PYTHON PRACTICAL

EXPERIMENT 1(a)

- AIM:

  Create a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.

PROGRAM:

```
'''
[a] Create a program that ask the user to enter theit name and their age.
print out a message addressed to them that tells them the year that they will turn 100 years old
'''

import datetime

Name = input("Enter your name: ")
Age = int(input("Enter your age: "))

Year = datetime.datetime.now().year
Remain = 100 - Age
Year_turn_to_100 = Year + Remain

print(Name," you will be 100 years old in ",Year_turn_to_100)
```

OUTPUT:

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: S:\VS code\python\college_practical\Practical_1.py =========
Enter your name: Sumit
Enter your age: 18
Sumit  you will be 100 years old in  2106
```

EXPERIMENT 1(b)

- AIM:

  Enter the number from user and depending on whether the number is even or odd, print out an appropriate message to the user.

PROGRAM:

```python
'''
[B] Enter the number form user and depending on whether the number is even or odd,
print out an appropriate message to the user.
'''

Number = int(input("Enter a number: "))

if Number % 2 == 0:
    print(Number, "is Even")
else:
    print(Number, "is Odd")
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter a number: 50
50 is Even
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter a number: 421
421 is Odd
PS S:\VS code\python>
```

EXPERIMENT 1(c)

- AIM:
  Write a program to generate the Fibonacci series.

PROGRAM:

```python
'''
[C] Write a program to generate the Fibonacci series.
'''

num_1 = int(input("Enrter the number of term you want in Fibonacci series"))
num1 = 0
num2 = 1
print(num1 ,"\n ", num2)

for i in range(num_1-2):
    next_num = num1 + num2
    print(next_num)
    num1 = num2
    num2 = next_num
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enrter the number of term you want in Fibonacci series8
0
  1
1
2
3
5
8
13
PS S:\VS code\python>
```

EXPERIMENT 1(d)

- AIM:

  Write a function that reverses the user defined value.

PROGRAM:

Practical.py

```python
'''
[D] write a function that reverse a user defined value
'''

def reverse(Value):
    reverse_number = 0
    Orignal_number = Value
    while(Value != 0):
        devider = Value % 10
        reverse_number = (reverse_number * 10) + devider
        Value //= 10
    print("reversed number of "+ str(Orignal_number) + " is " + str(reverse_number))

Value = int(input("Wnter the Value to reverse it: "))
reverse(Value)
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Wnter the Value to reverse it: 26354
reversed number of 26354 is 45362
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Wnter the Value to reverse it: 645321879523
reversed number of 645321879523 is 325978123546
PS S:\VS code\python>
```

EXPERIMENT 1(e)

- AIM:

  Write a function to check the input value is Armstrong and also write the function for Palindrome.

PROGRAM:

Practical.py

```python
'''
[E] Write a function to check the input value is Armstrong and also write the fuction for Palindrome.
'''
def is_armstrong_number(number):
    original_number = number
    armstrong_sum = 0
    while number != 0:
        digit = number % 10
        armstrong_sum += digit ** 3
        number //= 10
    if armstrong_sum == original_number:
        print(original_number, "is an Armstrong number")
    else:
        print(original_number, "is not an Armstrong number")

def is_palindrome_number(number):
    original_number = number
    reversed_number = 0
    while number != 0:
        digit = number % 10
        reversed_number = reversed_number * 10 + digit
        number //= 10
    if reversed_number == original_number:
        print(original_number, "is a palindrome number")
    else:
        print(original_number, "is not a palindrome number")

number = int(input("Enter any number: "))
is_armstrong_number(number)
is_palindrome_number(number)
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter any number: 45682
45682 is not an Armstrong number
45682 is not a palindrome number
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter any number: 153
153 is an Armstrong number
153 is not a palindrome number
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter any number: 121
121 is not an Armstrong number
121 is a palindrome number
PS S:\VS code\python>
```

EXPERIMENT 1(f)

- AIM:

  Write a recursive function to print the factorial for a given number.

PROGRAM:

Practical.py

```python
'''
[f] Write a recursive function to print the factorial a given number
'''

def factorial(User_input):
    if User_input == 1:
        return 1
    else:
        return User_input * factorial(User_input - 1)

User_input = int(input("Enter number for factorial: "))
print("Factorial of " + str(User_input) +  " is " + str(factorial(User_input)))
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter number for factorial: 0
Factorial of 0 is 0
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter number for factorial: 42
Factorial of 42 is 1405006117752879898543142606244511569936384000000000
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter number for factorial: 1
Factorial of 1 is 1
PS S:\VS code\python>
```

EXPERIMENT 2(a)

- AIM:
  Write a function that takes a character (i.e. a string of length 1) and returns True if it is a vowel, False otherwise.

PROGRAM:

Practical.py

