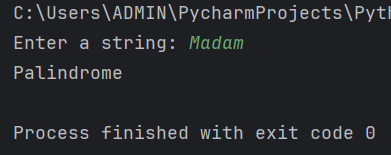
Python Journal

Q1) Write a program to determine if a given string is palindrome or not using combination of positive and negative indexing. Take the string as an input from the user.

-> String\_Methods.py # file containing all the string methods  
def palindromeCheck(s):   
 s = s.lower() # converts the entered input into lowercase  
 isPalindrome = True # flag variable initially set to True  
 length = len(s) // 2   
# we take half the length of input since we need to match first half with second half  
 for i in range(length):  
 if s[i] != s[-(i + 1)]:   
# if the first and second half don't match, we make isPalindrome to False, thus indicating not a palindrome  
 isPalindrome = False  
 if isPalindrome:  
 print("Palindrome")  
 else:  
 print("Not a palindrome")

-> String\_User\_Input.py # file for taking user input as string  
from String\_Methods import palindromeCheck # import file and its method  
string = input("Enter a string: ") # takes input from the user  
palindromeCheck(string) # function call

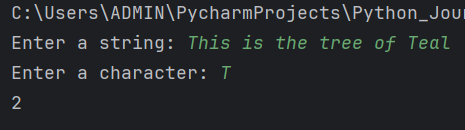
Output:



Q2) Without using count() demonstrate the use of for loop to determine the number of occurences of a given character in a string. Take the string and character from the user.

-> String\_Methods.py  
def occurencesOfChar(s, c):   
# method to check the occurences of character in a string  
 count = 0  
 for i in s:  
 if i == c:   
# if the occurence of character matches the index, we increment the count  
 count += 1  
 print(count)

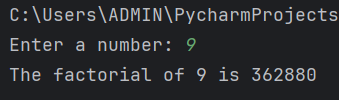
-> String\_User\_Input.py  
from String\_Methods import occurencesOfChar  
string = input("Enter a string: ") # takes input from the user  
ch = input("Enter a character: ")  
occurencesOfChar(string, ch) # function call

  
Output:

Q3) Without using readymade methods, write a program to find factorial of a given number. Take the number from the user.

-> Integer\_Methods.py # file containing all the integer methods  
def calculateFactorial(num): # method to calculate the factorial of a number  
 fact = 1  
 for i in range(1, num + 1):  
 fact = fact \* i  
 print(f"The factorial of {num} is {fact}")

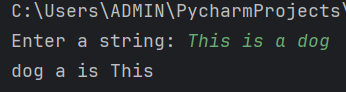
-> Integer\_User\_Input.py # file for taking user input as an integer  
from Integer\_Methods import calculateFactorial  
n = int(input("Enter a number: ")) # takes input from the user  
calculateFactorial(n) # function call

  
Output:

Q4) Without using any readymade methods, write a program in Python to reverse the sequence of words in a given string. Take the string from the user.

-> String\_Methods.py  
def reverseWords(string):   
# method to reverse the sequence of words in a string  
 st = string.split() # splits the words and creates a list  
 reverse\_str = st[::-1] # reverse the order of words using indexing  
 output = ' '.join(reverse\_str) # joins all the reverse words  
 print(output)

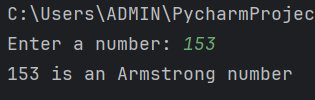
-> String\_User\_Input.py  
from String\_Methods import reverseWords  
s = input("Enter a string: ") # takes input from the user  
reverseWords(s) # function call

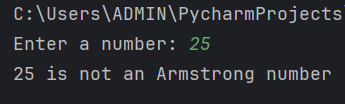
  
Output:

Q5) Without using any readymade methods, write a program in Python to check if the given number is an Armstrong number or not. Take the number from the user.

