**Assignment- OOP**

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

class stck:

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

self.nmbrs = []

def push(self, item):

self.nmbrs.append(item)

print("pushed given nmbr into the stack is ", item)

def pop(self):

if len(self.nmbrs) == 0:

print("Stack is empty")

return none

else:

popped\_item = self.nmbrs.pop()

print ("pop item from stack is ",popped\_item)

return popped\_item

try this:

my\_stk = stck()

my\_stk.push(1)

my\_stk.push(2)

my\_stk.push(3)

my\_stk.push(4)

my\_stk.pop()

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 # The data part of the node

self.next = None # Pointer to the next node, initially None

# Linked List class

class LinkedList:

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

self.head = None # Initialize the head of the list as None

# Method to display the data in the linked list

def display(self):

if self.head is None:

print("The linked list is empty.")

return

current\_node = self.head

print("Linked List data:")

while current\_node is not None:

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

current\_node = current\_node.next

print("None")

# Method to insert a new node at the end of the linked list

def insert(self, data):

new\_node = Node(data)

if self.head is None:

self.head = new\_node # If list is empty, new node is the head

else:

current\_node = self.head

while current\_node.next is not None: # Traverse to the last node

current\_node = current\_node.next

current\_node.next = new\_node # Insert the new node at the end

print(f"{data} inserted into the linked list.")

# Method to delete a node by value

def delete(self, data):

if self.head is None:

print("The linked list is empty, nothing to delete.")

return

# If the node to be deleted is the head

if self.head.data == data:

self.head = self.head.next # Move head to the next node

print(f"Node with data {data} deleted from the linked list.")

return

current\_node = self.head

while current\_node.next is not None:

if current\_node.next.data == data:

current\_node.next = current\_node.next.next # Bypass the node

print(f"Node with data {data} deleted from the linked list.")

return

current\_node = current\_node.next

print(f"Node with data {data} not found in the linked list.")

try this:

l1 = LinkedList()

l1.insert(10)

l1.insert(20)

l1.insert(30)

l1.display()

l1.delete(20)

l1.display()

l1.delete(40)

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 ShoppingCart:

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

self.cart = []

def add\_item(self, item\_name, price):

self.cart.append((item\_name, price))

print(item\_name, "added to the cart")

def rem\_item(self, item\_name):

for item in self.cart:

if item[0] == item\_name:

self.cart.remove(item)

print("Item", item\_name, "is deleted from cart")

return

print(item\_name, "not found in the cart")

def total\_price(self):

total = sum(item[1] for item in self.cart)

return total

def show\_cart(self):

if not self.cart:

print("The cart is empty.")

else:

print("Shopping Cart:")

for item in self.cart:

print(f"{item[0]} - ${item[1]}")

print(f"Total Price: ${self.total\_price()}")

try this:

cart = ShoppingCart()

cart.add\_item("Apple", 30)

cart.add\_item("orange", 20)

cart.add\_item("Banana", 10)

cart.show\_cart()

cart.rem\_item("Banana")

cart.show\_cart()

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

class stck:

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

self.nmbrs = []

def push(self, item):

self.nmbrs.append(item)

print("pushed given nmbr into the stack is ", item)

def pop(self):

if len(self.nmbrs) == 0:

print("Stack is empty")

return none

else:

popped\_item = self.nmbrs.pop()

print ("pop item from stack is ",popped\_item)

return popped\_item

def display(self):

print("Current Stack value is ", self.nmbrs)

try this:

my\_stk = stck()

my\_stk.push(1)

my\_stk.push(2)

my\_stk.push(3)

my\_stk.push(4)

my\_stk.display()

my\_stk.pop()

my\_stk.display()

**Assignment**

**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.  
Sample Output:  
25  
48

add = lambda x: x+15, multi = lambda x,y: x\*y  
 result1 = add(10), result2 = multi(6,8)

print (result1)

print (result2)

