LAB CYCLE - 1

Experiment No : 1

Date : 30/09/2024

Aim :

Write a program that prompts the user to enter his first name and last name and then displays a message.

Pseudocode :

1. Read first name.
2. Read last name.
3. Print “Greetings !!! First name Last name”

Method :

Function	Description	Syntax
input()	Allows user input (Returns a string value)	input(prompt)
print()	Print the specified message to the screen	print(object)

Source Code :

```
a=input("Enter your First name:")
b=input("Enter your Last name:")
print("Greetings !!!",a,b)
```

Output :

```
Enter your First Name: Shreyas
Enter your Last Name: S
Greetings!!! Shreyas S
```

Result :

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 30/09/2024

Aim :

Write a program to demonstrate different number data types in python.

Pseudocode :

1. Set x = 5, y = 7.4, z = 5 + 2j.
2. Print the type of x, y, and z.
3. Print x, y, and z with their respective types.

Method :

Function	Description	Syntax
type()	Returns the type of an object	type(object)

Source Code :

```
x,y,z=5,7.4,5+2j
print("Data Type of x=",type(x))
print("Data Type of y=",type(y))
print("Data Type of z=",type(z))
print("Integer:",x)
print("Float:",y)
print("Complex number:",z)
```

Output :

```
Data Type of x = <class 'int'>
Data Type of y = <class 'int'>
Data Type of z = <class 'int'>
Integer: 5
Float: 7.4
Complex Number: (5+2j)
```

Result :

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 30/09/2024

Aim :

Write a program to calculate the area of a circle by reading inputs from the user.

Pseudocode :

1. Input r as the radius.
2. area = 3.14 * r * r
3. Print area.

Source Code :

```
r=float(input("Enter the radius of the circle:"))
area=3.14*r*r
print("The Area of the circle =",area)
```

Output :

Enter the radius of the Circle: 7

The Area of the Circle = 153.86

Result :

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 30/09/2024

Aim :

Write a program to calculate the salary of an employee given his basic pay (to be entered by the user) .HRA = 10 percent of the basic pay, TA = 5 percent of the basic pay.

Pseudocode :

1. Input BP as basic pay.
2. HRA = 0.10 * BP
3. TA = 0.05 * BP
4. Calculate salary = BP + HRA + TA
5. Print salary.

Source Code :

```
P=float(input("Enter the Basic salary of the Employee:"))
HRA=0.10*BP
TA=0.05*BP
salary=BP+HRA+TA
print("The salary of the employee:",salary)
```

Output :

Enter the Basic salary of the Employee: 12000

The salary of the Employee: 13800

Result :

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 30/09/2024

Aim :

Write a Python program to perform arithmetic operations on two integer numbers.

Pseudocode :

1. Input x and y as two numbers
2. Calculate and print the sum, difference, product, division, and remainder of x and y.

Source Code :

```
x=int(input("Enter the first number:"))
y=int(input("Enter the second number:"))
print(f"Sum:{x}+{y}={x+y}")
print(f"Difference:{x}-{y}={x-y}")
print(f"Product:{x}*{y}={x*y}")
print(f"Division:{x}/{y}={x/y}")
print(f"Remainder:{x}%{y}={x%y}")
```

Output :

Enter the First Number: 10

Enter the Second Number: 2

Sum: 10 + 2 = 12

Difference: 10 – 2 = 8

Product: 10 * 2 = 20

Division: 10 / 2 = 5.0

Remainder: 10 % 2 = 0

Result :

The program is successfully executed and the output is verified.

Experiment No : 6

Date : 30/09/2024

Aim :

Write a Python program to get a string which is n (non-negative integer) copies of given string.

Pseudocode :

1. Input str as a string
2. Input n as the number of copies
3. If n < 0, print "Invalid"
4. Else, print str repeated n times.

Source Code :

```
str=input("Enter a string:")
n=int(input("Enter the number of copies of string:"))
if n<0 :
   print("Invalid")
else:
   print(str*n)
```

Output :

Enter a String: Apple

Enter the number of copies of String: 4

AppleAppleAppleApple

Result :

The program is successfully executed and the output is verified.

Experiment No : 7

Date : 30/09/2024

Aim :

Program to accept an integer n and compute n+nn+nnn.

