LINEAR SEARCH

def linear\_Search(list1, n, key):

for i in range(0, n):

if (list1[i] == key):

return i

return -1

list1 = [1 ,3, 5, 4, 7, 9]

key = 7

n = len(list1)

res = linear\_Search(list1, n, key)

if(res == -1):

Print(“2213711058039”)

print("Element not found")

else:

print("Element found at index: ", res)

OUTPUT:



BINARY SEARCH

def binary\_search(my\_list,elem):

low=0

high=len(my\_list)-1

mid=0

while low<=high:

mid=(high+low)//2

if my\_list[mid]<elem:

low=mid+1

elif my\_list[mid]>elem:

high=mid-1

else:

return mid

return -1

my\_list=[1,9,11,21,34,54,67,90]

Print(“2213711058039”)

elem\_to\_search =int(input(“enter search element:”)

print("the list is ")

print(my\_list)

my\_result = binary\_search(my\_list,elem\_to\_search)

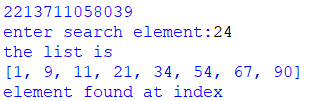
if my\_result != -1:

print("element not found at index",str(my\_result))

else:

print("element found at index")

OUTPUT:



DATA TYPES AND VARIABLES

NUMERIC:

a = 5

Print(“2213711058039”)

print("Type of a: ", type(a))

OUTPUT:



STRING:

String1 = 'Hello World'

Print(“2213711058039”)

print("String example: ")

print(String1)

OUTPUT:



LIST AND TUPLE

LIST:

List = [17,12,2,90,30,45,60]

Print(“2213711058039”)

print(List[0])

print(List[2])

print(len(List))

print('max of List', max(List))

print('min of T1', min(List))

print('sum of L1', sum(List))

print('List in sorted order', sorted(List))

List.append(100)

print("after append",List)

List.insert(2,30)

print('after insert',List)

c=List.count(30)

print ('count of 30',List)

List.extend([11,22,33])

print('after extend', List)

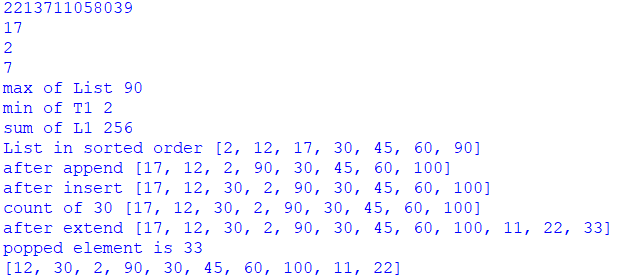
l=List.pop()

print("popped element is",l)

List.remove(17)

print(List)

OUTPUT:



TUPLE:

Tuple = [17,12,2,90,30,45,60]

print(“2213711058039”)

print(Tuple[0])

print(Tuple[2])

print(len(Tuple))

print('max of Tuple', max(Tuple))

print('min of Tuple', min(Tuple))

print('sum of Tuple', sum(Tuple))

print('Tuple in sorted order', sorted(Tuple))

c=Tuple.count(30)

print ('count of 30',c)

Tuple.extend([11,22,33])

print('after extend', Tuple)

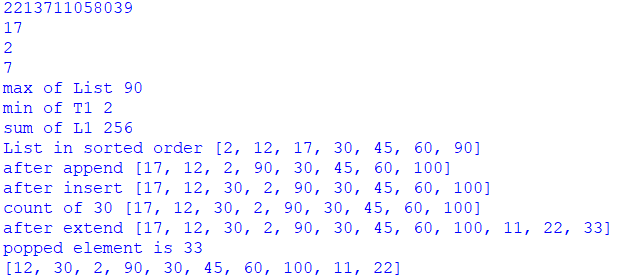
l=Tuple.pop()

print("popped element is",l)

Tuple.remove(17)

print(Tuple)

OUTPUT:



SET AND DICTIONARY

thisset = {"apple", "banana", "cherry"}

print(“2213711058039”)

print(len(thisset))

print(type(thisset))

thisset.add(32)

print(thisset)

set1=["car","bike","cherry"]

thisset.update(set1)

print(thisset)

removedValue = set1.remove("car")

print(set1)

print(thisset.union(set1))

print(thisset.intersection(set1))

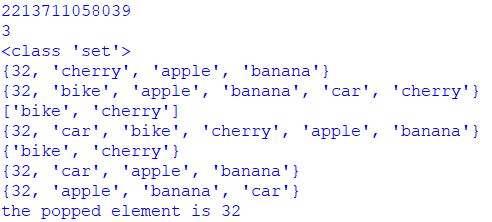
print(thisset.difference(set1))

print(thisset.symmetric\_difference(set1))

l=thisset.pop()

print("the popped element is",l)

OUTPUT:



DICTIONARIES:

dict1 = {1: "Python", 2: "Java", 3: "c++", 4: "html"}

dict2 = dict1.copy()

Print(“2213711058039”)

print(dict2)

dict1.clear()

print(dict1)

print(dict2.get(1))

print(dict2.items())

print(dict2.keys())

dict2.pop(4)

print(dict2)

dict2.popitem()

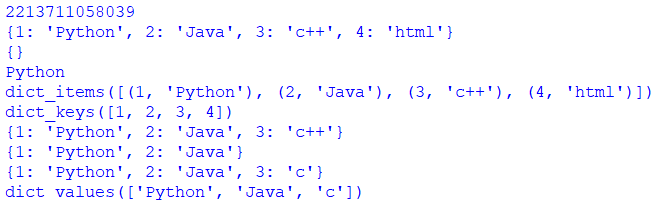
print(dict2)

dict2.update({3: "c"})

print(dict2)

print(dict2.values())

