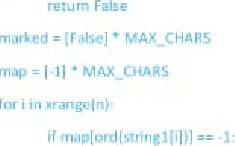
CAS0888 - PYTHON PROGRAMMING

inglr s rringn:

n • lenÄring2)





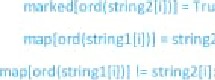
return

for

it markedlard(slringafrl'l••; True:

return

elif return False return True



True

strir.E21ii



2.iNPUT:

det od€\_sum - 0



num i"



even sum num \*\* 2

else:

odd sum += num \*\* 2 return [odd\_sum, even\_sum] numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9] result = sumsquare(numbers) print(result)

3.1NPUT:

def is\_happy(n):

def get\_next(num):

next num = O while num > O:

digit = num % 10 next\_num += digit \*\* 2

num //= 10 return next num slow = n fast = n while True:

slow = get\_next(slow) fast = get\_next(get\_next(fast)) if slow == fast: break return slow == 1 num = 19 print(is\_happy(num))

4.1NPUT: def is\_palindrome(x):

if x < O:

return False original =x reverse = O while x > O:

# 10

reverse = reverse \* 10 + digit

x 10

return original - reverse numl = 121 num2 = 123 print(is\_palindrome(numl)) print(is\_palindrome(num2))

5.1NPUT:

def calculate\_price(num\_fresh, num\_day\_old):

regular\_price = (num\_fresh + num\_day\_old) \* 185 discount = num\_day\_old \* 185 \* 0.60 total\_price = regular\_price - discount return regular\_price, discount, total\_price

def main():

try:

num\_fresh = int(input("Enter the number of fresh loaves purchased: ")) num\_day\_old = int(input("Enter the number of day-old loaves purchased:

if num\_fresh < O or num\_day\_old < O:

print("Please enter non-negative numbers of loaves.") else:

regular\_price, discount, total\_price = calculate\_price(num\_fresh, num\_day\_old) print("\nRegular Price: {:.2f} rupees" .format(regular\_price)) print("Discount: {:.2f} rupees" .format(discount))

print("Total Price: {:.2f} rupees" .format(total\_price)) except ValueError:

print("lnvalid input. Please enter valid integers.")

if namemain

main()

6.1NPUT: def max\_area(height):

left = O right = len(height) - 1 max area = O while left < right:

h = min(height[left], height[right]) w = right - left area = h \* w max\_area = max(max\_area, area) if height[left] < height[right]:

left += 1 else:

right -= 1 return max area heights = print(max\_area(heights))

7.1NPUT:

def count\_sorted\_vowel\_strings(n):

vowels = 'aeiou' def generate\_strings(curr\_str, index):

if len(curr\_str) == n.

return 1 count O

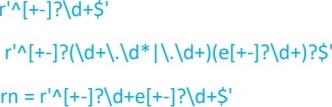
for i in range(index, len(vowels)):

count += generate\_strings(curr\_str + vowels[i], i) return count return generate\_strings(", O)

print(count\_sorted\_vowel\_strings(n))

8.1NPUT:

import re def isNumber(s):

s = s.strip() integer\_pattern = decimal\_pattern = exponential\_pattern if re.match(integer\_pattern, s):

return True if re.match(decimal\_pattern, s):

return True if re.match(exponential\_pattern, s):

return True return False

valid\_numbers = ["2", "0089 , " "+3. 14 "-.9", "2e10", "-902", "3e+7",

"53.5e93", "-123.456e789"] invalid\_numbers = ["abc " , "la", "le", "e3", "99e2 5 ," "95a54e53"] for num in valid numbers:

print(f"{num}: {isNumber(num)}") for num in invalid numbers:

print(f"{num}: {isNumber(num)}")

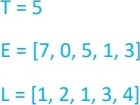
9.1NPUT:

def max\_guests\_within\_time\_limit(T, E, L):

max\_guests O

current\_guests = O for i in range(T):

current\_guests += Eli] - L[i] max\_guests = max(max\_guests, current\_guests) return max\_guests



result = max\_guests\_within\_time\_limit(T, E, L) print(result) IO.INPUT:

def modify\_string(s):

def circular\_distance(ch):

return (ord(ch) - ord('a') + 1) % 26 char\_frequency = {} modified\_string = for ch in s:

if ch in char\_frequency:

char\_frequency[ch] += 1 else:

char\_frequency[ch] = 1 for ch in s:

freq = char\_frequency[ch] distance = circular\_distance(ch) new\_char = chr((ord('a') + distance - 1) % 26 + ord('a')) modified\_string += new\_char return modified\_string input\_string = "abbccc" result modify\_string(input\_string)

print(result)