Pseudocode :

1. Input n as a string
2. Print n + nn + nnn
3. Calculate and print the sum of n, nn, and nnn

Source Code :

```
n=input("Enter the string:")
print(n,"+",n*2,"+",n*3)
sum=(int(n)+int(n*2)+int(n*3))
print("Sum=",sum)
```

Output :

Enter the String: 7

7 + 77 + 777

Sum = 861

Result :

The program is successfully executed and the output is verified.

Experiment No : 8

Date : 30/09/2024

Aim :

Find biggest of 3 numbers entered.

Pseudocode :

1. Input a, b, c as three numbers.
2. If a > b and a > c, set largest = a
3. Else if b > a and b > c, set largest = b
4. Else, set largest = c
5. Print largest.

Source Code :

```
a=int(input("Enter the First number:"))
b=int(input("Enter the Second number:"))
c=int(input("Enter the Third number:"))
if a>b and a>c:
   largest=a
elif b>a and b>c:
  largest=b
else:
   largest=c
print("The largest number is:",largest)
```

Output :

Enter the First Number: 345

Enter the Second Number: 123

Enter the Third Number: 256

The largest number is : 345

Result :

The program is successfully executed and the output is verified.

Experiment No : 9

Date : 07/10/2024

Aim :

Program to determine whether a year is a leap year or not.

Pseudocode :

1. Input year.
2. If year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
   Print the year is a leap year
3. Else, print the year is not a leap year.

Source Code :

```
year=int(input("Enter the year:"))
if  year%4==0 and (year%100!=0 or year%400==0):
  print(year,"is a leap year")
else:
   print(year,"is not a leap year")
```

Output :

Enter the Year : 2024

2024 is a Leap year

Enter the Year: 2025

2025 is not a Leap year

Result :

The program is successfully executed and the output is verified.

Experiment No : 10

Date : 07/10/2024

Aim :

Write a Python program to determine the rate of entry-ticket in a trade fair based on age as follows:

Age	Rate
<10	7
>=10 and <60	10
>=60	5

Pseudocode :

1. Input age
2. If age < 10, set rate = 7
3. Else if 10 <= age < 60, set rate = 10
4. Else, set rate = 5
5. Print the ticket rate for the given age.

Source Code :

```
age=int(input("Enter your age:"))
if age<10:
    rate=7
elif age>=10 and age<60:
    rate=10
else:
    rate=5
print("The rate of the ticker for age",age,"is",rate)
```

Output :

Enter your age: 8

The rate of the Ticket for the age 8 is : 7

Enter your age: 35

The rate of the Ticket for the age 35 is : 10

Enter your age: 82

The rate of the Ticket for the age 82 is : 5

Result :

The program is successfully executed and the output is verified.

Experiment No : 11

Date : 07/10/2024

Aim :

Write a Python program to solve a quadratic equation.

Pseudocode :

1. Import math
2. Input coefficients a, b, c
3. Calculate d = b*b - 4*a*c
4. If a == 0, print error message
5. Else if d > 0, calculate and print two real solutions
6. Else if d == 0, calculate and print one real solution
7. Else, calculate and print two complex solutions.

Method :

Function	Description	Syntax
sqrt()	Returns the square root of a given number.	math.sqrt(x)

Source Code :

```
import math
a=float(input("Enter the coefficient of a:"))
b=float(input("Enter the coefficient of b:"))
c=float(input("Enter the coefficient of c:"))
d=(b*b)-(4*a*c)
if a==0:
   print("Coefficient 'a' cannot be zero")
elif d>0:
  sol1=(-b + math.sqrt(d))/(2*a)
   sol2=(-b - math.sqrt(d))/(2*a)
   print("There are two solutions:",sol1,"and",sol2)
```

```
elif d==0:
   sol=-b/(2*a)
   print("There is only one real solution:",sol)
else:
   real=-b/(2*a)
   imag=math.sqrt(abs(d))/(2*a)
   print(f"The solutions are complex numbers:{real}+i{imag} and {real}-i{imag}")
```

Output :

Enter the coefficient of a : 1

Enter the coefficient of b : -3

Enter the coefficient of c : 2

There are two solutions **:** 2.0 and 1.0

Enter the coefficient of a : 1

Enter the coefficient of b : -4

Enter the coefficient of c : 4

There is only one real solution **:** 2.0

Enter the coefficient of a : 1

Enter the coefficient of b : 2

Enter the coefficient of c : 5

The solutions are complex numbers **:** -1.0 + i 2.0 and -1.0 – i 2.0

Result :

The program is successfully executed and the output is verified.