OUTPUT:



CONTROL STRUCTURE

CONDITIONAL STRUCTURE:

IF:

v = 5

t = 4

Print(“2213711058039”)

print("The initial value of v is", v, "and that of t is ",t)

if v > t :

print(v, "is bigger than ", t)

v -= 2

print("The new value of v is", v, "and the t is ",t)

IF ELSE:

if v > t :

Print(“2213711058039”)

print("v is greater than t")

else :

print("v is less than t")

IF LOOP:

if v > t :

Print(“2213711058039”)

print(v, "is bigger than ", t)

v -= 2

print("The new value of v is", v, "and the t is ",t)

if v < t :

print(v , "is smaller than ", t)

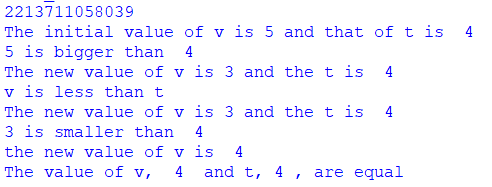
v += 1

print("the new value of v is ", v)

if v == t:

print("The value of v, ", v, " and t,", t, ", are equal")

OUTPUT:



LOOP CONTROL STRUCTURE:

FOR:

l = [2, 4, 7, 1, 6, 4]

for i in range(len(l)):

Print(“2213711058039”)

print(l[i], end = ", ")

print("\n")

for j in range(0,10):

print(j, end = "")

WHILE:

b = 9

a = 2

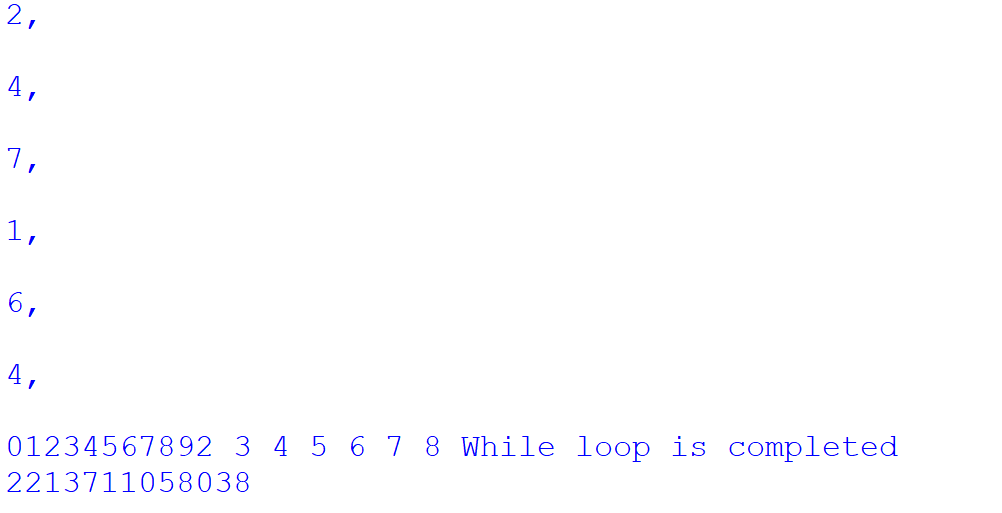
while a < b:

print(a, end = " ")

a = a + 1

print("While loop is completed")

OUTPUT:



FUNCTIONS

def my\_function(x):

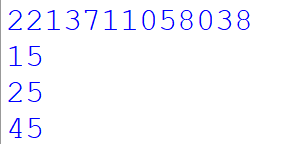
return 5 \* x

print(my\_function(3))

print(my\_function(5))

print(my\_function(9))

OUTPUT:



FACTORIAL

num = int(input("Enter a number: "))

factorial = 1

if num < 0:

print("Sorry, factorial does not exist for negative numbers")

elif num == 0:

print("The factorial of 0 is 1")

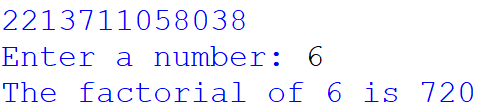
else:

for i in range(1,num + 1):

factorial = factorial\*i

print("The factorial of",num,"is",factorial)

OUTPUT:



FIBONACCI SEQUENCE

n = 10

num1 = 0

num2 = 1

next\_number = num2

count = 1

while count <= n:

print(next\_number, end=" ")

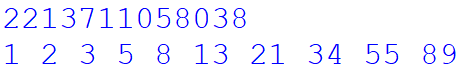
count += 1

num1, num2 = num2, next\_number

next\_number = num1 + num2

print()

OUTPUT:



CLASS

class MyClass:

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

self.value = value

def \_\_add\_\_(self, other):

if (other, MyClass):

return MyClass(self.value + other.value)

obj1 = MyClass(10)

obj2 = MyClass(20)

result = obj1 + obj2

Print(“2213711058039”)

print(result.value)

OUTPUT:



MODULE

def add(x, y):

return (x+y)

def subtract(x, y):

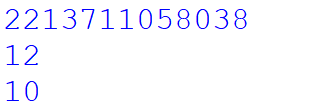
return (x-y)

import calc

print(calc.add(10, 2))

print(calc.subtract(12, 2))

OUTPUT:



EXCEPTIONAL HANDELING:

x = 5

y = "hello"

try:

z = x + y

except TypeError:

Print(“2213711058039”)

print("Error: cannot add an int and a str")

FINALLY:

try:

k = 5//0

print(k)

except ZeroDivisionError:

print("Can't divide by zero")

finally:

print('This is always executed')

RAISE:

try:

raise NameError("Hi there")

except NameError:

print ("An exception")

raise

OUTPUT:

