Assignment : 01

Title : Write a program to calculate Fibonacci numbers and find its step

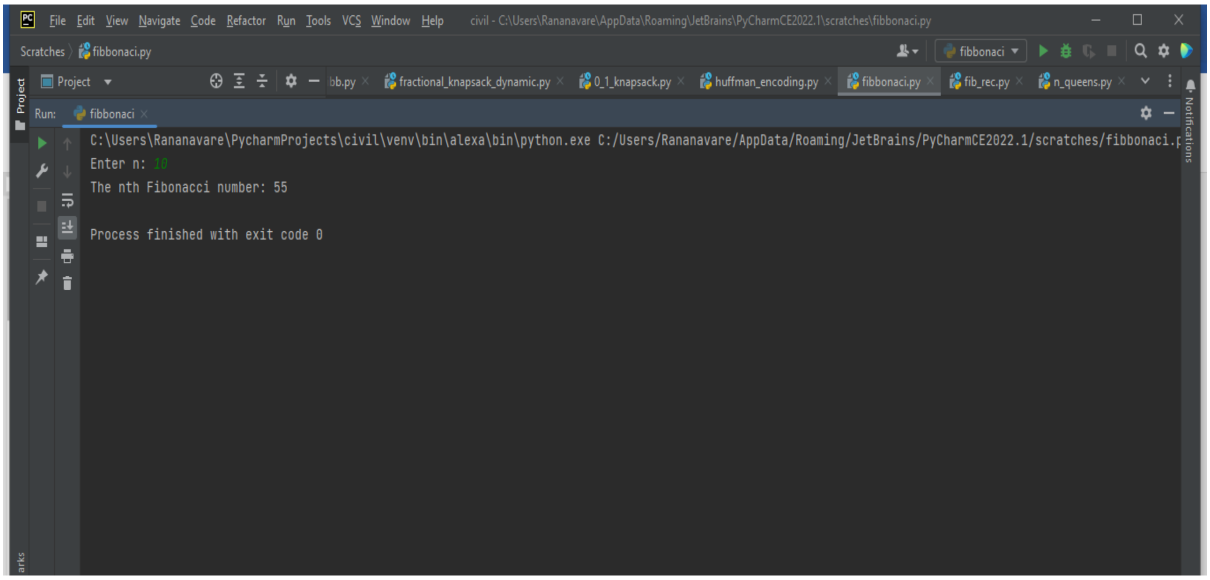
countRoll no: CO405

Program :

A) Non-Recursive

```
def fibonacci(n):
    """Return the nth Fibonacci number."""
    if n == 0:
        return 0

    # r[i] will contain the ith Fibonacci number
    r = [-1] * (n + 1)
    r[0] = 0
    r[1] = 1

    for i in range(2, n + 1):
        r[i] = r[i - 1] + r[i - 2]

    return r[n]


n = int(input('Enter n: '))

ans = fibonacci(n)
print('The nth Fibonacci number:', ans)
```

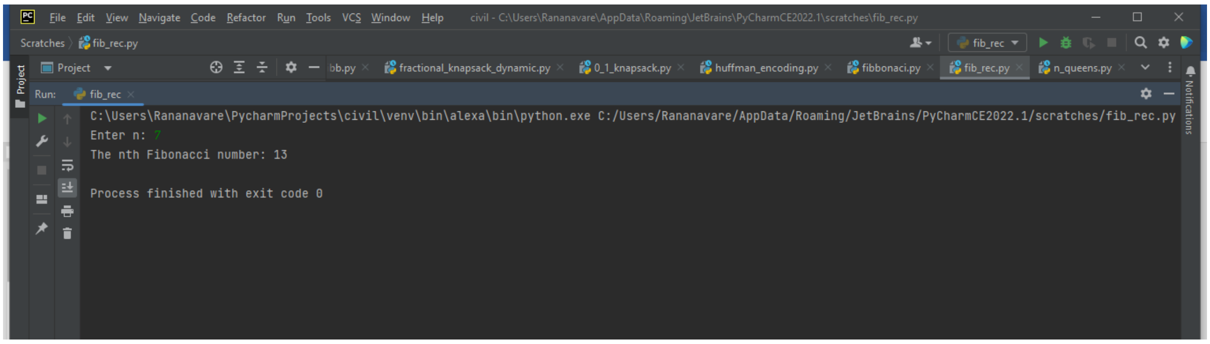
OUTPUT



B)Recursive

```
def fibonacci(n):
    """Return the nth Fibonacci number."""
    # r[i] will contain the ith Fibonacci number
    r = [-1] * (n + 1)
    return fibonacci_helper(n, r)


def fibonacci_helper(n, r):
    """Return the nth Fibonacci number and store the ith
Fibonacci number in
    r[i] for 0 <= i <= n."""
    if r[n] >= 0:
        return r[n]
if (n == 0 or n == 1):
        q = n
    else:
        q = fibonacci_helper(n - 1, r) + fibonacci_helper(n - 2,
r)
    r[n] = q
    return q
n = int(input('Enter n: '))
ans = fibonacci(n)
print('The nth Fibonacci number:', ans)
```

