Lesson 08

CSC140 Foundations of Computer Science

25 February 2020

Examine the code in these two programs. Do you see any unfamiliar uses of the Python language here?

Merge sort

Add comments to this program.

Modify this code to produce a program that can be used to measure the amount of work in a merge sort.

```
import random

SIZE = 8

def merge( a, b ):
    i  = 0
    j = 0
    result = []

    while i < len(a) and j < len(b):
        if a[i] < b[j]:
            result.append( a[i] )
            i += 1
        else:
            result.append( b[j] )
            j += 1

    while i < len(a):
        result.append( a[i] )
        i += 1

    while j < len(b):
        result.append( b[j] )
```

```
            j += 1

        return result

        # end of merge()

    def merge_sort_step( values, j ):
        result = []
        for i in range( 0, len(values), j ):
            a = values[i:i + j//2]
            b = values[i + j//2: i + j]
            c = merge( a, b )
            result.extend( c )
        return result

        # merge_sort_step()

    def merge_sort( values ):
        if len(values) <= 1:
            return values
        else:
            k = len(values) // 2
            prefix = merge_sort( values[0:k] )
            suffix = merge_sort( values[k:] )
            return merge( prefix, suffix )

    if __name__ == "__main__":
        a = [ random.randint(10,100) for i in range(SIZE) ]
        a.sort()
        print( "a = ", a )

        b = [ random.randint(10,100) for i in range(SIZE) ]
        b.sort()
        print( "b = ", b )

        c = merge( a, b )
        print( "c = ", c )

        d = [ random.randint(10,100) for i in range(2*SIZE) ]
        print( d )
        e = merge_sort_step( d, 2 )
        print( e )
        f = merge_sort_step( e, 4 )
        print( f )
        g = merge_sort_step( f, 8 )
        print( g )
```

```
    h = [ random.randint(10,100) for i in range(2*SIZE) ]
    print( h )
    i = merge_sort( h )
    print( i )
```

Weight class

Add comments to this program.

Using the Weight class as a model, add a definition of a class that models a length that is measured in feet and inches.

```
class Weight:
    OUNCES_IN_A_POUND = 16

    def __init__(self, pounds, ounces ):
        self.pounds = pounds + ounces // Weight.OUNCES_IN_A_POUND
        self.ounces = ounces % Weight.OUNCES_IN_A_POUND

    def add(self, other_weight):
        lbs = (self.pounds + other_weight.pounds +
            (self.ounces + other_weight.ounces) //
            Weight.OUNCES_IN_A_POUND)

        oz = ((self.ounces + other_weight.ounces) %
            Weight.OUNCES_IN_A_POUND)

        return Weight( lbs, oz )

    def __str__(self):
        return (str(self.pounds) + " lbs., " +
            str(self.ounces) + " oz.")

def main():
    a = Weight( 3, 14 )
    b = Weight( 2, 10 )
    print( "a = ", a )
    print( "b = ", b )
    print( "a + b = ", a.add(b) )

if __name__ == "__main__":
    main()
```