LAB CYCLE - 2

Experiment No : 1

Date : 15/10/2024

Aim :

Create a string from the given string where the first and last character are exchanged.

Pseudocode :

1. Input str as a string.
2. If length of str is greater than 1

   Set r as last character + middle characters + first character
3. Print str and r.

Source Code :

```
str=input("Enter a word:")
if len(str)>1:
  r=str[-1]+str[1:-1]+str[0]
  print(str,"=>",r)
```

Output :

Enter a Word: python

python => nythop

Result :

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 15/10/2024

Aim :

Get a string from an input string where all occurrences of the first character are replaced with ‘$’, except the first character.

Pseudocode :

1. Input s as a string.
2. Set result as first character + rest of string with first character replaced by '$'
3. Print s and result.

Method :

Function	Description	Syntax
replace()	Replaces all occurances of a character with another.	string.replace(old, new)

Source Code :

```
s = input("Enter a string:")
result=s[0]+s[1:].replace(s[0],'$')
print(s,"->",result)
```

Output :

```
Enter a String: onion
onion -> oni$n
```

Result :

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 15/10/2024

Aim :

Create a single string separated with space from two strings by swapping the character at position 1.

Pseudocode :

1. Input str1 and str2.
2. Set swap_str1 as first character of str1 + second character of str2 + rest of str1
3. Set swap_str2 as first character of str2 + second character of str1 + rest of str2
4. Set result as swap_str1 + " " + swap_str2
5. Print result.

Source Code :

```
str1=input("Enter the string1: ")
str2=input("Enter the string2: ")
swap_str1=str1[0]+str2[1]+str1[2:]
swap_str2=str2[0]+str1[1]+str2[2:]
result=swap_str1+" "+swap_str2
print(result)
```

Output :

```
Enter the String1: Hello
Enter the String2: World
Hollo Werld
```

Result :

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 15/10/2024

Aim :

Count the number of characters (character frequency) in a string.

Pseudocode :

1. Input s as a string
2. For each unique character in s

    Print the character and its count in s

Method :

Function	Description	Syntax
count()	Counts the occurrences of a substring.	string.count(‘substring’)

Source Code :

```
s=input("Enter a string: ")
for char in set(s):
  print(char,":",s.count(char))
```

Output :

Enter a String: Malayalam

m : 2

a : 4

l : 2

y : 1

Result :

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 15/10/2024

Aim :

Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’.

Pseudocode :

1. Input s as a string
2. If s ends with "ing":

   Set r as s + 'ly'
3. Else:

   Set r as s + 'ing'
4. Print r.

Method :

Function	Description	Syntax
endswith()	Checks if the string ends with a specific substring.	string.endswith(‘substring’)

Source Code :

```
s=input("Enter the string: ")
if s.endswith('ing'):
  r=s+'ly'
else:
  r=s+'ing'
print(r)
```

Output :

```
Enter the String: run
Running
Enter the String: crying
cryingly
```

Result : The program is successfully executed and the output is verified.

Experiment No : 6

Date : 15/10/2024

Aim :

Store a list of first names. Count the occurrences of ‘a’ within the list.

Pseudocode :

1. Input names
2. Set a_count = 0
3. For each name in names:

   Add count of 'a' in name to a_count
4. Print a_count.

Method :

Function	Description	Syntax
lower()	Converts a string to lower case.	string.lower()

Source Code :

```
names=input("Enter the Names:").split()
a_count=0
for name in names:
  a_count=a_count+name.lower().count('a')
print(f"The letter 'a' has occured {a_count} times in the list")
```

Output :

Enter the Names: Ram Sai Amar Mike

The letter ‘a’ has occurred 4 times in the list

Result :

The program is successfully executed and the output is verified.

Experiment No : 7

Date : 15/10/2024

Aim :

Write a python program to read two lists color-list1 and color-list2. Print out all colors from color-list1 not contained in color-list2.

Pseudocode :

1. Input list1 and list2
2. Set new as colors in list1 not in list2
3. Print new.

Source Code :

```
list1=input("Enter the colors in List1:").split()
list2=input("Enter the colors in List2:").split()
new=[color for color in list1 if color not in list2]
print("colors that are in list1 and not in list2 :",new)
```

Output :

Enter the colors in List1: red black yellow green blue

Enter the colors in List2: red blue black brown grey

Colors that are in List1 and not in List2 : [‘yellow’ , ‘green’]

Result :

The program is successfully executed and the output is verified.

