Neeraj Appai 3073

Algorithm Practical 09

Aim - Write a program to demonstrate Huffman Coding

Description

1) What is Huffman coding
-) Huffman coding is also known as encoding. It is an algorithm to compress data, basically. It is a baseless data compression algo.
2) This idea is to assign variable lenght code to input character
3) The most frequent character gets the smallest code & the least frequent character gets the biggest code
4) The bitsequence is assigned in such a way that the code assigned to one character is not the prefix of the code assigned to any other character to prevent ambiguity while encoding

2) Algorithm:
There are mainly 2 parts
i) Built a Huffman Tree from input characters
ii) Traverse the Huffman Tree and assign codes to characters

i) Input is a set of unique characters along with their frequencies
1. Create a leaf node for each unique character and build a min heap of all nodes
(Min - Heap - used as a priority queue. The value of frequency is used to compare, compare least

Neeraj Appan S073

frequency means root node)

2. Extract 2 nodes with min freq from the min heap

3. Create a new internal node with a frequency equal to the sum of the two nodes frequency
Make the first extracted child as its left child and other as right child
Add this node to min heap

4. Repeat step 2 and 3 until the heap contains only one node. This remaining node is the root node and the tree is complete

ii) Traverse the tree formed starting from the root
Maintain an auxillary array. While moving to left child, write 0 to array while moving to right child, write 1 to array print array when leaf is encountered

With this we can generate codes for characters
And using them we can compress the data

prac 9.py - E:/fffiiles/college pracs and projects/Algorithm/prac 9.py (3.8.3)

File Edit Format Run Options Window Help

```
# A Huffman Tree Node
class node:
    print("Neeraj Appari S073")
    def __init__(self, freq, symbol, left=None, right=None):

        # frequency of symbol
        self.freq = freq

        # symbol name (charecter)
        self.symbol = symbol

        # node left of current node
        self.left = left

        # node right of current node
        self.right = right

        # tree direction (0/1)
        self.huff = ''

# utility function to print huffman
# codes for all symbols in the newly
# created Huffman tree


def printNodes(node, val=''):
    # huffman code for current node
    newVal = val + str(node.huff)

    # if node is not an edge node
    # then traverse inside it
    if(node.left):
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/Algorithm/prac 9.py =====
Neeraj Appari S073
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
>>>
```

prac 9.py - E:/fffiiles/college pracs and projects/Algorithm/prac 9.py (3.8.3)

File Edit Format Run Options Window Help

```
def printNodes(node, val=''):
    # huffman code for current node
    newVal = val + str(node.huff)

    # if node is not an edge node
    # then traverse inside it
    if(node.left):
        printNodes(node.left, newVal)
    if(node.right):
        printNodes(node.right, newVal)

        # if node is edge node then
        # display its huffman code
    if(not node.left and not node.right):
        print(f"{node.symbol} -> {newVal}")


# charecters for huffman tree
chars = ['a', 'b', 'c', 'd', 'e', 'f']

# frequency of charecters
freq = [ 5, 9, 12, 13, 16, 45]

# list containing unused nodes
nodes = []

# converting ccharecters and frequencies
# into huffman tree nodes
for x in range(len(chars)):
    nodes.append(node(freq[x], chars[x]))

while len(nodes) > 1:
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/Algorithm/prac 9.py =====
Neeraj Appari S073
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
>>>
```

prac 9.py - E:/fffiiles/college pracs and projects/Algorithm/prac 9.py (3.8.3)

File Edit Format Run Options Window Help

```
# converting ccharecters and frequencies
# into huffman tree nodes
for x in range(len(chars)):
    nodes.append(node(freq[x], chars[x]))

while len(nodes) > 1:
    # sort all the nodes in ascending order
    # based on theri frequency
    nodes = sorted(nodes, key=lambda x: x.freq)

    # pick 2 smallest nodes
    left = nodes[0]
    right = nodes[1]

    # assign directional value to these nodes
    left.huff = 0
    right.huff = 1

    # combine the 2 smallest nodes to create
    # new node as their parent
    newNode = node(left.freq+right.freq, left.symbol+right.symbo

    # remove the 2 nodes and add their
    # parent as new node among others
    nodes.remove(left)
    nodes.remove(right)
    nodes.append(newNode)

# Huffman Tree is ready!
printNodes(nodes[0])
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:1
9) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for
more information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/
Algorithm/prac 9.py =====
Neeraj Appari S073
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
>>>
```

Neeraj Appani 5073

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical 1

Aim: Write a python program to perform matrix multiplication

Write up:

(i) Algorithm: Step 1: Start
Step 2: Initialize two matrices
Step 3: Take proper input for the matrix
Step 4: Initialize product matrix
Step 5: Multiply matrix in form of each row elements of first matrix to each column element of second
Step 6: Print the product matrix
Step 7: Stop

(ii) Explain time complexity (Big-O notation).

-> 1) Time complexity is the the complexity of an algorithm that describes the amount of computer time it takes to run an algorithm. It is

2) It is commonly estimated by counting the number of elementary operators performed by the algorithm, supposing each algorithm takes a fixed amount of time

3) Big O notation is the most common metric for calculating time complexity. It describes the execution time of a task in relation to the number of steps required to complete it. Big O notation is written as O(n) where O stands for order of magnitude and n represents what we are comparing the complexity of the task against.

prac 1.py - E:/fffiiles/college pracs and projects/Algorithm/prac 1.py (3.8.3)

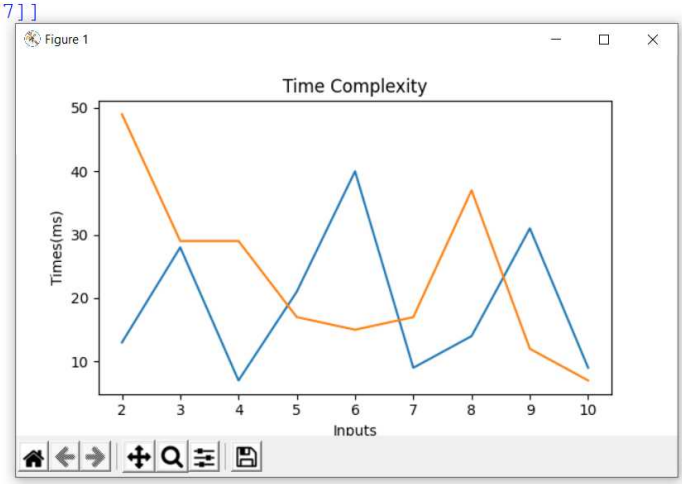
File Edit Format Run Options Window Help

```
import matplotlib.pyplot as plt
import numpy as np
print("Neeraj Appari S073\n")
u=np.array([(4,3,2),(5,4,7),(4,5,1)])
v=np.array([(2,5,1),(1,2,1),(1,1,0)])
print("Multiplication Matrix : \n")
e=np.dot(u,v)
print(e)
print("-"*25)

p=np.array([(5,4,5),(3,2,1),(0,7,5)])
q=np.array([(3,4,5),(1,1,1),(6,1,0)])
r=np.dot(p,q)
print(r)

x = [2,3,4,5,6,7,8,9,10]
y = np.concatenate(e);
z = np.concatenate(r);
plt.title('Time Complexity')
plt.xlabel('Inputs')
plt.ylabel('Times(ms)')
plt.plot(x,y)
plt.plot(x,z)

plt.show()
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
ithm/prac 1.py =====
Neeraj Appari S073

Multiplication Matrix :

[[13 28  7]
 [21 40  9]
 [14 31  9]]
--------------------------
[[49 29 29]
 [17 15 17]
 [37 12  7]]
```



Ln: 35 Col: 0

Neeraj Appari 5073 ~~Appari~~

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical No 2

Aim: Write a python program for selection Sort and perform Analysis of algorithm.

Description.

1) Start
2) Import Modules (random, matplotlib.pyplot)
3) Take random numbers from size of inputs of 10, 100, 1000 and 10000
4) Sort the numbers into arrays
5) plot the graph by contatenation list with x axis of sorted array and give labels.
6) Stop

2) a) Big O notation (O):
Big O notation is used to describe asymptotic upper bound

$$0 <= f(n) <= c\,g(n) \text{ for all } n >= n0$$

n = used to give upper bound an a function.
If a function is O(n), It is automatically O(n-square) as well. It is worst case

b) Big Omega notation (Ω):
Just like Big O notation provide an asymptotic upper bound, Ω notation provides a asymptotic lower bound. let (n) be define running time of an algorithm.

f(n) is said to be Ω(g(n)) if there exists positive constant C and n(0) such that

$0 <= C\ g(n) <= f(n)$ for all $n >= n0$

n - used to give lower bound on a function
If a function is O(n-square) it automatically Ω(n) as well

3) Big Theta notation (Θ):
let f(n) define running time of an algorithm.

$\Theta c = f(n) <= C1\ g(n)$ for $n >= n0$
$0 <= C2\ g(n) <= f(n)$ for $n >= n0$

Merging both equation we get.

$0 <= C2\ g(n) <= f(n) <= c1\ g(n)$ for $n >= n0$

The equation simply means, there exist positive constants C1 and C2 such that f(n) is sandwhich between C2 g(n) and C1 g(n)

prac 2.py - E:\fffiiles\college pracs and projects\Algorithm\prac 2.py (3.8.3)

File Edit Format Run Options Window Help

```
import matplotlib.pyplot as plt
import random
E = []

E = random.randint(0,12000)

print("Number For Input size 1:",E)

A = []
##n = int(input("Enter number of elements : "))
while len(A) < 10:
    rnd = random.randint(0,12000)
    if rnd in A:
        continue
    A += [rnd]
#print("Numbers For Input 10",A)
for i in range(len(A)):
    min_idx = i
    for j in range(i+1, len(A)):
        if A[min_idx] > A[j]:
            min_idx = j
    A[i], A[min_idx] = A[min_idx], A[i]
B=[]
while len(B) < 100:
    rnd = random.randint(0,12000)
    if rnd in B:
        continue
    B += [rnd]
#print("Numbers For Input 100",B)
for i in range(len(B)):
    min_idx = i
    for j in range(i+1, len(B)):
```

Ln: 71 Col: 0

prac 2.py - E:\fffiiles\college pracs and projects\Algorithm\prac 2.py (3.8.3)

File Edit Format Run Options Window Help

```
print ("Sorted array for input size 10:",A)
print ("Sorted array for input size 100:",B)
print ("Sorted array for input size 1000:",C)
print ("Sorted array for input size 10000:",D)


z = []
x = z+A

y=range(0,10)
plt.plot(x,y,label='Input Size =10')
z1 = []
x1 =z1+B

y1=range(0,100)
plt.plot(x1,y1,label='Input Size =100')
z2 = []
x2 = z2+C

y2=range(0,1000)
plt.plot(x2,y2,label='Input Size =1000')
z3 = []
x3 =z3+D

y3=range(0,10000)
plt.plot(x3,y3,label='Input Size =10000')

plt.title('Time Complexity For Different Input Sizes')
plt.xlabel('Inputs')
```

Ln: 71 Col: 0

prac 2.py - E:\fffiiles\college pracs and projects\Algorithm\prac 2.py (3.8.3)

File Edit Format Run Options Window Help

```
for i in range(len(B)):
    min_idx = i
    for j in range(i+1, len(B)):
        if B[min_idx] > B[j]:
            min_idx = j
    B[i], B[min_idx] = B[min_idx], B[i]

C=[]
while len(C) < 1000:
    rnd = random.randint(0,12000)
    if rnd in C:
        continue
    C += [rnd]
#print("Numbers For Input 1000",C)
for i in range(len(C)):
    min_idx = i
    for j in range(i+1, len(C)):
        if C[min_idx] > C[j]:
            min_idx = j
    C[i], C[min_idx] = C[min_idx], C[i]

D=[]
while len(D) < 10000:
    rnd = random.randint(0,12000)
    if rnd in D:
        continue
    D += [rnd]
#print("Numbers For Input 10000",D)
for i in range(len(D)):
    min_idx = i
    for j in range(i+1, len(D)):
        if D[min_idx] > D[j]:
```

Ln: 71 Col: 0

prac 2.py - E:\fffiiles\college pracs and projects\Algorithm\prac 2.py (3.8.3)

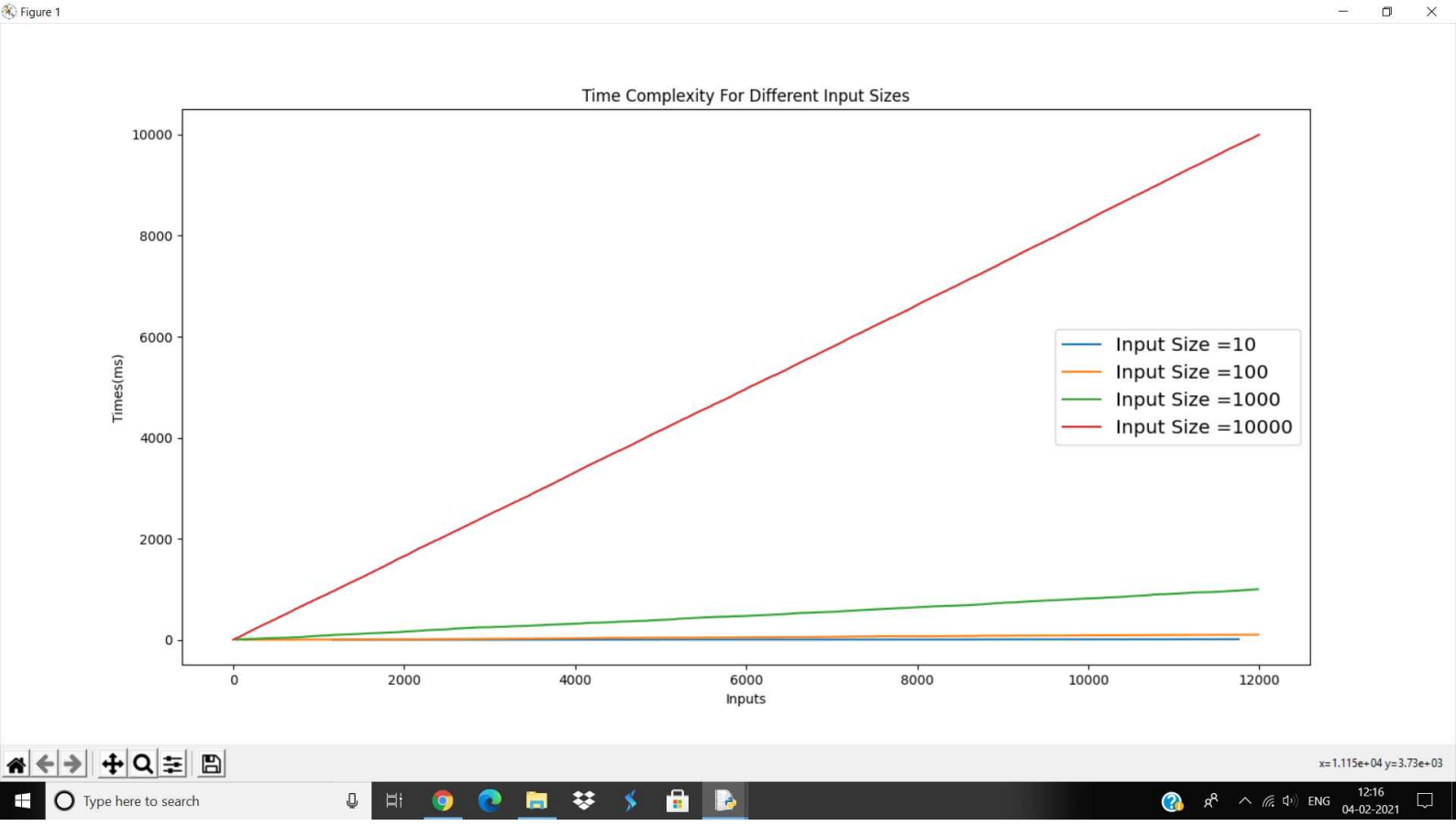
File Edit Format Run Options Window Help

```
z = []
x = z+A

y=range(0,10)
plt.plot(x,y,label='Input Size =10')
z1 = []
x1 =z1+B

y1=range(0,100)
plt.plot(x1,y1,label='Input Size =100')
z2 = []
x2 = z2+C

y2=range(0,1000)
plt.plot(x2,y2,label='Input Size =1000')
z3 = []
x3 =z3+D

y3=range(0,10000)
plt.plot(x3,y3,label='Input Size =10000')

plt.title('Time Complexity For Different Input Sizes')
plt.xlabel('Inputs')
plt.ylabel('Times(ms)')
legend = plt.legend(loc='center right', fontsize='x-large')

plt.show()
```

