**WEEK-9**

1).complete function to implement coin change making problem i.e. finding the minimum

number of coins of certain denominations that add up to given amount of money.

The only available coins are of values 1, 2, 3, 4

**Input Format:**

Integer input from stdin.

**Output Format:**

return the minimum number of coins required to meet the given target.

**Example Input**:

16

**Output:**

4

**Explanation:**

We need only 4 coins of value 4 each

**Example Input:**

25

**Output:**

7

**Explanation:**

We need 6 coins of 4 value, and 1 coin of 1 value

**PROGRAM:**

def coinChange(n):

dp = [float('inf')] \* (n + 1)

dp[0] = 0

coins = [1, 2, 3, 4]

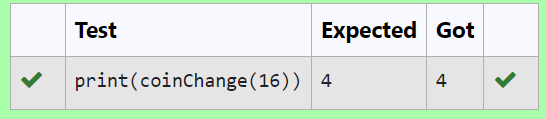
for coin in coins:

for i in range(coin, n + 1):

dp[i] = min(dp[i], dp[i - coin] + 1)

return dp[n]

**OUTPUT:**



2) An abundant number is a number for which the sum of its proper divisors is greater than

the number itself. Proper divisors of the number are those that are strictly lesser than the number.

**Input Format:**

Take input an integer from stdin

**Output Format:**

Return Yes if given number is Abundant. Otherwise, print No

**Example input:**

12

**Output:**

Yes

**Explanation**

The proper divisors of 12 are: 1, 2, 3, 4, 6, whose sum is 1 + 2 + 3 + 4 + 6 = 16. Since sum of

proper divisors is greater than the given number, 12 is an abundant number.

**Example input:**

13

**Output:**

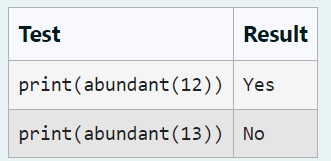
No

**Explanation**

The proper divisors of 13 is: 1, whose sum is 1. Since sum of proper divisors is not greater

than the given number, 13 is not an abundant number.

**For example:**

****

**PROGRAM:**

def abundant(number):

divisor\_sum=sum([divisor for divisor in range(1,number) if number % divisor==0])

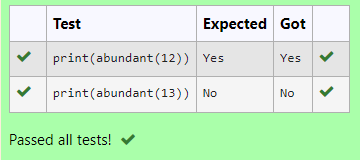
if divisor\_sum>number:

return "Yes"

else:

return "No"

**OUTPUT:**



3) Given a number with maximum of 100 digits as input, find the difference between the sum

of odd and even position digits.

**Input Format:**

Take a number in the form of String from stdin.

**Output Format:**

Print the difference between sum of even and odd digits

**Example input:**

1453

**Output:**

1

**Explanation:**

Here, sum of even digits is 4 + 3 = 7

sum of odd digits is 1 + 5 = 6.

Difference is 1.

Note that we are always taking absolute difference

**PROGRAM:**

def differenceSum(number\_str):

number\_str=str(number\_str)

odd\_sum = 0

even\_sum = 0

for index, char in enumerate(number\_str):

digit = int(char)

if (index + 1) % 2 == 0:

even\_sum += digit

else:

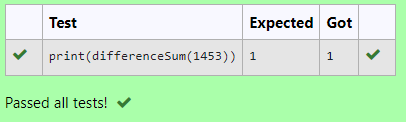
odd\_sum += digit

difference = abs(even\_sum - odd\_sum)

return difference

print(differenceSum(number\_str))

**OUTPUT:**



4) Write a code to check whether product of digits at even places is divisible by sum of digits

at odd place of a positive integer.

**Input Format:**

Take an input integer from stdin.

**Output Format**:

Print TRUE or FALSE.

**Example Input:**

1256

**Output**:

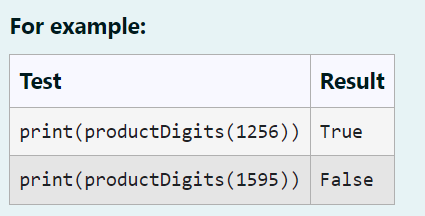
TRUE

**Example Input:**

1595

**Output:**

FALSE



**PROGRAM:**

def productDigits(number):

number\_str = str(number)

product\_even = 1

sum\_odd = 0

for i, digit\_char in enumerate(number\_str):

digit = int(digit\_char)

if (i+1)%2 == 0:

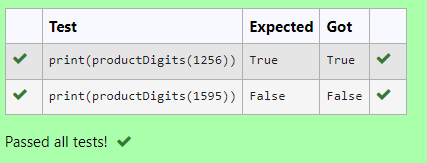
product\_even \*= digit

else:

sum\_odd += digit

return product\_even % sum\_odd == 0

**OUTPUT**:



5) An automorphic number is a number whose square ends with the number itself.

For example, 5 is an automorphic number because 5\*5 =25. The last digit is 5 which same

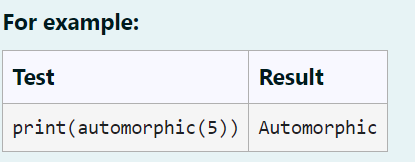
as the given number.

If the number is not valid, it should display “Invalid input”.

If it is an automorphic number display “Automorphic” else display “Not Automorphic”.

Input Format:

Take a Integer from Stdin Output Format: Print Automorphic if given number is Automorphic number,otherwise Not Automorphic Example input: 5 Output: Automorphic Example input: 25 Output: Automorphic Example input: 7 Output: Not Automorphic



**PROGRAM:**

def automorphic(n):

b=n\*\*2

if b%10==n:

return "Automorphic"

else:

return "Not Automorphic"

**OUTPUT:**

