IoT Assignment 3

Submitted By:

Name	: Md.Azharuddin
Roll No.	: 36725
Regd No.	: 1701105431
Branch	: CSE
Sem	: 7

Github Link: https://github.com/mdazharuddin1011999/IoT_Assignment_3

Function

Q1) Wap to create a list of prime Fibonacci series between user defined range, default range is 10 to 50, using default arguments, required arguments, keyword arguments and function

Program

```
def is_prime(n):
    if n<2: return False
    if n<4: return True
    if n%2 == 0 or n%3 == 0: return False
    i = 5
    while i*i <= n:
        if n%i == 0 or n%(i+2) == 0:
            return False
        i += 6
    return True

def fibonacci_range(start=10, end=50):
    series = []
    a=0
    b=1
    while (c:=a+b) <= end:
        a = b
        b = c
        if c >= start and is_prime(c):
            series.append(c)
    return series

i = int(input("Enter start number: "))
j = int(input("Enter end number: "))
print("All prime fibonacci numbers are: ", end="")
print(*fibonacci_range(start=i, end=j), sep=", ")
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 1.py
Enter start number: 1
Enter end number: 100
All prime fibonacci numbers are: 2, 3, 5, 13, 89
```

Q2) Wap to check whether a number is Armstrong number or not using default arguments, required argument, keyword arguments and function

Program:

```
def is_armstrong(number):
    n = len(number)
    summed = sum([int(i)**n for i in number])
    return True if int(number) == summed else False

num = input("Enter a number to check Armstrong: ")
if is_armstrong(number=num): print(num, "is Armstrong")
else: print(num, "is not Armstrong")
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 2.py
Enter a number to check Armstrong: 153
153 is Armstrong
```

Q3) Wap to check whether a number is palindrome number and divisible by 3 and 5 or not using default arguments, required argument, keyword arguments and function

Program:

```
def is_palindrome(number):
    if int(number)%15 == 0:
        rev = number[::-1]
        return True if rev == number else False
    else: return False

num = input("Enter a number to check palindrome divisible by 3 and 5: ")
if is_palindrome(number=num):
    print(num, "is a Palindrome divisible by 3 and 5")
else:
    print(num, "is not Palindrome divisible by 3 and 5")
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 3.py
Enter a number to check palindrome divisible by 3 and 5: 3553
3553 is not Palindrome divisible by 3 and 5
```

Q4) Wap to find repeated elements and no of repeated elements in the list of 20 user defined values and also remove redundant values and update list with unique values only using required arguments and pass by reference

Program:

```
from collections import Counter
def remove_repeat(values):
    counter = Counter(values)
    print("Repeated elements are: ", end="")
    for key in counter:
        if counter[key] > 1:
            print(key, end=", ")
    print("")
    repeat_count = 0
    for i in range(len(values)-1, -1, -1):
        if counter[values[i]] > 1:
            counter[values[i]] -= 1
            values.pop(i)
            repeat_count += 1
    return repeat_count

numbers = []
for i in range(20):
    numbers.append(int(input("Enter number_%d: "%(i+1))))
print("Original List:", numbers)
repeated = remove_repeat(values=numbers)
print("There were %d redundant numbers"%(repeated))
print("List after removal of repeated numbers:", numbers)
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 4.py
Enter number_1: 12
Enter number_2: 23
Enter number_3: 34
Enter number_4: 45
Enter number_5: 56
Enter number_6: 12
Enter number_7: 23
Enter number_8: 34
Enter number_9: 56
Enter number_10: 23
Enter number_11: 36
Enter number_12: 3423
Enter number_13: 567
Enter number_14: 24
Enter number_15: 768
Enter number_16: 15
Enter number_17: 6
Enter number_18: 242
Enter number_19: 65
Enter number_20: 567
Original List: [12, 23, 34, 45, 56, 12, 23, 34, 56, 23, 36, 3423, 567, 24, 768, 15, 6, 242, 65, 567]
Repeated elements are: 12, 23, 34, 56, 567,
There were 6 redundant numbers
List after removal of repeated numbers: [12, 23, 34, 45, 56, 36, 3423, 567, 24, 768, 15, 6, 242, 65]
```

Q5) Wap to create biodata of a student using variable length argument function

Program:

```
def bio(*args):
    header = ["Name", "Roll no.", "Regd no.", "Branch", "Stream", "Sem",
"Phone no.", "Address"]
    for head, data in zip(header, args):
        print("%-10s: %s"%(head, data))

bio("Md.Azharuddin", "36725", "1701105431", "CSE", "B.Tech", "7th",
"9078600498", "Arad Bazar, Balasore")
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 5.py
Name      : Md.Azharuddin
Roll no.  : 36725
Regd no.  : 1701105431
Branch    : CSE
Stream    : B.Tech
Sem       : 7th
Phone no. : 9078600498
Address   : Arad Bazar, Balasore
```