Ln: 71 Col: 0

Type here to search

ENG 12:16 04-02-2021



Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and projects\Algorithm\prac 2.py =====
Number For Input size 1: 9067
Sorted array for input size 10: [1162, 3563, 4636, 5187, 8498, 8668, 8997, 10364, 11181, 11762]
Sorted array for input size 100: [118, 525, 764, 912, 1660, 1714, 2279, 2298, 2308, 2367, 2373, 2417, 2448, 2495, 265
7, 2729, 2766, 3021, 3211, 3227, 3239, 3309, 3516, 3520, 3602, 3767, 3812, 3926, 3937, 4026, 4118, 4139, 4226, 4248,
4263, 4325, 4336, 4366, 4736, 4798, 4820, 5017, 5347, 5469, 5628, 5838, 5874, 5985, 6061, 6064, 6179, 6364, 6495, 680
5, 6879, 6881, 6933, 6987, 7098, 7099, 7144, 7171, 7275, 7289, 7313, 7501, 7508, 7512, 7575, 8251, 8284, 8300, 8433,
8522, 8623, 8675, 8704, 8709, 8712, 8768, 8894, 9066, 9172, 9244, 9731, 9835, 9897, 9945, 10278, 10516, 10674, 10784,
10802, 10909, 10974, 11205, 11443, 11677, 11816, 11991]
Sorted array for input size 1000: Squeezed text (77 lines).
Sorted array for input size 10000: Squeezed text (760 lines).
```

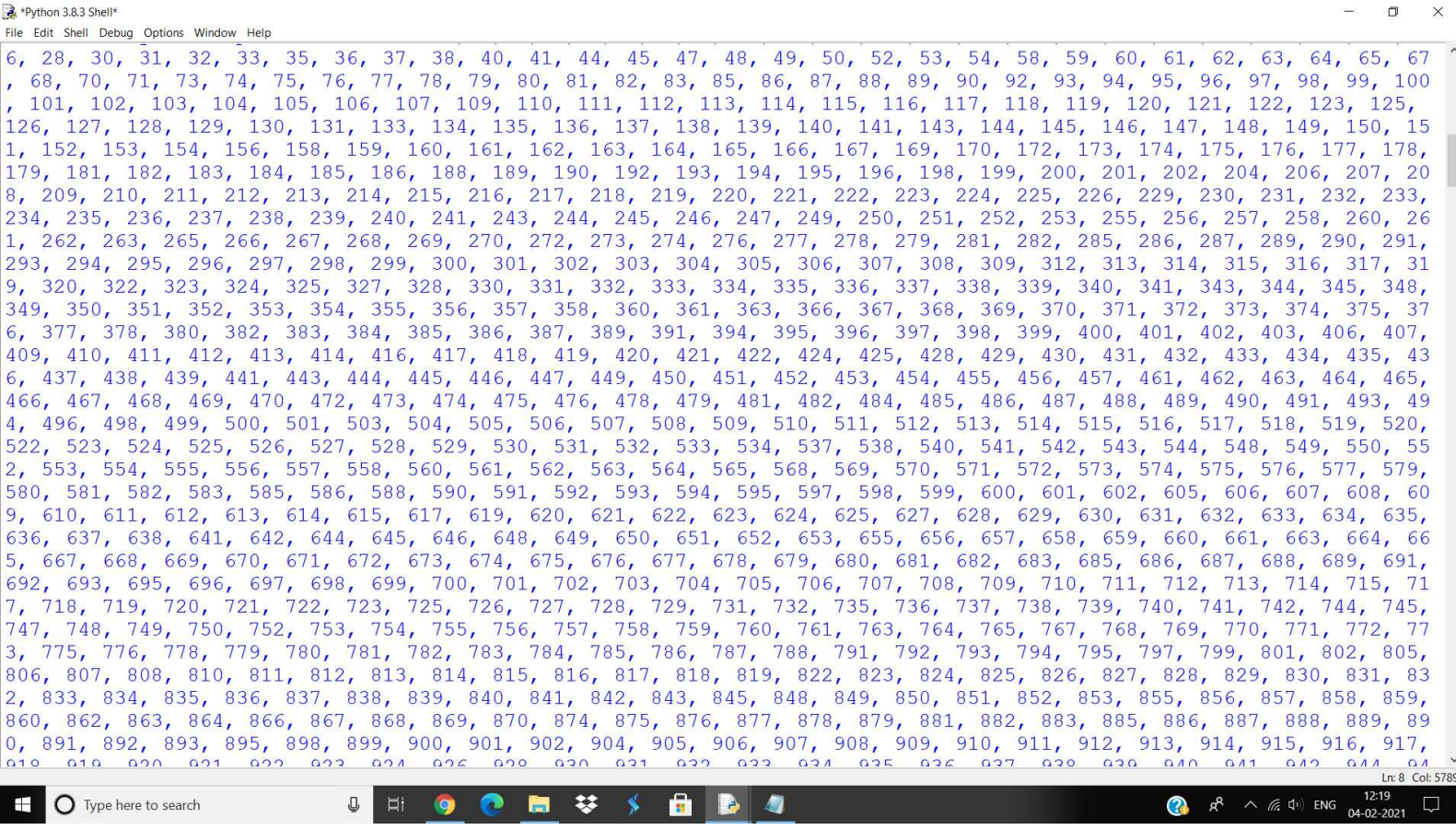
Ln: 5 Col: 0

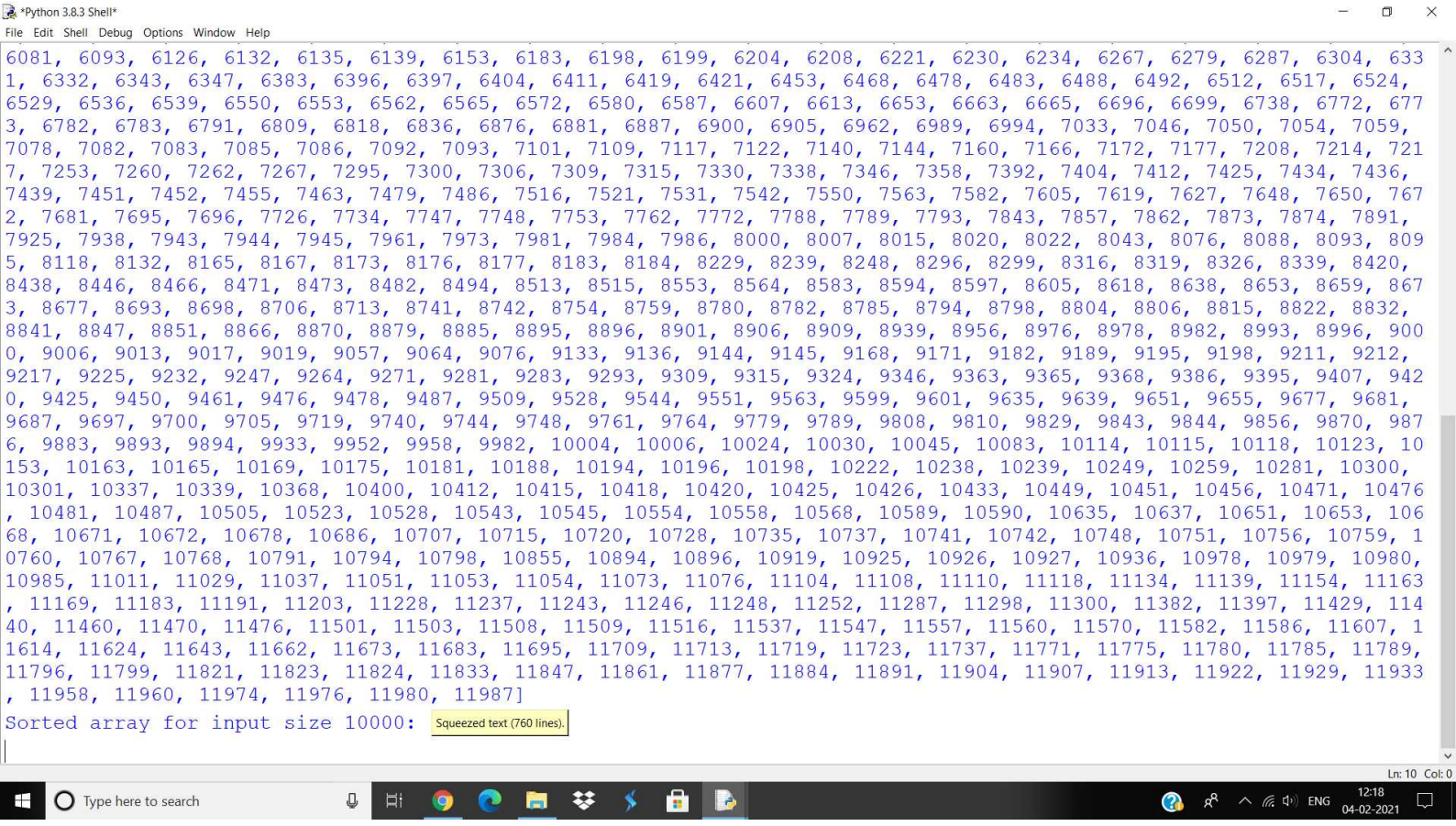
Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and projects\Algorithm\prac 2.py =====
Number For Input size 1: 9067
Sorted array for input size 10: [1162, 3563, 4636, 5187, 8498, 8668, 8997, 10364, 11181, 11762]
Sorted array for input size 100: [118, 525, 764, 912, 1660, 1714, 2279, 2298, 2308, 2367, 2373, 2417, 2448, 2495, 265
7, 2729, 2766, 3021, 3211, 3227, 3239, 3309, 3516, 3520, 3602, 3767, 3812, 3926, 3937, 4026, 4118, 4139, 4226, 4248,
4263, 4325, 4336, 4366, 4736, 4798, 4820, 5017, 5347, 5469, 5628, 5838, 5874, 5985, 6061, 6064, 6179, 6364, 6495, 680
5, 6879, 6881, 6933, 6987, 7098, 7099, 7144, 7171, 7275, 7289, 7313, 7501, 7508, 7512, 7575, 8251, 8284, 8300, 8433,
8522, 8623, 8675, 8704, 8709, 8712, 8768, 8894, 9066, 9172, 9244, 9731, 9835, 9897, 9945, 10278, 10516, 10674, 10784,
10802, 10909, 10974, 11205, 11443, 11677, 11816, 11991]
Sorted array for input size 1000: [38, 50, 53, 71, 73, 74, 76, 109, 129, 141, 157, 159, 167, 172, 181, 210, 213, 246,
260, 272, 279, 297, 305, 331, 337, 347, 354, 368, 372, 387, 413, 420, 485, 507, 511, 515, 535, 553, 563, 612, 642, 65
3, 655, 657, 663, 690, 744, 752, 760, 769, 778, 789, 796, 797, 802, 819, 834, 837, 841, 846, 852, 853, 863, 864, 873,
879, 895, 902, 911, 923, 928, 942, 961, 970, 976, 982, 995, 997, 999, 1001, 1015, 1060, 1073, 1076, 1099, 1102, 1117,
1123, 1146, 1147, 1154, 1157, 1158, 1159, 1163, 1181, 1202, 1205, 1216, 1260, 1272, 1279, 1303, 1343, 1356, 1378, 138
6, 1392, 1405, 1407, 1417, 1436, 1449, 1459, 1470, 1471, 1485, 1493, 1497, 1518, 1522, 1533, 1549, 1554, 1555, 1588,
1597, 1601, 1605, 1627, 1658, 1686, 1697, 1703, 1713, 1737, 1747, 1749, 1758, 1781, 1792, 1797, 1802, 1813, 1837, 186
2, 1867, 1889, 1896, 1922, 1930, 1935, 1945, 1949, 1961, 1968, 1977, 1983, 1995, 2017, 2019, 2025, 2030, 2043, 2056,
2064, 2069, 2070, 2096, 2113, 2119, 2126, 2136, 2140, 2141, 2154, 2176, 2177, 2184, 2193, 2218, 2219, 2223, 2231, 223
4, 2238, 2242, 2256, 2282, 2286, 2290, 2311, 2320, 2321, 2358, 2372, 2379, 2394, 2403, 2412, 2428, 2442, 2447, 2483,
2489, 2499, 2502, 2510, 2514, 2523, 2530, 2532, 2546, 2564, 2567, 2568, 2570, 2571, 2574, 2590, 2591, 2599, 2613, 262
9, 2631, 2646, 2652, 2668, 2710, 2711, 2723, 2735, 2737, 2743, 2752, 2763, 2775, 2778, 2790, 2794, 2811, 2828, 2842,
2850, 2855, 2866, 2987, 2998, 3019, 3024, 3038, 3039, 3048, 3076, 3081, 3112, 3180, 3181, 3190, 3202, 3215, 3219, 322
0, 3232, 3258, 3274, 3278, 3282, 3286, 3327, 3328, 3379, 3406, 3412, 3452, 3453, 3471, 3478, 3486, 3487, 3498, 3517,
3523, 3553, 3562, 3572, 3585, 3602, 3603, 3614, 3630, 3633, 3639, 3647, 3671, 3677, 3693, 3703, 3712, 3739, 3762, 376
4, 3769, 3777, 3822, 3829, 3845, 3850, 3854, 3882, 3883, 3905, 3912, 3923, 3946, 3950, 3983, 3986, 4025, 4037, 4046,
4050, 4072, 4085, 4086, 4088, 4099, 4105, 4120, 4128, 4138, 4141, 4142, 4146, 4160, 4223, 4252, 4255, 4281, 4297, 431
5, 4367, 4369, 4375, 4378, 4383, 4410, 4419, 4452, 4453, 4457, 4462, 4467, 4471, 4482, 4485, 4487, 4521, 4546, 4551,
4587, 4620, 4622, 4650, 4651, 4669, 4670, 4683, 4699, 4711, 4719, 4722, 4750, 4806, 4821, 4835, 4845, 4848, 4855, 485
9, 4861, 4867, 4882, 4897, 4909, 4918, 4930, 4935, 4941, 4948, 4964, 4982, 4984, 4985, 5005, 5036, 5042, 5062, 5079,
5088, 5104, 5127, 5139, 5143, 5147, 5149, 5153, 5163, 5179, 5181, 5182, 5184, 5189, 5190, 5193, 5201, 5218, 5225, 524
```

Ln: 10 Col: 0


Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
6, 28, 30, 31, 32, 33, 35, 36, 37, 38, 40, 41, 44, 45, 47, 48, 49, 50, 52, 53, 54, 58, 59, 60, 61, 62, 63, 64, 65, 67
, 68, 70, 71, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 96, 97, 98, 99, 100
, 101, 102, 103, 104, 105, 106, 107, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 125,
126, 127, 128, 129, 130, 131, 133, 134, 135, 136, 137, 138, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149, 150, 15
1, 152, 153, 154, 156, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 169, 170, 172, 173, 174, 175, 176, 177, 178,
179, 181, 182, 183, 184, 185, 186, 188, 189, 190, 192, 193, 194, 195, 196, 198, 199, 200, 201, 202, 204, 206, 207, 20
8, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 229, 230, 231, 232, 233,
234, 235, 236, 237, 238, 239, 240, 241, 243, 244, 245, 246, 247, 249, 250, 251, 252, 253, 255, 256, 257, 258, 260, 26
1, 262, 263, 265, 266, 267, 268, 269, 270, 272, 273, 274, 276, 277, 278, 279, 281, 282, 285, 286, 287, 289, 290, 291,
293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 312, 313, 314, 315, 316, 317, 31
9, 320, 322, 323, 324, 325, 327, 328, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 343, 344, 345, 348,
349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 360, 361, 363, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 37
6, 377, 378, 380, 382, 383, 384, 385, 386, 387, 389, 391, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 406, 407,
409, 410, 411, 412, 413, 414, 416, 417, 418, 419, 420, 421, 422, 424, 425, 428, 429, 430, 431, 432, 433, 434, 435, 43
6, 437, 438, 439, 441, 443, 444, 445, 446, 447, 449, 450, 451, 452, 453, 454, 455, 456, 457, 461, 462, 463, 464, 465,
466, 467, 468, 469, 470, 472, 473, 474, 475, 476, 478, 479, 481, 482, 484, 485, 486, 487, 488, 489, 490, 491, 493, 49
4, 496, 498, 499, 500, 501, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520,
522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 537, 538, 540, 541, 542, 543, 544, 548, 549, 550, 55
2, 553, 554, 555, 556, 557, 558, 560, 561, 562, 563, 564, 565, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 579,
580, 581, 582, 583, 585, 586, 588, 590, 591, 592, 593, 594, 595, 597, 598, 599, 600, 601, 602, 605, 606, 607, 608, 60
9, 610, 611, 612, 613, 614, 615, 617, 619, 620, 621, 622, 623, 624, 625, 627, 628, 629, 630, 631, 632, 633, 634, 635,
636, 637, 638, 641, 642, 644, 645, 646, 648, 649, 650, 651, 652, 653, 655, 656, 657, 658, 659, 660, 661, 663, 664, 66
5, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 685, 686, 687, 688, 689, 691,
692, 693, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 71
7, 718, 719, 720, 721, 722, 723, 725, 726, 727, 728, 729, 731, 732, 735, 736, 737, 738, 739, 740, 741, 742, 744, 745,
747, 748, 749, 750, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 763, 764, 765, 767, 768, 769, 770, 771, 772, 77
3, 775, 776, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 791, 792, 793, 794, 795, 797, 799, 801, 802, 805,
806, 807, 808, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 83
2, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 845, 848, 849, 850, 851, 852, 853, 855, 856, 857, 858, 859,
860, 862, 863, 864, 866, 867, 868, 869, 870, 874, 875, 876, 877, 878, 879, 881, 882, 883, 885, 886, 887, 888, 889, 89
0, 891, 892, 893, 895, 898, 899, 900, 901, 902, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917,
Ln: 8 Col: 5789
Type here to search
ENG
12:19
04-02-2021


Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
6081, 6093, 6126, 6132, 6135, 6139, 6153, 6183, 6198, 6199, 6204, 6208, 6221, 6230, 6234, 6267, 6279, 6287, 6304, 633
1, 6332, 6343, 6347, 6383, 6396, 6397, 6404, 6411, 6419, 6421, 6453, 6468, 6478, 6483, 6488, 6492, 6512, 6517, 6524,
6529, 6536, 6539, 6550, 6553, 6562, 6565, 6572, 6580, 6587, 6607, 6613, 6653, 6663, 6665, 6696, 6699, 6738, 6772, 677
3, 6782, 6783, 6791, 6809, 6818, 6836, 6876, 6881, 6887, 6900, 6905, 6962, 6989, 6994, 7033, 7046, 7050, 7054, 7059,
7078, 7082, 7083, 7085, 7086, 7092, 7093, 7101, 7109, 7117, 7122, 7140, 7144, 7160, 7166, 7172, 7177, 7208, 7214, 721
7, 7253, 7260, 7262, 7267, 7295, 7300, 7306, 7309, 7315, 7330, 7338, 7346, 7358, 7392, 7404, 7412, 7425, 7434, 7436,
7439, 7451, 7452, 7455, 7463, 7479, 7486, 7516, 7521, 7531, 7542, 7550, 7563, 7582, 7605, 7619, 7627, 7648, 7650, 767
2, 7681, 7695, 7696, 7726, 7734, 7747, 7748, 7753, 7762, 7772, 7788, 7789, 7793, 7843, 7857, 7862, 7873, 7874, 7891,
7925, 7938, 7943, 7944, 7945, 7961, 7973, 7981, 7984, 7986, 8000, 8007, 8015, 8020, 8022, 8043, 8076, 8088, 8093, 809
5, 8118, 8132, 8165, 8167, 8173, 8176, 8177, 8183, 8184, 8229, 8239, 8248, 8296, 8299, 8316, 8319, 8326, 8339, 8420,
8438, 8446, 8466, 8471, 8473, 8482, 8494, 8513, 8515, 8553, 8564, 8583, 8594, 8597, 8605, 8618, 8638, 8653, 8659, 867
3, 8677, 8693, 8698, 8706, 8713, 8741, 8742, 8754, 8759, 8780, 8782, 8785, 8794, 8798, 8804, 8806, 8815, 8822, 8832,
8841, 8847, 8851, 8866, 8870, 8879, 8885, 8895, 8896, 8901, 8906, 8909, 8939, 8956, 8976, 8978, 8982, 8993, 8996, 900
0, 9006, 9013, 9017, 9019, 9057, 9064, 9076, 9133, 9136, 9144, 9145, 9168, 9171, 9182, 9189, 9195, 9198, 9211, 9212,
9217, 9225, 9232, 9247, 9264, 9271, 9281, 9283, 9293, 9309, 9315, 9324, 9346, 9363, 9365, 9368, 9386, 9395, 9407, 942
0, 9425, 9450, 9461, 9476, 9478, 9487, 9509, 9528, 9544, 9551, 9563, 9599, 9601, 9635, 9639, 9651, 9655, 9677, 9681,
9687, 9697, 9700, 9705, 9719, 9740, 9744, 9748, 9761, 9764, 9779, 9789, 9808, 9810, 9829, 9843, 9844, 9856, 9870, 987
6, 9883, 9893, 9894, 9933, 9952, 9958, 9982, 10004, 10006, 10024, 10030, 10045, 10083, 10114, 10115, 10118, 10123, 10
153, 10163, 10165, 10169, 10175, 10181, 10188, 10194, 10196, 10198, 10222, 10238, 10239, 10249, 10259, 10281, 10300,
10301, 10337, 10339, 10368, 10400, 10412, 10415, 10418, 10420, 10425, 10426, 10433, 10449, 10451, 10456, 10471, 10476
, 10481, 10487, 10505, 10523, 10528, 10543, 10545, 10554, 10558, 10568, 10589, 10590, 10635, 10637, 10651, 10653, 106
68, 10671, 10672, 10678, 10686, 10707, 10715, 10720, 10728, 10735, 10737, 10741, 10742, 10748, 10751, 10756, 10759, 1
0760, 10767, 10768, 10791, 10794, 10798, 10855, 10894, 10896, 10919, 10925, 10926, 10927, 10936, 10978, 10979, 10980,
10985, 11011, 11029, 11037, 11051, 11053, 11054, 11073, 11076, 11104, 11108, 11110, 11118, 11134, 11139, 11154, 11163
, 11169, 11183, 11191, 11203, 11228, 11237, 11243, 11246, 11248, 11252, 11287, 11298, 11300, 11382, 11397, 11429, 114
40, 11460, 11470, 11476, 11501, 11503, 11508, 11509, 11516, 11537, 11547, 11557, 11560, 11570, 11582, 11586, 11607, 1
1614, 11624, 11643, 11662, 11673, 11683, 11695, 11709, 11713, 11719, 11723, 11737, 11771, 11775, 11780, 11785, 11789,
11796, 11799, 11821, 11823, 11824, 11833, 11847, 11861, 11877, 11884, 11891, 11904, 11907, 11913, 11922, 11929, 11933
, 11958, 11960, 11974, 11976, 11980, 11987]
Sorted array for input size 10000:
Squeezed text (760 lines).
Ln: 10 Col: 0
Type here to search
ENG
12:18
04-02-2021

