Name: Abhishek Kanoujia

DATA ENGINEERING BATCH 1

DAY 7 ASSIGNMENT

If statement

```
num = int(input("Enter a number: "))
if num > 0:
    print(f"{num} is a positive number.")
```

If-else statement

```
temperature = float(input("Enter the
temperature in Celsius: "))
if temperature > 30:
    print("It's a hot day.")
else:
    print("It's not a hot day.")
```

If-elif-else statement

```
grade = int(input("Enter your grade
(1-100): "))
if grade >= 90:
    print("A")
elif grade >= 80:
    print("B")
```

```
elif grade >= 70:
    print("C")
else:
    print("F")



# For Loop

print("Printing numbers from 1 to 5
using a for loop:")
for i in range(1, 6):
    print(i)


# While Loop

count = 0
while count < 3:
    print("This is loop iteration",
count + 1)
    count += 1
```

```python
# Nested Loop
print("Nested Loop Example:")
for i in range(3):
    for j in range(2):
        print(f"({i}, {j})", end=' ')
    print()

    # Break, Continue & Pass
    for char in "Python":
        if char == 'h':
            break
        elif char == 'o':
            continue
        else:
            pass
        print(char)



# Input and Output
name = input("Enter your name: ")
print("Hello, " + name + "!")
```

```
# Introduction to Lists
fruits = ["apple", "banana",
"orange"]
print("List of fruits:", fruits)



# List Methods and Slicing
fruits.append("grape")
print("After adding 'grape':",
fruits)



print("Sliced list:", fruits[1:3])


# Introduction to Dictionaries
& Dictionary Methods
student = {
    "name": "John",
    "age": 20,
    "grade": "A"
}
print("Student details:", student)
```

```
# Introduction to Set & Set Methods
numbers = {1, 2, 3, 4, 5}
print("Set of numbers:", numbers)
```

```
# Introduction to Map & Map Methods
square = lambda x: x*x
numbers = [1, 2, 3, 4, 5]
squared_numbers = map(square, numbers)
print("Squared numbers:", list(squared_numbers))
```

```python
# Grade Calculator

def calculate_grade(percentage):
    if percentage >= 90:
        return "A"
    elif percentage >= 80:
        return "B"
    elif percentage >= 70:
        return "C"
    elif percentage >= 60:
        return "D"
    else:
        return "F"

percentage = float(input("Enter your
percentage: "))
result = calculate_grade(percentage)
print(f"Your grade is: {result}")
```

```
# Simple To-Do List

todo_list = []

while True:
    print("\nTo-Do List Menu:")
    print("1. Add Task")
    print("2. View Tasks")
    print("3. Exit")

    choice = int(input("Enter your
choice: "))

    if choice == 1:
        task = input("Enter the task:
")
        todo_list.append(task)
        print(f"Task '{task}'
added.")
    elif choice == 2:
        print("To-Do List:",
todo_list if todo_list else "Empty")
    elif choice == 3:
        print("Exiting. Goodbye!")
        break
    else:
        print("Invalid choice.")
```

```
# Factorial Calculator

def calculate_factorial(n):
    result = 1
    while n > 1:
        result *= n
        n -= 1
    return result

number = int(input("Enter a number:
"))
factorial_result =
calculate_factorial(number)
print(f"The factorial of {number} is:
{factorial_result}")



# Mapping function example
numbers = [1, 2, 3, 4, 5]
squared_numbers = map(lambda x: x**2,
numbers)
print(list(squared_numbers))
```

```python
# Python function example
def greet(name):
    print(f"Hello, {name}!")

greet("John")


# Lambda expression example
square = lambda x: x**2
print(square(5))
```

```
# Sample string
sentence = "   This is a sample
sentence with sample words.   "

words = sentence.split()
print("1. Splitting the sentence into
words:", words)

# Joining the words into a sentence
joined_sentence = " ".join(words)
print("2. Joining the words into a
sentence:", joined_sentence)

# Replacing occurrences of "sample"
with "modified"
modified_sentence =
sentence.replace("sample",
"modified")
print("3. Replacing 'sample' with
'modified':", modified_sentence)

# Checking if the sentence starts
with "This" and ends with "words."
print("4. Does the sentence start
with 'This'? :",
sentence.startswith("This"))  # True
print("   Does the sentence end with
'words.'? :",
sentence.endswith("words."))  # True
```

```
# Finding the index of the first
occurrence of "sample"
index_of_sample =
sentence.find("sample")
print("5. Index of the first
occurrence of 'sample':",
index_of_sample)

# Counting occurrences of "sample"
sample_count =
sentence.count("sample")
print("6. Count of occurrences of
'sample':", sample_count)

# Converting the sentence to
uppercase and lowercase
upper_case_sentence =
sentence.upper()
lower_case_sentence =
sentence.lower()
print("7. Uppercase sentence:",
upper_case_sentence)
print("   Lowercase sentence:",
lower_case_sentence)

# Stripping leading and trailing
whitespaces
stripped_sentence = sentence.strip()
print("8. Stripped sentence:",
stripped_sentence)
```

```python
# Number functions example
num = -5.67
print(abs(num))
print(round(num))
print(max(2, 7, 1, 9))
print(min(4, 8, 3, 5))


# Date and Time functions example
from datetime import datetime, timedelta

current_time = datetime.now()
print(current_time)
future_time = current_time + timedelta(days=7)
print(future_time)



# Default argument values example
def greet(name, greeting="Hello"):
    print(f"{greeting}, {name}!")

greet("Alice")
```

```python
# Keyword arguments example
def show_info(name, age, city):
    print(f"Name: {name}, Age: {age},
City: {city}")

show_info(age=25, name="Bob",
city="New York")


# Special parameters example
def special_params(*args, **kwargs):
    print("Positional arguments:",
args)
    print("Keyword arguments:",
kwargs)

    special_params(1, 2, 3,
name="Alice", age=30)



# Arbitrary argument lists
example
def sum_numbers(*nums):
    return sum(nums)

result = sum_numbers(1, 2, 3, 4, 5)
print(result)
```

```python
# OOPS example
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        print(f"{self.name} says
Woof!")

my_dog = Dog("Buddy", 3)
print(my_dog.name)
my_dog.bark()


# Inheritance and Polymorphism
example
class Animal:
    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        print("Woof!")

class Cat(Animal):
    def speak(self):
```

```
        print("Meow!")

def animal_sound(animal):
    animal.speak()

dog = Dog()
cat = Cat()

animal_sound(dog)
animal_sound(cat)
```

```python
# OOPS example2
class BankAccount:
    # Constructor
    def __init__(self,
account_holder, balance=0):
        self._account_holder =
account_holder  # Protected attribute
        self.__balance = balance  #
Private attribute

    # Encapsulation - Getter for
balance
    def get_balance(self):
        return self.__balance

    # Encapsulation - Setter for
balance
    def set_balance(self,
new_balance):
        if new_balance >= 0:
            self.__balance =
new_balance
        else:
            print("Invalid balance
value.")

    # Polymorphism - Display balance
    def display_balance(self):
        print(f"Account balance for
{self._account_holder}:
```

```
${self.__balance}")

    # Inheritance - Method for
transactions
    def perform_transaction(self,
amount):
        print("Performing a generic
transaction")

    # Method Overriding - Deposit
method specific to BankAccount
    def deposit(self, amount):
        self.__balance += amount
        print(f"Deposited ${amount}.
New balance: ${self.__balance}")

    # Method Overriding - Withdraw
method specific to BankAccount
    def withdraw(self, amount):
        if amount <= self.__balance:
            self.__balance -= amount
            print(f"Withdrew
${amount}. New balance:
${self.__balance}")
        else:
            print("Insufficient
funds!")

# Inheritance - SavingsAccount
inherits from BankAccount
```

```
class SavingsAccount(BankAccount):
    def __init__(self,
account_holder, balance=0,
interest_rate=0.02):

super().__init__(account_holder,
balance)
        self.interest_rate =
interest_rate

    # Method Overriding - Deposit
method specific to SavingsAccount
    def deposit(self, amount):
        super().deposit(amount)
        self._calculate_interest()
        print(f"Interest calculated.
New balance: ${self.get_balance()}")

    # Method Overriding - Withdraw
method specific to SavingsAccount
    def withdraw(self, amount):
        super().withdraw(amount)
        self._calculate_interest()
        print(f"Interest calculated.
New balance: ${self.get_balance()}")

    # Encapsulation - Private method
for calculating interest
    def _calculate_interest(self):
        interest_amount =
```

```
self.get_balance() *
self.interest_rate

self.set_balance(self.get_balance() +
interest_amount)
        print(f"Interest added:
${interest_amount}")

# Creating bank accounts
account1 = BankAccount("Alice", 1000)
account2 = SavingsAccount("Bob", 500,
0.03)

# Performing transactions
account1.display_balance()
account1.deposit(500)
account1.withdraw(200)

account2.display_balance()
account2.deposit(100)
account2.withdraw(300)
```

```
# File handling example
# Writing to a file
with open("example.txt", "w") as
file:
    file.write("Hello, File!")

# Reading from a file
with open("example.txt", "r") as
file:
    content = file.read()
    print(content)
```

```
# Exception handling example
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero!")
except Exception as e:
    print(f"An error occurred: {e}")
else:
    print("Division successful.")
finally:
    print("Execution completed.")
```

```
# Examples
class Bird:
    # constructor
    def __init__(self, name):
        self.name = name

    def print_info(self):
        print('This bird is:',
self.name)

    def fly(self):
        print('The bird can fly')

# Parrot class inherits from Bird
class with all attributes.
class Parrot(Bird):
    def __init__(self, name, color,
charater):
        # call the constructor of the
parent class
        super().__init__(name)
        self.color = color
        self.charater = charater

    # Override method
    def print_info(self):
        print('This bird is:',
self.name)
        print('Color of bird is:',
self.color)
```

```
        print('Character of bird
is:', self.charater)

# Creating an instance of Parrot
class
obj_parrot = Parrot('parrot', 'red',
'good')

# Accessing methods from both Bird
and Parrot classes
obj_parrot.fly()
obj_parrot.print_info()
```