Experiment No : 8

Date : 15/10/2024

Aim :

Create a list of colors from comma-separated color names entered by the user. Display first and last colors.

Pseudocode :

1. Input colors as comma-separated list
2. Set First_color as first color in colors
3. Set Last_color as last color in colors
4. Print First_color and Last_color.

Method :

Function	Description	Syntax
split()	Splits a string into a list based on a separator.	string.split(“ , ”)

Source Code :

```
colors=input("Enter the colors name:").split(',')
print(colors)
First_color=colors[0]
Last_color=colors[-1]
print(f"First color:{First_color}\nLast color:{Last_color}")
```

Output :

Enter the colors name: red, black, blue, yellow

[‘red’, ’black’, ‘blue’, ‘yellow’]

First color: red

Last color: yellow

Result :

The program is successfully executed and the output is verified.

Experiment No : 9

Date : 15/10/2024

Aim :

Write a program to prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead.

Pseudocode :

1. Input integers as space-separated list
2. For each i in integers:
   If i > 100, set i to "over"
3. Print integers.

Source Code :

```
integers=input("Enter the list of integers:").split()
for i in range(len(integers)):
  if int(integers[i])>100:
integers[i]='over'
print(integers)
```

Output :

Enter the list of Integers : 12 123 34 445 2 78

[‘12’, ‘over’, ‘34’, ‘over’, ‘2’, ‘78’]

Result :

The program is successfully executed and the output is verified.

Experiment No : 10

Date : 15/10/2024

Aim :

From a list of integers, create a list after removing even numbers.

Pseudocode :

1. Input integers as space-separated list
2. Set odd = []
3. For each integer in integers:
   If integer is odd, add to odd
4. Print odd.

Source Code :

```
integers=input("Enter the list of integers:").split()
odd=[]
for i in range(len(integers)):
  if int(integers[i])%2!=0:
     odd.append(i)
print("List after removing Even numbers:",odd)
```

Output :

Enter the List of Integers: 0 1 2 3 4 5 6 12

List after removing Even Numbers: [1, 3, 5]

Result :

The program is successfully executed and the output is verified.

Experiment No : 11

Date : 15/10/2024

Aim :

Accept a list of words and return the length of the longest word.

Pseudocode :

1. Input words as space-separated list
2. Set longest = -1
3. For each word in words:

   If len(word) > longest, update longest and longest_word
4. Print longest_word and its length.

Method :

Function	Description	Syntax
len()	Returns the length of an object	len(object)

Source Code :

```
words=input(“Enter the words:”).split()
longest=-1
for i in words:
  if len(i)>longest
    longest=len(i)
    word=i
print(“The longest word:”,word)
print(f”The length of{word}:{longest}”)
```

Output :

Enter the Words: C Java HTML Python

The longest Word: Python

The length of Python: 6

Result :

The program is successfully executed and the output is verified.

Experiment No : 12

Date : 15/10/2024

Aim :

Write a program to prompt the user to enter two lists of integers and check

(a) Whether lists are of the same length.
(b) Whether the list sums to the same value.
(c) Whether any value occurs in both Lists.

Pseudocode :

1. Input lst1 and lst2
2. Check if len(lst1) == len(lst2)
3. Check if sum(lst1) == sum(lst2)
4. Find common elements in both lists
5. Print results of the checks and common elements.

Source Code :

```
lst1=[int(num) for num in input("Enter first list:").split()]
lst2=[int(num) for num in input("Enter second list:").split()]
length = len(lst1) == len(lst2)
lsum = sum(lst1) == sum(lst2)
common=set(lst1)&set(lst2)
if lenght:
  print("lists lenghts are same")
else:
  print("lists lenght are not same")
print(f"lists common elements: {common}")
if lsum:
  print("list sums to the same value")
else:
  print("list doesn't sums to the same value")
```

Output :

Enter First List: 1 2 3 4 5 6

Enter Second List: 4 6 5 7 8 9 11

Lists length are not Same

Lists common Elements: {4, 5, 6}

List doesn’t Sums to the same Value.

Result :

The program is successfully executed and the output is verified.

Experiment No : 13

Date : 21/10/2024

Aim :

Write a Python program to count the occurrences of each word in a line of text.

Hint: use split() function and dictionary.

Pseudocode :

1. Input sentence
2. Set freq_dict = {}
3. For each word in sentence:

Increment count of word in freq_dict

4. Print freq_dict

Source Code :

```
sentence=[word for word in input("enter a string:").lower().split()]
freq_dict={}
for word in sentence:
  if word in freq_dict:
     freq_dict[word]+=1
  else:
     freq_dict[word]=1
print("character occurance:")
for key,value  in freq_dict.items():
print(f"{key}:{value}")
```

Output :

Enter a String: There is a will there is a way.

Character Occurrence:

There: 2
is: 2
a: 2
will: 1
way: 1

Result :

The program is successfully executed and the output is verified.

Experiment No : 14

Date : 21/10/2024

Aim :

List comprehensions:

(a) Generate positive list of numbers from a given list of integers
(b) Square of N numbers
(c) Form a list of vowels selected from a given word
(d) Form a list ordinal value of each element of a word (Hint: use ord() to get ordinal values)

Pseudocode :

1. Given numbers
2. Set positive_numbers as list of positive numbers in numbers
3. Set squares as squares of first N numbers
4. Given word, set vowels as list of vowels in word
5. Given word, set ordinal_values as list of ordinal values of each character
6. Print results of each list comprehension.

Function	Description	Syntax
ord()	Returns the ASCII value of a character	ord(char)
range()	Generate a range of numbers	range(start, stop)

Source Code :

```
numbers = [-1, 2, -5, 7, -2, 18, 3]

print(numbers)

positive_numbers = [num for num in numbers if num > 0]

print(f"Positive numbers in the list  :", positive_numbers)

N = 5

squares = [num ** 2 for num in range(1, N + 1)]

print("Squares of first 6 numbers:", squares)
```

```
word = "comprehension"
vowels = [char for char in word if char in 'aeiou']
print(f"Vowels in the word: {word}", vowels)
word = "python"
ordinal_values = [ord(char) for char in word]
print(f'Ordinal Values: {word}: { ordinal_values }")
```

Output :

[-1, 2, -5, 7, -2, 18, 3]

Positive numbers in the List : [2, 7, 18, 3]

Squares of First 5 numbers: [1, 4, 9, 16, 25]

Vowels in the Word : Comprehension :: [‘o’, ‘e’, ‘e’, ‘i’, ‘o’]

Ordinal values : python : [112, 121, 116 104, 111, 110]

Result :

The program is successfully executed and the output is verified.

Experiment No : 15

Date : 21/10/2024

Aim :

Sort dictionary in ascending and descending order.

Pseudocode :

1. Given my_dict
2. Sort my_dict by keys (ascending and descending)
3. Sort my_dict by values (ascending and descending)
4. Print each sorted dictionary.

Method :

Function	Description	Syntax
sorted()	Sorts the elements of an iterable.	sorted(iterable, reverse= bool)
dict()	Creates a dictionary from a sequence of key-value pairs.	dict(sequence)
item()	Returns key-value pairs of dictionary as tubles.	dict.item()

Source Code :

```
my_dict = {'rice': 3, 'wheat': 6, 'barley': 10, 'corn': 4}

keys_asc = dict(sorted(my_dict.items()))

print("Sorted by keys (ascending):", keys_asc)

keys_desc = dict(sorted(my_dict.items(), reverse = True))

print("Sorted by keys (descending):", keys_desc)

values_asc = dict(sorted(my_dict.items(), key = lambda item: item[1]))

print("Sorted by values (ascending):", values_asc)

values_desc = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse = True))

print("Sorted by values (descending):", values_desc)
```

Output :

Sorted By Keys(Ascending): {‘barley’:10, ‘corn’: 4, ‘rice’: 3, ‘wheat’: 6}

Sorted By Keys(Descending): {‘wheat’: 6, ‘rice’: 3, ‘corn’: 4, ‘barley’:10}

Sorted By Values(Ascending): {‘rice’: 3, ‘corn’: 4, ‘wheat’: 6, ‘barley’: 10}

Sorted By Values(Descending): {‘barley’: 10, ‘wheat’: 6, ‘corn’: 4, ‘rice’: 3}

Result :

The program is successfully executed and the output is verified.

Experiment No : 16

Date : 21/10/2024

Aim :

Merge two dictionaries.