Q6) Wap to find factorial of a number using while loop, do while loop and for loop and keyword arguments function

Program:

```
def factorial_while(num):
    n = num
    fact = 1
    while n > 0:
        fact *= n
        n -= 1
    print("Factorial of %d using WHILE loop: %d"%(num, fact))

def factorial_for(num):
    fact = 1
    for i in range(1, num+1):
        fact *= i
    print("Factorial of %d using FOR loop: %d"%(num, fact))

def factorial_do_while(num):
    fact = 1
    n = num
    while True:
        if n <= 0:
            break
        fact *= n
        n -= 1
    print("Factorial of %d using EMULATED DO WHILE loop: %d"%(num, fact))

n = int(input("Enter a number: "))
factorial_while(num=n)
factorial_for(num=n)
factorial_do_while(num=n)
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 6.py
Enter a number: 6
Factorial of 6 using WHILE loop: 720
Factorial of 6 using FOR loop: 720
Factorial of 6 using EMULATED DO WHILE loop: 720
```

Q7) Wap to search position of element in the list of 20 user defined values using binary search and function

Program:

```
def binary_search(values, element):
    values.sort()
    print("Sorted list: ", values)
    low = 0
    high = len(values)-1
    while low <= high:
        mid = (low + high) // 2
        if values[mid] == element:
            return mid
        if values[mid] < element:
            low = mid+1
        else:
            high = mid-1
    return -1


numbers = []
for i in range(20):
    numbers.append(int(input("Enter number_%d: "%(i+1))))

element = int(input("Enter element to search: "))
position = binary_search(numbers, element)

if position >= 0:
    print(element, "was found at position", position, "(zero based index)
in the sorted list.")
else:
    print("Element not found")
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 7.py
Enter number_1: 12
Enter number_2: 15
Enter number_3: 17
Enter number_4: 19
Enter number_5: 25
Enter number_6: 27
Enter number_7: 28
Enter number_8: 31
Enter number_9: 32
Enter number_10: 36
Enter number_11: 39
Enter number_12: 43
Enter number_13: 49
Enter number_14: 51
Enter number_15: 57
Enter number_16: 74
Enter number_17: 76
Enter number_18: 88
Enter number_19: 91
Enter number_20: 98
Enter element to search: 36
Sorted list:  [12, 15, 17, 19, 25, 27, 28, 31, 32, 36, 39, 43, 49, 51, 57, 74, 76, 88, 91, 98]
36 was found at position 9 (zero based index) in the sorted list.
```

Q8) Wap to find area, perimeter of rectangle, square, triangle using keyword arguments with function

Program:

```
from math import sqrt

def area_of_rect(length, breadth):
    return length*breadth

def peri_of_rect(length, breadth):
    return 2*(length+breadth)

def area_of_square(side):
    return side*side

def peri_of_square(side):
    return 4*side

def is_triangle(a, b, c):
    if a+b<=c or a+c<=b or b+c<=a:
        return False
    return True
```

```
def area_of_tri(a, b, c):
    if is_triangle(a, b, c):
        s = (a+b+c)/2
        return sqrt(s*(s-a)*(s-b)*(s-c))
    return "Invalid Triangle"

def peri_of_tri(a, b, c):
    if is_triangle(a, b, c):
        return a+b+c
    return "Invalid Triangle"

l = int(input("Enter length of rectangle: "))
b = int(input("Enter breadth of rectangle: "))
print("Area of rectangle:", area_of_rect(length=l, breadth=b))
print("Perimeter of rectangle:", peri_of_rect(breadth=b, length=l))

s = int(input("Enter side of square: "))
print("Area of square:", area_of_square(side=s))
print("Perimeter of square:", peri_of_square(side=s))

a = int(input("Enter first side of triangle: "))
b = int(input("Enter second side of triangle: "))
c = int(input("Enter third side of triangle: "))
print("Perimeter of triangle:", peri_of_tri(a, b, c))
print("Area of triangle:", area_of_tri(a, b, c))
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 8.py
Enter length of rectangle: 12
Enter breadth of rectangle: 8
Area of rectangle: 96
Perimeter of rectangle: 40
Enter side of square: 20
Area of square: 400
Perimeter of square: 80
Enter first side of triangle: 8
Enter second side of triangle: 9
Enter third side of triangle: 10
Perimeter of triangle: 27
Area of triangle: 34.197039345533994
```

