

Recursive Algorithm

Peasant Method

```
peasantAlgorithm.py ✕
1 #!/usr/bin/env python2
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu Apr 16 20:39:06 2020
5
6 @author: farshad
7 @email : fzcomputerscience@gmail.com
8 """
9
10 def peasantMultiplication(x,y):
11     if(x==0):
12         return 0
13     x1 = x/2
14     y1 = y + y
15     result = peasantMultiplication(x1,y1)
16     if(x % 2 == 0):
17         return result
18     else:
19         return result + y
20
21
22 result = peasantMultiplication(2,3)
23 print(result)
24
25
```

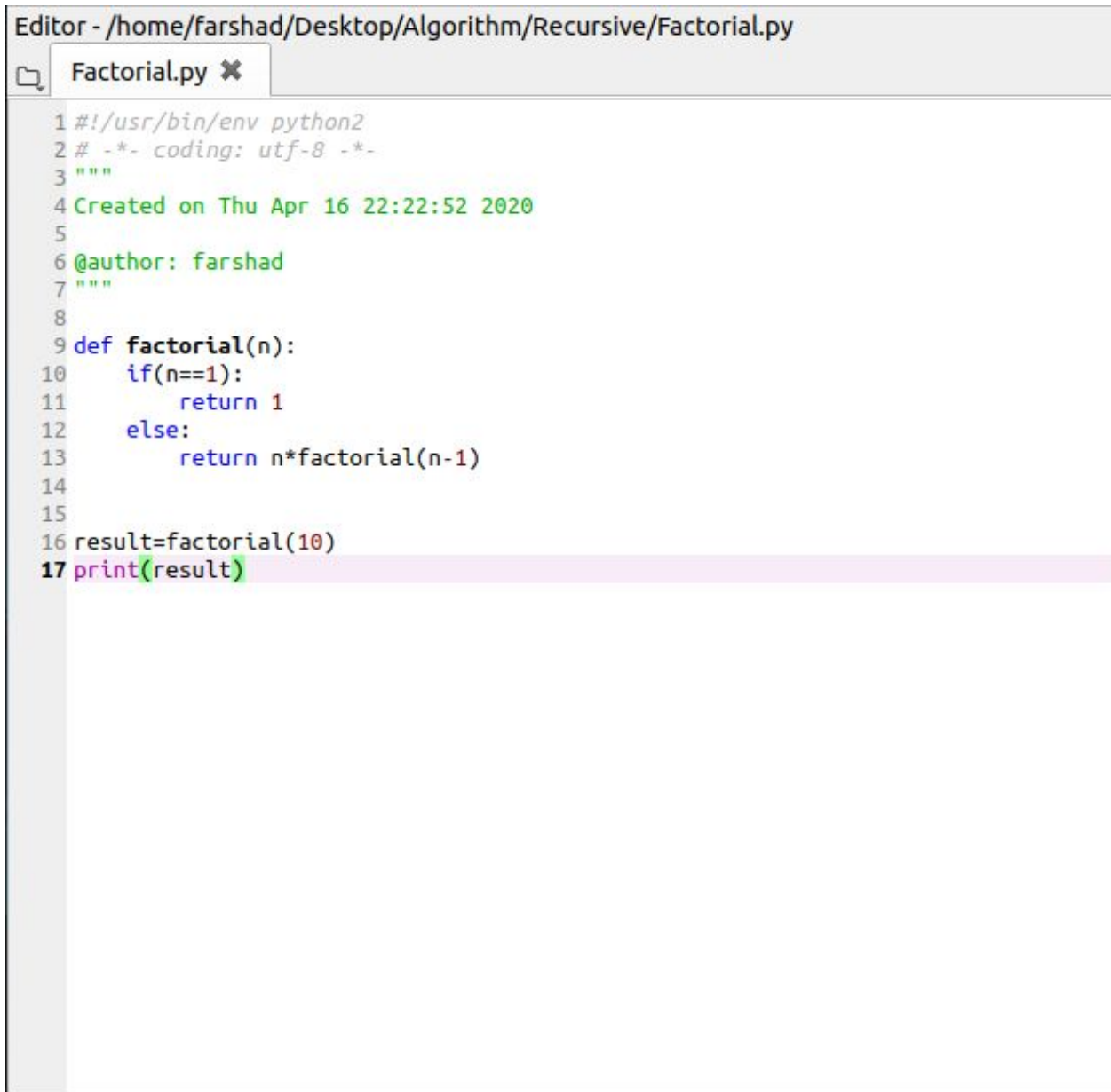
```
def peasantMultiplication(x,y):
    if(x==0):
        return 0
    x1 = x/2
    y1 = y + y
    result = peasantMultiplication(x1,y1)
    if(x % 2 == 0):
        return result
    else:
        return result + y
```

```
result = peasantMultiplication(2,3)
print(result)
```

Lattice Algorithm

- Not Ready Yet

Factorial of a Number

A screenshot of a code editor window titled "Editor - /home/farshad/Desktop/Algorithm/Recursive/Factorial.py". The editor shows a Python script for calculating the factorial of a number. The script includes a shebang line, a coding declaration, a docstring with creation date and author, a recursive function definition, and a main execution block. The line numbers 1 through 17 are visible on the left side of the editor. The line "17 print(result)" is highlighted with a light purple background.

```
1 #!/usr/bin/env python2
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu Apr 16 22:22:52 2020
5
6 @author: farshad
7 """
8
9 def factorial(n):
10     if(n==1):
11         return 1
12     else:
13         return n*factorial(n-1)
14
15
16 result=factorial(10)
17 print(result)
```

```
def factorial(n):
    if(n==1):
        return 1
    else:
        return n*factorial(n-1)
```

```
result=factorial(10)
print(result)
```

Merge Sort

Editor - /home/farshad/Desktop/Algorithm/Recursive/MergeSort.py

MergeSort.py X

```
1 """
2 Created on Sun Apr 26 16:59:08 2020
3 @author: farshad
4 @email: fzcomputerscience@gmail.com
5 """
6 def mergeSort(inputArray):
7     length = len(inputArray)
8     if(length<2):
9         return
10    else:
11        middle = length/2
12        subInputArrayLeft = inputArray[0:middle]
13        subInputArrayRight = inputArray[middle:length]
14        mergeSort(subInputArrayLeft)
15        mergeSort(subInputArrayRight)
16        merge(inputArray,subInputArrayLeft,subInputArrayRight,middle)
17
18 def merge(inputArray,subInputArrayLeft,subInputArrayRight,middle):
19     leftIndex = 0
20     rightIndex = 0
21     for i in range(len(inputArray)):
22         if(leftIndex>=middle):
23             inputArray[i] = subInputArrayRight[rightIndex]
24             rightIndex = rightIndex + 1
25         elif((rightIndex+middle)>=len(inputArray)):
26             inputArray[i] = subInputArrayLeft[leftIndex]
27             leftIndex = leftIndex + 1
28         elif(subInputArrayLeft[leftIndex]<=subInputArrayRight[rightIndex]):
29             inputArray[i] = subInputArrayLeft[leftIndex]
30             leftIndex = leftIndex + 1
31         elif(subInputArrayLeft[leftIndex]>subInputArrayRight[rightIndex]):
32             inputArray[i] = subInputArrayRight[rightIndex]
33             rightIndex = rightIndex + 1
34
35 numbers = [200,201,100,10,7,6,5,3,7,1,0,-1,0,-2,0,100,13]
36 mergeSort(numbers)
37 for i in range(len(numbers)):
38     print(numbers[i])
```

```
def mergeSort(inputArray):
    length = len(inputArray)
    if(length<2):
        return
    else:
        middle = length/2
        subInputArrayLeft = inputArray[0:middle]
        subInputArrayRight = inputArray[middle:length]
        mergeSort(subInputArrayLeft)
        mergeSort(subInputArrayRight)
        merge(inputArray,subInputArrayLeft,subInputArrayRight,middle)
```

```
def merge(inputArray,subInputArrayLeft,subInputArrayRight,middle):
    leftIndex = 0
    rightIndex = 0
    for i in range(len(inputArray)):
        if(leftIndex>=middle):
            inputArray[i] = subInputArrayRight[rightIndex]
            rightIndex = rightIndex + 1
        elif((rightIndex+middle)>=len(inputArray)):
            inputArray[i] = subInputArrayLeft[leftIndex]
            leftIndex = leftIndex + 1
        elif(subInputArrayLeft[leftIndex]<=subInputArrayRight[rightIndex]):
            inputArray[i] = subInputArrayLeft[leftIndex]
            leftIndex = leftIndex + 1
        elif(subInputArrayLeft[leftIndex]>subInputArrayRight[rightIndex]):
            inputArray[i] = subInputArrayRight[rightIndex]
            rightIndex = rightIndex + 1
```

```
numbers = [200,201,100,10,7,6,5,3,7,1,0,-1,0,-2,0,100,13]
mergeSort(numbers)
for i in range(len(numbers)):
    print(numbers[i])
```

References

Erickson, J. (2019). *Algorithms*. Etats-Unis: Jeff Erickson.