ProgrammingAssignment5

April 5, 2023

Author: J. Dittenber Assignment 5: iNeuron.ai FullStack Data Science Boot Camp

1. Write a Python Program to Find LCM?
2. Write a Python Program to Find HCF?
3. Write a Python Program to Convert Decimal to Binary, Octal and Hexadecimal?
4. Write a Python Program To Find ASCII value of a character?
5. Write a Python Program to Make a Simple Calculator with 4 basic mathematical operations?

1 1. Write a Python Program to Find LCM.

We can use some properties of integers as well as the Euclidean Algorithm.

First, note that we can calculate the $lcm(a, b)$ from the theorem:

For all

$$a, b \in \mathbb{Z}^+, a \cdot b = lcm(a, b) \cdot gcd(a, b).$$

Note that the HCF (highest common factor) is the same thing as the GCD (greatest common divisor) which is mentioned in the Euclidean Algorithm.

The Euclidean Algorithm allows one to generate a sequence of remainders and the remainder that is one term before a remainder equal to zero is the GCD, i.e. HCF

The procedure is like this: To find the $gcd(a, b)$ we first write a in terms of b assuming $a > b$.

For example, to find the $gcd(112, 64)$ we would start by writing a as such

$$112 = 1 \times 64 + 48$$

Next, we shift all of the values as such

$$64 = 1 \times 48 + 16$$

And again, until we reach a remainder of 0

$$48 = 3 \times 16 + 0$$

In other words, we use the algorithm to find the remainder in the second to last position (the last position being $r_n = 0$)

More formally,

$$a = q_o b + r_0$$
$$b = q_1 r_0 + r_1$$
$$r_o = q_2 r1 + r_2$$
$$\vdots$$
$$r_{n-1} = gcd(a, b)$$
$$r_n = 0$$

2 #2 will come first so that I can use it to solve #1

2.1 #2 Write a program to find the HCF

[1]:
```
def euclid(a, b):
    ''' We can find the gcd (equivalently, the HCF) by using the algorithm and␣
 ↪returning
    value of the remainder before r=0 using while
    '''
    while b != 0:
        r = b
        b = a % b
        a = r
    return a
```

[2]:
```
euclid(84,12)
```

[2]: 12

3 Finding the LCM

3.1 Note that we have for any $a, b \in \mathbb{Z}^+$ where $ a b$ we can get

3.2

$$a \cdot b = lcm(a, b) * gcd(a, b)$$

3.3 Thus we can rearrange this as such

3.3.1

$$lcm(a, b) = \frac{a \cdot b}{gcd(a, b)}$$

[3]:
```
def euclid_2(a,b):
    '''
    This function will take in two values
    then, it will use the formula above and call the function
    I wrote in the prevous block
    '''
```

```
    c,d = a,b
    gcd = euclid(a,b)
    return int((c*d)/gcd)
```

[4]:
```
euclid_2(9,12)
```

[4]: 36

3.4 3. Write a Python Program to Convert Decimal to Binary, Octal and Hexadecimal?

Decimal to Binary To convert from decimal to binary you must divide the number by 2 and store the remainder each time until the quotient is 0. Then write the remainders in reverse order.

Example: Convert 25 to binary.

1. $$25 \div 2 = 12r1$$
2. $$12 \div 2 = 6r0$$
3. $$6 \div 2 = 3r0$$
4. $$3 \div 2 = 1r1$$
5. $$1 \div 2 = 0r1$$

The remainders in reverse order:

$$11001$$

3.4.1 Decimal to Octal

To convert from decimal to octal you must divide the number by 8 and store the remainder each time until the quotient is 0. Then, write the remainders in reverse order.

Example: Convert 45 to octal.

1. $$45 \div 8 = 5r5$$
2. $$5 \div 8 = 0r5$$

The remainders in reverse order :

$$55$$

To convert from decimal to hexadecimal you must divide the number by 16 and store the remainder each time until the quotient is 0. Then, write the remainders in reverse order. If the remainder is greater than 9, then it is represented by a letter as -> A:10, B:11, C:12, etc.

Exmaple Convert 255 to hexadecimal.

1. $$255 \div 16 = 15r15$$
2. $$15 \div 16 = 0r15$$

Write the remainders in reverse order to get:

$$1515$$

Note $A : 10, B : 11, C : 12, D : 13, E : 14, F : 15$

Thus, we get

$$FF$$

[37]:
```
# The methods are using a static method because they are related to the␣
↪Converter class,but do not depend on any instance-specific state or behavior.

class Converter:
    @staticmethod
    def dec_to_base(n, base):
        conv = []
        while n >0:
            rem = n%base
            if rem < 10:
                conv.append(str(rem))
            else:
                conv.append(chr(rem-10 + ord('A')))
            n= n//base
        conv.reverse()
        return ''.join(conv)

    @staticmethod
    def dec_to_bin(n):
        return Converter.dec_to_base(n, 2)

    @staticmethod
    def dec_to_oct(n):
        return Converter.dec_to_base(n, 8)

    @staticmethod
    def dec_to_hex(n):
        return Converter.dec_to_base(n, 16)

    @staticmethod
    def dec_to_ascii(n):
        if n < 0 or n > 127:
```

```
            raise ValueError("Input value must be in the range of 0 to 127.") ␣
↪#valid ascii character range is (0,127)
        return chr(n)
```

[38]:
```
c = Converter()
```

[39]:
```
c.dec_to_bin(25)
```

[39]:
```
'11001'
```

[40]:
```
c.dec_to_oct(45)
```

[40]:
```
'55'
```

[41]:
```
c.dec_to_hex(357)
```

[41]:
```
'165'
```

4 4. Write a Python Program To Find ASCII value of a character?

This can be included in the converter class. Note, that there is an exception for the case that the input value is out of range.

[43]:
```
c.dec_to_ascii(126)
```

[43]:
```
'~'
```

5 5. Write a Python Program to Make a Simple Calculator with 4 basic mathematical operations?

[72]:
```
class Calculator:
    def __init__(self, *args):
        self.args = args

    def add(self):
        return sum(self.args)

    def sub(self):
        result = self.args[0]              #must pay attention to order of␣
↪operations
        for i in range(1, len(self.args)):
            result -= self.args[i]
        return result

    def mult(self):
```

```
        result = 1
        for arg in self.args:
            result *= arg
        return result

    def div(self):
        if 0 in self.args:
            return f"Cannot divide by zero."
        result = self.args[0]                #must pay attention to order of␣
↪operations
        for i in range(1, len(self.args)):
            result /= self.args[i]
        return result
```

[83]:
```
calc = Calculator(1000,10,10,10)
```

[84]:
```
calc.add()
```

[84]: 1030

[89]:
```
calc.sub() #note that substraction goes left to right
```

[89]: 970

[86]:
```
calc.mult()
```

[86]: 1000000

[88]:
```
calc.div() #note that the division goes left to right
```

[88]: 1.0

[ ]:
```

```