Q9) WAP to swap two numbers using anonymous or lambda function

Program:

```
swap = lambda x, y : (y,x)

x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
x, y = swap(x, y)
print("First number:", x)
print("Second number:", y)
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 9.py
Enter first number: 12
Enter second number: 56
First number: 56
Second number: 12
```

Q10) Wap to perform all arithmetic operations using anonymous or lambda function

Program:

```
add = lambda x,y : x+y
subtract = lambda x,y : x-y
multiply = lambda x,y : x*y
divide = lambda x,y : x/y

x = int(input("Enter first number: "))
y = int(input("Enter second number: "))

print("%d + %d = %d"%(x, y, add(x, y)))
print("%d - %d = %d"%(x, y, subtract(x, y)))
print("%d * %d = %d"%(x, y, multiply(x, y)))
print("%d / %d = %f"%(x, y, divide(x, y)))
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\function>python 10.py
Enter first number: 10
Enter second number: 20
10 + 20 = 30
10 - 20 = -10
10 * 20 = 200
10 / 20 = 0.500000
```

Module

Q1) WAP to create scientific calculator and perform all operations like sum, subtraction, division, multiplication, modulus, power, sqrt, cubic root, sinx, cos x, tanx, log x, exp x, absolute value of x using function and module

scicalc.py

```
from math import sqrt, sin, cos, tan, log, exp

def addition(a, b):
    return a+b
def subtraction(a, b):
    return a-b
def division(a, b):
    return a/b
def multiplication(a, b):
    return a*b
def modulus(a, b):
    return a%b
def power(a, b):
    return a**b
def square_root(n):
    return sqrt(n)
def cubic_root(n):
    return n**(1/3)
def sinx(x):
    return sin(x)
def cosx(x):
    return cos(x)
def tanx(x):
    return tan(x)
def logx(x):
    return log(x)
def expx(x):
    return exp(x)
def absolute(x):
    return abs(x)
if __name__ == "__main__":
    print("You are trying to run a module!")
```

main.py

```
import scicalc

x = int(input("Enter x: "))
y = int(input("Enter y: "))

print("%d + %d = %d"%(x, y, scicalc.addition(x, y)))
print("%d - %d = %d"%(x, y, scicalc.subtraction(x, y)))
print("%d * %d = %d"%(x, y, scicalc.multiplication(x, y)))
print("%d / %d = %f"%(x, y, scicalc.division(x, y)))
print("%d %% %d = %d"%(x, y, scicalc.modulus(x, y)))
print("%d ^ %d = %d"%(x, y, scicalc.power(x, y)))
print("sqrt(%d) = %f"%(x, scicalc.square_root(x)))
print("curt(%d) = %f"%(x, scicalc.cubic_root(x)))
print("sin(%d) = %f"%(x, scicalc.sinx(x)))
print("cos(%d) = %f"%(x, scicalc.cosx(x)))
print("tan(%d) = %f"%(x, scicalc.tanx(x)))
print("log(%d) = %f"%(x, scicalc.logx(x)))
print("exp(%d) = %f"%(x, scicalc.expx(x)))
print("|%d| = %d"%(x, scicalc.absolute(x)))
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\modules\1>python main.py
Enter x: 10
Enter y: 20
10 + 20 = 30
10 - 20 = -10
10 * 20 = 200
10 / 20 = 0.500000
10 % 20 = 10
10 ^ 20 = 100000000000000000000
sqrt(10) = 3.162278
curt(10) = 2.154435
sin(10) = -0.544021
cos(10) = -0.839072
tan(10) = 0.648361
log(10) = 2.302585
exp(10) = 22026.465795
|10| = 10
```

Q2) WAP to perform following computation on stack using function like push, pop, isempty, isfull, peak and use function and module

1. **Create an stack of user defined size using list**
2. **Insert 20 user defined values in the stack using is full and push operation in stack**
3. **Search for an user defined element in the stack using peak operation of stack**
4. **Delete 5 elements from stack using pop operation**
5. **Display all remaining elements of stack**

stack.py

```
class Stack:
    def __init__(self, size):
        self.stack = []
        if size < 0:
            size = 0
        self.size = size

    def push(self, n):
        if len(self.stack) < self.size:
            self.stack.append(n)

    def pop(self):
        if len(self.stack) > 0:
            return self.stack.pop(-1)

    def isEmpty(self):
        return len(self.stack) == 0

    def isFull(self):
        return len(self.stack) == self.size

    def peek(self):
        if len(self.stack) > 0:
            return self.stack[-1]

    def __str__(self):
        s = "["
        for i in range(len(self.stack)):
            s += str(self.stack[i])+", "
        return s+"]"

if __name__ == "__main__":
    print("You are trying to run a module!")
```

main.py

```
from stack import Stack

size = int(input("Enter size of stack: "))
s = Stack(size)
for i in range(20):
    n = int(input("Enter number_%d: "%(i+1)))
    if not s.isFull():
        s.push(n)
        print(n, "pushed into stack")
    else:
        print("Stack is already full")

elm = int(input("Enter element to search: "))
if s.peek() == elm:
    print(elm, "is in the top of stack")
else:
    print(elm, "is not in the top of stack")

for i in range(5):
    if not s.isEmpty():
        elm = s.pop()
        print(elm, "popped off stack")
    else:
        print("Stack is empty")

print("Remaining Elements:", s)
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\modules\2>python main.py
Enter size of stack: 15
Enter number_1: 12
12 pushed into stack
Enter number_2: 23
23 pushed into stack
Enter number_3: 34
34 pushed into stack
Enter number_4: 45
45 pushed into stack
Enter number_5: 56
56 pushed into stack
Enter number_6: 67
67 pushed into stack
Enter number_7: 78
78 pushed into stack
Enter number_8: 89
89 pushed into stack
```

```
Enter number_9: 90
90 pushed into stack
Enter number_10: 09
9 pushed into stack
Enter number_11: 98
98 pushed into stack
Enter number_12: 87
87 pushed into stack
Enter number_13: 76
76 pushed into stack
Enter number_14: 65
65 pushed into stack
Enter number_15: 54
54 pushed into stack
Enter number_16: 43
Stack is already full
Enter number_17: 32
Stack is already full
Enter number_18: 21
Stack is already full
Enter number_19: 12
Stack is already full
Enter number 20: 23
Enter element to search: 54
54 is in the top of stack
54 popped off stack
65 popped off stack
76 popped off stack
87 popped off stack
98 popped off stack
Remaining Elements: [12, 23, 34, 45, 56, 67, 78, 89, 90, 9, ]
```

Q3) WAP to perform following computation on queue using function like insert, delete, isempty, isfull, peak and use function and module

1. **Create an queue of user defined size using list**
2. **Insert 20 user defined values in the queue using is full and insert operation in stack**
3. **Search for an user defined element in the queue using peak operation of stack**
4. **Delete 5 elements from queue using delete operation**
5. **Display all remaining elements of queue**

queue.py

```python
class Queue:
    def __init__(self, size):
        self.queue = []
        if size < 0:
            size = 0
        self.size = size
    def insert(self, n):
        if len(self.queue) < self.size:
            self.queue.append(n)
    def delete(self):
        if len(self.queue) > 0:
            return self.queue.pop(0)
```

```
    def isEmpty(self):
        return len(self.queue) == 0
    def isFull(self):
        return len(self.queue) == self.size
    def peek(self):
        if len(self.queue) > 0:
            return self.queue[0]
    def __str__(self):
        s = "["
        for i in range(len(self.queue)):
            s += str(self.queue[i])+", "
        return s+"]"

if __name__ == "__main__":
    print("You are trying to run a module!")
```

main.py

```
from queue import Queue
size = int(input("Enter size of queue: "))
q = Queue(size)
for i in range(20):
    n = int(input("Enter number_%d: "%(i+1)))
    if not q.isFull():
        q.insert(n)
        print(n, "inserted into queue")
    else:
        print("Queue is already full")
elm = int(input("Enter element to search: "))
if q.peek() == elm:
    print(elm, "is the current element")
else:
    print(elm, "is not the current element")
for i in range(5):
    if not q.isEmpty():
        elm = q.delete()
        print(elm, "deleted from queue")
    else:
        print("Queue is empty")

print("Remaining Elements:", q)
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\modules\3>python main.py
Enter size of queue: 15
Enter number_1: 12
12 inserted into queue
Enter number_2: 23
23 inserted into queue
Enter number_3: 34
34 inserted into queue
Enter number_4: 45
45 inserted into queue
Enter number_5: 56
56 inserted into queue
Enter number_6: 67
67 inserted into queue
Enter number_7: 78
78 inserted into queue
Enter number_8: 89
89 inserted into queue
Enter number_9: 90
90 inserted into queue
Enter number_10: 09
9 inserted into queue
Enter number_11: 98
98 inserted into queue
Enter number_12: 87
87 inserted into queue
Enter number_13: 76
76 inserted into queue
Enter number_14: 65
65 inserted into queue
Enter number_15: 54
54 inserted into queue
Enter number_16: 43
Queue is already full
Enter number_17: 32
Queue is already full
Enter number_18: 21
Queue is already full
Enter number_19: 12
Queue is already full
Enter number_20: 23
Queue is already full
Enter element to search: 12
12 is the current element
12 deleted from queue
23 deleted from queue
34 deleted from queue
45 deleted from queue
56 deleted from queue
Remaining Elements: [67, 78, 89, 90, 9, 98, 87, 76, 65, 54, ]
```

Q4) WAP to perform following using module and function for deposit, withdraw, check balance for banking application

1. **Assign a fixed value as balance for account**
2. **Deposit user defined amount and display updated balance**
3. **Withdraw user defined money if user has sufficient fund otherwise show Insufficient fund**
4. **Check for balance**
5. **Maintain a fixed amount 3000 in the account**

account.py

```
class Account:
    def __init__(self, initial=0, minimum=0):
        if minimum < 0:
            minimum = 0
        if initial < minimum:
            initial = minimum
```

```
        self.balance = initial
        self.minimum = minimum
    def deposit(self, amount):
        self.balance += amount
        print("Deposited", amount, "successfully!")
    def withdraw(self, amount):
        after = self.balance-amount
        if after >= self.minimum:
            print("Withdrawn", amount, "successfully")
            self.balance = after
        else:
            print("Insufficient balance")
    def check_balance(self):
        return self.balance
    def set_minimum(self, minimum):
        self.minimum = minimum
    def get_minimum(self):
        return self.minimum

if __name__ == "__main__":
    print("You are trying to run a module!")
```

main.py

```
from account import Account

acc = Account(int(input("Enter initial account balance: ")))
deposit_amount = int(input("Enter amount to deposit: "))
acc.deposit(deposit_amount)
print("Current Balance: ", acc.check_balance())
withdraw_amount = int(input("Enter amount to withdraw: "))
acc.withdraw(withdraw_amount)
print("Current Balance: ", acc.check_balance())
acc.set_minimum(3000)
print("Minimum balance has been changed to 3000")
withdraw_amount = int(input("Enter amount to withdraw: "))
acc.withdraw(withdraw_amount)
print("Current Balance: ", acc.check_balance(), "--- Minimum required: ",
acc.get_minimum())
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\modules\4>python main.py
Enter initial account balance: 5000
Enter amount to deposit: 100
Deposited 100 successfully!
Current Balance:  5100
Enter amount to withdraw: 2100
Withdrawn 2100 successfully
Current Balance:  3000
Minimum balance has been changed to 3000
Enter amount to withdraw: 100
Insufficient balance
Current Balance:  3000 --- Minimum required:  3000
```

Q5) Wap to compute factorial, GCD, LCM, sqrt without using any library function, swap two numbers without using 3rd variable using function and module

mymath.py

```python
def factorial(n):
    fact = 1
    for i in range(1, n+1):
        fact *= i
    return fact

def gcd(a, b):
    if a==0: return b
    return gcd(b%a, a)

def lcm(a, b):
    hcf = gcd(a, b)
    return a*b//hcf

def sqrt(n):
    return n**(1/2)

def swap(a, b):
    return (b, a)

if __name__ == "__main__":
    print("You are trying to run a module!")
```

main.py

```
import mymath

n = int(input("Enter a number to find factorial: "))
print("%d! = %d"%(n, mymath.factorial(n)))

a = int(input("Enter a: "))
b = int(input("Enter b: "))
print("GCD of %d, %d = %d"%(a, b, mymath.gcd(a, b)))
print("LCM of %d, %d = %d"%(a, b, mymath.lcm(a, b)))

n = int(input("Enter a number to find square root: "))
print("Square root of %d is %f"%(n, mymath.sqrt(n)))

a = int(input("Enter a: "))
b = int(input("Enter b: "))
a, b = mymath.swap(a, b)
print("After Swapping: a =", a, "b =", b)
```

Output:

```
D:\Books & Pdf\B.Tech\7th sem\IOT\Assignment3\modules\5>python main.py
Enter a number to find factorial: 6
6! = 720
Enter a: 12
Enter b: 4
GCD of 12, 4 = 4
LCM of 12, 4 = 12
Enter a number to find square root: 2
Square root of 2 is 1.414214
Enter a: 12
Enter b: 78
After Swapping: a = 78 b = 12
```