", "doc", "bmp", "tif"]
print("File Extensions:", fileExtension)
```

---

## 15. READ AND WRITE DATA FROM / TO FILES IN PYTHON PROGRAMS

```
# Writing to a file
file_name = 'example.txt'

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

# Reading from a file
file_name = 'example.txt'

# Open file in read mode ('r')
with open(file_name, 'r') as file:
    content = file.read()  # Reads the entire file
    print(content)

# Appending to a file
```

```
file_name = 'example.txt'


# Open file in append mode ('a')
with open(file_name, 'a') as file:
    file.write("\n All the Students are active learners.")
```

---