```python
'''
[A] Write a function that takes a character  and  returns True if it is a vowel false otherwise
'''

def isVowel(character):
    vowels = ['a', 'e', 'i', 'o', 'u']
    if character.lower() in vowels:
        print(character,'is vowel')
    else:
        print(character,"is not vowel")

def is_vowel(character):
    vowel = 'aeiouAEIOU'
    return character in vowel


character = input("enter an one character: ")
isVowel(character)
print(f"{character} is vowel" if is_vowel(character) else f"{character} is not vowel")
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
enter an one character: u
u is vowel
u is vowel
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
enter an one character: io
io is not vowel
io is vowel
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
enter an one character: E
E is vowel
E is vowel
PS S:\VS code\python>
```

EXPERIMENT 2(b)

- AIM:
  Define a function that computes the length of a given list or string.

PROGRAM:

Practical.py

```python
'''
[B] Define a function that computes the length of a given list or string
'''
def lengthOfString(strings):
    count = 0
    for i in strings:
        count += 1
    return count

def lenghtOfList(lists):
    count = 0
    for i in lists:
        count += 1
    return count

strings = input("Enter a string: ")
print(f"Length of {strings} is {lengthOfString(strings)}")

lists = list()
intt = int(input("How much data you want to add in list: "))

while intt >= 1:
    item = input("enter the data: ")
    lists.append(item)
    intt -= 1

print(f"Length of {lists} is {lenghtOfList(lists)}")
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter a string: Sumit
Length of Sumit is 5
How much data you want to add in list: 5
enter the data: Sumit
enter the data: Duby
enter the data: 24
enter the data: Fyit
enter the data: A
Length of ['Sumit', 'Duby', '24', 'Fyit', 'A'] is 5
PS S:\VS code\python>
```

EXPERIMENT 2(c)

- AIM:

Define a procedure histogram() that takes a list of integers and prints a histogram to the screen. For example, histogram([4, 9, 7]) should print the following:

****

****

****

PROGRAM:

```python
'''
[C] Define a procedure histogram() that takes a list of integers and prints a histogram to the screen.
'''

def histogram(New_list):
    for i in New_list:
        print(i*"*")

new_list = list()
number_of_items = int(input("Enter the row number for histrogram: "))

for i in range(number_of_items):
    Entery = int(input(f"Enter the number of columns that should be in {i+1} row of histogram: "))
    new_list.append(Entery)

histogram(new_list)
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter the row number for histrogram: 8
Enter the number of columns that should be in 1 row of histogram: 5
Enter the number of columns that should be in 2 row of histogram: 4
Enter the number of columns that should be in 3 row of histogram: 3
Enter the number of columns that should be in 4 row of histogram: 2
Enter the number of columns that should be in 5 row of histogram: 1
Enter the number of columns that should be in 6 row of histogram: 2
Enter the number of columns that should be in 7 row of histogram: 3
Enter the number of columns that should be in 8 row of histogram: 4
*****
****
***
**
*
**
***
****
PS S:\VS code\python>
```

EXPERIMENT 3(a)

- AIM:
  A pangram is a sentence that contains all the letters of the English alphabet at least once, for example:
  The quick brown fox jumps over the lazy dog. Your task here is to write a function to check a sentence to see if it is a pangram or not.

PROGRAM:

Practical.py

```python

englishLanguageAlphabets ='abcdefghijklmnopqrstuvwxyz'

def Pangram(sentences):
    is_it = True
    for i in englishLanguageAlphabets:
        if i not in sentences.lower():
            is_it = False
    if is_it == True:
        print("Sentence is Pangram")
    else:
        print("Sentence is not pangram")

sentences = "The quick brown fox jumps over the lazy dog"

Pangram(sentences)

```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Sentence is Pangram
PS S:\VS code\python>
```

EXPERIMENT 3(b)

- AIM:

  Take a list, say for example this one:
  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] and write a program that prints out all the elements of the list that are less than 5.

PROGRAM:

Practical.py

```python
'''
[B] vWrite a program that prints out all the elements of the list that are less than 5
'''

lists = list()
numberOfItemsInListToHave = int(input("Enter how many items should be there in list: "))

for i in range(numberOfItemsInListToHave):
    entry = int(input("Enter the data(should be of int type): "))
    lists.append(entry)

new_list = list()
for i in lists:
    if i < 5:
        new_list.append(i)
print("number that are less than 5 are: ", new_list)
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\tempCodeRunnerFile.py"
Enter how many items should be there in list: 10
Enter the data(should be of int type): 45
Enter the data(should be of int type): 1
Enter the data(should be of int type): 3
Enter the data(should be of int type): 152
Enter the data(should be of int type): 0
Enter the data(should be of int type): 15
Enter the data(should be of int type): 2
Enter the data(should be of int type): 3
Enter the data(should be of int type): 56
Enter the data(should be of int type): 754
number that are less than 5 are:  [1, 3, 0, 2, 3]
PS S:\VS code\python>
```

EXPERIMENT 4(a):

- AIM:

  Write a program that takes two lists and returns True if they have at least one common member.

PROGRAM:

```python
'''
[A] write a program that takes two lists and returns true if they have at least one common member.
'''

list_1 = ["sumit", "Dubey", 20, "Anime lover", False, 10]
list_2 = ["FYIt", 'A', 24, "sumit", 20, True, "Dubey"]

set_1 = set(list_1)
set_2 = set(list_2)

value = len(set_1.intersection(set_2))

if value > 0:
    print("True, Both the list have one or more than one items common in them")
else:
    print("False, they don't have common items")
```

OUTPUT:

```
PS S:\VS code\python> python -u "s:\VS code\python\college_practical\practical_4.py"
True, Both the list have one or more than one items common in them
PS S:\VS code\python>
```

EXPERIMENT 4(b):

- AIM:

Write a Python program to print a specified list after removing the 0th, 2nd, 4th and 5th elements

PROGRAM:

```python
'''
[B] Write a Python program to print a specified list after removing the 0th, 2nd, 4th
and 5th elements.
'''

list_ = ["FYIt", 'A', 24, "sumit", True, "Dubey"]

list_.pop(0)
del list_[1]
list_.pop(2)
del list_[2]
print(list_)
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_4.py"
True, Both the list have one or more than one items common in them
['A', 'sumit']
```

EXPERIMENT 4©

- AIM:
  Write a Python program to clone or copy a list.

PROGRAM:

```python
'''
[C] Write a Python program to clone or copy a list
'''

original_list = [54,35.21,14,3]

copy_1 = original_list.copy()
copy_2 = list(original_list)
print(copy_1)
print(copy_2)
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_4.py"
[54, 35.21, 14, 3]
[54, 35.21, 14, 3]

Process finished with exit code 0
```

EXPERIMENT 5 (A)

- AIM:

  Write a python script to sort (ascending and descending) a dictionary by value.

PROGRAM:

```python
'''
[A] Write a Python script to sort (ascending and descending) a dictionary by value.
'''

def sortDictionaryByValue(dictionary, reverse=False):
    sorted_items = sorted(dictionary.items(), key=lambda item: item[1], reverse=reverse)
    sorted_dict = dict(sorted_items)
    return sorted_dict


myDictionary = {'a': 4, 'b': 7, 'c': 2, 'd': 9, 'e': 5}

asc = sortDictionaryByValue(myDictionary)
print("Sorted Dictionary in Ascending :", asc)

desc = sortDictionaryByValue(myDictionary, reverse=True)
print("Sorted Dictionary in Descending :", desc)
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_5.py"
Sorted Dictionary in Ascending : {'c': 2, 'a': 4, 'e': 5, 'b': 7, 'd': 9}
Sorted Dictionary in Descending : {'d': 9, 'b': 7, 'e': 5, 'a': 4, 'c': 2}
```

EXPERIMENT 5 (b)

- AIM:

  Write a Python script to concatenate following dictionaries to create a new one.
  Sample Dictionary :
  dic1={1:10, 2:20}
  dic2={3:30, 4:40}
  dic3={5:50,6:60}
  Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

PROGRAM:

Practical.py

```python
'''
[B] Write a Python script to concatenate following dictionaries to create a new one.
Sample Dictionary :
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
'''

Grandmother = {
    'Name': "Nirmala",
    'Son': 3,
    'Daughter': 0
}

Father = {
    'Son_1_Name': "Vinod",
    'Childs': 2
}

child = {
    'child_1_name': "Sumit"
}

family_tree = {}
family_tree.update(Grandmother)
family_tree.update(Father)
family_tree.update(child)
print(family_tree)
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_5.py"
{'Name': 'Nirmala', 'Son': 3, 'Daughter': 0, 'Son_1_Name': 'Vinod', 'Childs': 2, 'child_1_name': 'Sumit'}
```

EXPERIMENT 5 ©

- AIM:

  Write a Python program to sum all the items in a dictionary.

PROGRAM:

```python
'''
[C] Write a Python program to sum all the items in a dictionary
'''

def SumOfValues_ofDictionary(dictionarys):
    return sum(dictionarys.values())

sales_Profit = {
    'Transport': 420,
    'Goods': 632001,
    'Manufacturing': 2600,
    'Extra_charges': 316,
    'Other':830
}

print("Total Profit From the Sales is: "+ str(SumOfValues_ofDictionary(sales_Profit)))
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_5.py"
Total Profit From the Sales is: 636167

Process finished with exit code 0
```

EXPERIMENT 6 (a)

- AIM:

  Write a Python program to read an entire text file.

PROGRAM:

```python
'''
[A] Write a Python program to read an entire text file.
'''

try:
    file = open("S:\VS code\python\college_practical\sample.txt", "rt")
    try:
        contents = file.read()
        print("Content of the file: \n")
        print(contents)
    except IOError:
        print("Unable to read file due to some error")
    else:
        print("\n File has been closed")
    finally:
        file.close()
except FileNotFoundError:
    print("The specified file was not found.")
```

Output:

```
Content of the file:

Title: "Exploring the Marvels of Quantum Computing"

Quantum computing, a groundbreaking field at the intersection of quantum physics and computer science, promises to revolutionize the way we process information.
Unlike classical computers, which use bits to represent information as either 0 or 1, quantum computers leverage quantum bits or qubits.
Qubits, owing to the principles of superposition and entanglement in quantum mechanics, can exist in multiple states simultaneously, exponentially expanding the computational
The potential applications of quantum computing are vast and diverse.
In fields such as cryptography, drug discovery, optimization problems, and artificial intelligence, quantum computers hold the promise of solving complex problems that are cur
For instance, quantum computers could revolutionize cryptography by efficiently factoring large numbers, which is crucial for secure communication.
However, quantum computing is not without its challenges.
Maintaining qubits in a coherent state, minimizing errors caused by decoherence and noise, and scaling up the number of qubits are significant hurdles that researchers are act
Despite these challenges, the rapid progress in quantum computing research has led to the development of quantum algorithms and prototypes of quantum computers by companies li
With continued advancements, quantum computing is poised to unlock new frontiers in science, technology, and beyond, offering unparalleled computational power to tackle the mo

 File has been closed
```

EXPERIMENT 6 (b)

- AIM:

  Write a Python program to append text to a file and display the text.

PROGRAM:

Practical.py

```python
'''
[B] Write a Python program to append text to a file and display the text.
'''
try:
    with open(r"S:\VS code\python\college_practical\sample_2.txt", "a+") as file:
        add_content = "\nThis is new content that is added using append method"
        file.write(add_content)
        file.seek(0)
        read_content = file.read()
        print("Content of the file:\n", read_content)
        print("\nFile has been closed")
except FileNotFoundError:
    print("The file is not found.")
except IOError:
    print("An error occurred while reading or writing the file.")
finally:
    file.close()
```

Output:

```
Content of the file:
 Lorem ipsum dolor sit amet consectetur, adipisicing elit.
Incidunt qui dolore consequuntur, impedit voluptas, eligendi voluptatem sed alias itaque cupiditate officia perferendis ipsa nulla deserunt iste?
Distinctio cum facilis accusantium nobis rerum voluptatibus, ad nostrum quod corrupti ex sapiente maxime pariatur sed perspiciatis aspernatur facere aliquam laborum?
Quasi cum perspiciatis omnis delectus sapiente minima beatae pariatur architecto et, id odio ullam fugiat recusandae officiis, neque aut quod aliquam nihil magni?
This is new content that is added using append method
This is new content that is added using append method
This is new content that is added using append method

File has been closed
```

EXPERIMENT 6 (c)

- AIM:

  Write a Python program to read last n lines of a file.

PROGRAM:

Practical.py

```python
'''
[C] Write a Python program to read last n lines of a file.
'''

def read_last_n_lines(file_path, n):
    try:
        with open("S:\VS code\python\college_practical\sample.txt", "r") as file:
            lines = file.readlines()
            last_n_lines = lines[-n:]
            return last_n_lines
    except FileNotFoundError:
        print("The specified file was not found.")
    except IOError:
        print("An error occurred while reading the file.")

file_path = "sample.txt"
n = 5

last_n_lines = read_last_n_lines(file_path, n)
if last_n_lines:
    print(f"Last {n} lines of the file:")
    for line in last_n_lines:
        print(line.strip())
```

Output:

```
Last 5 lines of the file:
For instance, quantum computers could revolutionize cryptography by efficiently factoring large numbers, which is crucial for secure communication.
However, quantum computing is not without its challenges.
Maintaining qubits in a coherent state, minimizing errors caused by decoherence and noise, and scaling up the number of qubits are significant hurdles that researchers are a
Despite these challenges, the rapid progress in quantum computing research has led to the development of quantum algorithms and prototypes of quantum computers by companies
With continued advancements, quantum computing is poised to unlock new frontiers in science, technology, and beyond, offering unparalleled computational power to tackle the

Process finished with exit code 0
```

EXPERIMENT 7 (a)

- AIM:

  Design a class that store the information of student and display the same.

PROGRAM:

Practical.py

```python
'''
[A] Design a class that store the information of student and display the same
'''

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

    def display_info(self):
        print("Student Information:")
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Age: {self.age}")
        print(f"Grade: {self.grade}")

student1 = Student("Sumit Dubey", 24, 19, "A")
student2 = Student("Shivam Dubey", 22, 18, "O")

student1.display_info()
print()
student2.display_info()
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_7.py"
Student Information:
Name: Sumit Dubey
Roll Number: 24
Age: 19
Grade: A

Student Information:
Name: Shivam Dubey
Roll Number: 22
Age: 18
Grade: O
```

EXPERIMENT 7 (b)

- AIM:

  Implement the concept of inheritance using python.

PROGRAM:

Practical.py

```python
'''
[B] Implement the concept of inheritance using python
'''

class Person:
    def __init__(self, fname, lname):
        self.fname = fname
        self.lname = lname

    def Fullname(self):
        print("FullName: ")
        print(self.fname+" "+ self.lname)

class Student(Person):
    pass

std1 = Student("Sumit", "Dubey")
std1.Fullname()
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_7.py"
FullName:
Sumit Dubey
```

EXPERIMENT 7 (c)

- AIM:

  Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a constructor which takes the parameters x and y (these should
  all be numbers).

  i. Write a method called add which returns the sum of the attributes x and y.
  ii. Write a class method called multiply, which takes a single number parameter a and returns the product of a and MULTIPLIER.
  iii. Write a static method called subtract, which takes two number parameters, b
  and c, and returns b - c.
  iv. Write a method called value which returns a tuple containing the values of x
  and y. Make this method into a property, and write a setter and a deleter for manipulating the values of x and y.

PROGRAM:

Practical.py

```python
'''
[C] Create a class called Numbers, which has a single class attribute called
MULTIPLIER, and a constructor which takes the parameters x and y (these should
all be numbers).

i. Write a method called add which returns the sum of the attributes x and y.
ii. Write a class method called multiply, which takes a single number
parameter a and returns the product of a and MULTIPLIER.
iii. Write a static method called subtract, which takes two number parameters, b
and c, and returns b - c.
iv. Write a method called value which returns a tuple containing the values of x
and y. Make this method into a property, and write a setter and a deleter for
manipulating the values of x and y.

'''

class Numbers:
    MULTIPLIER = 10

    def __init__(self, x, y):
        self._x = x
        self._y = y

    def add(self):
        return self._x + self._y

```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_7.py"
Enter a number: 20
Enter a number: 20
Sum: 40
Product: 30
Difference: 0
Value: (20, 20)
New value after setting: (20, 20)

Process finished with exit code 0
```

EXPERIMENT 8 (a)

- AIM:

  Open a new file in IDLE (“New Window” in the “File” menu) and save it as geometry.py in the directory where you keep the files you create for this course.
  Then copy the functions you wrote for calculating volumes and areas in the “Control Flow and Functions” exercise into this file and save it.
  Now open a new file and save it in the same directory. You should now be able
  to import your own module like this:
  import geometry

  Try and add print dir(geometry) to the file and run it.
  Now write a function pointyShapeVolume(x, y, squareBase) that calculates the
  volume of a square pyramid if squareBase is True and of a right circular cone if
  squareBase is False. x is the length of an edge on a square if squareBase is True
  and the radius of a circle when squareBase is False. y is the height of the object.
  First use squareBase to distinguish the cases. Use the circleArea and squareArea
  from the geometry module to calculate the base areas.

PROGRAM:

- Geometry.py

```python
import math

def squareArea(side):
    return side ** 2

def rectangleArea(length, width):
    return length * width

def circleArea(radius):
    return math.pi * radius ** 2

def triangleArea(base, height):
    return 0.5 * base * height

def cubeVolume(side):
    return side ** 3

def rectangularPrismVolume(length, width, height):
    return length * width * height

def cylinderVolume(radius, height):
    return math.pi * radius ** 2 * height

def pyramidVolume(base_area, height):
    return (1/3) * base_area * height

def coneVolume(base_area, height):
    return (1/3) * base_area * height

def pointyShapeVolume(x, y, squareBase):
    if squareBase:
        base_area = squareArea(x)
        volume = pyramidVolume(base_area, y)
    else:
        base_area = circleArea(x)
        volume = coneVolume(base_area, y)
    return volume
```

- Practical.py

Practical.py

```python
'''
[A] Open a new file in IDLE ("New Window" in the "File" menu) and save it as
geometry.py in the directory where you keep the files you create for this course.
Then copy the functions you wrote for calculating volumes and areas in the
"Control Flow and Functions" exercise into this file and save it.
Now open a new file and save it in the same directory. You should now be able
to import your own module like this:
import geometry

Try and add print dir(geometry) to the file and run it.
Now write a function pointyShapeVolume(x, y, squareBase) that calculates the
volume of a square pyramid if squareBase is True and of a right circular cone if
squareBase is False. x is the length of an edge on a square if squareBase is True
and the radius of a circle when squareBase is False. y is the height of the object.
First use squareBase to distinguish the cases. Use the circleArea and squareArea
from the geometry module to calculate the base areas.
'''
import geometry

print(dir(geometry))
print("Area of Square: ", str(geometry.squareArea(6)))
print("Area of circle: ", str(geometry.circleArea(8)))
print(geometry.pointyShapeVolume(4,2,True))
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_8.py"
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'circleArea', 'coneVolume', 'cubeVolume', 'cylinderVolume', 'math',
Area of Square:  36
Area of circle:  201.06192982974676
10.666666666666666
```

EXPERIMENT 8 (b)

- AIM:

  Write a program to implement exception handling.

PROGRAM:

Practical.py