Pseudocode :

1. Given dict1 and dict2
2. Merge dict2 into dict1
3. Print merged dictionary.

Method :

Function	Description	Syntax
update()	Update or merge dictionary with new key-value pairs.	dict1.update(dict2)

Source Code :

```
dict1 = {'rice': 3, 'wheat': 5}
dict2 = {'ragi': 2, 'jowar': 4}
print(dict1)
print(dict2)
dict1.update(dict2)
print(f"Merged :{dict1}")
```

Output :

{‘rice’: 3, ‘wheat’: 5}

{‘ragi’: 2, ‘jowar’: 4}

Merged Dictionary: {‘rice’: 3, ‘wheat’: 5, ‘ragi’: 2, ‘jowar’: 4}

Result :

The program is successfully executed and the output is verified.

LAB CYCLE – 3

Experiment No : 1

Date : 04/11/2024

Aim :

Write a program to find the factorial of a number.

Pseudocode :

1. Input n

2. Set fact = 1

3. If n < 0, print an error message

4. Else

   Loop from i = 1 to n

   Multiply fact by i

   print “fact”

Source Code :

```
n=int(input("Enter the number to find its Factorial:"))
fact=1
for i in range(1,n+1):
    fact*=i
if n<0:
    print("Enter a value greater than 0")
else:
    print(f"factorial of {n} is:{fact}")
```

Output :

Enter the number to find its Factorial : 6

The Factorial of 6 is : 720

Result :

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 04/11/2024

Aim :

Generate Fibonacci series of N terms.

Pseudocode :

1. Input n
2. Set a = 0, b = 1
3. If n < 0, print an error message
4. Else

    Loop n times

    Append a to fibonacci

    Update a, b to b, a + b
5. Print fibonacci

Source Code :

```
n=int(input("Enter the number to find its fibonacci series:"))
a,b = 0,1
fibonacci=[]
if(n<0):
    print("Enter the number greater than 0")
else:
    for i in range(n):
        fibonacci.append(a)
        a,b = b,a+b
print(f"Fibonacci Series of {n}:{fibonacci}")
```

Output :

Enter the number to find its Fibonacci Series : 10

The Fibonacci series of 10 : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

Result :

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 04/11/2024

Aim :

Write a program to find the sum of all items in a list. [Using for loop]

Pseudocode :

1. Input n
2. Split n into numbers
3. Set sum = 0
4. For each number in numbers

    Add number (converted to integer) to sum
5. Print sum

Source Code :

```
n=input("Enter the numbers:")
numbers=n.split()
sum=0
print(numbers)
for j in numbers:
    sum +=int(j)
print("sum of numbers=",sum)
```

Output :

Enter the Numbers: 1 2 3 4 5 6 7

[‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’]

Sum of Numbers = 28

Result :

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 04/11/2024

Aim :

Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.

Pseudocode :

1. Import math
2. Input start and end
3. Set result = []
4. Loop i from sqrt(start) to sqrt(end)
   Calculate square = i * i
   If square has all even digits and is 4-digit
   Append square to result
5. Print result

Source Code :

```
import math
start = int(input("Enter the start of the range:"))
end = int(input("Enter the end of the range:"))
result = []
for i in range(math.isqrt(start), math.isqrt(end) + 1):
   square = i * i
  if 1000 <= square <= 9999:
      Even = True
      for digit in str(square):
          if  int(digit) % 2 != 0:
```

```
            Even = False
             break
    if Even:
        result.append(square)
print("Four-digit perfect squares with  all even digits:", result)
```

Output :

Enter the Starting Range: 1000

Enter the Ending Range: 9999

Four-digit Perfect squares with all even Digits: [4624, 6084, 6400, 8464]

Result :

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 04/11/2024

Aim :

Write a program using a for loop to print the multiplication table of n, where n is entered by the user.

Pseudocode :

1. Input n
2. Print Multiplication Table of n
3. Loop i from 1 to 10

    Print i * n

Source Code :

```
n=int(input("Enter a number to print its Multiplication Table:"))
print("Multiplication Table of ",n,":")
for i in range(1,11):
    print(f"{i}*{n} = {n*i}")
```

Output :

Enter a Number to Print its Multiplication Table: 7

Multiplication Table of 7 :

1 * 7 = 7

2 * 7 = 14

3 * 7 = 21

4 * 7 = 28

5 * 7 = 35

6 * 7 = 42

7 * 7 = 49

8 * 7 = 56

9 * 7 = 63

10 * 7 = 70

Result :

The program is successfully executed and the output is verified.