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
1030, 1031, 1032, 1033, 1035, 1036, 1037, 1038, 1040, 1041, 1043, 1044, 1045, 1046, 1049, 1051, 1052, 1054, 1055, 105
8, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1069, 1070, 1071, 1073, 1074, 1075, 1076, 1079, 1081, 1082, 1083,
1084, 1086, 1087, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1106, 110
7, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1124, 1125, 1126, 1127,
1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1138, 1140, 1142, 1143, 1144, 1145, 1147, 1148, 1149, 1151, 115
2, 1154, 1156, 1157, 1159, 1160, 1161, 1162, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1177, 1178,
1180, 1181, 1182, 1183, 1184, 1186, 1187, 1188, 1189, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 120
1, 1202, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1221, 1223, 1224,
1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1241, 1242, 1243, 1244, 1245, 1246, 124
7, 1248, 1249, 1250, 1251, 1253, 1254, 1256, 1257, 1258, 1259, 1260, 1261, 1263, 1264, 1265, 1266, 1267, 1268, 1269,
1270, 1272, 1273, 1274, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1284, 1285, 1286, 1287, 1289, 1290, 1291, 1294, 129
5, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1314, 1315, 1316, 1317, 1318,
1319, 1320, 1321, 1322, 1323, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 134
0, 1341, 1342, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1352, 1353, 1355, 1356, 1358, 1359, 1360, 1361, 1362, 1363,
1364, 1366, 1367, 1369, 1370, 1372, 1373, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1385, 1386, 1387, 1388, 139
0, 1391, 1392, 1393, 1394, 1396, 1397, 1398, 1399, 1401, 1402, 1403, 1404, 1405, 1408, 1409, 1410, 1411, 1412, 1413,
1415, 1416, 1417, 1419, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1432, 1433, 1434, 1435, 1436, 143
7, 1439, 1440, 1441, 1442, 1444, 1445, 1446, 1447, 1448, 1449, 1451, 1452, 1453, 1455, 1456, 1458, 1460, 1461, 1462,
1463, 1464, 1465, 1466, 1467, 1468, 1469, 1471, 1472, 1475, 1476, 1477, 1478, 1479, 1480, 1484, 1485, 1486, 1487, 148
8, 1489, 1490, 1492, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1505, 1506, 1507, 1508, 1510, 1511,
1512, 1514, 1515, 1516, 1518, 1519, 1520, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1530, 1531, 1532, 1533, 1534, 153
5, 1536, 1537, 1538, 1540, 1541, 1542, 1543, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1556, 1557,
1560, 1561, 1562, 1563, 1564, 1565, 1566, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 158
0, 1581, 1583, 1584, 1585, 1586, 1587, 1588, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1600, 1601, 1602, 1604,
1605, 1606, 1607, 1608, 1609, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1621, 1622, 1623, 1624, 1625, 1626, 162
7, 1628, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1644, 1645, 1646, 1647, 1648,
1649, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1659, 1660, 1661, 1662, 1663, 1664, 1666, 1667, 1668, 1669, 1670, 167
1, 1672, 1675, 1676, 1679, 1680, 1681, 1682, 1683, 1685, 1686, 1687, 1688, 1689, 1690, 1692, 1693, 1694, 1695, 1696,
1697, 1699, 1700, 1701, 1703, 1704, 1706, 1707, 1708, 1709, 1710, 1711, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 172
0, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1737, 1738, 1739, 1740, 1742,
1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1753, 1756, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1766, 176
7, 1768, 1769, 1770, 1771, 1773, 1774, 1775, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788,
```

Ln: 8 Col: 5789

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1753, 1756, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1766, 176
7, 1768, 1769, 1770, 1771, 1773, 1774, 1775, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788,
1789, 1790, 1791, 1792, 1793, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1809, 181
1, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1833,
1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1848, 1849, 1850, 1851, 1852, 1854, 185
6, 1857, 1858, 1859, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876,
1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1895, 1896, 189
8, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1912, 1913, 1914, 1916, 1917, 1918, 1919,
1920, 1921, 1922, 1923, 1926, 1927, 1929, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1939, 1940, 1941, 1942, 1943, 194
4, 1945, 1946, 1947, 1949, 1950, 1951, 1952, 1953, 1954, 1956, 1957, 1959, 1960, 1961, 1962, 1963, 1966, 1968, 1969,
1970, 1971, 1972, 1973, 1974, 1975, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1989, 1993, 199
4, 1995, 1996, 1997, 1999, 2000, 2002, 2004, 2005, 2008, 2009, 2010, 2011, 2013, 2014, 2015, 2016, 2017, 2018, 2019,
2020, 2021, 2022, 2023, 2024, 2025, 2027, 2028, 2029, 2031, 2032, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2042, 204
3, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2065,
2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 208
6, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105,
2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2119, 2120, 2121, 2123, 2125, 2126, 2128, 212
9, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149,
2150, 2152, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 217
1, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2182, 2184, 2185, 2186, 2188, 2189, 2190, 2192, 2193, 2194,
2196, 2197, 2198, 2199, 2202, 2204, 2205, 2207, 2208, 2209, 2210, 2212, 2213, 2214, 2216, 2217, 2218, 2219, 2221, 222
3, 2225, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2245, 2246, 2247,
2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 227
1, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291,
2292, 2294, 2295, 2296, 2297, 2298, 2300, 2301, 2302, 2303, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2314, 231
5, 2317, 2318, 2319, 2320, 2321, 2323, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338,
2339, 2340, 2343, 2344, 2345, 2347, 2348, 2349, 2351, 2352, 2353, 2355, 2356, 2357, 2359, 2360, 2361, 2363, 2364, 236
5, 2366, 2367, 2368, 2369, 2370, 2372, 2373, 2374, 2376, 2378, 2381, 2384, 2385, 2387, 2388, 2389, 2391, 2392, 2393,
2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2403, 2404, 2406, 2407, 2409, 2411, 2412, 2413, 2414, 2416, 2417, 241
8, 2419, 2420, 2421, 2422, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438,
2439, 2440, 2441, 2443, 2444, 2445, 2446, 2447, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2459, 2460, 2461, 2462, 246
3, 2464, 2465, 2469, 2470, 2471, 2472, 2473, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2486, 2487, 2488
```

Ln: 8 Col: 5789

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
4082, 4083, 4084, 4086, 4087, 4088, 4089, 4090, 4091, 4092, 4093, 4094, 4095, 4097, 4098, 4099, 4100, 4101, 4102, 410
3, 4105, 4106, 4107, 4108, 4109, 4110, 4111, 4112, 4113, 4114, 4115, 4116, 4117, 4118, 4119, 4120, 4121, 4122, 4123,
4125, 4126, 4127, 4128, 4129, 4130, 4133, 4134, 4135, 4136, 4137, 4138, 4139, 4140, 4141, 4142, 4143, 4144, 4146, 414
7, 4149, 4150, 4151, 4152, 4153, 4154, 4155, 4156, 4157, 4158, 4159, 4160, 4161, 4162, 4163, 4164, 4165, 4166, 4167,
4168, 4169, 4170, 4172, 4174, 4175, 4176, 4177, 4178, 4181, 4184, 4185, 4186, 4190, 4191, 4192, 4194, 4195, 4196, 419
7, 4198, 4199, 4200, 4201, 4202, 4203, 4204, 4205, 4206, 4207, 4208, 4209, 4211, 4212, 4213, 4215, 4217, 4218, 4220,
4221, 4222, 4223, 4224, 4225, 4226, 4227, 4228, 4229, 4230, 4231, 4232, 4233, 4235, 4237, 4238, 4239, 4240, 4241, 424
3, 4244, 4245, 4246, 4247, 4248, 4249, 4250, 4251, 4252, 4253, 4254, 4255, 4256, 4258, 4259, 4263, 4264, 4265, 4266,
4267, 4268, 4270, 4272, 4273, 4274, 4275, 4276, 4277, 4278, 4279, 4281, 4282, 4283, 4284, 4287, 4289, 4290, 4291, 429
2, 4293, 4294, 4296, 4297, 4298, 4299, 4301, 4302, 4303, 4304, 4305, 4307, 4308, 4309, 4310, 4311, 4313, 4314, 4315,
4316, 4317, 4319, 4320, 4321, 4323, 4325, 4326, 4327, 4328, 4330, 4331, 4332, 4333, 4334, 4335, 4337, 4338, 4339, 434
0, 4341, 4342, 4343, 4344, 4345, 4346, 4347, 4348, 4349, 4350, 4352, 4354, 4356, 4357, 4358, 4359, 4360, 4361, 4362,
4363, 4364, 4366, 4367, 4368, 4369, 4370, 4371, 4372, 4373, 4374, 4375, 4377, 4378, 4381, 4382, 4383, 4384, 4385, 438
6, 4387, 4389, 4390, 4391, 4392, 4393, 4394, 4395, 4396, 4397, 4398, 4399, 4400, 4401, 4402, 4405, 4406, 4407, 4408,
4409, 4410, 4412, 4413, 4415, 4416, 4417, 4418, 4419, 4420, 4421, 4423, 4424, 4425, 4426, 4427, 4428, 4429, 4430, 443
1, 4433, 4434, 4435, 4436, 4437, 4438, 4439, 4440, 4441, 4443, 4444, 4445, 4446, 4448, 4449, 4450, 4451, 4452, 4454,
4456, 4457, 4460, 4461, 4462, 4463, 4464, 4466, 4467, 4469, 4470, 4471, 4473, 4474, 4475, 4476, 4477, 4478, 4479, 448
1, 4482, 4483, 4484, 4486, 4487, 4489, 4492, 4493, 4494, 4495, 4496, 4497, 4498, 4499, 4500, 4501, 4502, 4503, 4506,
4507, 4508, 4509, 4511, 4512, 4513, 4514, 4516, 4517, 4519, 4520, 4522, 4523, 4526, 4527, 4529, 4530, 4531, 4532, 453
4, 4535, 4536, 4539, 4540, 4541, 4542, 4543, 4544, 4546, 4547, 4548, 4549, 4550, 4551, 4552, 4553, 4554, 4555, 4556,
4558, 4559, 4562, 4563, 4564, 4565, 4566, 4567, 4568, 4569, 4571, 4572, 4573, 4574, 4576, 4577, 4578, 4579, 4580, 458
1, 4582, 4584, 4585, 4586, 4587, 4590, 4591, 4592, 4594, 4597, 4599, 4600, 4601, 4602, 4604, 4605, 4606, 4607, 4608,
4609, 4610, 4611, 4612, 4613, 4614, 4616, 4617, 4618, 4619, 4620, 4621, 4622, 4623, 4624, 4626, 4628, 4629, 4631, 463
2, 4633, 4634, 4635, 4636, 4637, 4640, 4642, 4643, 4644, 4645, 4646, 4647, 4648, 4649, 4650, 4651, 4652, 4653, 4655,
4656, 4657, 4658, 4659, 4661, 4662, 4663, 4664, 4665, 4666, 4667, 4668, 4669, 4670, 4671, 4672, 4674, 4675, 4676, 467
7, 4678, 4679, 4680, 4681, 4682, 4683, 4685, 4686, 4687, 4688, 4690, 4691, 4692, 4693, 4694, 4695, 4696, 4697, 4698,
4699, 4700, 4701, 4702, 4703, 4705, 4706, 4707, 4708, 4709, 4710, 4711, 4712, 4713, 4714, 4715, 4716, 4717, 4718, 471
9, 4720, 4721, 4722, 4724, 4725, 4726, 4728, 4730, 4731, 4732, 4733, 4734, 4735, 4736, 4737, 4738, 4740, 4742, 4743,
4745, 4747, 4748, 4749, 4750, 4751, 4752, 4754, 4755, 4756, 4757, 4758, 4759, 4760, 4761, 4762, 4763, 4765, 4766, 476
7, 4768, 4770, 4772, 4773, 4774, 4775, 4776, 4777, 4778, 4779, 4780, 4781, 4782, 4783, 4784, 4785, 4786, 4787, 4788,
4789, 4790, 4791, 4793, 4794, 4795, 4796, 4797, 4798, 4800, 4802, 4803, 4804, 4805, 4807, 4808, 4810, 4811, 4813, 481
4, 4816, 4817, 4818, 4819, 4820, 4821, 4822, 4823, 4824, 4825, 4826, 4827, 4828, 4829, 4830, 4831, 4832, 4833, 4834,
```

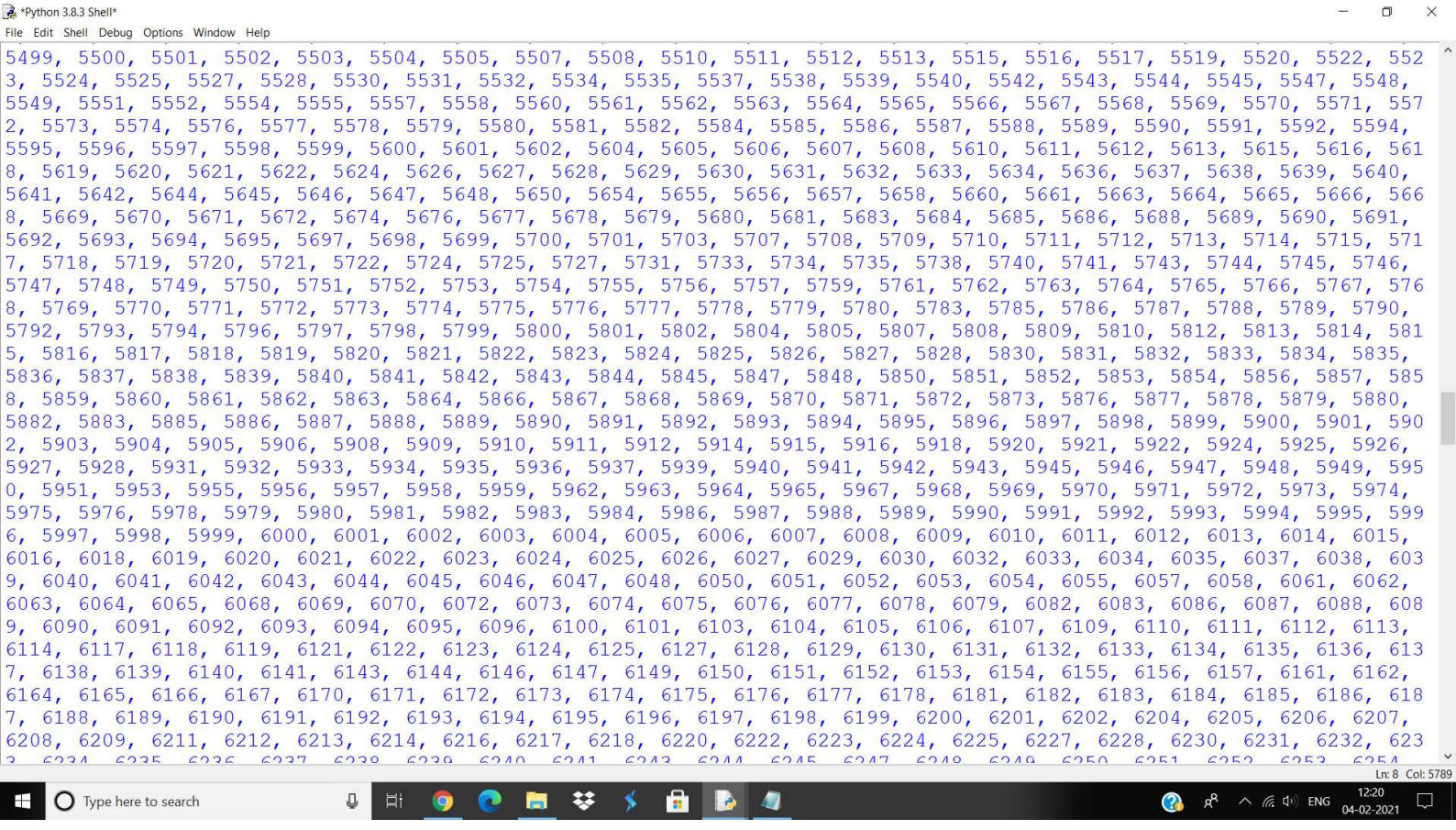
Ln: 8 Col: 5789

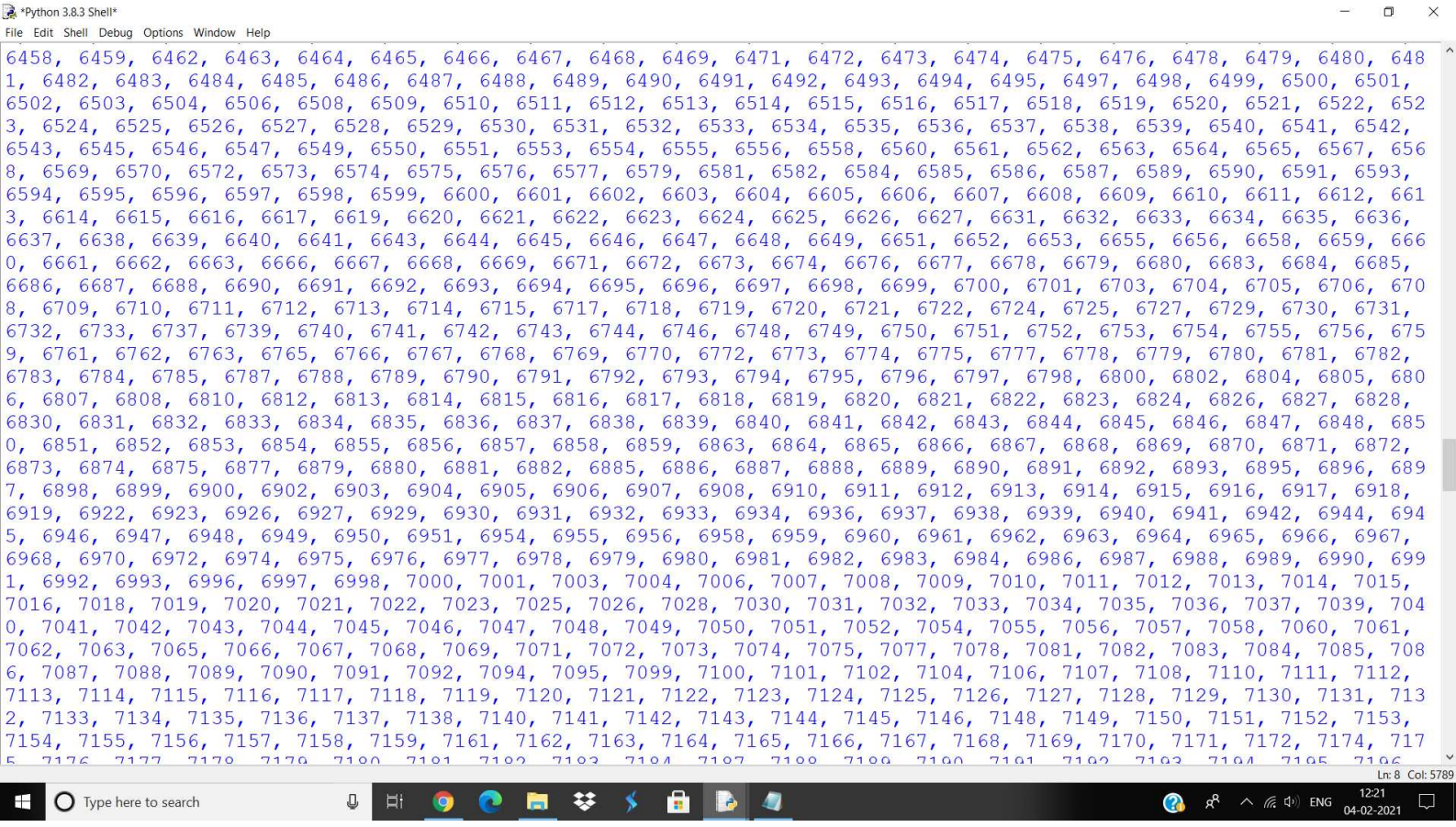
Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
8, 2559, 2560, 2561, 2562, 2564, 2565, 2566, 2567, 2568, 2569, 2571, 2572, 2573, 2574, 2575, 2577, 2578, 2579, 2581,
2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2593, 2594, 2595, 2596, 2598, 2600, 2602, 2604, 2606, 260
7, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2618, 2619, 2620, 2622, 2623, 2624, 2625, 2626, 2627, 2629, 2630, 2632,
2633, 2634, 2635, 2636, 2637, 2638, 2639, 2641, 2642, 2643, 2644, 2646, 2647, 2648, 2649, 2650, 2652, 2653, 2654, 265
5, 2656, 2657, 2658, 2659, 2660, 2661, 2663, 2664, 2666, 2667, 2669, 2670, 2671, 2672, 2673, 2674, 2676, 2677, 2679,
2680, 2681, 2682, 2683, 2684, 2685, 2687, 2688, 2689, 2690, 2691, 2692, 2693, 2694, 2695, 2696, 2697, 2698, 2699, 270
0, 2701, 2702, 2703, 2704, 2706, 2708, 2709, 2710, 2712, 2714, 2715, 2716, 2717, 2718, 2719, 2720, 2721, 2723, 2724,
2725, 2726, 2727, 2728, 2730, 2731, 2732, 2734, 2735, 2736, 2737, 2738, 2739, 2740, 2741, 2742, 2743, 2744, 2746, 274
7, 2748, 2750, 2752, 2753, 2754, 2755, 2756, 2757, 2758, 2759, 2760, 2761, 2762, 2763, 2764, 2765, 2766, 2767, 2768,
2769, 2770, 2771, 2774, 2775, 2776, 2777, 2778, 2779, 2780, 2782, 2783, 2784, 2785, 2786, 2788, 2789, 2790, 2791, 279
2, 2793, 2795, 2797, 2798, 2799, 2800, 2802, 2805, 2806, 2807, 2808, 2810, 2811, 2812, 2813, 2814, 2815, 2816, 2819,
2820, 2821, 2823, 2825, 2826, 2827, 2828, 2829, 2830, 2831, 2832, 2833, 2834, 2835, 2836, 2837, 2838, 2841, 2843, 284
4, 2845, 2846, 2847, 2849, 2850, 2851, 2852, 2853, 2855, 2856, 2857, 2858, 2859, 2860, 2861, 2862, 2863, 2864, 2866,
2867, 2868, 2869, 2870, 2871, 2872, 2873, 2874, 2875, 2876, 2877, 2879, 2880, 2881, 2882, 2883, 2884, 2885, 2886, 288
7, 2889, 2890, 2891, 2892, 2893, 2894, 2895, 2896, 2897, 2898, 2900, 2901, 2902, 2904, 2905, 2907, 2908, 2909, 2910,
2911, 2912, 2914, 2915, 2917, 2918, 2920, 2921, 2922, 2923, 2924, 2925, 2926, 2927, 2928, 2929, 2931, 2932, 2933, 293
4, 2935, 2937, 2938, 2939, 2940, 2941, 2942, 2943, 2944, 2945, 2946, 2947, 2948, 2949, 2950, 2952, 2953, 2954, 2955,
2956, 2957, 2959, 2960, 2962, 2963, 2964, 2965, 2968, 2970, 2971, 2972, 2973, 2974, 2976, 2977, 2978, 2979, 2981, 298
4, 2985, 2986, 2987, 2988, 2989, 2990, 2991, 2992, 2993, 2994, 2996, 2997, 2998, 2999, 3000, 3002, 3003, 3004, 3005,
3006, 3007, 3008, 3009, 3010, 3011, 3012, 3013, 3014, 3015, 3017, 3018, 3019, 3020, 3022, 3024, 3025, 3026, 3028, 302
9, 3030, 3031, 3032, 3033, 3034, 3035, 3036, 3037, 3038, 3039, 3040, 3041, 3042, 3045, 3046, 3047, 3048, 3050, 3051,
3052, 3053, 3054, 3055, 3056, 3057, 3059, 3060, 3061, 3062, 3063, 3065, 3066, 3068, 3069, 3070, 3071, 3072, 3074, 307
5, 3076, 3078, 3079, 3080, 3081, 3082, 3084, 3087, 3089, 3090, 3092, 3093, 3094, 3096, 3097, 3099, 3100, 3101, 3102,
3103, 3104, 3105, 3107, 3109, 3110, 3112, 3113, 3115, 3116, 3117, 3118, 3119, 3120, 3121, 3122, 3123, 3124, 3125, 312
6, 3127, 3129, 3130, 3131, 3133, 3134, 3135, 3136, 3137, 3138, 3139, 3142, 3145, 3146, 3147, 3148, 3150, 3151, 3152,
3153, 3154, 3155, 3156, 3157, 3158, 3159, 3160, 3161, 3163, 3164, 3165, 3166, 3168, 3169, 3170, 3171, 3172, 3174, 317
6, 3177, 3178, 3180, 3181, 3182, 3183, 3184, 3185, 3186, 3187, 3190, 3192, 3193, 3195, 3196, 3197, 3198, 3199, 3200,
3201, 3203, 3204, 3205, 3206, 3207, 3208, 3209, 3211, 3212, 3213, 3214, 3215, 3216, 3217, 3218, 3219, 3221, 3223, 322
4, 3225, 3226, 3227, 3228, 3229, 3230, 3231, 3232, 3233, 3234, 3236, 3237, 3238, 3239, 3240, 3242, 3243, 3244, 3245,
3246, 3248, 3249, 3251, 3252, 3253, 3255, 3256, 3257, 3259, 3260, 3261, 3262, 3265, 3266, 3267, 3268, 3269, 3270, 327
1, 3272, 3273, 3275, 3276, 3277, 3278, 3279, 3280, 3281, 3282, 3283, 3284, 3285, 3286, 3287, 3288, 3290, 3291, 3292,
3293, 3294, 3295, 3299, 3300, 3301, 3302, 3303, 3304, 3305, 3306, 3307, 3309, 3311, 3312, 3313, 3314, 3318, 3319, 332
```

Ln: 8 Col: 5789


Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
5499, 5500, 5501, 5502, 5503, 5504, 5505, 5507, 5508, 5510, 5511, 5512, 5513, 5515, 5516, 5517, 5519, 5520, 5522, 552
3, 5524, 5525, 5527, 5528, 5530, 5531, 5532, 5534, 5535, 5537, 5538, 5539, 5540, 5542, 5543, 5544, 5545, 5547, 5548,
5549, 5551, 5552, 5554, 5555, 5557, 5558, 5560, 5561, 5562, 5563, 5564, 5565, 5566, 5567, 5568, 5569, 5570, 5571, 557
2, 5573, 5574, 5576, 5577, 5578, 5579, 5580, 5581, 5582, 5584, 5585, 5586, 5587, 5588, 5589, 5590, 5591, 5592, 5594,
5595, 5596, 5597, 5598, 5599, 5600, 5601, 5602, 5604, 5605, 5606, 5607, 5608, 5610, 5611, 5612, 5613, 5615, 5616, 561
8, 5619, 5620, 5621, 5622, 5624, 5626, 5627, 5628, 5629, 5630, 5631, 5632, 5633, 5634, 5636, 5637, 5638, 5639, 5640,
5641, 5642, 5644, 5645, 5646, 5647, 5648, 5650, 5654, 5655, 5656, 5657, 5658, 5660, 5661, 5663, 5664, 5665, 5666, 566
8, 5669, 5670, 5671, 5672, 5674, 5676, 5677, 5678, 5679, 5680, 5681, 5683, 5684, 5685, 5686, 5688, 5689, 5690, 5691,
5692, 5693, 5694, 5695, 5697, 5698, 5699, 5700, 5701, 5703, 5707, 5708, 5709, 5710, 5711, 5712, 5713, 5714, 5715, 571
7, 5718, 5719, 5720, 5721, 5722, 5724, 5725, 5727, 5731, 5733, 5734, 5735, 5738, 5740, 5741, 5743, 5744, 5745, 5746,
5747, 5748, 5749, 5750, 5751, 5752, 5753, 5754, 5755, 5756, 5757, 5759, 5761, 5762, 5763, 5764, 5765, 5766, 5767, 576
8, 5769, 5770, 5771, 5772, 5773, 5774, 5775, 5776, 5777, 5778, 5779, 5780, 5783, 5785, 5786, 5787, 5788, 5789, 5790,
5792, 5793, 5794, 5796, 5797, 5798, 5799, 5800, 5801, 5802, 5804, 5805, 5807, 5808, 5809, 5810, 5812, 5813, 5814, 581
5, 5816, 5817, 5818, 5819, 5820, 5821, 5822, 5823, 5824, 5825, 5826, 5827, 5828, 5830, 5831, 5832, 5833, 5834, 5835,
5836, 5837, 5838, 5839, 5840, 5841, 5842, 5843, 5844, 5845, 5847, 5848, 5850, 5851, 5852, 5853, 5854, 5856, 5857, 585
8, 5859, 5860, 5861, 5862, 5863, 5864, 5866, 5867, 5868, 5869, 5870, 5871, 5872, 5873, 5876, 5877, 5878, 5879, 5880,
5882, 5883, 5885, 5886, 5887, 5888, 5889, 5890, 5891, 5892, 5893, 5894, 5895, 5896, 5897, 5898, 5899, 5900, 5901, 590
2, 5903, 5904, 5905, 5906, 5908, 5909, 5910, 5911, 5912, 5914, 5915, 5916, 5918, 5920, 5921, 5922, 5924, 5925, 5926,
5927, 5928, 5931, 5932, 5933, 5934, 5935, 5936, 5937, 5939, 5940, 5941, 5942, 5943, 5945, 5946, 5947, 5948, 5949, 595
0, 5951, 5953, 5955, 5956, 5957, 5958, 5959, 5962, 5963, 5964, 5965, 5967, 5968, 5969, 5970, 5971, 5972, 5973, 5974,
5975, 5976, 5978, 5979, 5980, 5981, 5982, 5983, 5984, 5986, 5987, 5988, 5989, 5990, 5991, 5992, 5993, 5994, 5995, 599
6, 5997, 5998, 5999, 6000, 6001, 6002, 6003, 6004, 6005, 6006, 6007, 6008, 6009, 6010, 6011, 6012, 6013, 6014, 6015,
6016, 6018, 6019, 6020, 6021, 6022, 6023, 6024, 6025, 6026, 6027, 6029, 6030, 6032, 6033, 6034, 6035, 6037, 6038, 603
9, 6040, 6041, 6042, 6043, 6044, 6045, 6046, 6047, 6048, 6050, 6051, 6052, 6053, 6054, 6055, 6057, 6058, 6061, 6062,
6063, 6064, 6065, 6068, 6069, 6070, 6072, 6073, 6074, 6075, 6076, 6077, 6078, 6079, 6082, 6083, 6086, 6087, 6088, 608
9, 6090, 6091, 6092, 6093, 6094, 6095, 6096, 6100, 6101, 6103, 6104, 6105, 6106, 6107, 6109, 6110, 6111, 6112, 6113,
6114, 6117, 6118, 6119, 6121, 6122, 6123, 6124, 6125, 6127, 6128, 6129, 6130, 6131, 6132, 6133, 6134, 6135, 6136, 613
7, 6138, 6139, 6140, 6141, 6143, 6144, 6146, 6147, 6149, 6150, 6151, 6152, 6153, 6154, 6155, 6156, 6157, 6161, 6162,
6164, 6165, 6166, 6167, 6170, 6171, 6172, 6173, 6174, 6175, 6176, 6177, 6178, 6181, 6182, 6183, 6184, 6185, 6186, 618
7, 6188, 6189, 6190, 6191, 6192, 6193, 6194, 6195, 6196, 6197, 6198, 6199, 6200, 6201, 6202, 6204, 6205, 6206, 6207,
6208, 6209, 6211, 6212, 6213, 6214, 6216, 6217, 6218, 6220, 6222, 6223, 6224, 6225, 6227, 6228, 6230, 6231, 6232, 623
Ln: 8 Col: 5789
Type here to search
ENG
12:20
04-02-2021

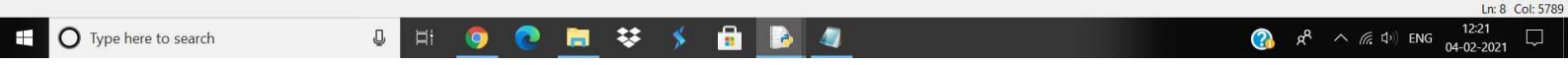

Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
6458, 6459, 6462, 6463, 6464, 6465, 6466, 6467, 6468, 6469, 6471, 6472, 6473, 6474, 6475, 6476, 6478, 6479, 6480, 648
1, 6482, 6483, 6484, 6485, 6486, 6487, 6488, 6489, 6490, 6491, 6492, 6493, 6494, 6495, 6497, 6498, 6499, 6500, 6501,
6502, 6503, 6504, 6506, 6508, 6509, 6510, 6511, 6512, 6513, 6514, 6515, 6516, 6517, 6518, 6519, 6520, 6521, 6522, 652
3, 6524, 6525, 6526, 6527, 6528, 6529, 6530, 6531, 6532, 6533, 6534, 6535, 6536, 6537, 6538, 6539, 6540, 6541, 6542,
6543, 6545, 6546, 6547, 6549, 6550, 6551, 6553, 6554, 6555, 6556, 6558, 6560, 6561, 6562, 6563, 6564, 6565, 6567, 656
8, 6569, 6570, 6572, 6573, 6574, 6575, 6576, 6577, 6579, 6581, 6582, 6584, 6585, 6586, 6587, 6589, 6590, 6591, 6593,
6594, 6595, 6596, 6597, 6598, 6599, 6600, 6601, 6602, 6603, 6604, 6605, 6606, 6607, 6608, 6609, 6610, 6611, 6612, 661
3, 6614, 6615, 6616, 6617, 6619, 6620, 6621, 6622, 6623, 6624, 6625, 6626, 6627, 6631, 6632, 6633, 6634, 6635, 6636,
6637, 6638, 6639, 6640, 6641, 6643, 6644, 6645, 6646, 6647, 6648, 6649, 6651, 6652, 6653, 6655, 6656, 6658, 6659, 666
0, 6661, 6662, 6663, 6666, 6667, 6668, 6669, 6671, 6672, 6673, 6674, 6676, 6677, 6678, 6679, 6680, 6683, 6684, 6685,
6686, 6687, 6688, 6690, 6691, 6692, 6693, 6694, 6695, 6696, 6697, 6698, 6699, 6700, 6701, 6703, 6704, 6705, 6706, 670
8, 6709, 6710, 6711, 6712, 6713, 6714, 6715, 6717, 6718, 6719, 6720, 6721, 6722, 6724, 6725, 6727, 6729, 6730, 6731,
6732, 6733, 6737, 6739, 6740, 6741, 6742, 6743, 6744, 6746, 6748, 6749, 6750, 6751, 6752, 6753, 6754, 6755, 6756, 675
9, 6761, 6762, 6763, 6765, 6766, 6767, 6768, 6769, 6770, 6772, 6773, 6774, 6775, 6777, 6778, 6779, 6780, 6781, 6782,
6783, 6784, 6785, 6787, 6788, 6789, 6790, 6791, 6792, 6793, 6794, 6795, 6796, 6797, 6798, 6800, 6802, 6804, 6805, 680
6, 6807, 6808, 6810, 6812, 6813, 6814, 6815, 6816, 6817, 6818, 6819, 6820, 6821, 6822, 6823, 6824, 6826, 6827, 6828,
6830, 6831, 6832, 6833, 6834, 6835, 6836, 6837, 6838, 6839, 6840, 6841, 6842, 6843, 6844, 6845, 6846, 6847, 6848, 685
0, 6851, 6852, 6853, 6854, 6855, 6856, 6857, 6858, 6859, 6863, 6864, 6865, 6866, 6867, 6868, 6869, 6870, 6871, 6872,
6873, 6874, 6875, 6877, 6879, 6880, 6881, 6882, 6885, 6886, 6887, 6888, 6889, 6890, 6891, 6892, 6893, 6895, 6896, 689
7, 6898, 6899, 6900, 6902, 6903, 6904, 6905, 6906, 6907, 6908, 6910, 6911, 6912, 6913, 6914, 6915, 6916, 6917, 6918,
6919, 6922, 6923, 6926, 6927, 6929, 6930, 6931, 6932, 6933, 6934, 6936, 6937, 6938, 6939, 6940, 6941, 6942, 6944, 694
5, 6946, 6947, 6948, 6949, 6950, 6951, 6954, 6955, 6956, 6958, 6959, 6960, 6961, 6962, 6963, 6964, 6965, 6966, 6967,
6968, 6970, 6972, 6974, 6975, 6976, 6977, 6978, 6979, 6980, 6981, 6982, 6983, 6984, 6986, 6987, 6988, 6989, 6990, 699
1, 6992, 6993, 6996, 6997, 6998, 7000, 7001, 7003, 7004, 7006, 7007, 7008, 7009, 7010, 7011, 7012, 7013, 7014, 7015,
7016, 7018, 7019, 7020, 7021, 7022, 7023, 7025, 7026, 7028, 7030, 7031, 7032, 7033, 7034, 7035, 7036, 7037, 7039, 704
0, 7041, 7042, 7043, 7044, 7045, 7046, 7047, 7048, 7049, 7050, 7051, 7052, 7054, 7055, 7056, 7057, 7058, 7060, 7061,
7062, 7063, 7065, 7066, 7067, 7068, 7069, 7071, 7072, 7073, 7074, 7075, 7077, 7078, 7081, 7082, 7083, 7084, 7085, 708
6, 7087, 7088, 7089, 7090, 7091, 7092, 7094, 7095, 7099, 7100, 7101, 7102, 7104, 7106, 7107, 7108, 7110, 7111, 7112,
7113, 7114, 7115, 7116, 7117, 7118, 7119, 7120, 7121, 7122, 7123, 7124, 7125, 7126, 7127, 7128, 7129, 7130, 7131, 713
2, 7133, 7134, 7135, 7136, 7137, 7138, 7140, 7141, 7142, 7143, 7144, 7145, 7146, 7148, 7149, 7150, 7151, 7152, 7153,
7154, 7155, 7156, 7157, 7158, 7159, 7161, 7162, 7163, 7164, 7165, 7166, 7167, 7168, 7169, 7170, 7171, 7172, 7174, 717
Ln: 8 Col: 5789
Type here to search
ENG
12:21
04-02-2021