```python
'''
[B] Write a program to implement exception handling.
'''

def divide(x, y):
    try:
        result = x / y
        print("Result of division:", result)
    except ZeroDivisionError:
        print("Error: Division by zero")
    except TypeError:
        print("Error: Unsupported operand type")
    except Exception as e:
        print("An unexpected error occurred:", e)

print("Case 1:")
divide(10, 2)

print("\nCase 2:")
divide(10, 0)

print("\nCase 3:")
divide(10, '2')

print("\nCase 4:")
divide(10, '0')
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_8.py"
Case 1:
Result of division: 5.0

Case 2:
Error: Division by zero

Case 3:
Error: Unsupported operand type

Case 4:
Error: Unsupported operand type

Process finished with exit code 0
```

EXPERIMENT 9 (a)

- AIM:

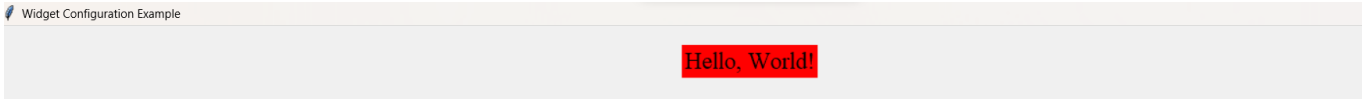
  Try to configure the widget with various options like: bg=”red”, family=”times”,
  size=18

PROGRAM:

Practical.py

```python
'''
[A] Try to configure the widget with various options like: bg=”red”, family=”times”,
size=18
'''

import tkinter as tk

def configure_widget(widget):
    widget.config(bg="red", font=("Times", 18))

root = tk.Tk()
root.title("Widget Configuration Example")

label = tk.Label(root, text="Hello, World!")

configure_widget(label)

label.pack(padx=20, pady=20)

root.mainloop()
```

Output:



EXPERIMENT 9 (b)

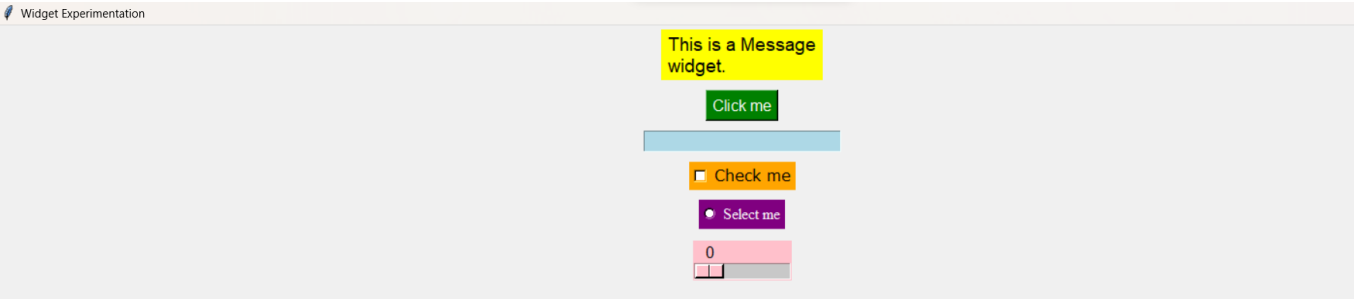
- AIM:
  Try to change the widget type and configuration options to experiment with other widget types like Message, Button, Entry, Checkbutton, Radiobutton, Scale
  etc.

PROGRAM:

Practical.py

```python
'''
[B] Try to change the widget type and configuration options to experiment with
other widget types like Message, Button, Entry, Checkbutton, Radiobutton, Scale
etc.
'''
import tkinter as tk

def configure_widget(widget, **kwargs):
    widget.config(**kwargs)

root = tk.Tk()
root.title("Widget Experimentation")

message = tk.Message(root, text="This is a Message widget.", width=200)
configure_widget(message, bg="yellow", font=("Arial", 14))

button = tk.Button(root, text="Click me")
configure_widget(button, bg="green", fg="white", font=("Helvetica", 12))

entry = tk.Entry(root)
configure_widget(entry, bg="lightblue", font=("Courier", 12))

checkbutton = tk.Checkbutton(root, text="Check me")
configure_widget(checkbutton, bg="orange", font=("Verdana", 12))

radiobutton = tk.Radiobutton(root, text="Select me")
configure_widget(radiobutton, bg="purple", fg="white", font=("Times", 12))

scale = tk.Scale(root, from_=0, to=100, orient=tk.HORIZONTAL)
configure_widget(scale, bg="pink", font=("Arial", 12))

message.pack(pady=5)
button.pack(pady=5)
entry.pack(pady=5)
checkbutton.pack(pady=5)
radiobutton.pack(pady=5)
scale.pack(pady=5)

root.mainloop()
```

Output:



EXPERIMENT 10 (a)

- AIM:
  Design a simple database application that stores the records and retrieve the Same.

PROGRAM:

Practical.py

