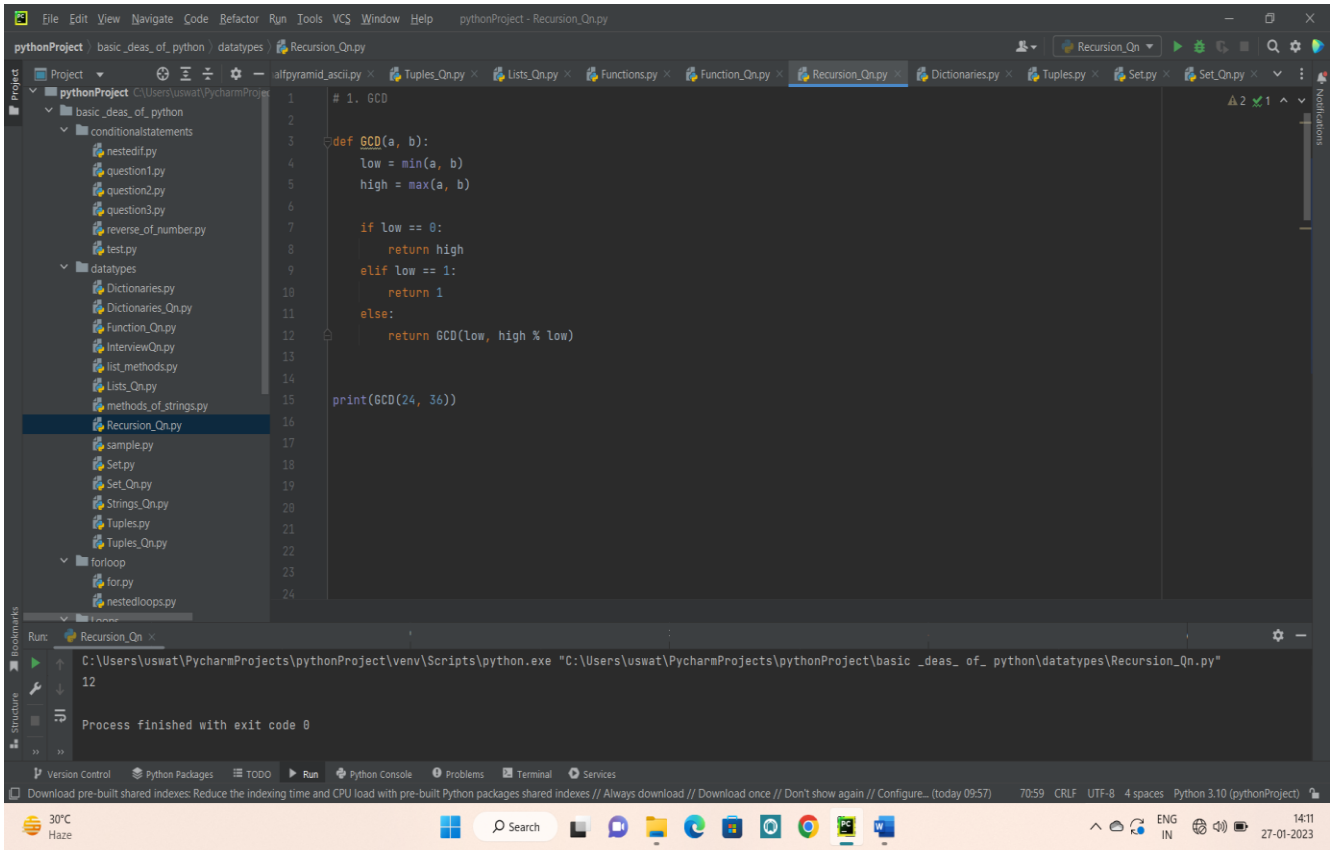
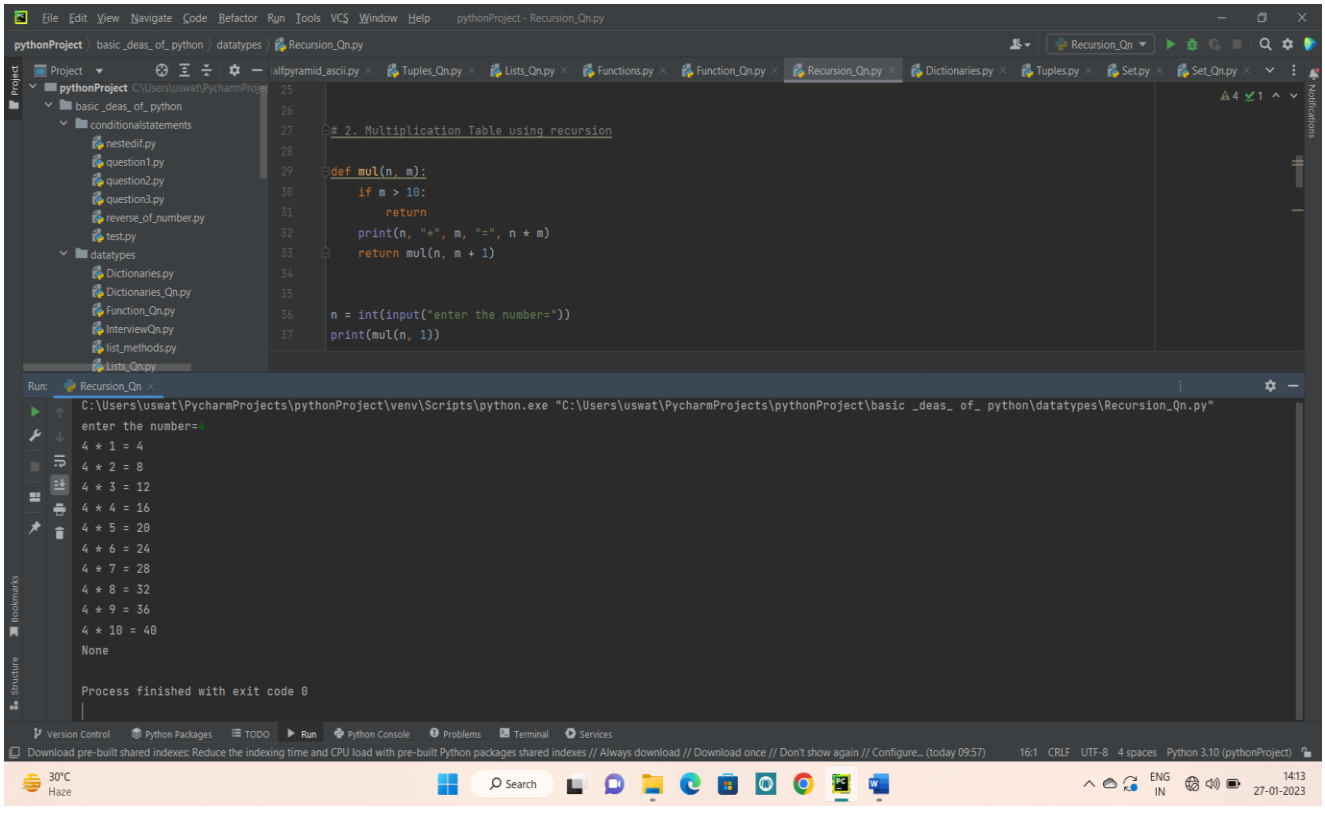
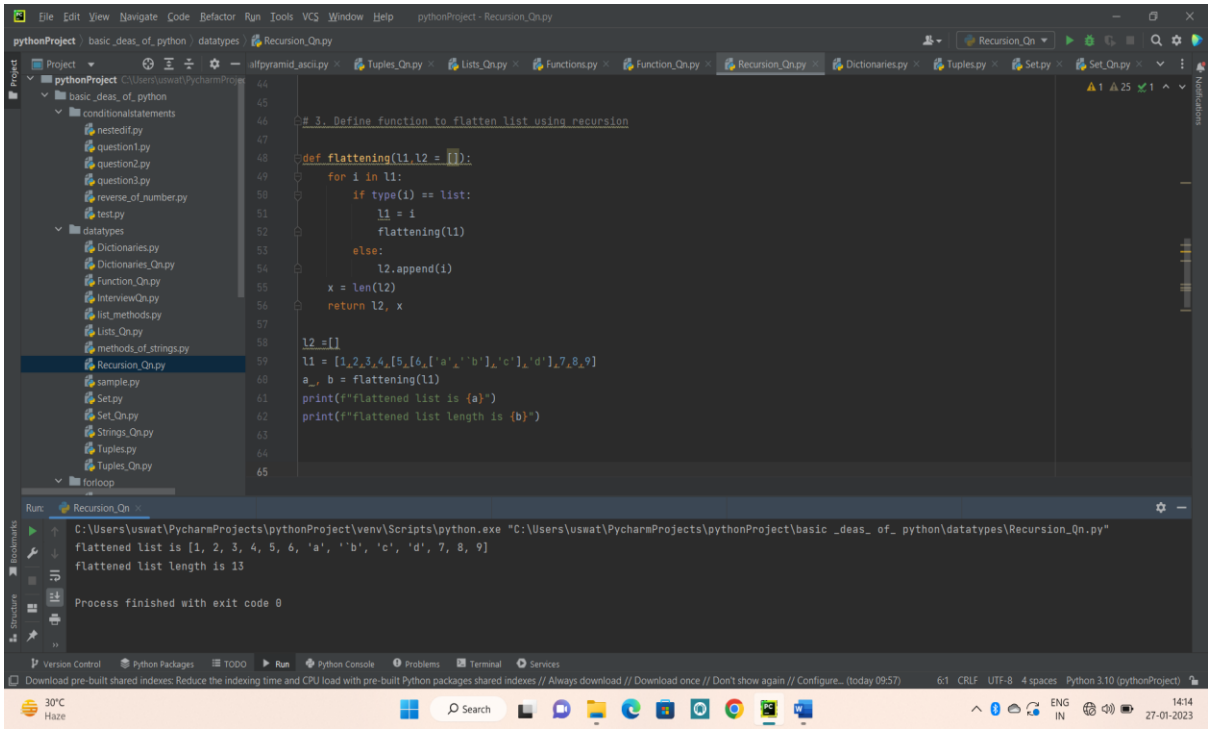
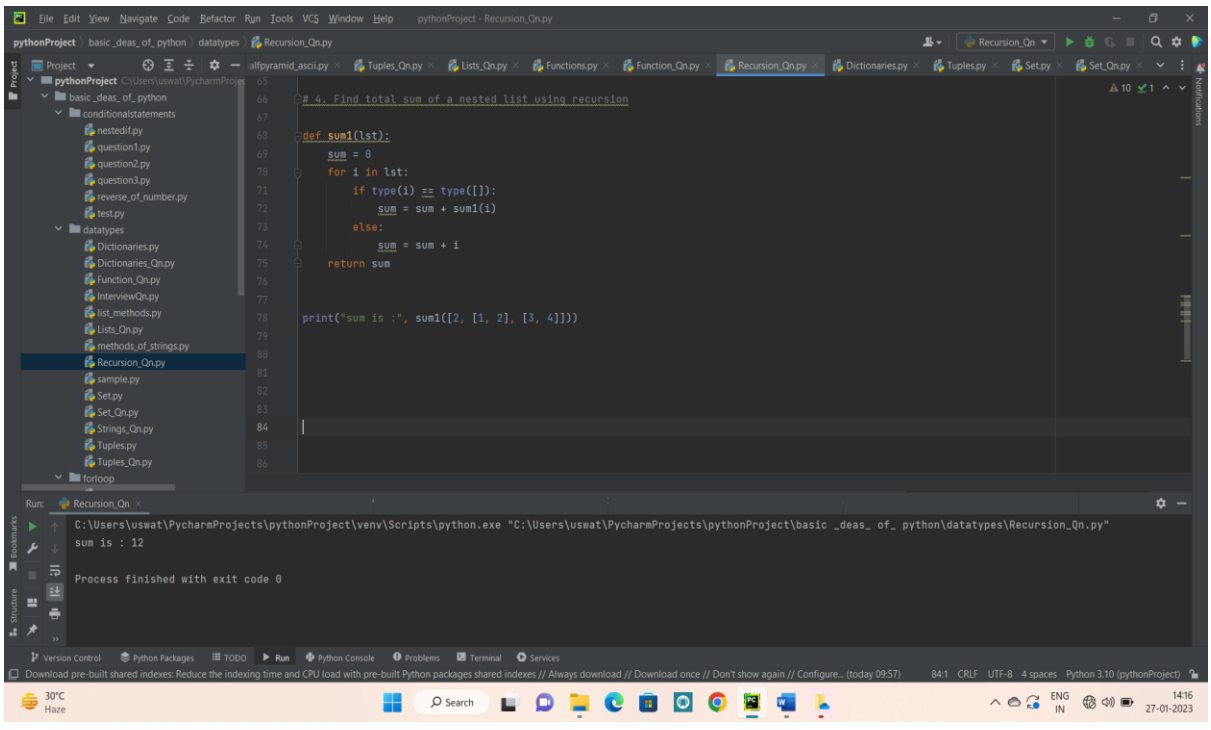
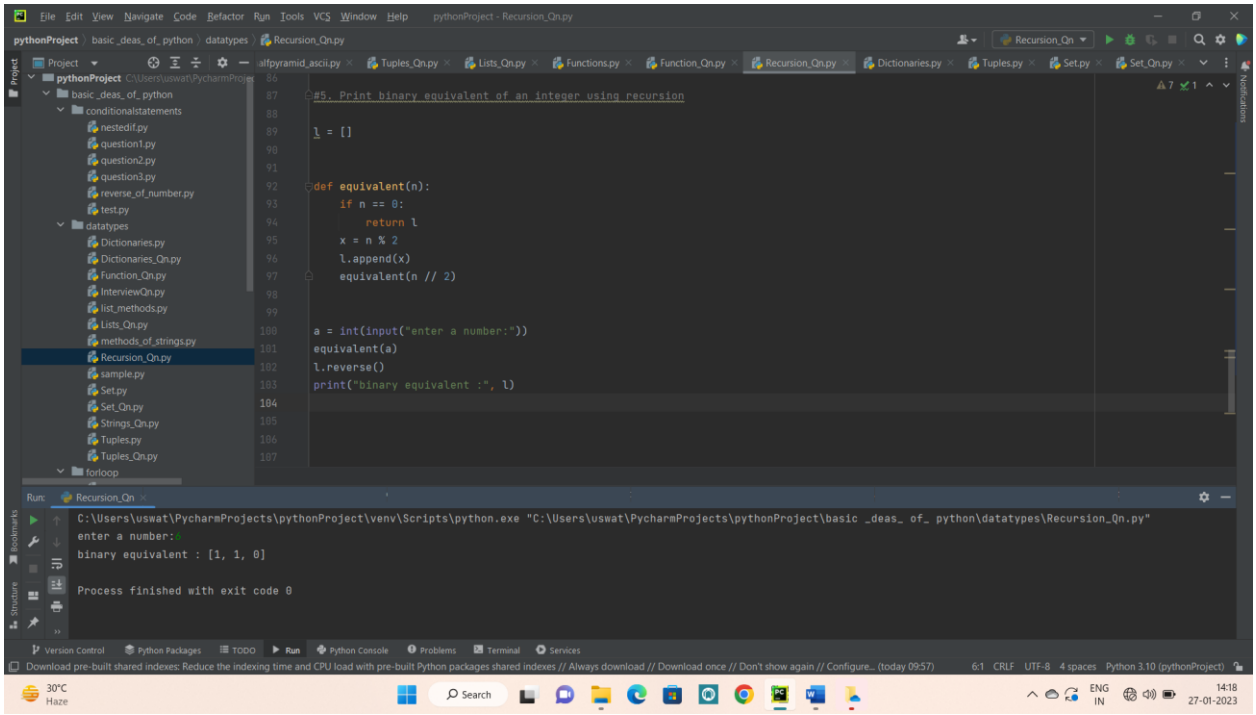
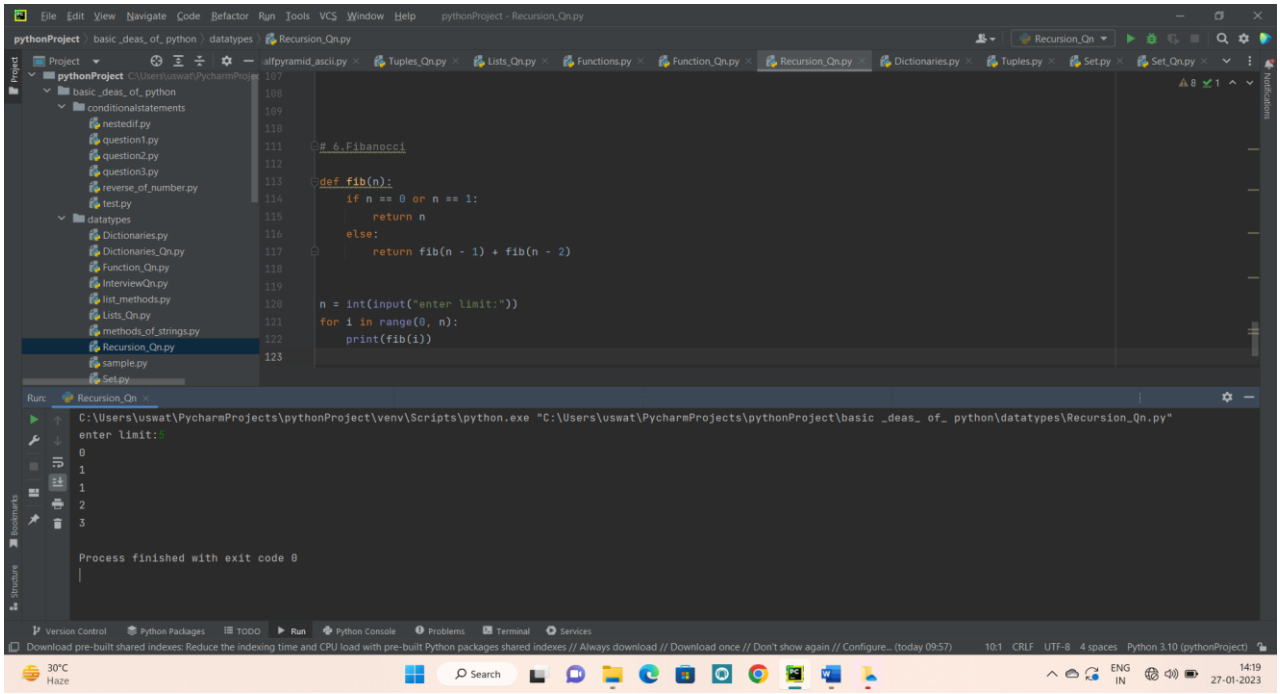

File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - Recursion_Qn.py
# 1. GCD
def GCD(a, b):
low = min(a, b)
high = max(a, b)
if low == 0:
return high
elif low == 1:
return 1
else:
return GCD(low, high % low)
print(GCD(24, 36))
C:\Users\uswat\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\Users\uswat\PycharmProjects\pythonProject\basic _deas_ of_ python\datatypes\Recursion_Qn.py"
12
