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):

coins=[1,2,3,4]

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

dp[0]=0

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

for coin in coins:

if coin<=i:

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

return dp[n]

output:

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | print(coinChange(16)) | 4 | 4 |  |

Passed all tests!

**Correct**

2. 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

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(automorphic(5)) | Automorphic |

Program:

def automorphic(number):

if number<0:

return "invalid input"

square=number\*number

number\_str=str(number)

square\_str=str(square)

if square\_str.endswith(number\_str):

return "Automorphic"

else:

return "Not Automorphic"

output:

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | print(automorphic(5)) | Automorphic | Automorphic |  |
|  | print(automorphic(7)) | Not Automorphic | Not Automorphic |  |

Passed all tests!

**Correct**

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):

number\_str = str(number)

even\_sum = 0

odd\_sum = 0

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

if i % 2 == 0:

even\_sum += int(digit)

else:

odd\_sum += int(digit)

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

return difference

try:

number = input()

print(differenceSum(int(number)), end="")

except EOFError:

print(end="")

except ValueError:

print(end="")

output:

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | print(differenceSum(1453)) | 1 | 1 |  |

Passed all tests!

**Correct**

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

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(productDigits(1256)) | True |
| print(productDigits(1595)) | False |

Program:

def productDigits(num):

# Convert the number to a string to iterate over its digits

num\_str = str(num)

# Initialize variables to store the product of digits at even places

# and the sum of digits at odd places

product\_even = 1

sum\_odd = 0

# Iterate over the digits

for i in range(len(num\_str)):

digit = int(num\_str[i])

# Check if the digit is at an even place

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

# Multiply the digit with the current product of digits at even places

product\_even \*= digit

else:

# Add the digit to the sum of digits at odd places

sum\_odd += digit

# Check if the product of digits at even places is divisible by the sum of digits at odd places

if product\_even % sum\_odd == 0:

return True

else:

return False

output:

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | print(productDigits(1256)) | True | True |  |
|  | print(productDigits(1595)) | False | False |  |

Passed all tests!

**Correct**

5. An e-commerce company plans to give their customers a special discount for Christmas.

They are planning to offer a flat discount. The discount value is calculated as the sum of all

the prime digits in the total bill amount.

Write an algorithm to find the discount value for the given total bill amount.

Constraints

1 <= orderValue< 10e100000

Input

The input consists of an integer orderValue, representing the total bill amount.

Output

Print an integer representing the discount value for the given total bill amount.

Example Input

578

Output

12

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(christmasDiscount(578)) | 12 |

Program;

def is\_prime(num):

if num < 2:

return False

for i in range(2, int(num \*\* 0.5) + 1):

if num % i == 0:

return False

return True

def christmasDiscount(orderValue):

# Convert the orderValue to a string to iterate over its digits

order\_str = str(orderValue)

# Initialize discount value

discount = 0

# Iterate over the digits

for digit in order\_str:

if is\_prime(int(digit)):

discount += int(digit)

return discount

output:

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | print(christmasDiscount(578)) | 12 | 12 |  |

Passed all tests!

**Correct**