```python
'''
[A] Design a simple database application that stores the records and retrieve the
same
'''

import sqlite3

def create_connection(db_file):
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print("Connected to SQLite database.")
    except sqlite3.Error as e:
        print("Error:", e)
    return conn

def create_table(conn, create_table_sql):
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
        print("Table created successfully.")
    except sqlite3.Error as e:
        print("Error:", e)

def insert_record(conn, record):
    sql = ''' INSERT INTO records(name, age, city)
              VALUES(?,?,?) '''
    cur = conn.cursor()
    cur.execute(sql, record)
    conn.commit()
```

```python
    print("Record inserted successfully.")

def retrieve_records(conn):
    cur = conn.cursor()
    cur.execute("SELECT * FROM records")

    rows = cur.fetchall()

    print("\nRecords:")
    for row in rows:
        print(row)

def main():
    database = "mydatabase.db"

    create_table_sql = """ CREATE TABLE IF NOT EXISTS records (
                                id INTEGER PRIMARY KEY,
                                name TEXT NOT NULL,
                                age INTEGER,
                                city TEXT
                            ); """

    conn = create_connection(database)
    if conn is not None:
        create_table(conn, create_table_sql)

        records_to_insert = [
            ('Sumit', 19, 'Mumbai'),
            ('Shivam', 18, 'Dombivali'),
            ('Himayu', 17, 'Kalyan'),
            ('Siddarth', 20, 'Ambarnath')
        ]
        for record in records_to_insert:
            insert_record(conn, record)

        retrieve_records(conn)

        conn.close()
        print("\nConnection to SQLite database closed.")
    else:
        print("Error: Unable to establish database connection.")

if __name__ == '__main__':
    main()
```

Output:

```
C:\Users\SUMIT\AppData\Local\Programs\Python\Python312\python.exe "S:\VS code\python\college_practical\practical_10.py"
Connected to SQLite database.
Table created successfully.
Record inserted successfully.
Record inserted successfully.
Record inserted successfully.
Record inserted successfully.

Records:
(1, 'Alice', 30, 'New York')
(2, 'Bob', 25, 'Los Angeles')
(3, 'Charlie', 35, 'Chicago')
(4, 'Sumit', 19, 'Mumbai')
(5, 'Shivam', 18, 'Dombivali')
(6, 'Himayu', 17, 'Kalyan')
(7, 'Siddarth', 20, 'Ambarnath')
```

EXPERIMENT 10 (b)

- AIM:
  Design a database application to search the specified record from the database.

PROGRAM:

Practical.py

```python
'''
[B] Design a database application to search the specified record from the database.
'''
import sqlite3

def create_connection(db_file):
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print("Connected to SQLite database.")
    except sqlite3.Error as e:
        print("Error:", e)
    return conn

def create_table(conn, create_table_sql):
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
        print("Table created successfully.")
    except sqlite3.Error as e:
        print("Error:", e)

def insert_record(conn, record):
    sql = ''' INSERT INTO records(name, age, city)
              VALUES(?,?,?) '''
    cur = conn.cursor()
    cur.execute(sql, record)
    conn.commit()
    print("Record inserted successfully.")
```

```python
def retrieve_records(conn):
    cur = conn.cursor()
    cur.execute("SELECT * FROM records")

    rows = cur.fetchall()

    print("\nRecords:")
    for row in rows:
        print(row)

def search_records(conn, criteria):
    cur = conn.cursor()
    cur.execute("SELECT * FROM records WHERE name=?", (criteria,))

    rows = cur.fetchall()

    print("\nSearch Results:")
    for row in rows:
        print(row)

def main():
    database = "mydatabase.db"

    create_table_sql = """ CREATE TABLE IF NOT EXISTS records (
                                id INTEGER PRIMARY KEY,
                                name TEXT NOT NULL,
                                age INTEGER,
                                city TEXT
                            ); """

    conn = create_connection(database)
    if conn is not None:
        create_table(conn, create_table_sql)

        records_to_insert = [
            ('Alice', 30, 'New York'),
            ('Bob', 25, 'Los Angeles'),
            ('Charlie', 35, 'Chicago')
        ]
        for record in records_to_insert:
            insert_record(conn, record)

        retrieve_records(conn)

        search_criteria = 'Bob'
        search_records(conn, search_criteria)

        conn.close()
        print("\nConnection to SQLite database closed.")
    else:
        print("Error: Unable to establish database connection.")

if __name__ == '__main__':
    main()
```

Output:

```
Connected to SQLite database.
Table created successfully.
Record inserted successfully.
Record inserted successfully.
Record inserted successfully.