**2.** Write a Python program to create a function that takes one argument, and that argument will be multiplied with an unknown given number.  
Sample Output:  
Double the number of 15 = 30  
Triple the number of 15 = 45  
Quadruple the number of 15 = 60  
Quintuple the number 15 = 75

def multi\_func(val):

return lambda x: x\*val

res1 = multi\_func(2)

res2 = multi\_func(3)

res3 = multi\_func(4)

res4 = multi\_func(5)

num = 15

print("Double the number of ", num, f"= {res1(num)}")

print("Triple the number of ", num, f"= {res2(num)}")

print("Quadruple the number of ", num, f"= {res3(num)}")

print("Quintuple the number of ", num, f"= {res4(num)}")

**3.** Write a Python program to sort a list of tuples using Lambda.  
Original list of tuples:  
[('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]  
Sorting the List of Tuples:  
[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]

sorted\_subject\_mark = sorted(subject\_mark, key = lambda x : x[1])

**4.** Write a Python program to sort a list of dictionaries using Lambda.  
Original list of dictionaries :  
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]  
Sorting the List of dictionaries :  
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}]

sorted\_phones = sorted(phones, key = lambda x: x['color'])

1. Write a Python program to filter a list of integers using Lambda.  
   Original list of integers:  
   newlist = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Even numbers from the said list:  
[2, 4, 6, 8, 10] -🡪 even\_numbers = list(filter(lambda x: x%2==0, newlist))

Odd numbers from the said list:  
[1, 3, 5, 7, 9] 🡪 odd\_numbers = list(filter(lambda x: x%2 != 0, newlist))

1. Write a Python program to square and cube every number in a given list of integers using Lambda.  
   Original list of integers:  
   [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
   Square every number of the said list:  
   [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]  
   Cube every number of the said list:  
   [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

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

sqres = list(map(lambda x: x\*\*2, newlist))

cubes = list(map(lambda x: x\*\*3, newlist))

print("Original list of integers:")

print(newlist)

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

print(sqres)

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

print(cubes)

1. Write a Python program to find if a given string starts with a given character using Lambda.  
   Sample Output:  
   True  
   False

starts\_with = lambda x: x.startswith("H")

print(starts\_with("Hello"))

print(starts\_with("hello"))

1. Write a Python program to extract year, month, date and time using Lambda.  
   Sample Output:  
   2020-01-15 09:03:32.744178  
   2020  
   1  
   15  
   09:03:32.744178

extract\_date = lambda dt: dt.year

extract\_month = lambda dt: dt.month

extract\_day = lambda dt: dt.day

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

from datetime import datetime

dt = datetime.now()

print(extract\_date(dt))

print(extract\_month(dt))

print(extract\_day(dt))

print(extract\_time(dt))

1. Write a Python program to check whether a given string is a number or not using Lambda.  
   Sample Output:  
   True  
   True  
   False  
   True  
   False  
   True  
   Print checking numbers:  
   True  
   True

strings = ["123", "45.67", "hello", "-123", "12.34.56", "789"]

def is\_number(s):

if s.replace('.', '', 1).isdigit() and s.count('.') <= 1:

return True

return False

print("Print checking numbers:")

for i in strings:

res10 = is\_number(i)

print(res10, end="\n\n")

**10.** Write a Python program to create Fibonacci series up to n using Lambda.  
Fibonacci series upto 2:  
[0, 1]  
Fibonacci series upto 5:  
[0, 1, 1, 2, 3]  
Fibonacci series upto 6:  
[0, 1, 1, 2, 3, 5]  
Fibonacci series upto 9:  
[0, 1, 1, 2, 3, 5, 8, 13, 21]

from functools import reduce

fib\_series1 = lambda n: reduce(lambda x, \_ : x + [x[-1] + x[-2]], range(n-2), [0,1])

print("Fibonacci series upto 2:", fib\_series(2))

print("Fibonacci series upto 5:", fib\_series(5))

print("Fibonacci series upto 6:", fib\_series(5))

print("Fibonacci series upto 9:", fib\_series(9))

**11.** Write a Python program to find the intersection of two given arrays using Lambda.  
Original arrays:  
[1, 2, 3, 5, 7, 8, 9, 10]  
[1, 2, 4, 8, 9]  
Intersection of the said arrays: [1, 2, 8, 9]

intersction1 = list(filter(lambda x: x in arr1, arr2))

print("Intersection of the said arrays:",intersction1)