INDEX

INDEX

ASSIGNMENT

Q1. Write a python program to find whether the given no. is seven or odd.

Program:

```
# This is program of finding Even or odd
def main():
    num = int(input("Enter  number: "))
    if num%2==0:
        print("This is even number")
    else:
        print("This is odd number")
if __name__ == "__main__":
    main()
```

Output:

```
Enter  number: 10
This is even number
```

Q2.Write a python program to compute the distance between two points taking input from the user(python Theorem).

Program:

```
import math
def distance(x1,x2,y1,y2):
    return math.sqrt((x2-x1)**2 + (y2-y1)**2);
def main():
    print("Enter coordinates of point 1: ")
    x1 = float(input("Enter x1: "))
    x2 = float(input("Enter x2: "))

    print("Enter coordinates of point 2: ")
    y1 = float(input("Enter y1: "))
    y2 = float(input("Enter y2: "))
    dist = distance(x1,x2,y1,y2)
    print(f"Distance between the two points: {dist}")
main()
```

Output:

```
Enter coordinates of point 1:
Enter x1: 2
Enter x2: 3
Enter coordinates of point 2:
Enter y1: 4
Enter y2: 5
Distance between the two points: 1.4142135623730951
```

ASSIGNMENT

Q3.Write a python program to find factorial of given number.

Program:

```
def factorial(n):
   if n<0:
      return "Factorial is not defined for negative numbers !"
   elif n==0 or n==1:
      return 1
   else:
      return n * factorial ( n-1)
num = int(input("Enter a number: "))
result = factorial (num)
print(f"The factorial of {num} is : {result} " )
```

Output:

```
Enter a number: 5
The factorial of 5 is : 120
```

Q4.Write a python program to design a calculator.

Program:

```
first= int(input("enter first number: "))
operator = input("+,-,*,/,%: ")
second = int(input("Enter second number: "))
if operator == "+":
   result = first + second
   print(f"Addition: {result}")
elif operator =="-":
   result = first - second
   print(f"Substraction: {result}")
elif operator =="*":
   result = first * second
   print(f"Multiple: {result}")
elif operator =="/":
   result = first // second
   print(f"Division {result}")
elif operator =="%":
   result = first % second
   print(f"Mod: {result}")
else:
   print("Invalid operation!")
```

Output:

```
enter first number: 10
```

ASSIGNMENT

```
+,-,*,/,%: +
Enter second number: 20
Addition: 30
```

Q5.Write a python class to implement pow(x,n).

Program:

```
#This is program of power exponentiation with operator and function
def custom_operator(base,exponent):
    #using result =1 bcz power of 0 is equal to 1
    result = 1
    # underscore(_) is used to repeat the code
    for _ in range(exponent):
        result *=base
    return result
def power_function(base,exponent):
    return base**exponent
b = float(input("Enter the Base number : "))
e = int(input("Enter the exponent number: "))
using_operator = custom_operator(b,e)
using_function = power_function(b,e)
print(f"{b} raised to the power {e} is : {using_operator}","(Using operator)")
print(f"{b} raised to the power {e} is : {using_function}","(Using power function)")
```

Output:

```
Enter the Base number : 3
Enter the exponent number: 2
3.0 raised to the power 2 is : 9.0 (Using operator)
3.0 raised to the power 2 is : 9.0 (Using power function)
```

Q6.Write a python to use split and join methods in the string and trace a birthday with a dictionary data structure.

Program:

```
# This is split and join method and trace birthday
def main():
    sentence =input("Enter string to split: ")
    words = sentence.split()  #Split the sentence into words
    new_sentence = '-'.join(words) #Join the words with '-'
    print("Original Sentence: ",sentence)
    print("Modified sentence: ",new_sentence)
    #Using a dictionary to store birthdays
    birthdays = {
        "sumit" : "10 March",
        "khushboo" : "26 July",
        "manan" : "11 May",
        "udit": "15 Auguest"
        }
    while True:
        name = input("Enter a name( or 'exit' to quit): ")
        if name.lower() == "exit":
            print("Program is terminated")
            break
        elif name in birthdays:
            print(f"{name}'s birthday is on {birthdays[name]}")
        else:
            print(f"{name}'s birthday is not in the dictionary.")
if __name__ == "__main__":
    main()
```

Output:

```
Enter string to split: i am a programmer
Original Sentence:  i am a programmer
Modified sentence:  i-am-a-programmer
Enter a name( or 'exit' to quit): exit
Program is terminated
```

Q7.Write a python program to print each line of a file in reverse order.

Program:

```
def print_lines_in_reverse(file_path):
    try:
        with open(file_path,'r') as file:   #with is ensure the file is close after reading
            lines = file.readlines()
            lines.reverse()
            for line in lines:
                print(line.strip()[::-1])
    except FileNotFoundError:
        print("File not found!")
def main():
    file_path = input("Enter the path of the file: ")
    print_lines_in_reverse(file_path)
if __name__ == "__main__":
    main()
```

Output:

ASSIGNMENT

```
Enter the path of the file: sample.txt

!taerg si nohtyP
lawohk timus si sihT
```

Q8.Write a python program to compute the number of character, words, and lines in files.

Program:

```
def count_and_read(file_path):
    try:
        with open(file_path,'r') as file:   #with is ensure the file is close after reading
            content = file.readlines()
            num_char = sum(len(line) for line in content) #characters in file
            num_words = sum(len(line.split()) for line in content)
            num_lines = len(content)
            num_lines_actual = sum(1 for line in content if line.strip())
            print("Number of Characters: ",num_char)
            print("Number of words: ",num_words)
            print("Number of lines: ",num_lines_actual)
    except FileNotFoundError:
        print("File not found!")
def main():
    file_path = input("Enter the path of the file: ")
    count_and_read(file_path)
if __name__ == "__main__":
    main()
```

Output:

```
Enter the path of the file: output.txt
Number of Characters:  39
Number of words:  7
Number of lines:  2
```

Q9.Write a python program for finding unique and duplicate items of a list.

Program:

```
def find_unique_and_duplicate(items):
    unique_items = []
    duplicate_items = []
    seen = set()   #set is used to store unique items
    for item in items:
        if item not in seen:
            unique_items.append(item)
            seen.add(item)
        else:
            duplicate_items.append(item)
    return unique_items, duplicate_items
def main():
    input_string = input("Enter a list of items separated by spaces: ")
    items = input_string.split()

    unique_items, duplicate_items = find_unique_and_duplicate(items)
    print("Unique items:", unique_items)
    print("Duplicate items:", duplicate_items)
if __name__ == "__main__":
    main()
```

Output:

```
Enter a list of items separated by spaces: mca bca ba mca bsc
Unique items: ['mca', 'bca', 'ba', 'bsc']
Duplicate items: ['mca']
```

Q10.Write a python program to slice with list.

Program:

```
my_list = [1,2,3,4,5,6,7,8,9,10]
start_index = 4
end_index = 8
slice_list = my_list[start_index:end_index]
print("Original List : ",my_list)
print("Sliced List : ",slice_list)
```

Output:

```
Original List :  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sliced List :  [5, 6, 7, 8]
```

Q11.Write a python program to display the calendar of given month and year.

Program:

```
import calendar
year = int(input("Enter year: "))
month = int(input("Enter Month b/w 1 to 12 : "))
if 1<= month <=12 :
    cal = calendar.month(year,month)
    print("\n Calendar for {} / {} \n " .format(month,year))
```

ASSIGNMENT

```
    print(cal)
else:
    print("Invalid month. Please enter a month between 1 and 12.")
```

Output:

```
Enter year: 2023
Enter Month b/w 1 to 12 : 09

 Calendar for 9 / 2023

   September 2023
Mo Tu We Th Fr Sa Su
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

Q12.Write a python program to demonstrate working with tuples in python.

Program:

```
def main():
    #creating tuple
    fruits = ("apple","mango","pear","orange","grapes")
    #access items in tuples
    print("First fruit: ",fruits[0])
    print("Third fruit: ",fruits[2])
    #Slincing a tuple
    print("Sliced fruits: ",fruits[2:5])
    #Length of tuple
    print("Number of fruits: ",len(fruits))
    #checking if an elements is present or not in tuple
    if "apple" in fruits:
        print("Yes,'apple' is in the tuple")
    else:
        print("'apple' is not in the tuple")
    # Concatenating tuples
    more_fruits = ("mango", "kiwi", "pineapple")
    all_fruits = fruits + more_fruits
    print("All fruits:", all_fruits)
if __name__ == "__main__":
    main()
```

ASSIGNMENT

Output:

```
First fruit:  apple
Third fruit:  pear
Sliced fruits:  ('pear', 'orange', 'grapes')
Number of fruits:  5
Yes,'apple' is in the tuple
All fruits: ('apple', 'mango', 'pear', 'orange', 'grapes', 'mango', 'kiwi', 'pineapple')
```

Q13.Write a python program to count the number of characters in the string and store them in a dictionary data structure.

Program:

```
def count_characters(string):
    char_count = {}

    for char in string:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1
    return char_count
def main():
    input_string = input("Enter a string: ")
    char_counts = count_characters(input_string)
    print("Character counts:")
    for char, count in char_counts.items():
        print(f"'{char}': {count}")
if __name__ == "__main__":
    main()
```

Output:

```
Enter a string: KHUSHBOO
Character counts:
'K': 1
'H': 2
'U': 1
'S': 1
'B': 1
'O': 2
```

Q14.Write a python program to define a module and import a specific function in that module to another program.

Program:

```
#main file here import module file
```

ASSIGNMENT

```
from my_module import file
def main():
    name = input("Enter your name: ")
    files = file(name)
    print(files)

main()
```

Output:

```
Enter your name: sumit khowal
Hello sumit khowal !
```

Q15.Write a script named filecopy.py. This script should prompt the user for the names of two text files. The contents of the first file should be input and written to the second file.

Program:

```
# program of copy items ot anoter file
def main():
    input_file_name=input("Enter the name of the input file: ")
    try:
        with open(input_file_name,'r') as input_file:
            content = input_file.read()
            output_file_name=input("Enter the name of the output file: ")
            with open(output_file_name,'w') as output_file:
                output_file.write(content)
            print("File contents copied successfully.")
    except FileNotFoundError:
        print("Input file not found!")

main()
```

Output:

```
Enter the name of the input file: sample.txt
Enter the name of the output file: output.txt
File contents copied successfully.
```

Q16.Write a python program to use classes and methods.

Program:

```
class BankAccount:
    def __init__(self,account_number,holder_name,balance = 0.0):
        self.account_number = account_number
```

ASSIGNMENT

```
        self.holder_number = holder_name
        self.balance = balance
        #the 'self' keyword is used to refer to the instance of the class itself.
     #creating deposit Method
    def deposit(self,amount):
        if amount > 0:
            self.balance += amount
            print(f" Deposited ${amount} into account {self.account_number} . New
balance: ${self.balance:.2f}")
        else:
            print("Invalid deposit amount. Amount must be greater than zero.")
    #creating withdraw method
    def withdraw (self,amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdraw ${amount} from account {self.account_number} .New
balance : ${self.balance : .2f}")
        else:
            print("Invalid withdrawal amount or insufficient funds.")
    #creating display method
    def display_balance(self):
        print(f"Account {self.account_number} balance: ${self.balance:.2f}")
# Create two bank accounts
account1 = BankAccount("1003", "Sumit")
account2 = BankAccount("1002", "Roshni", 1000.0)
# Perform transactions
account1.display_balance()
d_amount = int(input("Enter amount to deposit : "))
account1.deposit(d_amount)
w_amount = int(input("Enter amount to withdraw : "))
account1.withdraw(w_amount)
account2.display_balance()
account2.deposit(1000.0)
account2.withdraw(1500.0)
account1.display_balance()
account2.display_balance()
```

Output:

```
Account 1003 balance: $0.00
Enter amount to deposit : 500
 Deposited $500 into account 1003 . New balance: $500.00
Enter amount to withdraw : 10
Withdraw $10 from account 1003 .New balance : $ 490.00
```

ASSIGNMENT

```
Account 1002 balance: $1000.00
 Deposited $1000.0 into account 1002 . New balance: $2000.00
Withdraw $1500.0 from account 1002 .New balance : $ 500.00
Account 1003 balance: $490.00
Account 1002 balance: $500.00
```

Q17.Write a python program to methods with self.

Program:

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.speed = 0

    def start(self):
        print(f"{self.year} {self.make} {self.model} is starting.")

    def accelerate(self):
        self.speed += 10
        print(f"Accelerating... Current speed: {self.speed} mph")
    def brake(self):
        if self.speed > 0:
            self.speed -= 10
            print(f"Braking... Current speed: {self.speed} mph")
        else:
            print("The car is already at a complete stop.")
# Create an instance of the Car class
car_make = input("Enter car make : ")
car_model = input("Enter model: ")
car_year = input("Enter Year: ")
my_car = Car(car_make, car_model, car_year)

# Perform actions on the car object
my_car.start()
my_car.accelerate()
my_car.accelerate()
my_car.accelerate()
my_car.brake()
```

Output:

```
Enter car make : Toyato
Enter model: camary
```

ASSIGNMENT

```
Enter Year: 2023
2023 Toyato camary is starting.
Accelerating... Current speed: 10 mph
Accelerating... Current speed: 20 mph
Accelerating... Current speed: 30 mph
Braking... Current speed: 20 mph
```

Q18.Write a python program to count the no. of object.

Program:

```
class count:
    counter = 0
    def __init__(self):
        count.counter +=1

c1 = count()
c2 = count()
c3 = count()
c4 = count()
print("Number of Objects created : " ,count.counter)
```

Output:

```
Number of Objects created :  4
```

Q19.Write a python program to use of constructor.

Program:

```
class Student:
    def __init__(self,name,age,grade):
        self.name = name
        self.age = age
        self.grade = grade

    def displayInfo(self):
        print(f"Name of Student: {self.name}")
        print(f"Age of Student : {self.age}")
        print(f"Gade : {self.grade}")

s1 = Student("Sumit Khowal",23,"A+")
s2 = Student("Roshni",22,"B")
print("Student 1 : ")
s1.displayInfo()
print("Student 2 : ")
s2.displayInfo()
```

ASSIGNMENT

Output:
```
Student 1 :
Name of Student: Sumit Khowal
Age of Student : 23
Gade : A+
Student 2 :
Name of Student: Roshni
Age of Student : 22
Gade : B
```

Q20.Write a python to use of inheritance.

Program:
```
class Animal:
    def __init__(self,name):
        self.name = name
    def speak(self):
        pass

 #the pass statement is used to indicate that this method is intentionally empty
class Dog(Animal):
    def speak(self):
        return f"{self.name} says bark!"

class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow!"

dog = Dog("Dazy")
cat = Cat("Minky")
print(dog.speak())
print(cat.speak())
```

Output:
```
Dazy says bark!
Minky says Meow!
```