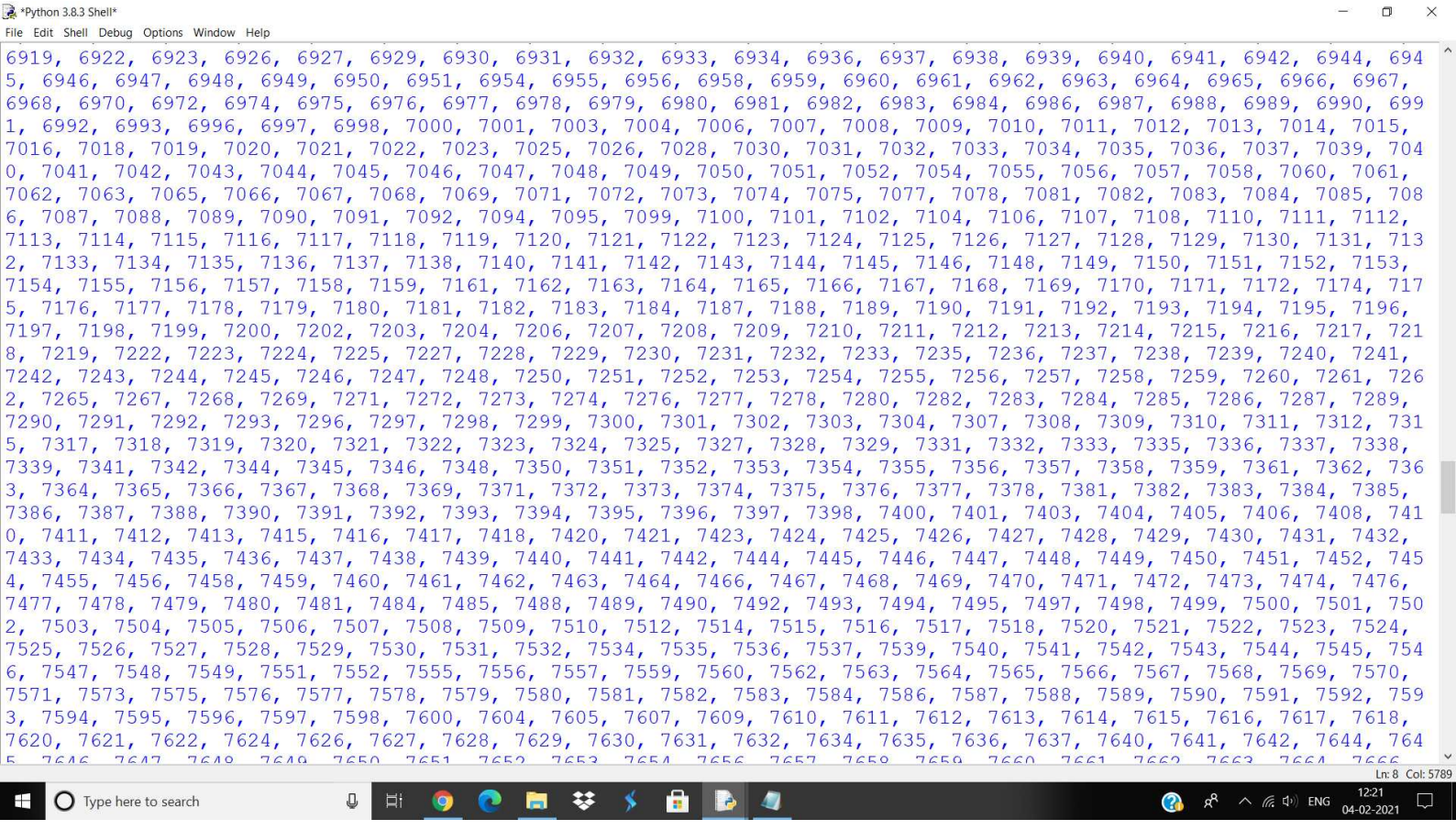
Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
2, 7744, 7745, 7746, 7747, 7748, 7750, 7751, 7752, 7753, 7754, 7758, 7759, 7760, 7761, 7762, 7763, 7764, 7765, 7766,
7768, 7769, 7770, 7773, 7774, 7775, 7776, 7777, 7779, 7780, 7781, 7782, 7783, 7784, 7785, 7786, 7788, 7789, 7790, 779
1, 7792, 7793, 7794, 7795, 7796, 7797, 7799, 7800, 7801, 7803, 7804, 7805, 7806, 7807, 7808, 7809, 7811, 7813, 7814,
7815, 7816, 7817, 7818, 7819, 7821, 7822, 7823, 7825, 7826, 7827, 7828, 7829, 7830, 7831, 7832, 7833, 7834, 7836, 783
7, 7838, 7840, 7841, 7842, 7844, 7845, 7846, 7847, 7848, 7849, 7850, 7852, 7853, 7854, 7855, 7858, 7859, 7860, 7861,
7862, 7863, 7864, 7865, 7866, 7868, 7869, 7870, 7871, 7873, 7875, 7878, 7879, 7880, 7881, 7882, 7883, 7884, 7885, 788
7, 7888, 7889, 7891, 7892, 7893, 7894, 7895, 7897, 7898, 7899, 7901, 7902, 7903, 7904, 7905, 7906, 7907, 7908, 7909,
7910, 7911, 7912, 7913, 7914, 7915, 7916, 7917, 7918, 7919, 7920, 7921, 7922, 7923, 7924, 7925, 7926, 7927, 7928, 792
9, 7930, 7931, 7933, 7934, 7935, 7936, 7937, 7939, 7940, 7941, 7943, 7944, 7945, 7947, 7948, 7949, 7950, 7951, 7952,
7953, 7954, 7955, 7956, 7957, 7958, 7959, 7960, 7961, 7962, 7963, 7964, 7965, 7966, 7967, 7968, 7969, 7970, 7971, 797
2, 7973, 7974, 7975, 7976, 7977, 7978, 7979, 7980, 7981, 7982, 7983, 7985, 7986, 7987, 7989, 7990, 7991, 7992, 7993,
7994, 7995, 7996, 7997, 7998, 7999, 8000, 8001, 8002, 8003, 8005, 8007, 8008, 8009, 8010, 8011, 8012, 8013, 8014, 801
5, 8016, 8017, 8018, 8019, 8020, 8021, 8022, 8023, 8024, 8025, 8027, 8028, 8029, 8030, 8031, 8032, 8033, 8034, 8036,
8037, 8038, 8039, 8040, 8041, 8042, 8045, 8046, 8047, 8048, 8049, 8050, 8051, 8052, 8053, 8054, 8055, 8056, 8057, 805
8, 8059, 8060, 8061, 8062, 8063, 8064, 8066, 8067, 8068, 8069, 8070, 8071, 8073, 8074, 8075, 8076, 8078, 8079, 8080,
8081, 8082, 8083, 8084, 8085, 8087, 8089, 8090, 8091, 8092, 8094, 8095, 8096, 8097, 8098, 8100, 8101, 8103, 8104, 810
5, 8106, 8107, 8108, 8109, 8110, 8112, 8113, 8114, 8117, 8118, 8119, 8120, 8121, 8122, 8124, 8125, 8127, 8128, 8129,
8131, 8132, 8133, 8134, 8135, 8136, 8137, 8138, 8139, 8140, 8141, 8142, 8144, 8145, 8146, 8148, 8149, 8151, 8152, 815
4, 8155, 8156, 8157, 8158, 8159, 8160, 8161, 8162, 8163, 8164, 8165, 8166, 8167, 8169, 8171, 8173, 8174, 8175, 8177,
8178, 8179, 8180, 8182, 8183, 8184, 8185, 8186, 8187, 8188, 8189, 8190, 8191, 8192, 8193, 8194, 8195, 8199, 8200, 820
1, 8202, 8203, 8204, 8205, 8207, 8210, 8212, 8213, 8215, 8216, 8217, 8219, 8220, 8221, 8222, 8223, 8224, 8226, 8227,
8228, 8229, 8230, 8231, 8232, 8233, 8234, 8235, 8236, 8238, 8239, 8240, 8241, 8243, 8244, 8245, 8246, 8247, 8248, 824
9, 8250, 8251, 8252, 8254, 8255, 8256, 8257, 8258, 8259, 8260, 8262, 8264, 8265, 8266, 8267, 8268, 8270, 8271, 8272,
8273, 8274, 8275, 8276, 8277, 8278, 8279, 8280, 8281, 8282, 8283, 8284, 8285, 8286, 8287, 8288, 8289, 8291, 8293, 829
5, 8296, 8297, 8298, 8299, 8301, 8302, 8304, 8305, 8306, 8307, 8308, 8310, 8313, 8314, 8315, 8317, 8318, 8319, 8321,
8322, 8323, 8325, 8326, 8330, 8331, 8332, 8333, 8334, 8335, 8336, 8337, 8339, 8340, 8341, 8342, 8343, 8345, 8346, 834
7, 8348, 8349, 8351, 8352, 8354, 8355, 8356, 8357, 8358, 8359, 8360, 8362, 8363, 8364, 8365, 8366, 8367, 8368, 8369,
8370, 8371, 8372, 8373, 8374, 8375, 8376, 8377, 8378, 8379, 8380, 8381, 8382, 8383, 8384, 8385, 8387, 8389, 8390, 839
1, 8392, 8393, 8394, 8395, 8396, 8397, 8398, 8399, 8400, 8402, 8405, 8406, 8407, 8408, 8409, 8410, 8412, 8413, 8414,
8415, 8416, 8419, 8420, 8421, 8422, 8423, 8425, 8426, 8427, 8429, 8431, 8432, 8433, 8435, 8436, 8437, 8440, 8442, 844
3, 8444, 8446, 8447, 8448, 8449, 8450, 8451, 8452, 8453, 8454, 8455, 8456, 8457, 8458, 8459, 8460, 8461, 8462, 8463,
8464, 8465, 8466, 8467, 8468, 8469, 8470, 8471, 8472, 8473, 8474, 8476, 8477, 8478, 8480, 8481, 8482, 8484, 8485, 848
```

Ln: 8 Col: 5789




Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
6919, 6922, 6923, 6926, 6927, 6929, 6930, 6931, 6932, 6933, 6934, 6936, 6937, 6938, 6939, 6940, 6941, 6942, 6944, 694
5, 6946, 6947, 6948, 6949, 6950, 6951, 6954, 6955, 6956, 6958, 6959, 6960, 6961, 6962, 6963, 6964, 6965, 6966, 6967,
6968, 6970, 6972, 6974, 6975, 6976, 6977, 6978, 6979, 6980, 6981, 6982, 6983, 6984, 6986, 6987, 6988, 6989, 6990, 699
1, 6992, 6993, 6996, 6997, 6998, 7000, 7001, 7003, 7004, 7006, 7007, 7008, 7009, 7010, 7011, 7012, 7013, 7014, 7015,
7016, 7018, 7019, 7020, 7021, 7022, 7023, 7025, 7026, 7028, 7030, 7031, 7032, 7033, 7034, 7035, 7036, 7037, 7039, 704
0, 7041, 7042, 7043, 7044, 7045, 7046, 7047, 7048, 7049, 7050, 7051, 7052, 7054, 7055, 7056, 7057, 7058, 7060, 7061,
7062, 7063, 7065, 7066, 7067, 7068, 7069, 7071, 7072, 7073, 7074, 7075, 7077, 7078, 7081, 7082, 7083, 7084, 7085, 708
6, 7087, 7088, 7089, 7090, 7091, 7092, 7094, 7095, 7099, 7100, 7101, 7102, 7104, 7106, 7107, 7108, 7110, 7111, 7112,
7113, 7114, 7115, 7116, 7117, 7118, 7119, 7120, 7121, 7122, 7123, 7124, 7125, 7126, 7127, 7128, 7129, 7130, 7131, 713
2, 7133, 7134, 7135, 7136, 7137, 7138, 7140, 7141, 7142, 7143, 7144, 7145, 7146, 7148, 7149, 7150, 7151, 7152, 7153,
7154, 7155, 7156, 7157, 7158, 7159, 7161, 7162, 7163, 7164, 7165, 7166, 7167, 7168, 7169, 7170, 7171, 7172, 7174, 717
5, 7176, 7177, 7178, 7179, 7180, 7181, 7182, 7183, 7184, 7187, 7188, 7189, 7190, 7191, 7192, 7193, 7194, 7195, 7196,
7197, 7198, 7199, 7200, 7202, 7203, 7204, 7206, 7207, 7208, 7209, 7210, 7211, 7212, 7213, 7214, 7215, 7216, 7217, 721
8, 7219, 7222, 7223, 7224, 7225, 7227, 7228, 7229, 7230, 7231, 7232, 7233, 7235, 7236, 7237, 7238, 7239, 7240, 7241,
7242, 7243, 7244, 7245, 7246, 7247, 7248, 7250, 7251, 7252, 7253, 7254, 7255, 7256, 7257, 7258, 7259, 7260, 7261, 726
2, 7265, 7267, 7268, 7269, 7271, 7272, 7273, 7274, 7276, 7277, 7278, 7280, 7282, 7283, 7284, 7285, 7286, 7287, 7289,
7290, 7291, 7292, 7293, 7296, 7297, 7298, 7299, 7300, 7301, 7302, 7303, 7304, 7307, 7308, 7309, 7310, 7311, 7312, 731
5, 7317, 7318, 7319, 7320, 7321, 7322, 7323, 7324, 7325, 7327, 7328, 7329, 7331, 7332, 7333, 7335, 7336, 7337, 7338,
7339, 7341, 7342, 7344, 7345, 7346, 7348, 7350, 7351, 7352, 7353, 7354, 7355, 7356, 7357, 7358, 7359, 7361, 7362, 736
3, 7364, 7365, 7366, 7367, 7368, 7369, 7371, 7372, 7373, 7374, 7375, 7376, 7377, 7378, 7381, 7382, 7383, 7384, 7385,
7386, 7387, 7388, 7390, 7391, 7392, 7393, 7394, 7395, 7396, 7397, 7398, 7400, 7401, 7403, 7404, 7405, 7406, 7408, 741
0, 7411, 7412, 7413, 7415, 7416, 7417, 7418, 7420, 7421, 7423, 7424, 7425, 7426, 7427, 7428, 7429, 7430, 7431, 7432,
7433, 7434, 7435, 7436, 7437, 7438, 7439, 7440, 7441, 7442, 7444, 7445, 7446, 7447, 7448, 7449, 7450, 7451, 7452, 745
4, 7455, 7456, 7458, 7459, 7460, 7461, 7462, 7463, 7464, 7466, 7467, 7468, 7469, 7470, 7471, 7472, 7473, 7474, 7476,
7477, 7478, 7479, 7480, 7481, 7484, 7485, 7488, 7489, 7490, 7492, 7493, 7494, 7495, 7497, 7498, 7499, 7500, 7501, 750
2, 7503, 7504, 7505, 7506, 7507, 7508, 7509, 7510, 7512, 7514, 7515, 7516, 7517, 7518, 7520, 7521, 7522, 7523, 7524,
7525, 7526, 7527, 7528, 7529, 7530, 7531, 7532, 7534, 7535, 7536, 7537, 7539, 7540, 7541, 7542, 7543, 7544, 7545, 754
6, 7547, 7548, 7549, 7551, 7552, 7555, 7556, 7557, 7559, 7560, 7562, 7563, 7564, 7565, 7566, 7567, 7568, 7569, 7570,
7571, 7573, 7575, 7576, 7577, 7578, 7579, 7580, 7581, 7582, 7583, 7584, 7586, 7587, 7588, 7589, 7590, 7591, 7592, 759
3, 7594, 7595, 7596, 7597, 7598, 7600, 7604, 7605, 7607, 7609, 7610, 7611, 7612, 7613, 7614, 7615, 7616, 7617, 7618,
7620, 7621, 7622, 7624, 7626, 7627, 7628, 7629, 7630, 7631, 7632, 7634, 7635, 7636, 7637, 7640, 7641, 7642, 7644, 764
Ln: 8 Col: 5789
Type here to search
ENG
12:21
04-02-2021

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
2, 8673, 8674, 8675, 8676, 8677, 8681, 8682, 8683, 8684, 8685, 8688, 8689, 8690, 8691, 8692, 8693, 8694, 8696, 8697,
8698, 8700, 8701, 8702, 8703, 8704, 8706, 8707, 8708, 8709, 8710, 8711, 8713, 8715, 8716, 8718, 8719, 8720, 8721, 872
2, 8724, 8725, 8726, 8727, 8728, 8729, 8730, 8731, 8732, 8733, 8734, 8735, 8738, 8742, 8743, 8744, 8745, 8746, 8747,
8748, 8749, 8750, 8751, 8752, 8753, 8754, 8755, 8756, 8757, 8758, 8759, 8760, 8761, 8762, 8763, 8764, 8766, 8767, 876
8, 8770, 8771, 8772, 8773, 8774, 8775, 8776, 8777, 8778, 8779, 8780, 8781, 8782, 8783, 8784, 8785, 8786, 8788, 8789,
8790, 8791, 8792, 8794, 8795, 8796, 8797, 8798, 8800, 8801, 8802, 8803, 8805, 8807, 8808, 8809, 8810, 8811, 8812, 881
4, 8815, 8816, 8817, 8819, 8820, 8821, 8822, 8824, 8825, 8826, 8827, 8829, 8830, 8832, 8833, 8834, 8835, 8836, 8837,
8838, 8839, 8840, 8841, 8842, 8843, 8844, 8845, 8847, 8848, 8849, 8850, 8851, 8852, 8854, 8855, 8856, 8857, 8858, 885
9, 8860, 8862, 8863, 8864, 8865, 8866, 8867, 8868, 8869, 8870, 8872, 8873, 8874, 8875, 8876, 8878, 8879, 8881, 8882,
8883, 8884, 8885, 8886, 8887, 8888, 8889, 8891, 8892, 8894, 8897, 8898, 8899, 8900, 8901, 8902, 8903, 8904, 8905, 890
7, 8908, 8909, 8910, 8911, 8912, 8913, 8914, 8915, 8916, 8917, 8918, 8920, 8921, 8922, 8923, 8925, 8927, 8928, 8929,
8932, 8933, 8934, 8935, 8936, 8937, 8938, 8939, 8940, 8941, 8942, 8943, 8945, 8946, 8947, 8948, 8949, 8950, 8951, 895
2, 8953, 8954, 8955, 8956, 8957, 8958, 8959, 8960, 8961, 8962, 8963, 8964, 8965, 8966, 8967, 8968, 8969, 8970, 8971,
8972, 8973, 8974, 8975, 8976, 8977, 8978, 8979, 8980, 8981, 8982, 8984, 8985, 8986, 8987, 8988, 8989, 8990, 8991, 899
2, 8993, 8994, 8995, 8996, 8998, 8999, 9000, 9001, 9003, 9004, 9005, 9006, 9008, 9010, 9011, 9012, 9013, 9015, 9016,
9017, 9018, 9019, 9020, 9021, 9023, 9024, 9025, 9026, 9027, 9028, 9030, 9031, 9032, 9033, 9034, 9036, 9037, 9038, 903
9, 9040, 9041, 9042, 9043, 9045, 9046, 9047, 9048, 9049, 9050, 9051, 9052, 9053, 9054, 9055, 9056, 9057, 9058, 9061,
9062, 9064, 9065, 9066, 9067, 9068, 9070, 9072, 9073, 9074, 9075, 9076, 9077, 9078, 9079, 9080, 9081, 9082, 9083, 908
4, 9085, 9086, 9087, 9088, 9089, 9090, 9091, 9094, 9095, 9098, 9099, 9100, 9101, 9102, 9103, 9104, 9105, 9106, 9107,
9109, 9110, 9111, 9112, 9113, 9114, 9116, 9117, 9118, 9119, 9120, 9122, 9123, 9124, 9125, 9127, 9128, 9129, 9130, 913
1, 9132, 9133, 9135, 9136, 9137, 9138, 9139, 9140, 9141, 9143, 9144, 9145, 9146, 9147, 9148, 9149, 9151, 9152, 9153,
9154, 9155, 9156, 9157, 9158, 9159, 9160, 9161, 9162, 9163, 9164, 9166, 9167, 9168, 9169, 9170, 9171, 9172, 9173, 917
5, 9176, 9177, 9178, 9179, 9180, 9181, 9183, 9184, 9186, 9187, 9188, 9189, 9191, 9192, 9193, 9194, 9195, 9196, 9198,
9199, 9200, 9201, 9202, 9203, 9204, 9205, 9206, 9207, 9210, 9211, 9212, 9213, 9214, 9216, 9217, 9218, 9219, 9220, 922
1, 9222, 9223, 9224, 9225, 9226, 9227, 9228, 9230, 9231, 9232, 9233, 9234, 9235, 9236, 9237, 9238, 9239, 9240, 9241,
9242, 9243, 9244, 9245, 9246, 9247, 9248, 9250, 9251, 9252, 9253, 9254, 9255, 9257, 9258, 9259, 9260, 9261, 9263, 926
4, 9265, 9266, 9267, 9268, 9270, 9271, 9272, 9273, 9275, 9276, 9277, 9278, 9280, 9281, 9282, 9283, 9284, 9285, 9287,
9288, 9289, 9290, 9291, 9292, 9293, 9294, 9295, 9296, 9297, 9298, 9299, 9301, 9302, 9306, 9307, 9308, 9310, 9311, 931
2, 9313, 9314, 9315, 9316, 9317, 9318, 9320, 9321, 9322, 9323, 9324, 9326, 9327, 9328, 9329, 9330, 9331, 9332, 9333,
9334, 9336, 9337, 9338, 9339, 9341, 9343, 9344, 9345, 9347, 9348, 9349, 9350, 9351, 9352, 9353, 9354, 9355, 9356, 935
7, 9359, 9360, 9361, 9362, 9363, 9364, 9365, 9366, 9367, 9368, 9370, 9371, 9372, 9374, 9376, 9377, 9378, 9379, 9380,
9382, 9383, 9384, 9385, 9386, 9387, 9388, 9389, 9390, 9392, 9393, 9394, 9395, 9397, 9398, 9401, 9402, 9403, 9404, 940
```