Process finished with exit code 0


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - Recursion_Qn.py
# 2. Multiplication Table using recursion
def mul(n, m):
if m > 10:
return
print(n, "*", m, "=", n * m)
return mul(n, m + 1)
n = int(input("enter the number="))
print(mul(n, 1))
C:\Users\uswat\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\Users\uswat\PycharmProjects\pythonProject\basic _deas_ of_ python\datatypes\Recursion_Qn.py"
enter the number=4
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
None
Process finished with exit code 0


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - Recursion_Qn.py
# 3. Define function to flatten list using recursion
def flattening(l1,l2 = []):
for i in l1:
if type(i) == list:
l1 = i
flattening(l1)
else:
l2.append(i)
x = len(l2)
return l2, x
l2 =[]
l1 = [1,2,3,4,[5,[6,['a','b'],'c'],'d'],7,8,9]
a_, b = flattening(l1)
print(f"flattened list is {a}")
print(f"flattened list length is {b}")
C:\Users\uswat\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\Users\uswat\PycharmProjects\pythonProject\basic _deas_ of_ python\datatypes\Recursion_Qn.py"
flattened list is [1, 2, 3, 4, 5, 6, 'a', 'b', 'c', 'd', 7, 8, 9]
flattened list length is 13
Process finished with exit code 0


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - Recursion_Qn.py
# 4. Find total sum of a nested list using recursion
def sum1(lst):
sum = 0
for i in lst:
if type(i) == type([]):
sum = sum + sum1(i)
else:
sum = sum + i
return sum
print("sum is :", sum1([2, [1, 2], [3, 4]]))
C:\Users\uswat\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\Users\uswat\PycharmProjects\pythonProject\basic _deas_ of_ python\datatypes\Recursion_Qn.py"
sum is : 12
Process finished with exit code 0


#5. Print binary equivalent of an integer using recursion
l = []
def equivalent(n):
if n == 0:
return l
x = n % 2
l.append(x)
equivalent(n // 2)
a = int(input("enter a number:"))
equivalent(a)
l.reverse()
print("binary equivalent :", l)
C:\Users\uswat\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\Users\uswat\PycharmProjects\pythonProject\basic _deas_ of_ python\datatypes\Recursion_Qn.py"
enter a number:8
binary equivalent : [1, 1, 0]
Process finished with exit code 0


# 6.Fibanocci
def fib(n):
if n == 0 or n == 1:
return n
else:
return fib(n - 1) + fib(n - 2)
n = int(input("enter limit:"))
for i in range(0, n):
print(fib(i))
C:\Users\uswat\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\Users\uswat\PycharmProjects\pythonProject\basic _deas_ of_ python\datatypes\Recursion_Qn.py"
enter limit:5
0
1
1
2
3
Process finished with exit code 0