**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.\_items = []

def push(self, item):

"""Add an item to the top of the stack."""

self.\_items.append(item)

def pop(self):

"""Remove and return the item from the top of the stack. Raise an exception if the stack is empty."""

if self.is\_empty():

raise IndexError("pop from an empty stack")

return self.\_items.pop()

def peek(self):

if self.is\_empty():

raise IndexError("peek from an empty stack")

return self.\_items[-1]

def is\_empty(self):

"""Return True if the stack is empty, False otherwise."""

return len(self.\_items) == 0

def size(self):

return len(self.\_items)

def \_\_str\_\_(self):

return str(self.\_items)

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.
2. 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):

"""Initialize an empty shopping cart."""

self.items = {}

def add\_item(self, item\_name, price, quantity=1):

if item\_name in self.items:

self.items[item\_name]['quantity'] += quantity

else:

self.items[item\_name] = {'price': price, 'quantity': quantity}

print(f"Added {quantity} of {item\_name} to the cart.")

def remove\_item(self, item\_name, quantity=1):

if self.items[item\_name]['quantity'] <= quantity:

del self.items[item\_name]

print(f"Removed {item\_name} from the cart.")

else:

self.items[item\_name]['quantity'] -= quantity

print(f"Removed {quantity} of {item\_name} from the cart.")

def calculate\_total(self):

total = 0

for item, details in self.items.items():

total += details['price'] \* details['quantity']

return total

def display\_cart(self):

if not self.items:

print("Shopping cart is empty.")

return

print("Shopping cart items:")

for item, details in self.items.items():

print(f"{item}: ${details['price']} x {details['quantity']} = ${details['price'] \* details['quantity']:.2f}")

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

class Stack:

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

self.stack = []

def push(self, item):

self.stack.append(item)

print(f"Pushed {item} onto the stack.")

def pop(self):

if self.is\_empty():

print("Stack is empty. Cannot pop.")

return None

return self.stack.pop()

def peek(self):

if self.is\_empty():

print("Stack is empty. Nothing to peek.")

return None

return self.stack[-1]

def is\_empty(self):

return len(self.stack) == 0

def display(self):

if self.is\_empty():

print("Stack is empty.")

else:

print("Stack elements are:")

for item in reversed(self.stack):

print(item)

**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

**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 multiply\_number(num, factor):

return num \* factor

def main():

number = 15

multipliers = {

"Double": 2,

"Triple": 3,

"Quadruple": 4,

"Quintuple": 5

}

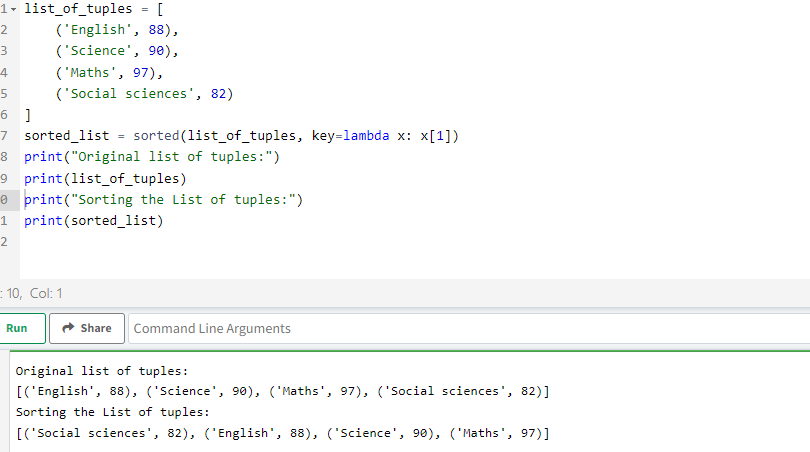
for name, factor in multipliers.items():

result = multiply\_number(number, factor)

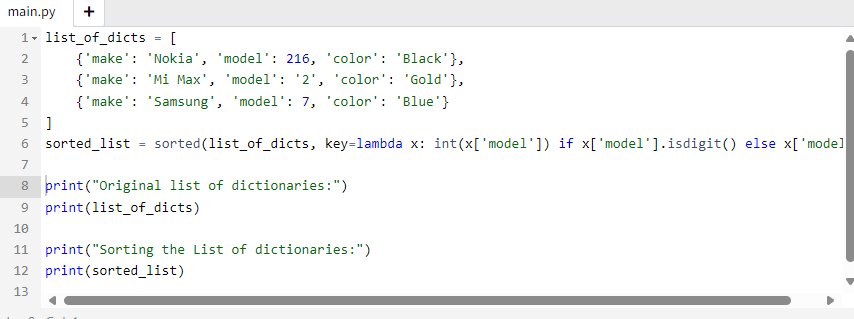
print(f"{name} the number of {number} = {result}")