OUTPUT



Assignment : 02

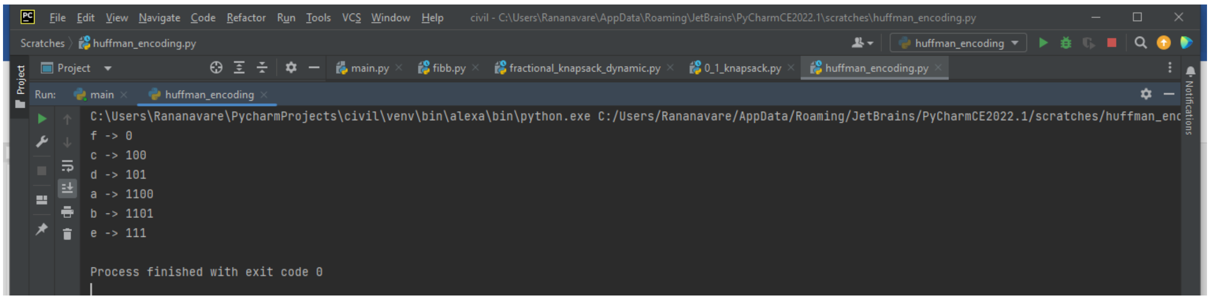
Title : Write program to implement Huffman Encoding using a greed strategy.

Roll no: CO405

Program :

```
 import heapq
class node:
   def___init_(self,freq,symbol,left=None,right=None):
      self.freq=freq
      self.symbol=symbo
      lself.left=left
      self.right=right
      self.huff= ''

   def___lt__(self,nxt):
      return self.freq<nxt.freq

def printnodes(node,val=''):
   newval=val+str(node.huff)

   if node.left:
      printnodes(node.left,newval)if
   node.right:
      printnodes(node.right,newval)

   if not node.left and not node.right:
      print("{} -> {}".format(node.symbol,newval))

if ___name__=="__main__":
   chars = ['a', 'b', 'c', 'd', 'e', 'f']
   freq = [ 5, 9, 12, 13, 16, 45]
   nodes=[]

   for i in range(len(chars)):
      heapq.heappush(nodes, node(freq[i],chars[i]))

   while len(nodes)>1:
      left=heapq.heappop(nodes)
      right=heapq.heappop(nodes)left.huff
      = 0
      right.huff = 1
 newnode = node(left.freq + right.freq , left.symbol + right.symbol , left , right)
      heapq.heappush(nodes, newnode)
   printnodes(nodes[0])
```

OUTPUT :


File Edit View Navigate Code Refactor Run Tools VCS Window Help
civil - C:\Users\Rananavare\AppData\Roaming\JetBrains\PyCharmCE2022.1\scratches\huffman_encoding.py
Scratches
huffman_encoding.py
huffman_encoding
Project
main.py
fibb.py
fractional_knapsack_dynamic.py
0_1_knapsack.py
huffman_encoding.py
Run:
main
huffman_encoding
C:\Users\Rananavare\PycharmProjects\civil\venv\bin\alexa\bin\python.exe C:/Users/Rananavare/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/huffman_enc
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
Process finished with exit code 0

Assignment : 03

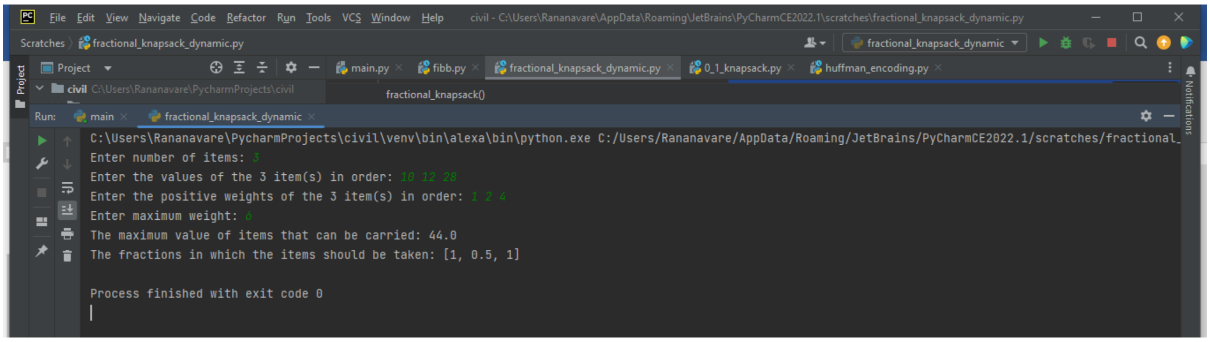
Title : Write a program to solve a fractional Knapsack problem using a greedy method.

Roll no: CO405

Program :

```
def fractional_knapsack(value, weight, capacity):

    index = list(range(len(value)))

   ratio = [v / w for v, w in zip(value, weight)]
   index.sort(key=lambda i: ratio[i], reverse=True)

   max_value = 0
   fractions = [0] * len(value)
   for i in index:
      if weight[i] <= capacity:
         fractions[i] = 1
         max_value +=
         value[i]capacity -=
         weight[i]
      else:
         fractions[i] = capacity / weight[i]
         max_value += value[i] * capacity /
         weight[i]break

   return max_value, fractions

n = int(input('Enter number of items: '))
value = input('Enter the values of the {} item(s) in order: '
           .format(n)).split()
value = [int(v) for v in
value]
weight = input('Enter the positive weights of the {} item(s) in order: '
            .format(n)).split()
weight = [int(w) for w in weight]
capacity = int(input('Enter maximum weight: '))

max_value, fractions = fractional_knapsack(value, weight, capacity)
print('The maximum value of items that can be carried:', max_value)
print('The fractions in which the items should be taken:', fractions)
```

OUTPUT:

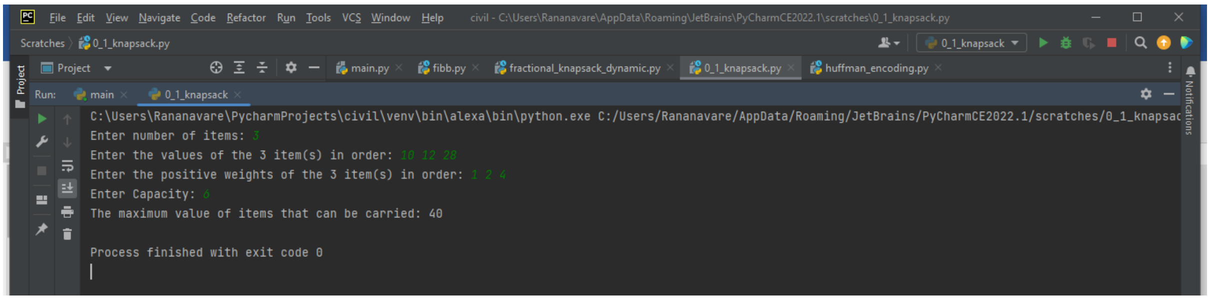

C:\Users\Rananavare\PycharmProjects\civil\venv\bin\alexa\bin\python.exe C:/Users/Rananavare/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/fractional_
Enter number of items: 3
Enter the values of the 3 item(s) in order: 10 12 28
Enter the positive weights of the 3 item(s) in order: 1 2 4
Enter maximum weight: 6
The maximum value of items that can be carried: 44.0
The fractions in which the items should be taken: [1, 0.5, 1]
Process finished with exit code 0

Assignment:04

Title : Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch andbound strategy.
Roll no:CO405
Program :

```
def knapsack(value, weight, capacity):
    n = len(value) - 1

    m = [[-1] * (capacity + 1) for _ in range(n + 1)]

    for w in range(capacity + 1):
        m[0][w] = 0

    for i in range(1, n + 1):
        for w in range(capacity + 1):
            if weight[i] > w:
                m[i][w] = m[i -
            1][w]else:
                m[i][w] = max(m[i - 1][w - weight[i]] + value[i],
                              m[i - 1][w])

    return m[n][capacity]


n = int(input('Enter number of items: '))
value = input('Enter the values of the {} item(s) in order: '
              .format(n)).split()
value = [int(v) for v in
value]
value.insert(0, None) # so that the value of the ith item is at value[i]
weight = input('Enter the positive weights of the {} item(s) in order: '
               .format(n)).split()
weight = [int(w) for w in weight]
weight.insert(0, None) # so that the weight of the ith item is at weight[i]capacity
= int(input('Enter Capacity: '))

ans = knapsack(value, weight, capacity)
print('The maximum value of items that can be carried:', ans)
```

OUTPUT



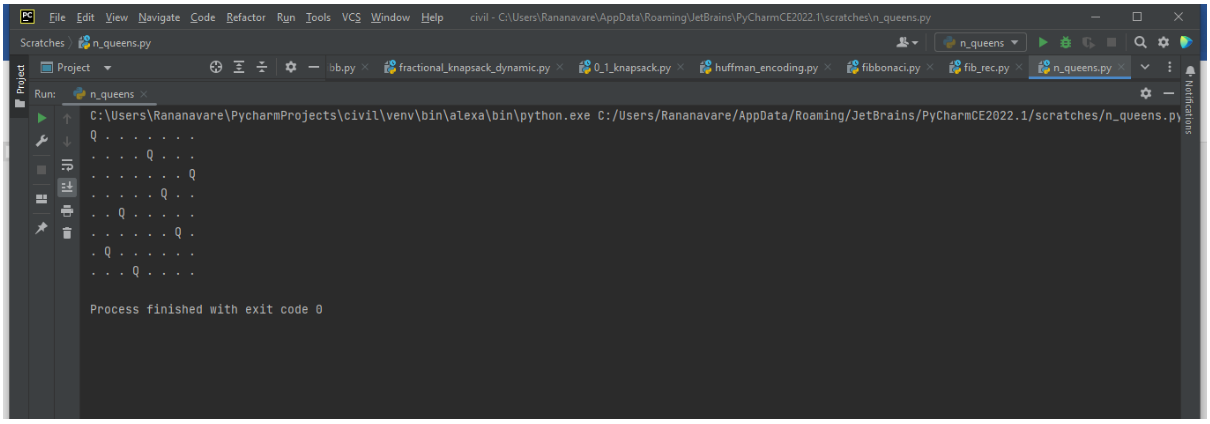
File Edit View Navigate Code Refactor Run Tools VCS Window Help
civil - C:\Users\Rananavare\AppData\Roaming\JetBrains\PyCharmCE2022.1\scratches\0_1_knapsack.py
Scratches 0_1_knapsack.py
0_1_knapsack
Project
main.py
fibb.py
fractional_knapsack_dynamic.py
0_1_knapsack.py
huffman_encoding.py
Run:
main
0_1_knapsack
C:\Users\Rananavare\PycharmProjects\civil\venv\bin\alexa\bin\python.exe C:/Users/Rananavare/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/0_1_knapsac
Enter number of items: 3
Enter the values of the 3 item(s) in order: 10 12 28
Enter the positive weights of the 3 item(s) in order: 1 2 4
Enter Capacity: 6
The maximum value of items that can be carried: 40
Process finished with exit code 0
Notifications

Assignment:05

Title : Design n Queens Matrix having First Queen placed using backtracking to place remaining Queens to generate the final n Queens Matrix.
Roll noCO405
Program :

```
N = 8  # Size of the chessboard

def is_safe(board, row, col):
    # Check if there is a Queen in the same column
    for i in range(row):
        if board[i][col] == 1:
            return False

    # Check upper left diagonal
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check upper right diagonal
    for i, j in zip(range(row, -1, -1), range(col, N)):
        if board[i][j] == 1:
            return False

    return True

def solve_n_queens(board, row):
    if row >= N:
        return True

    for col in range(N):
        if is_safe(board, row, col):
            board[row][col] = 1  # Place the Queen
            if solve_n_queens(board, row + 1):  # Recur to place rest of the Queens
                return True
            board[row][col] = 0  # If placing Queen doesn't lead to a solution, backtrack
    return False

# Initialize the chessboard with the first Queen already placed
chessboard = [[0 for _ in range(N)] for _ in range(N)]
chessboard[0][0] = 1

# Solve the 8-Queens problem using backtracking
if solve_n_queens(chessboard, 1):
    # Print the solution
    for row in chessboard:
        print(' '.join(['Q' if x else '.' for x in row]))
else:
    print("No solution exists.")
```

OUTPUT


File Edit View Navigate Code Refactor Run Tools VCS Window Help
civil - C:\Users\Rananavare\AppData\Roaming\JetBrains\PyCharmCE2022.1\scratches\n_queens.py
Scratches
n_queens.py
Project
n_queens
bb.py
fractional_knapsack_dynamic.py
0_1_knapsack.py
huffman_encoding.py
fibbonaci.py
fib_rec.py
n_queens.py
Run:
n_queens
C:\Users\Rananavare\PycharmProjects\civil\venv\bin\alexa\bin\python.exe C:/Users/Rananavare/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/n_queens.py
Q . . . .
. . . . Q . . .
. . . . Q
. . . . . Q . .
. . Q . . . . .
. . . . . . Q .
. Q . . . . . .
. . . Q . . . .
Process finished with exit code 0
Notifications