Ln: 8 Col: 5789

Type here to search

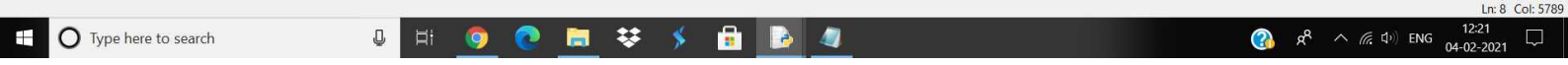
ENG 12:21 04-02-2021

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
6, 9827, 9828, 9829, 9830, 9831, 9832, 9833, 9834, 9835, 9836, 9837, 9838, 9839, 9841, 9842, 9843, 9844, 9845, 9846,
9847, 9848, 9849, 9850, 9851, 9852, 9853, 9854, 9855, 9856, 9857, 9858, 9860, 9861, 9862, 9863, 9864, 9865, 9866, 986
8, 9869, 9871, 9872, 9873, 9874, 9875, 9876, 9877, 9878, 9879, 9880, 9881, 9882, 9883, 9884, 9885, 9886, 9887, 9889,
9891, 9892, 9893, 9894, 9895, 9896, 9897, 9898, 9900, 9901, 9902, 9903, 9904, 9905, 9906, 9907, 9908, 9909, 9910, 991
1, 9913, 9914, 9915, 9916, 9918, 9919, 9921, 9922, 9924, 9925, 9926, 9927, 9929, 9930, 9931, 9932, 9933, 9934, 9935,
9936, 9937, 9938, 9939, 9941, 9942, 9943, 9944, 9948, 9949, 9950, 9951, 9953, 9955, 9956, 9957, 9958, 9959, 9961, 996
2, 9963, 9964, 9966, 9968, 9969, 9971, 9972, 9973, 9974, 9975, 9977, 9978, 9979, 9980, 9982, 9983, 9984, 9985, 9987,
9988, 9989, 9990, 9991, 9992, 9993, 9994, 9995, 9996, 9997, 9998, 10000, 10001, 10002, 10003, 10004, 10005, 10006, 10
007, 10008, 10009, 10010, 10011, 10012, 10013, 10014, 10015, 10016, 10017, 10018, 10019, 10020, 10021, 10022, 10023,
10024, 10025, 10026, 10027, 10028, 10029, 10030, 10031, 10033, 10034, 10035, 10036, 10037, 10038, 10039, 10040, 10041
, 10042, 10043, 10044, 10045, 10046, 10047, 10048, 10049, 10050, 10051, 10053, 10054, 10055, 10056, 10058, 10059, 100
60, 10061, 10062, 10063, 10064, 10065, 10066, 10068, 10069, 10070, 10071, 10072, 10073, 10074, 10075, 10076, 10077, 1
0078, 10079, 10081, 10082, 10083, 10085, 10086, 10087, 10088, 10089, 10090, 10091, 10092, 10093, 10094, 10095, 10096,
10097, 10098, 10099, 10100, 10101, 10102, 10103, 10105, 10106, 10107, 10108, 10109, 10110, 10112, 10113, 10114, 10115
, 10116, 10118, 10119, 10120, 10121, 10122, 10123, 10124, 10125, 10126, 10127, 10128, 10129, 10130, 10131, 10132, 101
33, 10134, 10135, 10136, 10137, 10138, 10139, 10140, 10141, 10143, 10144, 10145, 10146, 10147, 10148, 10149, 10150, 1
0151, 10153, 10154, 10155, 10156, 10158, 10159, 10160, 10161, 10162, 10164, 10165, 10166, 10167, 10169, 10170, 10171,
10172, 10173, 10174, 10175, 10177, 10178, 10179, 10180, 10181, 10182, 10183, 10185, 10186, 10188, 10189, 10190, 10191
, 10192, 10193, 10195, 10196, 10197, 10198, 10200, 10201, 10203, 10204, 10205, 10206, 10207, 10209, 10210, 10213, 102
14, 10215, 10217, 10218, 10219, 10220, 10221, 10222, 10223, 10225, 10226, 10227, 10228, 10229, 10230, 10231, 10232, 1
0233, 10234, 10235, 10237, 10239, 10240, 10241, 10243, 10244, 10245, 10246, 10247, 10248, 10250, 10251, 10252, 10254,
10255, 10256, 10257, 10258, 10259, 10260, 10262, 10263, 10264, 10265, 10266, 10267, 10269, 10270, 10271, 10272, 10273
, 10275, 10276, 10277, 10278, 10279, 10280, 10281, 10282, 10283, 10285, 10286, 10287, 10289, 10290, 10291, 10292, 102
93, 10296, 10297, 10298, 10300, 10301, 10302, 10303, 10304, 10305, 10306, 10307, 10309, 10310, 10311, 10312, 10313, 1
0314, 10315, 10316, 10317, 10318, 10319, 10321, 10322, 10323, 10324, 10325, 10326, 10327, 10328, 10329, 10330, 10331,
10332, 10333, 10334, 10336, 10338, 10339, 10341, 10342, 10343, 10344, 10346, 10347, 10349, 10350, 10351, 10352, 10353
, 10355, 10357, 10358, 10359, 10360, 10361, 10362, 10363, 10364, 10365, 10366, 10367, 10368, 10371, 10372, 10373, 103
75, 10376, 10377, 10378, 10379, 10382, 10383, 10384, 10385, 10386, 10387, 10388, 10389, 10390, 10391, 10392, 10393, 1
0395, 10396, 10397, 10398, 10399, 10400, 10401, 10402, 10403, 10404, 10405, 10406, 10408, 10409, 10410, 10411, 10412,
10413, 10414, 10415, 10416, 10417, 10418, 10419, 10420, 10421, 10422, 10423, 10424, 10425, 10426, 10427, 10428, 10429
, 10430, 10433, 10434, 10436, 10438, 10440, 10441, 10442, 10443, 10444, 10445, 10446, 10447, 10450, 10451, 10452, 104
53, 10454, 10455, 10456, 10457, 10458, 10460, 10461, 10462, 10464, 10465, 10466, 10467, 10468, 10469, 10471, 10472, 1
```

Ln: 8 Col: 5789

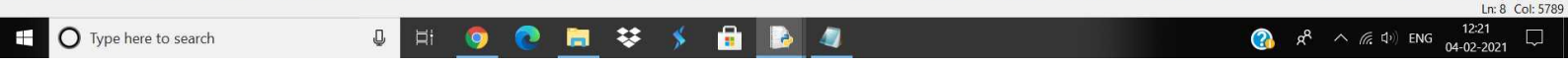


Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
9242, 9243, 9244, 9245, 9246, 9247, 9248, 9250, 9251, 9252, 9253, 9254, 9255, 9257, 9258, 9259, 9260, 9261, 9263, 926
4, 9265, 9266, 9267, 9268, 9270, 9271, 9272, 9273, 9275, 9276, 9277, 9278, 9280, 9281, 9282, 9283, 9284, 9285, 9287,
9288, 9289, 9290, 9291, 9292, 9293, 9294, 9295, 9296, 9297, 9298, 9299, 9301, 9302, 9306, 9307, 9308, 9310, 9311, 931
2, 9313, 9314, 9315, 9316, 9317, 9318, 9320, 9321, 9322, 9323, 9324, 9326, 9327, 9328, 9329, 9330, 9331, 9332, 9333,
9334, 9336, 9337, 9338, 9339, 9341, 9343, 9344, 9345, 9347, 9348, 9349, 9350, 9351, 9352, 9353, 9354, 9355, 9356, 935
7, 9359, 9360, 9361, 9362, 9363, 9364, 9365, 9366, 9367, 9368, 9370, 9371, 9372, 9374, 9376, 9377, 9378, 9379, 9380,
9382, 9383, 9384, 9385, 9386, 9387, 9388, 9389, 9390, 9392, 9393, 9394, 9395, 9397, 9398, 9401, 9402, 9403, 9404, 940
6, 9407, 9408, 9409, 9411, 9412, 9413, 9414, 9415, 9416, 9418, 9419, 9420, 9422, 9423, 9424, 9427, 9428, 9429, 9430,
9431, 9433, 9434, 9435, 9436, 9437, 9438, 9439, 9440, 9441, 9442, 9443, 9445, 9446, 9447, 9448, 9449, 9452, 9453, 945
4, 9457, 9458, 9459, 9461, 9462, 9463, 9464, 9465, 9466, 9467, 9468, 9469, 9470, 9471, 9472, 9473, 9474, 9475, 9476,
9477, 9478, 9479, 9480, 9482, 9483, 9484, 9485, 9486, 9487, 9488, 9489, 9491, 9492, 9494, 9495, 9496, 9497, 9498, 949
9, 9500, 9501, 9503, 9504, 9505, 9508, 9509, 9510, 9511, 9515, 9516, 9517, 9518, 9519, 9520, 9521, 9522, 9523, 9524,
9525, 9527, 9528, 9529, 9530, 9531, 9532, 9533, 9534, 9535, 9536, 9537, 9538, 9539, 9540, 9541, 9542, 9544, 9545, 954
6, 9547, 9548, 9549, 9550, 9551, 9552, 9553, 9555, 9558, 9559, 9561, 9564, 9565, 9566, 9568, 9569, 9570, 9571, 9572,
9575, 9576, 9578, 9579, 9580, 9582, 9583, 9585, 9586, 9587, 9588, 9589, 9590, 9591, 9592, 9593, 9594, 9595, 9596, 959
7, 9599, 9600, 9601, 9602, 9603, 9604, 9605, 9606, 9607, 9608, 9609, 9610, 9611, 9612, 9613, 9614, 9615, 9616, 9618,
9619, 9620, 9621, 9622, 9623, 9624, 9626, 9627, 9628, 9630, 9631, 9632, 9633, 9634, 9635, 9636, 9639, 9640, 9641, 964
2, 9643, 9644, 9646, 9647, 9648, 9649, 9650, 9651, 9652, 9653, 9654, 9655, 9656, 9657, 9658, 9659, 9661, 9662, 9663,
9664, 9665, 9666, 9667, 9668, 9669, 9670, 9671, 9673, 9674, 9675, 9676, 9677, 9678, 9679, 9680, 9681, 9682, 9683, 968
5, 9686, 9687, 9688, 9690, 9691, 9693, 9694, 9695, 9697, 9698, 9699, 9701, 9702, 9704, 9706, 9707, 9708, 9710, 9711,
9712, 9713, 9714, 9715, 9716, 9717, 9718, 9720, 9721, 9722, 9723, 9725, 9727, 9728, 9729, 9731, 9732, 9733, 9734, 973
6, 9737, 9738, 9740, 9741, 9742, 9743, 9744, 9745, 9746, 9747, 9748, 9749, 9751, 9753, 9755, 9756, 9757, 9759, 9760,
9761, 9762, 9763, 9764, 9765, 9766, 9767, 9768, 9769, 9770, 9771, 9772, 9773, 9774, 9775, 9776, 9778, 9779, 9780, 978
1, 9782, 9783, 9784, 9785, 9786, 9787, 9788, 9790, 9791, 9792, 9793, 9794, 9795, 9796, 9797, 9798, 9799, 9800, 9801,
9802, 9805, 9806, 9807, 9809, 9810, 9811, 9812, 9813, 9814, 9815, 9816, 9817, 9819, 9820, 9822, 9823, 9824, 9825, 982
6, 9827, 9828, 9829, 9830, 9831, 9832, 9833, 9834, 9835, 9836, 9837, 9838, 9839, 9841, 9842, 9843, 9844, 9845, 9846,
9847, 9848, 9849, 9850, 9851, 9852, 9853, 9854, 9855, 9856, 9857, 9858, 9860, 9861, 9862, 9863, 9864, 9865, 9866, 986
8, 9869, 9871, 9872, 9873, 9874, 9875, 9876, 9877, 9878, 9879, 9880, 9881, 9882, 9883, 9884, 9885, 9886, 9887, 9889,
9891, 9892, 9893, 9894, 9895, 9896, 9897, 9898, 9900, 9901, 9902, 9903, 9904, 9905, 9906, 9907, 9908, 9909, 9910, 991
1, 9913, 9914, 9915, 9916, 9918, 9919, 9921, 9922, 9924, 9925, 9926, 9927, 9929, 9930, 9931, 9932, 9933, 9934, 9935,
9936, 9937, 9938, 9939, 9941, 9942, 9943, 9944, 9948, 9949, 9950, 9951, 9953, 9955, 9956, 9957, 9958, 9959, 9961, 996
2, 9963, 9964, 9966, 9968, 9969, 9971, 9972, 9973, 9974, 9975, 9977, 9978, 9979, 9980, 9982, 9983, 9984, 9985, 9987
```

Ln: 8 Col: 5789

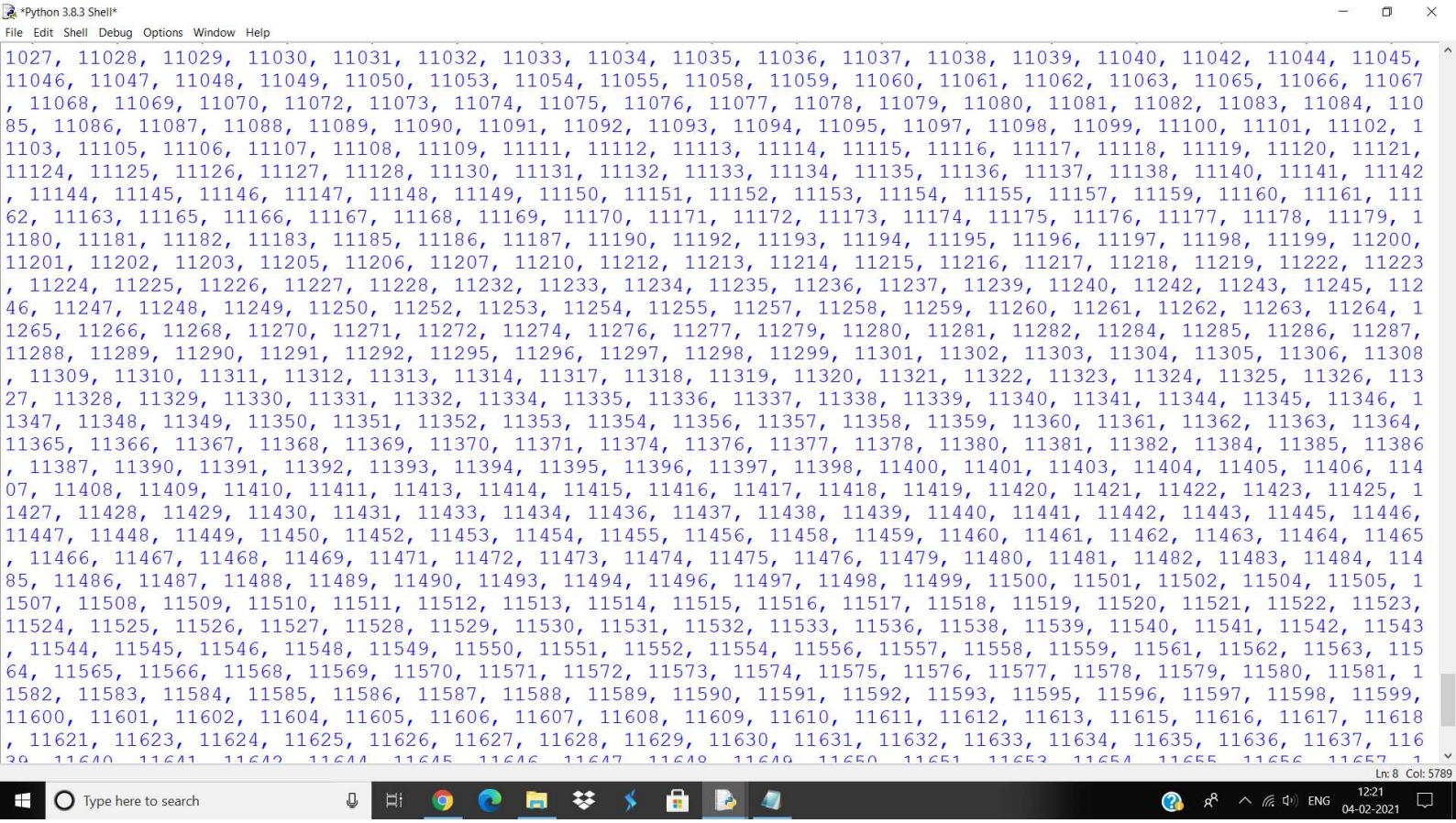


Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
0635, 10636, 10637, 10638, 10639, 10641, 10642, 10644, 10645, 10646, 10647, 10648, 10649, 10650, 10652, 10653, 10654,
10655, 10656, 10657, 10658, 10659, 10660, 10662, 10663, 10664, 10665, 10666, 10667, 10668, 10669, 10672, 10673, 10674
, 10675, 10676, 10677, 10678, 10679, 10680, 10681, 10682, 10683, 10684, 10685, 10688, 10689, 10690, 10691, 10692, 106
93, 10694, 10695, 10697, 10698, 10699, 10700, 10701, 10702, 10703, 10705, 10706, 10707, 10708, 10709, 10710, 10711, 1
0712, 10713, 10714, 10715, 10716, 10717, 10718, 10719, 10720, 10721, 10722, 10723, 10724, 10725, 10726, 10727, 10728,
10729, 10731, 10732, 10733, 10734, 10735, 10736, 10737, 10738, 10739, 10740, 10741, 10742, 10743, 10745, 10746, 10747
, 10748, 10750, 10751, 10755, 10756, 10757, 10758, 10759, 10760, 10764, 10765, 10766, 10767, 10768, 10769, 10770, 107
71, 10772, 10774, 10775, 10776, 10777, 10778, 10779, 10780, 10781, 10782, 10784, 10785, 10786, 10787, 10788, 10789, 1
0790, 10793, 10794, 10795, 10796, 10798, 10799, 10802, 10803, 10804, 10806, 10807, 10808, 10809, 10810, 10811, 10812,
10813, 10814, 10815, 10816, 10818, 10819, 10821, 10822, 10823, 10824, 10825, 10826, 10827, 10828, 10829, 10830, 10831
, 10832, 10833, 10836, 10838, 10839, 10840, 10841, 10842, 10843, 10844, 10845, 10846, 10847, 10848, 10849, 10850, 108
51, 10852, 10853, 10854, 10855, 10856, 10857, 10858, 10859, 10860, 10862, 10863, 10864, 10865, 10866, 10867, 10868, 1
0869, 10870, 10871, 10872, 10873, 10875, 10876, 10877, 10879, 10880, 10881, 10882, 10883, 10884, 10885, 10886, 10887,
10888, 10889, 10890, 10891, 10892, 10893, 10894, 10895, 10897, 10899, 10900, 10901, 10902, 10903, 10904, 10905, 10906
, 10907, 10908, 10910, 10912, 10913, 10914, 10917, 10918, 10919, 10920, 10921, 10922, 10923, 10924, 10925, 10927, 109
30, 10931, 10932, 10934, 10935, 10936, 10937, 10938, 10939, 10940, 10941, 10942, 10943, 10944, 10945, 10946, 10948, 1
0949, 10950, 10951, 10952, 10953, 10954, 10955, 10956, 10957, 10958, 10959, 10960, 10961, 10962, 10963, 10964, 10966,
10967, 10968, 10969, 10970, 10971, 10972, 10973, 10974, 10975, 10976, 10978, 10979, 10980, 10985, 10986, 10987, 10988
, 10989, 10990, 10991, 10992, 10993, 10994, 10995, 10996, 10998, 10999, 11000, 11001, 11002, 11003, 11004, 11005, 110
06, 11007, 11008, 11009, 11010, 11011, 11012, 11015, 11016, 11017, 11018, 11019, 11020, 11022, 11024, 11025, 11026, 1
1027, 11028, 11029, 11030, 11031, 11032, 11033, 11034, 11035, 11036, 11037, 11038, 11039, 11040, 11042, 11044, 11045,
11046, 11047, 11048, 11049, 11050, 11053, 11054, 11055, 11058, 11059, 11060, 11061, 11062, 11063, 11065, 11066, 11067
, 11068, 11069, 11070, 11072, 11073, 11074, 11075, 11076, 11077, 11078, 11079, 11080, 11081, 11082, 11083, 11084, 110
85, 11086, 11087, 11088, 11089, 11090, 11091, 11092, 11093, 11094, 11095, 11097, 11098, 11099, 11100, 11101, 11102, 1
1103, 11105, 11106, 11107, 11108, 11109, 11111, 11112, 11113, 11114, 11115, 11116, 11117, 11118, 11119, 11120, 11121,
11124, 11125, 11126, 11127, 11128, 11130, 11131, 11132, 11133, 11134, 11135, 11136, 11137, 11138, 11140, 11141, 11142
, 11144, 11145, 11146, 11147, 11148, 11149, 11150, 11151, 11152, 11153, 11154, 11155, 11157, 11159, 11160, 11161, 111
62, 11163, 11165, 11166, 11167, 11168, 11169, 11170, 11171, 11172, 11173, 11174, 11175, 11176, 11177, 11178, 11179, 1
1180, 11181, 11182, 11183, 11185, 11186, 11187, 11190, 11192, 11193, 11194, 11195, 11196, 11197, 11198, 11199, 11200,
11201, 11202, 11203, 11205, 11206, 11207, 11210, 11212, 11213, 11214, 11215, 11216, 11217, 11218, 11219, 11222, 11223
, 11224, 11225, 11226, 11227, 11228, 11232, 11233, 11234, 11235, 11236, 11237, 11239, 11240, 11242, 11243, 11245, 112
46, 11247, 11248, 11249, 11250, 11252, 11253, 11254, 11255, 11257, 11258, 11259, 11260, 11261, 11262, 11263, 11264, 1
```

Ln: 8 Col: 5789


Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
1027, 11028, 11029, 11030, 11031, 11032, 11033, 11034, 11035, 11036, 11037, 11038, 11039, 11040, 11042, 11044, 11045,
11046, 11047, 11048, 11049, 11050, 11053, 11054, 11055, 11058, 11059, 11060, 11061, 11062, 11063, 11065, 11066, 11067
, 11068, 11069, 11070, 11072, 11073, 11074, 11075, 11076, 11077, 11078, 11079, 11080, 11081, 11082, 11083, 11084, 110
85, 11086, 11087, 11088, 11089, 11090, 11091, 11092, 11093, 11094, 11095, 11097, 11098, 11099, 11100, 11101, 11102, 1
1103, 11105, 11106, 11107, 11108, 11109, 11111, 11112, 11113, 11114, 11115, 11116, 11117, 11118, 11119, 11120, 11121,
11124, 11125, 11126, 11127, 11128, 11130, 11131, 11132, 11133, 11134, 11135, 11136, 11137, 11138, 11140, 11141, 11142
, 11144, 11145, 11146, 11147, 11148, 11149, 11150, 11151, 11152, 11153, 11154, 11155, 11157, 11159, 11160, 11161, 111
62, 11163, 11165, 11166, 11167, 11168, 11169, 11170, 11171, 11172, 11173, 11174, 11175, 11176, 11177, 11178, 11179, 1
1180, 11181, 11182, 11183, 11185, 11186, 11187, 11190, 11192, 11193, 11194, 11195, 11196, 11197, 11198, 11199, 11200,
11201, 11202, 11203, 11205, 11206, 11207, 11210, 11212, 11213, 11214, 11215, 11216, 11217, 11218, 11219, 11222, 11223
, 11224, 11225, 11226, 11227, 11228, 11232, 11233, 11234, 11235, 11236, 11237, 11239, 11240, 11242, 11243, 11245, 112
46, 11247, 11248, 11249, 11250, 11252, 11253, 11254, 11255, 11257, 11258, 11259, 11260, 11261, 11262, 11263, 11264, 1
1265, 11266, 11268, 11270, 11271, 11272, 11274, 11276, 11277, 11279, 11280, 11281, 11282, 11284, 11285, 11286, 11287,
11288, 11289, 11290, 11291, 11292, 11295, 11296, 11297, 11298, 11299, 11301, 11302, 11303, 11304, 11305, 11306, 11308
, 11309, 11310, 11311, 11312, 11313, 11314, 11317, 11318, 11319, 11320, 11321, 11322, 11323, 11324, 11325, 11326, 113
27, 11328, 11329, 11330, 11331, 11332, 11334, 11335, 11336, 11337, 11338, 11339, 11340, 11341, 11344, 11345, 11346, 1
1347, 11348, 11349, 11350, 11351, 11352, 11353, 11354, 11356, 11357, 11358, 11359, 11360, 11361, 11362, 11363, 11364,
11365, 11366, 11367, 11368, 11369, 11370, 11371, 11374, 11376, 11377, 11378, 11380, 11381, 11382, 11384, 11385, 11386
, 11387, 11390, 11391, 11392, 11393, 11394, 11395, 11396, 11397, 11398, 11400, 11401, 11403, 11404, 11405, 11406, 114
07, 11408, 11409, 11410, 11411, 11413, 11414, 11415, 11416, 11417, 11418, 11419, 11420, 11421, 11422, 11423, 11425, 1
1427, 11428, 11429, 11430, 11431, 11433, 11434, 11436, 11437, 11438, 11439, 11440, 11441, 11442, 11443, 11445, 11446,
11447, 11448, 11449, 11450, 11452, 11453, 11454, 11455, 11456, 11458, 11459, 11460, 11461, 11462, 11463, 11464, 11465
, 11466, 11467, 11468, 11469, 11471, 11472, 11473, 11474, 11475, 11476, 11479, 11480, 11481, 11482, 11483, 11484, 114
85, 11486, 11487, 11488, 11489, 11490, 11493, 11494, 11496, 11497, 11498, 11499, 11500, 11501, 11502, 11504, 11505, 1
1507, 11508, 11509, 11510, 11511, 11512, 11513, 11514, 11515, 11516, 11517, 11518, 11519, 11520, 11521, 11522, 11523,
11524, 11525, 11526, 11527, 11528, 11529, 11530, 11531, 11532, 11533, 11536, 11538, 11539, 11540, 11541, 11542, 11543
, 11544, 11545, 11546, 11548, 11549, 11550, 11551, 11552, 11554, 11556, 11557, 11558, 11559, 11561, 11562, 11563, 115
64, 11565, 11566, 11568, 11569, 11570, 11571, 11572, 11573, 11574, 11575, 11576, 11577, 11578, 11579, 11580, 11581, 1
1582, 11583, 11584, 11585, 11586, 11587, 11588, 11589, 11590, 11591, 11592, 11593, 11595, 11596, 11597, 11598, 11599,
11600, 11601, 11602, 11604, 11605, 11606, 11607, 11608, 11609, 11610, 11611, 11612, 11613, 11615, 11616, 11617, 11618
, 11621, 11623, 11624, 11625, 11626, 11627, 11628, 11629, 11630, 11631, 11632, 11633, 11634, 11635, 11636, 11637, 116
Ln: 8 Col: 5789
Type here to search
ENG
12:21
04-02-2021

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
, 11387, 11390, 11391, 11392, 11393, 11394, 11395, 11396, 11397, 11398, 11400, 11401, 11403, 11404, 11405, 11406, 114
07, 11408, 11409, 11410, 11411, 11413, 11414, 11415, 11416, 11417, 11418, 11419, 11420, 11421, 11422, 11423, 11425, 1
1427, 11428, 11429, 11430, 11431, 11433, 11434, 11436, 11437, 11438, 11439, 11440, 11441, 11442, 11443, 11445, 11446,
11447, 11448, 11449, 11450, 11452, 11453, 11454, 11455, 11456, 11458, 11459, 11460, 11461, 11462, 11463, 11464, 11465
, 11466, 11467, 11468, 11469, 11471, 11472, 11473, 11474, 11475, 11476, 11479, 11480, 11481, 11482, 11483, 11484, 114
85, 11486, 11487, 11488, 11489, 11490, 11493, 11494, 11496, 11497, 11498, 11499, 11500, 11501, 11502, 11504, 11505, 1
1507, 11508, 11509, 11510, 11511, 11512, 11513, 11514, 11515, 11516, 11517, 11518, 11519, 11520, 11521, 11522, 11523,
11524, 11525, 11526, 11527, 11528, 11529, 11530, 11531, 11532, 11533, 11536, 11538, 11539, 11540, 11541, 11542, 11543
, 11544, 11545, 11546, 11548, 11549, 11550, 11551, 11552, 11554, 11556, 11557, 11558, 11559, 11561, 11562, 11563, 115
64, 11565, 11566, 11568, 11569, 11570, 11571, 11572, 11573, 11574, 11575, 11576, 11577, 11578, 11579, 11580, 11581, 1
1582, 11583, 11584, 11585, 11586, 11587, 11588, 11589, 11590, 11591, 11592, 11593, 11595, 11596, 11597, 11598, 11599,
11600, 11601, 11602, 11604, 11605, 11606, 11607, 11608, 11609, 11610, 11611, 11612, 11613, 11615, 11616, 11617, 11618
, 11621, 11623, 11624, 11625, 11626, 11627, 11628, 11629, 11630, 11631, 11632, 11633, 11634, 11635, 11636, 11637, 116
39, 11640, 11641, 11642, 11644, 11645, 11646, 11647, 11648, 11649, 11650, 11651, 11653, 11654, 11655, 11656, 11657, 1
1658, 11659, 11661, 11664, 11665, 11666, 11667, 11669, 11670, 11671, 11672, 11674, 11675, 11676, 11677, 11678, 11679,
11681, 11683, 11685, 11686, 11688, 11689, 11690, 11691, 11692, 11693, 11694, 11696, 11697, 11698, 11699, 11700, 11701
, 11702, 11703, 11704, 11705, 11707, 11708, 11709, 11710, 11711, 11712, 11713, 11716, 11717, 11718, 11719, 11722, 117
23, 11724, 11725, 11726, 11727, 11728, 11731, 11733, 11734, 11735, 11736, 11738, 11739, 11740, 11741, 11742, 11744, 1
1745, 11746, 11749, 11750, 11751, 11752, 11753, 11754, 11756, 11757, 11758, 11759, 11760, 11763, 11764, 11765, 11766,
11767, 11768, 11769, 11770, 11772, 11773, 11774, 11775, 11776, 11778, 11781, 11782, 11783, 11784, 11785, 11786, 11787
, 11789, 11790, 11791, 11794, 11795, 11796, 11797, 11798, 11799, 11800, 11801, 11802, 11803, 11804, 11805, 11808, 118
09, 11810, 11811, 11812, 11813, 11814, 11815, 11816, 11817, 11818, 11819, 11820, 11821, 11822, 11824, 11825, 11826, 1
1827, 11828, 11829, 11830, 11832, 11834, 11835, 11836, 11837, 11840, 11841, 11842, 11843, 11845, 11846, 11847, 11848,
11849, 11851, 11852, 11853, 11854, 11855, 11856, 11861, 11862, 11863, 11864, 11865, 11867, 11868, 11869, 11870, 11872
, 11873, 11874, 11876, 11877, 11878, 11879, 11881, 11882, 11883, 11884, 11885, 11887, 11890, 11891, 11893, 11894, 118
95, 11896, 11897, 11898, 11899, 11900, 11902, 11904, 11905, 11906, 11907, 11909, 11911, 11912, 11913, 11915, 11916, 1
1917, 11918, 11919, 11920, 11921, 11922, 11924, 11925, 11926, 11927, 11928, 11929, 11931, 11932, 11933, 11934, 11935,
11936, 11937, 11938, 11939, 11940, 11941, 11942, 11943, 11944, 11945, 11947, 11948, 11949, 11950, 11951, 11952, 11953
, 11954, 11955, 11956, 11957, 11958, 11959, 11960, 11961, 11962, 11963, 11964, 11965, 11966, 11967, 11968, 11969, 119
70, 11972, 11973, 11974, 11975, 11976, 11977, 11978, 11979, 11980, 11983, 11984, 11985, 11986, 11987, 11988, 11990, 1
1992, 11994, 11995, 11996, 11998, 11999, 12000]
```

Ln: 8 Col: 5789

Neeraj Appari 5073

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical - 3

Aim : 1] Write a code to create Binary Tree, where nodes are given as input by user. Display the tree
2] Write a code to traverse Binary tree in Pre-order, post-order and in-order

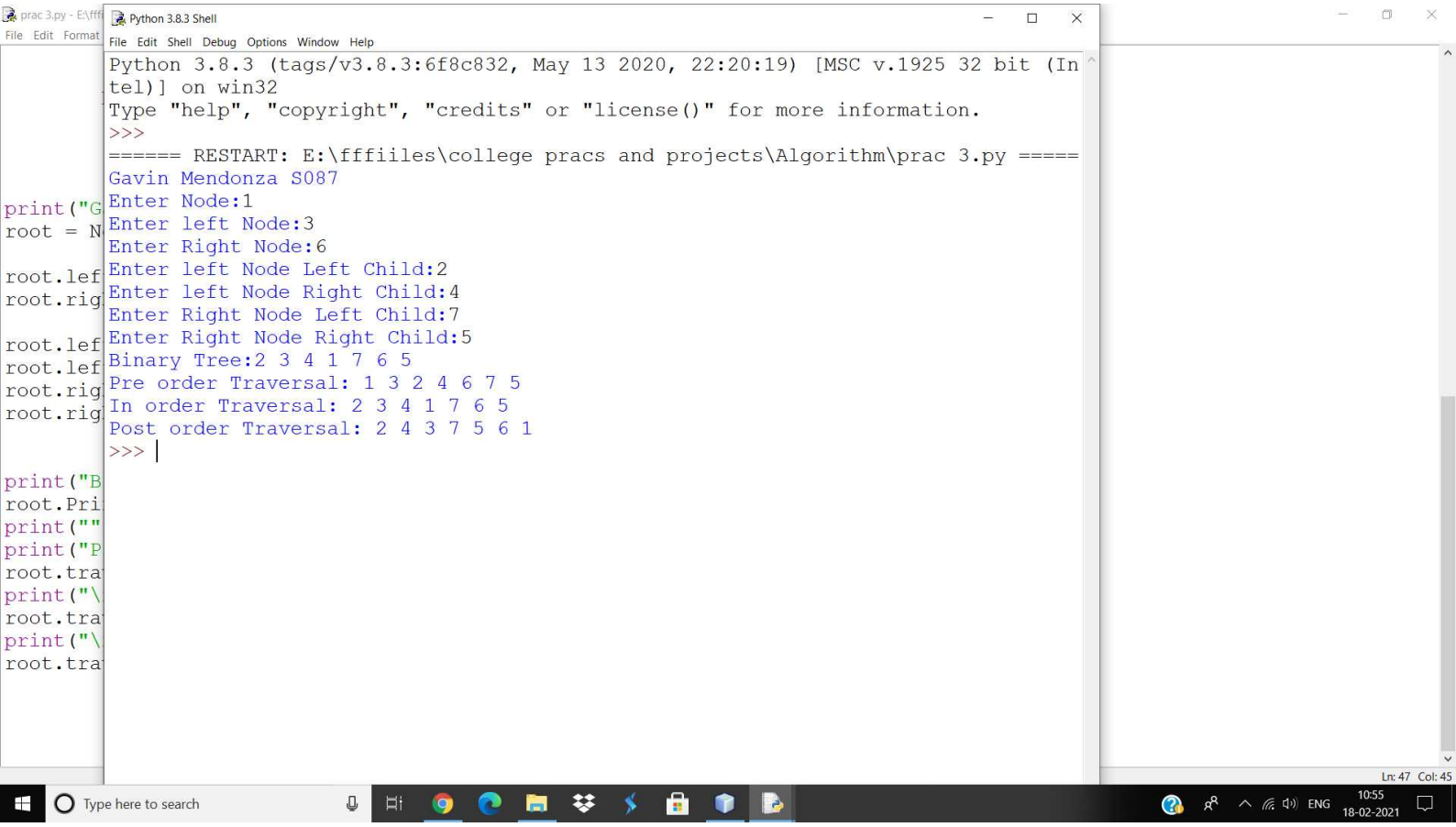
Algorithm - 1] Start
2] Create Root Node
3] Take key and value pair and use init_
4) Take input from user for Node, left Node, right node and their child nodes
5] Print Binary tree, Preorder, inorder and post order

Binary tree - A binary tree is a tree type non-linear data structure with a maximum of two children of each parent

Inorder - Output will produced sorted key in ascending order

Pre-order - The root node is visited first, then left subtree and finally right subtree

Post-order - First we traverse left subtree, then right subtree and finally root Node


Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and projects\Algorithm\prac 3.py =====
Gavin Mendonza S087
Enter Node:1
Enter left Node:3
Enter Right Node:6
Enter left Node Left Child:2
Enter left Node Right Child:4
Enter Right Node Left Child:7
Enter Right Node Right Child:5
Binary Tree:2 3 4 1 7 6 5
Pre order Traversal: 1 3 2 4 6 7 5
In order Traversal: 2 3 4 1 7 6 5
Post order Traversal: 2 4 3 7 5 6 1
>>>
Ln: 47 Col: 45
Type here to search
ENG
10:55
18-02-2021

prac 3.py - E:\fffiiles\college pracs and projects\Algorithm\prac 3.py (3.8.3)

File Edit Format Run Options Window Help

```
class Node:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.val = key

    # Traverse preorder
    def traversePreOrder(self):
        print(self.val, end=' ')
        if self.left:
            self.left.traversePreOrder()
        if self.right:
            self.right.traversePreOrder()

    # Traverse inorder
    def traverseInOrder(self):
        if self.left:
            self.left.traverseInOrder()
        print(self.val, end=' ')
        if self.right:
            self.right.traverseInOrder()

    # Traverse postorder
    def traversePostOrder(self):
        if self.left:
            self.left.traversePostOrder()
        if self.right:
            self.right.traversePostOrder()
        print(self.val, end=' ')

    def PrintTree(self):
        if self.left:
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22
:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and proj
ects\Algorithm\prac 3.py =====
Neeraj Appari S073
Enter Node:1
Enter left Node:3
Enter Right Node:6
Enter left Node Left Child:2
Enter left Node Right Child:4
Enter left Node Left Child:5
Enter Right Node Right Child:7
Binary Tree:2 3 4 1 5 6 7
Pre order Traversal: 1 3 2 4 6 5 7
In order Traversal: 2 3 4 1 5 6 7
Post order Traversal: 2 4 3 5 7 6 1
>>>
```

prac 3.py - E:\fffiiles\college pracs and projects\Algorithm\prac 3.py (3.8.3)

File Edit Format Run Options Window Help

```
        print(self.val, end=' ')

    def PrintTree(self):
        if self.left:
            self.left.PrintTree()
        print(self.val, end=' '),
        if self.right:
            self.right.PrintTree()


print("Neeraj Appari S073")
root = Node(int(input("Enter Node:")))

root.left = Node(int(input("Enter left Node:")))
root.right = Node(int(input("Enter Right Node:")))

root.left.left = Node(int(input("Enter left Node Left Child:")))
root.left.right = Node(int(input("Enter left Node Right Child:")))
root.right.left = Node(int(input("Enter left Node Left Child:")))
root.right.right = Node(int(input("Enter Right Node Right Child:"

print("Binary Tree:",end="")
root.PrintTree()
print("")
print("Pre order Traversal: ", end="")
root.traversePreOrder()
print("\nIn order Traversal: ", end="")
root.traverseInOrder()
print("\nPost order Traversal: ", end="")
root.traversePostOrder()
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22
:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and proj
ects\Algorithm\prac 3.py =====
Neeraj Appari S073
Enter Node:1
Enter left Node:3
Enter Right Node:6
Enter left Node Left Child:2
Enter left Node Right Child:4
Enter left Node Left Child:5
Enter Right Node Right Child:7
Binary Tree:2 3 4 1 5 6 7
Pre order Traversal: 1 3 2 4 6 5 7
In order Traversal: 2 3 4 1 5 6 7
Post order Traversal: 2 4 3 5 7 6 1
>>>
```

Neeraj Appari 5073 A

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical 4

Aim: Create a threaded Binary tree

Algorithm :-
1) Start
2) Create Node for threaded binary tree
3) Create utility function to return the leftmost node in given binary tree
4) Create iterative function to perform inorder traversal
5) Take values from user to convert binary tree into threaded binary tree.
6) stop

Description
1) The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion
2) A binary tree is made threaded by making all right child pointers that would be normally be NULL point to the inorder successor of the Node

prac 4.py - E:/fffiiles/college pracs and projects/Algorithm/prac 4.py (3.8.3)

File Edit Format Run Options Window Help

```
# Convert a binary tree into a threaded binary tree
def convertToThreaded(root):

    # stores previously visited node
    prev = None
    populateNext(root, prev)


if __name__ == '__main__':

    ''' Construct the following tree
              5
            /   \
           /     \
          2       7
         / \     / \
        /   \   /   \
       1    4 6     9
           /       / \
          /       /   \
         3       8    10
    '''
    print("Neeraj Appari S073")
    root = Node(int(input("Enter Node:")))

    root.left = Node(int(input("Enter left Node:")))
    root.right = Node(int(input("Enter Right Node:")))

    root.left.left = Node(int(input("Enter left Node Left Child:")))
    root.left.right = Node(int(input("Enter left Node Right Child:")))
    root.right.left = Node(int(input("Enter Right Node Left Child:")))
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020,
22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license(
)" for more information.
>>>
====== RESTART: E:/fffiiles/college pracs and pr
ojects/Algorithm/prac 4.py =====
Neeraj Appari S073
Enter Node:5
Enter left Node:2
Enter Right Node:7
Enter left Node Left Child:1
Enter left Node Right Child:4
Enter Right Node Left Child:6
Enter Right Node Right Child:9
Enter Left Node Right Child Node Left Child:3
Enter Right Node Right Child Node Left Child:8
Enter Right Node Right Child Node Right Child:10
1 2 3 4 5 6 7 8 9 10
>>>
```

prac 4.py - E:/fffiiles/college pracs and projects/Algorithm/prac 4.py (3.8.3)

File Edit Format Run Options Window Help

```
''' Construct the following tree
          5
        /   \
       /     \
      2       7
     / \     / \
    /   \   /   \
   1    4 6     9
       /       / \
      /       /   \
     3       8    10
'''
print("Neeraj Appari S073")
root = Node(int(input("Enter Node:")))

root.left = Node(int(input("Enter left Node:")))
root.right = Node(int(input("Enter Right Node:")))

root.left.left = Node(int(input("Enter left Node Left Child:")))
root.left.right = Node(int(input("Enter left Node Right Child:")))
root.right.left = Node(int(input("Enter Right Node Left Child:")))
root.right.right = Node(int(input("Enter Right Node Right Child:")))
root.left.right.left = Node(int(input("Enter Left Node Right Child Node Left Child:")))
root.right.right.left = Node(int(input("Enter Right Node Right Child Node Left Child:")))
root.right.right.right = Node(int(input("Enter Right Node Right Child Node Right Child:")))


convertToThreaded(root)
traverse(root)
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8
.3:6f8c832, May 13 2020
, 22:20:19) [MSC v.1925
32 bit (Intel)] on win3
2
Type "help", "copyright
", "credits" or "licens
e()" for more informati
on.
>>>
====== RESTART: E:/fffi
iles/college pracs and
projects/Algorithm/prac
4.py =====
Neeraj Appari S073
Enter Node:5
Enter left Node:2
Enter Right Node:7
Enter left Node Left Ch
ild:1
Enter left Node Right C
hild:4
Enter Right Node Left C
hild:6
Enter Right Node Right
Child:9
Enter Left Node Right C
hild Node Left Child:3
Enter Right Node Right
Child Node Left Child:8
Enter Right Node Right
Child Node Right Child:
```

Type here to search

ENG 08:53 24-02-2021

prac 4.py - E:\fffiiles\college pracs and projects\Algorithm\prac 4.py (3.8.3)

File Edit Format Run Options Window Help

```
# Threaded binary tree node
class Node:
    def __init__(self, data, left=None, right=None):
        self.data = data
        self.left = left
        self.right = right

        # true if the right pointer of a node points to its inorder successor
        self.isThreaded = False


# Utility function to return the leftmost node in a given binary tree
def leftMostNode(root):

    node = root
    while node and node.left:
        node = node.left
    return node


# Iterative function to perform inorder traversal on a threaded binary tree
def traverse(root):

    # base case
    if root is None:
        return

    # start from the leftmost node
    curr = leftMostNode(root)
    while curr:

        # print the current node
```

Ln: 1 Col: 0

prac 4.py - E:\fffiiles\college pracs and projects\Algorithm\prac 4.py (3.8.3)

File Edit Format Run Options Window Help

```
        # otherwise, visit the leftmost child in the right subtree
        else:
            curr = leftMostNode(curr.right)


# Function to convert a binary tree into a threaded binary tree
# using inorder traversal
def populateNext(curr, prev):

    # base case: empty tree
    if curr is None:
        return prev

    # recur for the left subtree
    prev = populateNext(curr.left, prev)

    # if the current node is not the root node of a binary tree
    # and has a None right child
    if prev and prev.right is None:

        # set the right child of the previous node to point to the current node
        prev.right = curr

        # set thread flag to true
        prev.isThreaded = True

    # update previous node
    prev = curr

    # recur for the right subtree
    prev = populateNext(curr.right, prev)
    return prev
```

Ln: 1 Col: 0

Neeraj Appari 5073 Appas

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical 5

Aim: Create an Expression tree

Algorithm - 1) Start
2) Take method for postfix convert
3) Take class for expression tree and methods for inorder, preorder and postorder
4) Take method for Expression tree
5) print the values in inorder, postorder and preorder

Description-
1) Expression tree is a binary tree in which eleval node corresponds to the operand so for expresion tree
2) Expression tree is a representation of expressions arranged in a tree-like data structure.
3) Tree with leaves as operands of the expression and nodes contain the operators

prac 5.py - E:/fffiiles/college pracs and projects/Algorithm/prac 5.py (3.8.3)

File Edit Format Run Options Window Help

```
    def __inorder_helper(self, node):
        if node:
            self.__inorder_helper(node.left)
            print (node.value)
            self.__inorder_helper(node.right)

    def preorder(self):
        self.__preorderUtil(self.__root)

    def __preorderUtil(self, node):
        if node:
            print (node.value)
            self.__preorderUtil(node.left)
            self.__preorderUtil(node.right)

    def postorder(self):
        self.__postorderUtil(self.__root)

    def __postorderUtil(self, node):
        if node:
            self.__postorderUtil(node.left)
            self.__postorderUtil(node.right)
            print (node.value)

def buildExpressionTree(infix):
    postfix = postfixConvert(infix)

    stack = []

    for char in postfix:
        if char not in operatorPrecedence:
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f
8c832, May 13 2020, 22:20:19
) [MSC v.1925 32 bit (Intel)
] on win32
Type "help", "copyright", "c
redits" or "license()" for m
ore information.
>>>
====== RESTART: E:/fffiiles/
college pracs and projects/A
lgorithm/prac 5.py =====
Neeraj Appari S073
In Order:
5
+
3
*
6
Post Order:
5
3
+
6
*
Pre Order:
*
+
5
3
6
>>>
```


prac 5.py - E:/fffiiles/college pracs and projects/Algorithm/prac 5.py (3.8.3)
File Edit Format Run Options Window Help
break
postfix.append(stack.pop())
stack.append(char)
while len(stack) != 0:
postfix.append(stack.pop())
return postfix
class Node(object):
def __init__(self, value):
self.value = value
self.left = None
self.right = None
class ExressionTree(object):
def __init__(self, root=None):
self.__root = root
def inorder(self):
self.__inorder_helper(self.__root)
def __inorder_helper(self, node):
if node:
self.__inorder_helper(node.left)
print (node.value)
self.__inorder_helper(node.right)
def preorder(self):
self.__preorderUtil(self.__root)
def __preorderUtil(self, node):

Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/Algorithm/prac 5.py =====
Neeraj Appari S073
In Order:
5
+
3
*
6
Post Order:
5
3
+
6
*
Pre Order:
*
+
5
3
6
>>>


Type here to search
ENG
13:49
24-02-2021

prac 5.py - E:/fffiiles/college pracs and projects/Algorithm/prac 5.py (3.8.3)

File Edit Format Run Options Window Help

```
            self.__postorderUtil(node.left)
            self.__postorderUtil(node.right)
            print (node.value)

def buildExpressionTree(infix):
    postfix = postfixConvert(infix)

    stack = []

    for char in postfix:
        if char not in operatorPrecedence:
            node = Node(char)
            stack.append(node)
        else:
            node = Node(char)
            right = stack.pop()
            left = stack.pop()
            node.right = right
            node.left = left
            stack.append(node)

    return ExressionTree(stack.pop())

print("Neeraj Appari S073")
print ("In Order:")
buildExpressionTree("(5+3)*6").inorder()
print ("Post Order:")
buildExpressionTree("(5+3)*6").postorder()
print ("Pre Order:")
buildExpressionTree("(5+3)*6").preorder()
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f
8c832, May 13 2020, 22:20:19
) [MSC v.1925 32 bit (Intel)
] on win32
Type "help", "copyright", "c
redits" or "license()" for m
ore information.
>>>
====== RESTART: E:/fffiiles/
college pracs and projects/A
lgorithm/prac 5.py =====
Neeraj Appari S073
In Order:
5
+
3
*
6
Post Order:
5
3
+
6
*
Pre Order:
*
+
5
3
6
>>>
```

prac 5.py - E:/fffiiles/college pracs and projects/Algorithm/prac 5.py (3.8.3)

File Edit Format Run Options Window Help

```
operatorPrecedence = {
    '(' : 0,
    ')' : 0,
    '+' : 1,
    '-' : 1,
    '*' : 2,
    '/' : 2
}

def postfixConvert(infix):
    stack = []
    postfix = []

    for char in infix:
        if char not in operatorPrecedence:
            postfix.append(char)
        else:
            if len(stack) == 0:
                stack.append(char)
            else:
                if char == "(":
                    stack.append(char)
                elif char == ")":
                    while stack[len(stack) - 1] != "(":
                        postfix.append(stack.pop())
                    stack.pop()
                elif operatorPrecedence[char] > operatorPrecedence[stack[len(stack) - 1]
                    stack.append(char)
                else:
                    while len(stack) != 0:
                        if stack[len(stack) - 1] == '(':
                            break
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f
8c832, May 13 2020, 22:20:19
) [MSC v.1925 32 bit (Intel)
] on win32
Type "help", "copyright", "c
redits" or "license()" for m
ore information.
>>>
====== RESTART: E:/fffiiles/
college pracs and projects/A
lgorithm/prac 5.py =====
Neeraj Appari S073
In Order:
5
+
3
*
6
Post Order:
5
3
+
6
*
Pre Order:
*
+
5
3
6
>>>
```

Neeraj Appar 5073 Appar

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical No 6

Aim: Insert and delete nodes in Binary tree

Algorithm:
Step 1: Start
Step 2: To create a class node in Binary tree
Step 3: Create a constructor to create new node
Step 4: Create a class for inorder
Step 5: Create a class for inserting the values
Step 6: Create a class for minValueNode
Step 7: Create a Class for deleteNode
Step 8: Take inputs for inserting Nodes into tree
Step 9: Print the Binary tree
Step 10: Take inputs for deleting nodes
Step 11: Print the binary tree
Step 12: stop

AI Description:
Binary tree - A tree is called binary tree if each node has zero child, one or one child or two children
Empty tree is also a valid binary tree we can visualize a binary tree as consisting of a root and two disjoint binary tree called the left and right subtrees of root

prac 6.py - E:\fffiiles\college pracs and projects\Algorithm\prac 6.py (3.8.3)

File Edit Format Run Options Window Help

```
                        temp = root.left
                        root = None
                        return temp


                temp = minValueNode(root.right)


                root.key = temp.key


                root.right = deleteNode(root.right, temp.key)

        return root

print("Neeraj Appari S073")
root = None
root = insert(root,int(input("Enter Node:")))
root = insert(root,int(input("Enter Left Node:")))
root = insert(root,int(input("Enter Right Node:")))
root = insert(root,int(input("Enter left Node Left Child:")))
root = insert(root,int(input("Enter left Node Right Child:")))
root = insert(root,int(input("Enter Right Node Left Child:")))
root = insert(root,int(input("Enter Right Node Right Child:")))

print ("Inorder traversal of the given tree")
inorder(root)


root = deleteNode(root,int(input("Enter which node to delete:")))
print ("Inorder traversal of the modified tree")
inorder(root)
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22
:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and proj
ects\Algorithm\prac 6.py =====
Neeraj Appari S073
Enter Node:1
Enter Left Node:4
Enter Right Node:6
Enter left Node Left Child:2
Enter left Node Right Child:3
Enter Right Node Left Child:5
Enter Right Node Right Child:7
Inorder traversal of the given tree
1
2
3
4
5
6
7
Enter which node to delete:4
Inorder traversal of the modified tree
1
2
3
5
6
7
>>>
```

prac 6.py - E:\fffiiles\college pracs and projects\Algorithm\prac 6.py (3.8.3)

File Edit Format Run Options Window Help

```
class Node:

        def __init__(self, key):
                self.key = key
                self.left = None
                self.right = None
def inorder(root):
        if root is not None:
                inorder(root.left)
                print (root.key),
                inorder(root.right)
def insert(node, key):


        if node is None:
                return Node(key)


        if key < node.key:
                node.left = insert(node.left, key)
        else:
                node.right = insert(node.right, key)


        return node


def minValueNode(node):
        current = node


        while(current.left is not None):
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22
:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and proj
ects\Algorithm\prac 6.py =====
Neeraj Appari S073
Enter Node:1
Enter Left Node:4
Enter Right Node:6
Enter left Node Left Child:2
Enter left Node Right Child:3
Enter Right Node Left Child:5
Enter Right Node Right Child:7
Inorder traversal of the given tree
1
2
3
4
5
6
7
Enter which node to delete:4
Inorder traversal of the modified tree
1
2
3
5
6
7
>>>
```

prac 6.py - E:\fffiiles\college pracs and projects\Algorithm\prac 6.py (3.8.3)

File Edit Format Run Options Window Help

```
        while(current.left is not None):
                current = current.left

        return current

def deleteNode(root, key):


        if root is None:
                return root

        if key < root.key:
                root.left = deleteNode(root.left, key)


        elif(key > root.key):
                root.right = deleteNode(root.right, key)

        else:


                if root.left is None:
                        temp = root.right
                        root = None
                        return temp

                elif root.right is None:
                        temp = root.left
                        root = None
                        return temp
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22
:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
====== RESTART: E:\fffiiles\college pracs and proj
ects\Algorithm\prac 6.py =====
Neeraj Appari S073
Enter Node:1
Enter Left Node:4
Enter Right Node:6
Enter left Node Left Child:2
Enter left Node Right Child:3
Enter Right Node Left Child:5
Enter Right Node Right Child:7
Inorder traversal of the given tree
1
2
3
4
5
6
7
Enter which node to delete:4
Inorder traversal of the modified tree
1
2
3
5
6
7
>>>
```

Neeraj Appan 5073 A

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical No 7

Aim: ~~Create~~ Create Graph using adjacency matrix

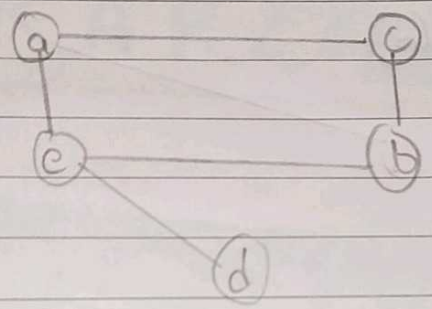
Algorithm: 1) Start
2) Take Class vertex
3) Add objects _init_, addNeighbor, getConnections, getVertexID, setVertexID, setVisited, __str__
4) Take Class Graph
5) Add objects __init__, setVertex, getVertex, addEdge, getVertices, printMatrix and getEdges
6) Take values in setVertex object
7) Combine the values in addEdge Vertex
8) print Matrix and print edges
9) Stop

Description

1) Adjacency matrix is a 2D array of size V x V where V is the number of vertices in a graph. ~~let the 2D array be adj[][]~~ a ~~slot~~ adj[i][j] = 1 indicates ~~that there~~ is an edge
2) It is a square matrix used to represent a finite graph
3) The element of the matrix indicate whether pairs of vertices are adjacent or not in the graph
4) It is also called Connection matrix

Neeraj Appan 5073 Appy

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science



	a	b	c	d	e
a	0	0	1	0	1
b	0	0	0	0	1
c	1	0	0	0	0
d	0	0	0	0	1
e	1	1	0	1	0

prac 7.py - E:/fffiiles/college pracs and projects/Algorithm/prac 7.py (3.8.3)

File Edit Format Run Options Window Help

```
            for u in range(0, self.numVertices):
                row = []
                for v in range(0, self.numVertices):
                    row.append(self.adjMatrix[u][v])
                    print (row)

    def getEdges(self):
        edges = []
        for v in range(0, self.numVertices):
            for u in range(0, self.numVertices):
                if self.adjMatrix[u][v] != -1:
                    vid = self.vertices[v].getVertexID()
                    wid = self.vertices[u].getVertexID()
                    edges.append((vid, wid, self.adjMatrix[u][v]))
                    return edges

if __name__ == '__main__':
    G = Graph(5)
    G.setVertex(0, 'a')
    G.setVertex(1, 'b')
    G.setVertex(2, 'c')
    G.setVertex(3, 'd')
    G.setVertex(4, 'e')
    print ('Graph data:')
    G.addEdge('a', 'e', 10)
    G.addEdge('a', 'c', 20)
    G.addEdge('c', 'b', 30)
    G.addEdge('b', 'e', 40)
    G.addEdge('e', 'd', 50)
    G.addEdge('f', 'e', 60)
    print (G.printMatrix())
    print (G.getEdges())
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
>>>
====== RESTART: E:/fffiiles/college pracs an
d projects/Algorithm/prac 7.py =====
Neeraj Appari S073
Graph data:
[-1]
[-1, -1]
[-1, -1, 20]
[-1, -1, 20, -1]
[-1, -1, 20, -1, 10]
[-1]
[-1, -1]
[-1, -1, 30]
[-1, -1, 30, -1]
[-1, -1, 30, -1, 40]
[20]
[20, 30]
[20, 30, -1]
[20, 30, -1, -1]
[20, 30, -1, -1, -1]
[-1]
[-1, -1]
[-1, -1, -1]
[-1, -1, -1, -1]
[-1, -1, -1, -1, 50]
[10]
[10, 40]
[10, 40, -1]
[10, 40, -1, 50]
[10, 40, -1, 50, -1]
None
[('a', 'c', 20)]
```

prac 7.py - E:/fffiiles/college pracs and projects/Algorithm/prac 7.py (3.8.3)

File Edit Format Run Options Window Help

```
class Vertex:
    print("Neeraj Appari S073")
    def __init__(self, node):
        self.id = node
        self.visited = False

    def addNeighbor(self, neighbor, G):
        G.addEdge(self.id, neighbor)

    def getConnections(self, G):
        return G.adjMatrix[self.id]

    def getVertexID(self):
        return self.id

    def setVertexID(self, id):
        self.id = id

    def setVisited(self):
        self.visited = True

    def __str__(self):
        return str(self.id)

class Graph:
    def __init__(self, numVertices, cost=0):
        self.adjMatrix = [[-1] * numVertices for _ in range(numVertices)]
        self.numVertices = numVertices
        self.vertices = []
        for i in range(0, numVertices):
            newVertex = Vertex(i)
            self.vertices.append(newVertex)
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 20
20, 22:20:19) [MSC v.1925 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "lice
nse()" for more information.
>>>
====== RESTART: E:/fffiiles/college pracs an
d projects/Algorithm/prac 7.py =====
Neeraj Appari S073
Graph data:
[-1]
[-1, -1]
[-1, -1, 20]
[-1, -1, 20, -1]
[-1, -1, 20, -1, 10]
[-1]
[-1, -1]
[-1, -1, 30]
[-1, -1, 30, -1]
[-1, -1, 30, -1, 40]
[20]
[20, 30]
[20, 30, -1]
[20, 30, -1, -1]
[20, 30, -1, -1, -1]
[-1]
[-1, -1]
[-1, -1, -1]
[-1, -1, -1, -1]
[-1, -1, -1, -1, 50]
[10]
[10, 40]
```

Type here to search

ENG 19:38 04-03-2021

prac 7.py - E:/fffiiles/college pracs and projects/Algorithm/prac 7.py (3.8.3)

File Edit Format Run Options Window Help

```
            self.vertices.append(newVertex)
    def setVertex(self, vtx, id):
        if 0 <= vtx < self.numVertices:
            self.vertices[vtx].setVertexID(id)

    def getVertex(self, n):
        for vertxin in range(0, self.numVertices):
            if n == self.vertices[vertxin].getVertexID():
                return vertxin
        return -1

    def addEdge(self, frm, to, cost=0):
        if self.getVertex(frm) != -1 and self.getVertex(to) != -1:
            self.adjMatrix[self.getVertex(frm)][self.getVertex(to)] = cost
            self.adjMatrix[self.getVertex(to)][self.getVertex(frm)] = cost

    def getVertices(self):
        vertices = []
        for vertxin in range(0, self.numVertices):
            vertices.append(self.vertices[vertxin].getVertexID())
            return vertices

    def printMatrix(self):
        for u in range(0, self.numVertices):
            row = []
            for v in range(0, self.numVertices):
                row.append(self.adjMatrix[u][v])
                print (row)

    def getEdges(self):
        edges = []
        for v in range(0, self.numVertices):
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
>>>
====== RESTART: E:/fffiiles/college pracs and projects/Algorithm/prac 7.py =====
Neeraj Appari S073
Graph data:
[-1]
[-1, -1]
[-1, -1, 20]
[-1, -1, 20, -1]
[-1, -1, 20, -1, 10]
[-1]
[-1, -1]
[-1, -1, 30]
[-1, -1, 30, -1]
[-1, -1, 30, -1, 40]
[20]
[20, 30]
[20, 30, -1]
[20, 30, -1, -1]
[20, 30, -1, -1, -1]
[-1]
[-1, -1]
[-1, -1, -1]
[-1, -1, -1, -1]
[-1, -1, -1, -1, 50]
[10]
[10, 40]
[10, 40, -1]
[10, 40, -1, 50]
[10, 40, -1, 50, -1]
None
[('a', 'c', 20)]
```

11/3/21

Neeraj Appari 5073

SHETH L.U.J. COLLEGE OF ARTS &
SIR M.V. COLLEGE OF SCIENCE & COMMERCE
Department of Computer Science

Practical - 08

Aim - Write a program to demonstrate DFS

* Describe DFS -

Depth - first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node selecting some arbitary node as the node in the case of graph and explores as far as possible along each branch before backtracking

Example

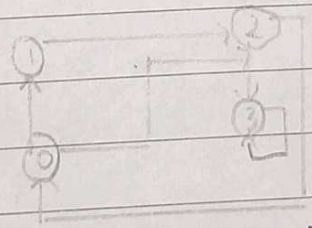
Input : n = 4, e = 6

0 - 1, 0 - 2, 1 → 2, 2 → 0, 2 → 3, 3 → 3

Output : DFS from vertex 1 : 1 2 0 3

Explanation :

DFS diagram

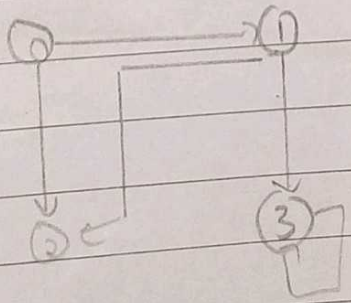


Input n = 4, e = 6

2 → 0, 0 → 2, 1 → 2, 1 → 2, 0 → 1, 3 → 3, 1 → 3

Output : DFS from vertex 2 : 2 0 1 3

DFS Diagram



prac 8.py - E:/fffiiles/college pracs and projects/Algorithm/prac 8.py (3.8.3)

File Edit Format Run Options Window Help

```
from collections import defaultdict
class Graph:
    def __init__(self):
        self.graph = defaultdict(list)
    def addEdge(self, u, v):
        self.graph[u].append(v)
    def DFSUtil(self, v, visited):
        visited.add(v)
        print(v, end=' ')

        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited)
    def DFS(self, v):

        visited = set()

        self.DFSUtil(v, visited)


g = Graph()
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)

print("Neeraj Appari S073")
print("Following is DFS from (starting from vertex 2)")
g.DFS(2)
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [
MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/Algo
rithm/prac 8.py =====
Neeraj Appari S073
Following is DFS from (starting from vertex 2)
2 0 1 3
>>>
```

Appan Meeraj S013

Algorithm Practical 10

Aim: WAP to find fibonacci series using dynamic programming

§ Algorithm:
1) Start
2) take fib function with parameter n and loop and main function
3) Use if loop for the conditions
4) return the variable lookup
5) take the initializing variable
6) print the fibonnaci number
7) Stop

Writeups
*1) Dynamic Programming
-> Dynamic Programming (DP) is mainly an optimization over plain recursion.
2) Whenever we see a recursive solution that has repeatedly calling the same inputs we can optimize it using DP
3) Value can be stored in two ways
i) Memorization (Top Down)
-> It is a small modification that it looks into a lookup table to be before computing solution.
ii) Tabulation (Bottom up)
-> It builts a b table in bottom up fashion and returns the last entry from table

*2) Fibonacci Series
-> It is the sum of two preceding numbers starting from 0 and 1.

prac 10.py - E:/fffiiles/college pracs and projects/Algorithm/prac 10.py (3.8.3)

File Edit Format Run Options Window Help

```
# Function to calculate nth Fibonacci number
def fib(n, lookup):

    # Base case
    if n == 0 or n == 1 :
        lookup[n] = n

    # If the value is not calculated previously then calcu
    if lookup[n] is None:
        lookup[n] = fib(n-1 , lookup)  + fib(n-2 , lookup)

    # return the value corresponding to that value of n
    return lookup[n]
# end of function

# Driver program to test the above function
def main():
    n = 34
    # Declaration of lookup table
    # Handles till n = 100
    lookup = [None]*(101)
    print ("Fibonacci Number is ", fib(n, lookup))

if __name__=="__main__":
    print("Neeraj Appari S073")
    main()
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [
v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
formation.
>>>
===== RESTART: E:/fffiiles/college pracs and projects/Algor
m/prac 10.py =====
Neeraj Appari S073
Fibonacci Number is  5702887
>>>
```

Appan Neeraj 5073

Algorithm Practical 11

Aim: WAP to demonstrate merge sort algorithm using divide and conquer technique

Algorithm
1) Start
2) take merge functions with parameters arr, l, m and r
3) initialize variables n1 and n2
4) create arrays L and R
5) Use for loop to copy data in array
6) merge the arrays using while loop
7) using mergesort a function for the sub-arrays in sort
8) Take the values in arr variables to sort
9) print the sorted array
10) Stop

Writeups
-) Merge Sort
-) It is a divide and conquer algorithm based on the idea of breaking down a list into several sub-lists until each sublist consists of a single element and merging those sublists in a manner that results into sort list

prac 11.py - E:/fffiiles/college pracs and projects/Algorithm/prac 11.py (3.8.3)

File Edit Format Run Options Window Help

```
print ("Neeraj Appari S073")
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r- m

    # create temp arrays
    L = [0] * (n1)
    R = [0] * (n2)

    # Copy data to temp arrays L[] and R[]
    for i in range(0 , n1):
        L[i] = arr[l + i]

    for j in range(0 , n2):
        R[j] = arr[m + 1 + j]

    # Merge the temp arrays back into arr[l..r]
    i = 0     # Initial index of first subarray
    j = 0     # Initial index of second subarray
    k = l     # Initial index of merged subarray

    while i < n1 and j < n2 :
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    # Copy the remaining elements of L[], if there
    # are any
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MS
C v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more i
nformation.
>>>
===== RESTART: E:/fffiiles/college pracs and projects/Algorit
hm/prac 11.py =====
Neeraj Appari S073
Given array is
12
11
13
2
6
9
18


Sorted array is
2
6
9
11
12
13
18
>>>
```

prac 11.py - E:/fffiiles/college pracs and projects/Algorithm/prac 11.py (3.8.3)

File Edit Format Run Options Window Help

```
        while j < n2:
            arr[k] = R[j]
            j += 1
            k += 1

# l is for left index and r is right index of the
# sub-array of arr to be sorted
def mergeSort(arr,l,r):

    if l < r:

        # Same as (l+r)//2, but avoids overflow for
        # large l and h
        m = (l+(r-1))//2

        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)


# Driver code to test above
arr = [12, 11, 13, 2, 6, 9, 18]
n = len(arr)
print ("Given array is")
for i in range(n):
    print ("%d" %arr[i]),

mergeSort(arr,0,n-1)
print ("\n\nSorted array is")
for i in range(n):
    print ("%d" %arr[i]),
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MS
C v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more i
nformation.
>>>
===== RESTART: E:/fffiiles/college pracs and projects/Algorit
hm/prac 11.py =====
Neeraj Appari S073
Given array is
12
11
13
2
6
9
18


Sorted array is
2
6
9
11
12
13
18
>>>
```