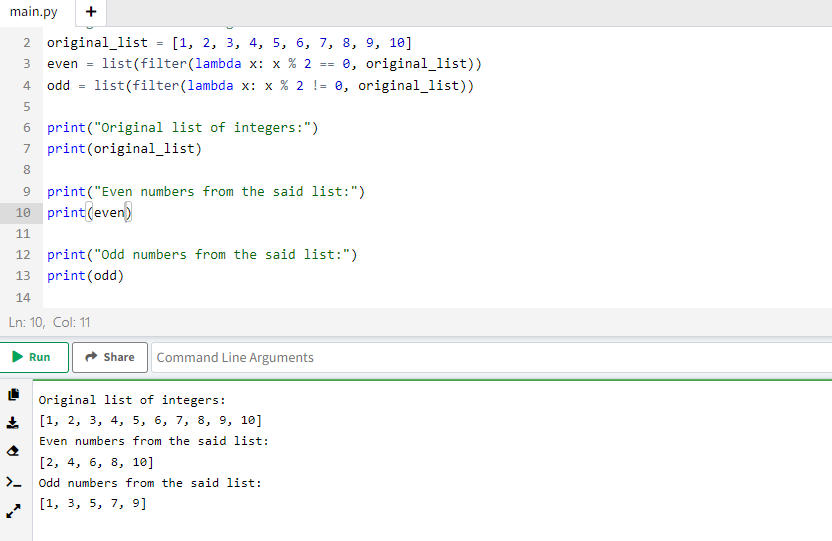
if \_\_name\_\_ == "\_\_main\_\_":

main()

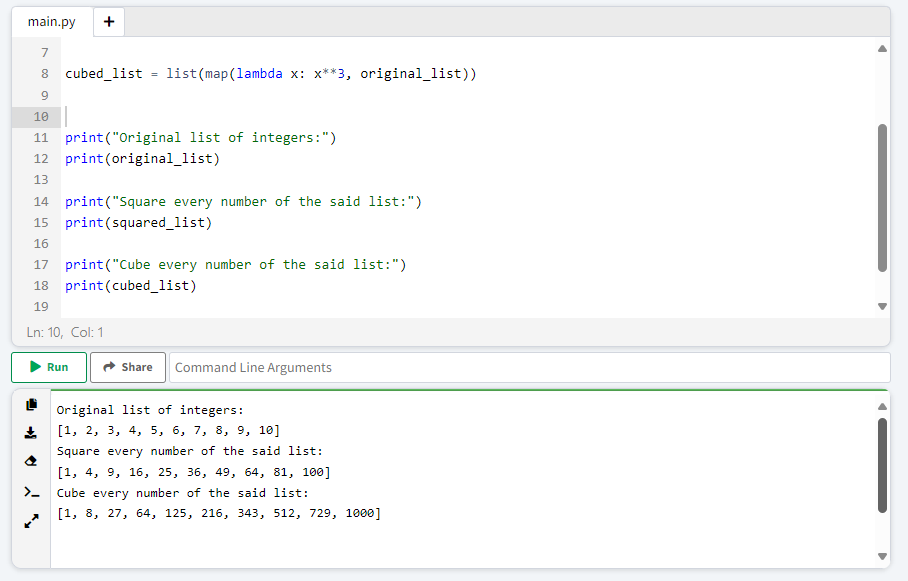
**3.** Write a Python program to sort a list of tuples using Lambda.  
Original list of tuples:  
[('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]  
Sorting the List of Tuples:  
[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]  


**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'}]



**5.** Write a Python program to filter a list of integers using Lambda.  
Original list of integers:  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
Even numbers from the said list:  
[2, 4, 6, 8, 10]  
Odd numbers from the said list:  
[1, 3, 5, 7, 9]  


**6.** 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]  
Ans:



**7.** Write a Python program to find if a given string starts with a given character using Lambda.  
Sample Output:  
True  
False  
starts\_with = lambda string, char: string.startswith(char)

strings\_to\_check = ["hello", "world"]

characters\_to\_check = ["h", "w", "a", "e"]

for s in strings\_to\_check:

for c in characters\_to\_check:

print(f"Does '{s}' start with '{c}'? {starts\_with(s, c)}")

**8.** 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  
from datetime import datetime

Ans:-

datetime\_str = "2020-01-15 09:03:32.744178"

dt = datetime.strptime(datetime\_str, "%Y-%m-%d %H:%M:%S.%f")

extract\_year = lambda dt: dt.year

extract\_month = lambda dt: dt.month

extract\_day = lambda dt: dt.day

extract\_time = lambda dt: dt.strftime("%H:%M:%S.%f")

print(datetime\_str)

print(extract\_year(dt))

print(extract\_month(dt))

print(extract\_day(dt))

print(extract\_time(dt))

**9.** 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  
Ans:

def is\_number(s):

try:

float(s)

return True

except ValueError:

return False

is\_number\_lambda = lambda s: True if is\_number(s) else False

strings\_to\_check = ["123", "45.67", "-89", "abc", "12abc", "0.1"]

print("Print checking numbers:")

for s in strings\_to\_check:

print(is\_number\_lambda(s))

**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]

Ans:

from functools import reduce

fibonacci\_up\_to\_n = lambda n: reduce(

lambda fib, \_: fib + [fib[-1] + fib[-2]],

range(n - 2),

[0, 1]

)

def print\_fibonacci\_series(n):

if n <= 0:

print("Fibonacci series up to", n, ":", [])

elif n == 1:

print("Fibonacci series up to", n, ":", [0])

else:

print("Fibonacci series up to", n, ":", fibonacci\_up\_to\_n(n))

**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]

Ans:

array1 = [1, 2, 3, 5, 7, 8, 9, 10]

array2 = [1, 2, 4, 8, 9]

set\_array2 = set(array2)

intersection = list(filter(lambda x: x in set\_array2, array1))

print("Original arrays:")

print("Array 1:", array1)

print("Array 2:", array2)

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