**CE156 COURSEWORK SPRING 2023**

**Exercise 1:**

from datetime import datetime, date  
  
# User Given input for their date of birth  
user\_dob = input("Enter your date of birth (mm/dd/yyyy): ")  
  
#try-except block is used to handle the case where the user enters an invalid date format.  
#enables us to more effectively manage issues that may happen at runtime and improve user experience.  
try:  
 # Convert the input string to a date object  
 dob = datetime.strptime(user\_dob, "%m/%d/%Y").date()  
  
 # Get today's date  
 today = date.today()  
  
 # Check if the user has had their birthday this year  
 birthday\_thisyear = (today.month, today.day) >= (dob.month, dob.day)  
  
 # Calculate the user's age  
 user\_age = today.year - dob.year - int(not birthday\_thisyear)  
  
 # Output the user's age  
 print("Your age is:", user\_age)  
  
 # Output European format Date  
 dob\_europe = dob.strftime("%d/%m/%Y")  
 print("Date of birth (European format):", dob\_europe)  
  
except ValueError:  
 # Handle invalid input  
 print("Invalid date format, please enter in the format mm/dd/yyyy.")

**Output:**

**Enter your date of birth (mm/dd/yyyy): 12/01/1998**

**Your age is: 24**

**Date of birth (European format): 01/12/1998**

**Process finished with exit code 0**

**Exercise 2:**

def is\_all\_prime\_number(num):  
 *"""Returns True if n is a prime; else, False."""* # Chunk of code to heck for special cases  
 if num <= 1:  
 return False  
 elif num <= 3:  
 return True  
 # Check if whether num is divisible by 2 or 3  
 elif num % 2 == 0 or num % 3 == 0:  
 return False  
 # Checking for other numbers divisors up to sqrt(n)  
 i = 5  
 while i \* i <= num:  
 if num % i == 0 or num % (i + 2) == 0:  
 return False  
 i += 6  
 return True  
  
def get\_all\_non\_primes(a, b):  
 *"""List of all non-prime numbers between a and b is returned (inclusive)"""* # Swap a and b if a > b  
 if a > b:  
 a, b = b, a  
 # Empty List used for all non prime numbers  
 non\_primes = []  
 # Checking each number between a and b  
 for n in range(a, b + 1):  
 # If n is not prime, add it to the list  
 if not is\_all\_prime\_number(n):  
 non\_primes.append(n)  
 return non\_primes  
  
 # Get input from user and check validity  
while True:  
 try:  
 # Get two integers from user  
 x1 = int(input("Enter first positive integer: "))  
 x2 = int(input("Enter second positive integer: "))  
 # Check that both integers are positive  
 if x1 <= 0 or x2 <= 0:  
 print("Both integers must be positive")  
 else:  
 # Exit loop if input is valid  
 break  
 except ValueError:  
 print("Invalid input, please enter integers only")  
  
 # Get list of non-primes and output 10 per line  
non\_primes = get\_all\_non\_primes(x1, x2)  
print("All non-prime numbers between "+str(x1)+" and "+str(x2))  
for i in range(0, len(non\_primes), 10):  
 # Output each sublist of 10 non-primes on a separate line  
 print(\*non\_primes[i:i + 10])

**Output:**

**Enter first positive integer: 46**

**Enter second positive integer: 80**

**All non-prime numbers between 46 and 80**

**46 48 49 50 51 52 54 55 56 57**

**58 60 62 63 64 65 66 68 69 70**

**72 74 75 76 77 78 80**

**Process finished with exit code 0**

**Exercise 3:**

#########################################################################################  
# Exercise 3 (A)  
#########################################################################################  
  
def is\_str\_palindrome(s: str) -> bool:  
 *"""  
 Determines whether or not the input string is a palindrome.  
 If a string reads the same forward as it does backward, it is referred to as a palindrome..* ***:parameter*** *s: input string to be checked* ***:return****: True if "s" is a palindrome, else False  
 """* # Lowercase the string and eliminate any spaces or other special characters.  
 s = ''.join(filter(str.isalnum, s.lower()))  
 # Compare the reversed string with the original string  
 return s == s[::-1]

**print(is\_str\_palindrome("racecar")) # True**

**print(is\_str\_palindrome("hello")) # False**

**OUTPUT 3(A)**

**True**

**False**

#########################################################################################  
# Exercise 3 (B)  
#########################################################################################  
  
def up\_most\_frequent\_letter(s: str) -> str:  
 *"""  
 returns the input string's most frequent letter or digit..  
 Characters that are neither letters nor digits are ignored.  
 """* # Convert string to uppercase  
 s = s.upper()  
 # Get rid of all characters that aren't letters or numbers.  
 s = ''.join(filter(str.isalnum, s))  
 # Making a dictionary to keep track of the frequency of each letter or number.  
 frq\_dct = {}  
 for char in s:  
 if char in frq\_dct:  
 frq\_dct[char] += 1  
 else:  
 frq\_dct[char] = 1  
 # Find the letter or numeral that appears the most frequently.  
 most\_frqt = max(frq\_dct, key=frq\_dct.get)  
 return most\_frqt

**print(up\_most\_frequent\_letter("abbbcdeee"))**

**print(up\_most\_frequent\_letter("1a1b1c1d1e1f"))**

**OUTPUT 3(B)**

**E**

**1**

#########################################################################################  
# Exercise 3 (C)  
#########################################################################################  
  
  
def count\_char(s: str) -> dict:  
 *"""  
 Returns a dictionary after counting the number of letters, spaces, and digits in the provided string.  
 """* count\_let = 0  
 count\_sp = 0  
 count\_dig = 0  
 # Count the number of letters, spaces, and digits in the string  
 for ch in s:  
 if ch.isalpha():  
 count\_let += 1  
 elif ch.isspace():  
 count\_sp += 1  
 elif ch.isdigit():  
 count\_dig += 1  
 # Create a dictionary with the counts  
 output\_counts = {'letters': count\_let, 'spaces': count\_sp, 'digits': count\_dig}  
 return output\_counts

**print(count\_char("Hello World! 123"))**

**print(count\_char("Lorem ipsum dolor sit amet"))**

**OUTPUT 3(C)**

**{'letters': 10, 'spaces': 2, 'digits': 3}**

**{'letters': 21, 'spaces': 4, 'digits': 0}**