Records:
(1, 'Alice', 30, 'New York')
(2, 'Bob', 25, 'Los Angeles')
(3, 'Charlie', 35, 'Chicago')
(4, 'Sumit', 19, 'Mumbai')
(5, 'Shivam', 18, 'Dombivali')
(6, 'Himayu', 17, 'Kalyan')
(7, 'Siddarth', 20, 'Ambarnath')
(8, 'Alice', 30, 'New York')
(9, 'Bob', 25, 'Los Angeles')
(10, 'Charlie', 35, 'Chicago')

Search Results:
(2, 'Bob', 25, 'Los Angeles')
(9, 'Bob', 25, 'Los Angeles')

Connection to SQLite database closed.
```

EXPERIMENT 10 (c)

- AIM:
  Design a database application to that allows the user to add, delete and modify the records.

PROGRAM:

Practical.py

```python
'''
[C] Design a database application to that allows the user to add, delete and modify
the records
'''
import sqlite3

def create_connection(db_file):
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print("Connected to SQLite database.")
    except sqlite3.Error as e:
        print("Error:", e)
    return conn

def create_table(conn, create_table_sql):
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
        print("Table created successfully.")
    except sqlite3.Error as e:
        print("Error:", e)

def insert_record(conn, record):
    sql = ''' INSERT INTO records(name, age, city)
              VALUES(?,?,?) '''
    cur = conn.cursor()
    cur.execute(sql, record)
    conn.commit()
    print("Record inserted successfully.")

```

```python
def retrieve_records(conn):
    cur = conn.cursor()
    cur.execute("SELECT * FROM records")

    rows = cur.fetchall()

    print("\nRecords:")
    for row in rows:
        print(row)

def delete_record(conn, record_id):
    sql = 'DELETE FROM records WHERE id=?'
    cur = conn.cursor()
    cur.execute(sql, (record_id,))
    conn.commit()
    print("Record deleted successfully.")

def update_record(conn, record_id, new_data):
    sql = ''' UPDATE records
              SET name = ? ,
                  age = ? ,
                  city = ?
              WHERE id = ?'''
    cur = conn.cursor()
    cur.execute(sql, (*new_data, record_id))
    conn.commit()
    print("Record updated successfully.")

def main():
    database = "mydatabase.db"
```

```python

    create_table_sql = """ CREATE TABLE IF NOT EXISTS records (
                                id INTEGER PRIMARY KEY,
                                name TEXT NOT NULL,
                                age INTEGER,
                                city TEXT
                            ); """

    conn = create_connection(database)
    if conn is not None:
        # Create a table
        create_table(conn, create_table_sql)

        records_to_insert = [
            ('Alice', 30, 'New York'),
            ('Bob', 25, 'Los Angeles'),
            ('Charlie', 35, 'Chicago')
        ]
        for record in records_to_insert:
            insert_record(conn, record)

        retrieve_records(conn)

        new_record = ('David', 28, 'San Francisco')
        insert_record(conn, new_record)
        print("\nAfter adding a new record:")
        retrieve_records(conn)

        record_id_to_delete = 2
        delete_record(conn, record_id_to_delete)
        print("\nAfter deleting a record:")
        retrieve_records(conn)

        record_id_to_update = 3
        new_data = ('Charlie Brown', 40, 'Boston')
        update_record(conn, record_id_to_update, new_data)
        print("\nAfter updating a record:")
        retrieve_records(conn)

        conn.close()
        print("\nConnection to SQLite database closed.")
    else:
        print("Error: Unable to establish database connection.")

if __name__ == '__main__':
    main()
```

Output:

```
Connected to SQLite database.
Table created successfully.
Record inserted successfully.
Record inserted successfully.
Record inserted successfully.

Records:
(1, 'Alice', 30, 'New York')
(2, 'Bob', 25, 'Los Angeles')
(3, 'Charlie', 35, 'Chicago')
(4, 'Sumit', 19, 'Mumbai')
(5, 'Shivam', 18, 'Dombivali')
(6, 'Himayu', 17, 'Kalyan')
(7, 'Siddarth', 20, 'Ambarnath')
(8, 'Alice', 30, 'New York')
(9, 'Bob', 25, 'Los Angeles')
(10, 'Charlie', 35, 'Chicago')
(11, 'Alice', 30, 'New York')
(12, 'Bob', 25, 'Los Angeles')
(13, 'Charlie', 35, 'Chicago')
Record inserted successfully.
```

```
Records:
(1, 'Alice', 30, 'New York')
(2, 'Bob', 25, 'Los Angeles')
(3, 'Charlie', 35, 'Chicago')
(4, 'Sumit', 19, 'Mumbai')
(5, 'Shivam', 18, 'Dombivali')
(6, 'Himayu', 17, 'Kalyan')
(7, 'Siddarth', 20, 'Ambarnath')
(8, 'Alice', 30, 'New York')
(9, 'Bob', 25, 'Los Angeles')
(10, 'Charlie', 35, 'Chicago')
(11, 'Alice', 30, 'New York')
(12, 'Bob', 25, 'Los Angeles')
(13, 'Charlie', 35, 'Chicago')
(14, 'David', 28, 'San Francisco')
Record deleted successfully.

After deleting a record:

Records:
(1, 'Alice', 30, 'New York')
(3, 'Charlie', 35, 'Chicago')
(4, 'Sumit', 19, 'Mumbai')
```