Experiment No : 6

Date : 04/11/2024

Aim :

Write a program to display alternate prime numbers till N.

Pseudocode :

1. Import math
2. Input n
3. Set Prime_numbers = []
4. Loop num from 2 to n
   Check if num is prime
   If prime, append num to Prime_numbers
5. Print every second element in Prime_numbers.

Source Code :

```
import math
n = int(input("Enter the Range: "))
Prime_numbers = []
for num in range(2, n + 1):
    prime = True
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            prime = False
            break
    if prime:
        Prime_numbers.append(num)
print(f"Alternative Prime Numbers till {n} are:")
```

```
for p in range(0, len(Prime_numbers), 2):
    print(Prime_numbers[p])
```

Output :

Enter the Range : 30

Alternative Prime Numbers till 30 are :

[2, 5, 11, 17, 23]

Result :

The program is successfully executed and the output is verified.

Experiment No : 7

Date : 11/11/2024

Aim :

Write a program to compute and display the sum of all integers that are divisible by 6 but not by 4, and that lie below a user-given upper limit.

Pseudocode :

1. Input limit
2. Set sum = 0
3. Loop i from 1 to limit

   If i is divisible by 6 and not by 4

   Add i to sum
4. Print sum

Source Code :

```
limit=int(input("Enter the limit:"))
sum=0
for i in range(1,limit):
    if i%6==0 and i%4!=0:
        sum+=i
print(f"The sum of all integers upto {limit} that are divisible by 6 and not by 4:",sum)
```

Output :

```
Enter the limit: 20
The Sum of all integers upto 20 that are divisible by 6 and not by 4 :  24
```

Result :

The program is successfully executed and the output is verified.

Experiment No : 8

Date : 11/11/2024

Aim :

Calculate the sum of the digits of each number within a specified range.

Print the sum only if it is prime.

Pseudocode :

1. Input limit1 and limit2

2. Loop num from limit1 to limit2

    Calculate sum of digits of num

    If sum is prime, print num and its digit sum

Source Code :

```
limit1 = int(input("Enter the starting range:"))
limit2 = int(input("Enter the Ending range:"))
for num in range(limit1, limit2+1):
    digit_sum = 0
    for digit in str(num):
        digit_sum+=int(digit)
    if digit_sum > 1:
        prime=True
        for i in range(2, int(digit_sum ** 0.5) + 1):
            if digit_sum % i == 0:
                prime = False
                break
        if prime:
            print(f"Number = {num} ; Sum of Digits = {digit_sum}")
```

Output :

Enter the Starting Range: 20

Enter the Ending Range: 30

Number = 20 ; Sum of Digits = 2

Number = 21 ; Sum of Digits = 3

Number = 23 ; Sum of Digits = 25

Number = 25 ; Sum of Digits = 7

Number = 29 ; Sum of Digits = 11

Number = 30 ; Sum of Digits = 3

Result :

The program is successfully executed and the output is verified.

Experiment No : 9

Date : 11/11/2024

Aim :

A number is input through the keyboard. Write a program to determine if it’s palindromic.

Pseudocode :

1. Input number
2. If number equals its reverse, print "palindromic"
3. Else, print "not palindromic"

Source Code :

```
number = input("Enter a number:")
if number == number[::-1]:
    print("The number is palindromic.")
else:
    print("The number is not palindromic.")
```

Output :

Enter a Number : 1234

The Number is not Palindromic.

Enter a Number : 6006

The Number is Palindromic.

Result :

The program is successfully executed and the output is verified.

Experiment No : 10

Date : 11/11/2024

Aim :

Write a program to generate all factors of a number.

Pseudocode :

1. Input number

2. Set divisor = 1

3. While divisor <= number

    If number is divisible by divisor, print divisor

    Increment divisor

Source Code :

```
number = int(input("Enter a number to find its factors: "))
divisor = 1
factors=[]
factors.append(divisor)
print(f"Factors of {number} are:")
while divisor <= number:
   if number % divisor == 0:
      print(divisor)
   divisor += 1
```

Output :

Enter a Number to find its Factors: 50

The Factors of 50 are: [1, 2, 5, 10, 25, 50]

Result :

The program is successfully executed and the output is verified.

Experiment No : 11

Date : 11/11/2024

Aim :

Write a program to find whether the given number is an Armstrong number or not.