-> Integer\_Methods.py:  
def armstrongCheck(num):   
# method to check if the number is armstrong or not  
 temp = num   
# temporary variable which holds same value as the entered input  
 sum = 0  
 while num > 0:  
 rem = num % 10  
 sum = sum + (rem \* rem \* rem)  
 num = num // 10  
 if sum == temp:  
 print(f"{temp} is an Armstrong number")  
 else:  
 print(f"{temp} is not an Armstrong number")

-> Integer\_User\_Input.py  
from Integer\_Methods import armstrongCheck  
n = int(input("Enter a number: ")) # takes input from the user  
armstrongCheck(n) # function call

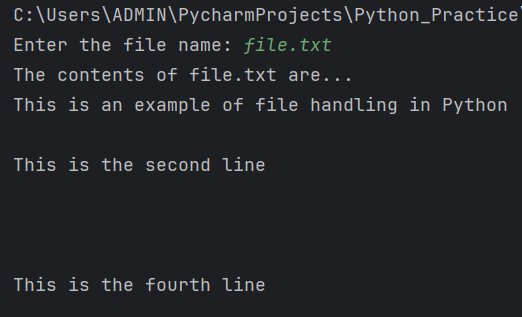
  
Output:



Q6) Without using readline() demonstrate a way in Python to read a multiline file line by line.

-> File\_Methods.py # file containing all the file handling methods  
def readMultiLines(file):  
# method to read a multiline file line by line without using readlines()  
 try:  
 f = open(file)   
# opens the specified file in 'r' mode which is default mode f.seek(0) # starts the file pointer from 0 print(f"The contents of {file} are...") for i in f: print(i) # prints the contents line by line  
 except FileNotFoundError: # if file not found  
 print("File does not exist")

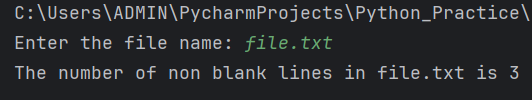
-> File\_User\_Input.py # file for taking user input as string (filename)  
from File\_Methods import readMultiLines  
fname = input("Enter the file name: ")readMultiLines(fname)

  
Output:

Q7) Using readlines() demonstrate a way to return the total number of NON BLANK lines in a file.

-> File\_Methods.py  
def nonBlankLines(file):  
# method to return the total number of non-blank lines in a file  
 try:  
 f = open(file)  
 lines = f.readlines() # reads all the lines from the specified file  
 count = 0  
 for i in lines:  
 if i.strip():   
# removes extra whitespaces thus returning only non-blank lines  
 count += 1  
 print(f"The number of non blank lines in {file} is {count}")  
 except FileNotFoundError: # if file not found  
 print("File does not exist")

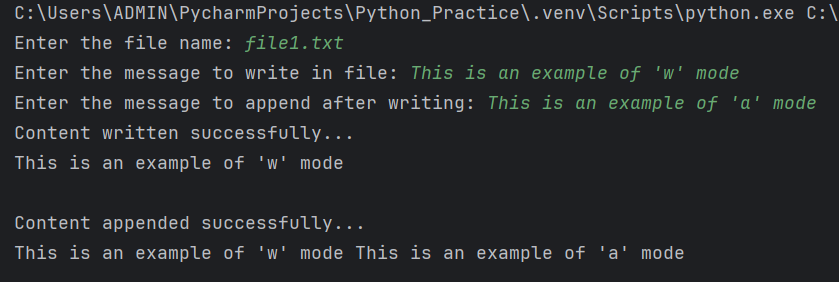
-> File\_User\_Input.py  
from File\_Methods import nonBlankLines  
fname = input("Enter the file name: ")  
nonBlankLines(fname)

  
Output:

Q8) Using file writing methods, write a message from the user in a file. Show use of write when the file is in 'w' mode and 'a' mode.

-> File\_Methods.py  
def writeReadDemo(file, msg, ap\_msg):  
# method to demonstrate the usage of 'w' and 'a' mode in file  
 try:  
 f = open(file, 'w') # opens the specified file in 'w' mode  
 f.write(msg) # writes the user defined message in the file  
 print("Content written successfully...")  
 f = open(file)  
 print(f.read()) # prints the contents of file after writing  
 f = open(file, 'a') # opens the specified file in 'a' mode  
 f.write(ap\_msg)  
 print("Content appended successfully...")  
 f = open(file)  
 print(f.read())  
 except FileNotFoundError: # if file not found  
 print("File does not exist")

-> File\_User\_Input.py  
from File\_Methods import writeReadDemo  
fname = input("Enter the file name: ")  
message = input("Enter the message to write in file: ")  
append\_msg = input("Enter the message to append after writing: ")  
writeReadDemo(fname, message, append\_msg)

  
Output: