**Assignment- OOP**

1. Write a Python program to create a class representing a stack data structure. Include methods for pushing and popping elements.

class Stack:

def \_\_init\_\_(self):

self.stack = []

def push(self,item):

self.stack.append(item)

print(f"Pushed {item} on stack")

def pop(self):

if len(self.stack) == 0:

print("stack is empty.")

else:

item = self.stack.pop()

print(f"Popped {item} from stack")

def show(self):

if len(self.stack) == 0:

print("stack is empty.")

else:

print(self.stack)

if \_\_name\_\_ == '\_\_main\_\_':

st1 = Stack()

st1.show()

st1.push(1)

st1.push(2)

st1.push(3)

st1.push(4)

st1.show()

st1.pop()

st1.show()

st1.pop()

st1.show()

st1.push(4)

st1.show()

st1.pop()

st1.pop()

st1.pop()

st1.show()

1. Write a Python program to create a class representing a linked list data structure. Include methods for displaying linked list data, inserting and deleting nodes.

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

class LinkedList:

def \_\_init\_\_(self):

self.head = None

def insert(self, data):

new\_node = Node(data)

if self.head is None:

self.head = new\_node

else:

current = self.head

while current.next:

current = current.next

current.next = new\_node

def delete(self, key):

current = self.head

if current is None:

print("The list is empty.")

return

if current.data == key:

self.head = current.next

current = None

return

prev = None

while current and current.data != key:

prev = current

current = current.next

if current is None:

print(f"Node with value {key} not found.")

return

prev.next = current.next

current = None

def display(self):

current = self.head

if current is None:

print("The list is empty.")

return

while current:

print(current.data, end=" -> ")

current = current.next

print("None")

linked\_list = LinkedList()

linked\_list.insert(10)

linked\_list.insert(20)

linked\_list.insert(30)

linked\_list.insert(40)

print("Linked List after insertion:")

linked\_list.display()

linked\_list.delete(20)

print("\nLinked List after deletion of 20:")

linked\_list.display()

linked\_list.delete(50)

1. Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.

class Cart:

def \_\_init\_\_(self):

self.cart = {}

def addItem(self, item, price):

self.cart.update({item:price})

print(f"{item} added to cart with price {price}")

def removeItem(self, item):

if item in self.cart:

item1 = self.cart.pop(item)

print(f"{item} removed from cart")

else:

print(f"{item} not found in cart")

def totalPrice(self):

self.sum = 0

for item in self.cart:

self.sum = self.sum + self.cart[item]

print("Total price of all items in cart =",self.sum)

def showCart(self):

if len(self.cart)>0:

print("Items\t Price")

for item in self.cart:

print(item,"\t",self.cart[item])

else:

print("Cart is empty")

if \_\_name\_\_ == '\_\_main\_\_':

c1 = Cart()

c1.showCart()

c1.addItem('Shooes',1000)

c1.showCart()

c1.totalPrice()

c1.addItem('Mobile',10000)

c1.showCart()

c1.totalPrice()

c1.removeItem('Shooes')

c1.showCart()

c1.totalPrice()

**Assignment 4**

1. Write a Python program to create a lambda function that adds 15 to a given number passed in as an argument, also create a lambda function that multiplies argument x with argument y and prints the result.

add15 = lambda x : x + 15

print(add15(10))

multi = lambda x,y : x \* y

print(multi(8,6))

1. Write a Python program to create a function that takes one argument, and that argument will be multiplied with an unknown given number.

def func(num):

print(f"Double the number of {num} = {num\*2}")

print(f"Triple the number of {num} = {num\*3}")

print(f"Quadruple the number of {num} = {num\*4}")

print(f"Quintuple the number of {num} = {num\*5}")

func(15)

1. Write a Python program to sort a list of tuples using Lambda.

l1 = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]

l1\_sorted = sorted(l1 , key = lambda x: x[1])

print("Original list of tuples:")

print(l1)

print("Sorting the List of Tuples:")

print(l1\_sorted)

1. Write a Python program to sort a list of dictionaries using Lambda.

l1 = [{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]

l1\_sorted = sorted(l1, key = lambda x : x['color'])

print("Original list of dictionaries :")

print(l1)

print("Sorting the List of dictionaries :")

print(l1\_sorted)

1. Write a Python program to filter a list of integers using Lambda.

l1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

evenl = []

oddl = []

fltoddeven = lambda i : evenl.append(i) if i%2 == 0 else oddl.append(i)

for i in l1:

fltoddeven(i)

print("Original list of integers:")

print(l1)

print("Even numbers from the said list:")

print(evenl)

print("Odd numbers from the said list:")

print(oddl)

1. Write a Python program to square and cube every number in a given list of integers using Lambda.

l1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

sqrl = [x\*\*2 for x in l1]

cubel = [x\*\*3 for x in l1]

print("Original list of integers:")

print(l1)

print("Square every number of the said list:")

print(sqrl)

print("Cube every number of the said list:")

print(cubel)

1. Write a Python program to find if a given string starts with a given character using Lambda.

val = lambda str1,c : str1.startswith(c)

print(val('Python','P'))

1. Write a Python program to extract year, month, date and time using Lambda.

from datetime import datetime

now = datetime.now()

extract\_year = lambda x: x.year

extract\_month = lambda x: x.month

extract\_day = lambda x: x.day

extract\_time = lambda x: x.time()

print(now)

print(extract\_year(now))

print(extract\_month(now))

print(extract\_day(now))

print(extract\_time(now))

1. Write a Python program to check whether a given string is a number or not using Lambda.

l1 = ['123','45','abc','32','xy']

is\_number = lambda str1: str1.isdigit()

for str1 in l1:

print(is\_number(str1))

1. Write a Python program to create Fibonacci series up to n using Lambda.

from functools import reduce

fib\_series = lambda n: reduce(lambda x, \_: x + [x[-1] + x[-2]], range(n-2), [0, 1])

n = 5

print(f"Fibonacci series upto {n}:")

print(fib\_series(n))

1. Write a Python program to find the intersection of two given arrays using Lambda.

l1 = [1, 2, 3, 5, 7, 8, 9, 10]

l2 = [1, 2, 4, 8, 9]

intersect = list(filter(lambda x:x in l1,l2))

print("Original arrays:")

print(l1)

print(l2)

print("Intersection of the said arrays:")

print(intersect)