Pseudocode :

1. Input number
2. Set original_number = number, sum_of_powers = 0
3. While number > 0

    Extract last digit,

    raise it to the power of the number of digits,

    add to sum_of_powers

    Remove last digit
4. If sum_of_powers equals original_number, print "Armstrong number"
5. Else, print "not an Armstrong number"

Source Code :

```
number = int(input("Enter a number to check if it is an Armstrong number: "))
original_number = number
num_digits = len(str(number))
sum_of_powers = 0
while number > 0:
    digit = number % 10
    sum_of_powers += digit ** num_digits
    number //= 10
if sum_of_powers == original_number:
    print(f"{original_number} is an Armstrong number.")
```

```
else:
    print(f"{original_number} is not an Armstrong number.")
```

Output :

Enter a number to check if it is an Armstrong Number: 370

370 is an Armstrong Number

Enter a number to check if it is an Armstrong Number: 750

750 is not an Armstrong Number

Result :

The program is successfully executed and the output is verified.

Experiment No : 12

Date : 11/11/2024

Aim :

Display the given pyramid with the step number accepted from the user.

1
2 4
3 6 9
4 8 12 16

Pseudocode :

1. Input n

2. Loop i from 1 to n

   Loop j from 1 to i

   Print i * j, end with space

3. Print new line after each row.

Source Code :

```
n = int(input("Enter the number of steps: "))
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(i * j, end=" ")
    print()
```

Output :

Enter the Number of steps: 6

1

2 4

3 6 9

4 8 12 16

5 10 15 20 25

6 12 18 24 30 36

Result :

The program is successfully executed and the output is verified.

Experiment No : 13

Date : 11/11/2024

Aim :

Construct the following pattern using nested loop.

```
*
**
***
****
*****
****
***
**
*
```

Pseudocode :

1. Input n
2. Loop i from 1 to n

   Print i asterisks in a row
3. Loop i from n-1 to 1

   Print i asterisks in a row

Source Code :

```
n = int(input("Enter the number of rows: " ))
for i in range(1, n + 1):
    for j in range(i):
        print("*", end=" ")
    print()
for i in range(n-1, 0, -1):
    for j in range(i):
        print("*", end=" ")
    print()
```

Output :

Enter the Number of rows : 6

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

Result :

The program is successfully executed and the output is verified.

LAB CYCLE - 4

Experiment No : 1

Date : 14/11/2024

Aim :

Write a program to print the Fibonacci series using recursion.

Pseudocode :

1. Define a function fibonacci(n)

   If n <= 1, return n

   Else, return fibonacci(n-1) + fibonacci(n-2)

2. Input number as the number of terms

3. If number <= 0, print error message

4. Else

   Loop from i = 0 to number - 1

   Print fibonacci(i)

Source Code :

```
def fibonacci(n):
    if n<=1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)
number=int(input("Enter the number of terms:"))
if number<=0:
    print("Enter a positive number")
else:
    print("Fibonacci Series:")
```

```
for i in range(number):
        print(fibonacci(i))
```

Output :

Enter the number of terms: 7

Fibonacci Series :

0

1

2

3

5

8

Result :

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 14/11/2024

Aim :

Write the to implement a menu-driven calculator. Use separate functions for the different operations.

Pseudocode :

1. Define functions for add, subtract, multiply, and divide

2. While True

   Display menu options

   Input choice

   If choice is 5, exit

   If choice is 1 to 4

   Input num1 and num2

3 .Perform the corresponding operation based on choice

4. Else, print invalid choice message.

Source Code :

```
def add(x,y):
    return x + y
def subtract(x,y):
    return x - y
def multiply(x,y):
    return x * y
def divide(x,y):
    if y==0:
        return "Error! Division by Zero"
    return x/y
```

```
while  True:
      print("\n1.ADDITION")
      print("2.SUBTRACTION")
      print("3.MULTIPLICATION")
      print("4.DIVISION")
      print("5.EXIT\n")
      choice = input("   Select an Operation::")
      if choice=='5':
            print("Exiting")
            break
      if choice in ('1','2','3','4'):
            num1=float(input("Enter the First Number:"))
            num2=float(input("Enter the Second Number:"))

            if choice == '1':
                  print(f"\nResult: {num1} + {num2} = {add(num1,num2)}")
            elif choice == '2':
                  print(f"\nResult: {num1} - {num2} = {subtract(num1,num2)}")
            elif choice == '3':
                  print(f"\nResult: {num1} * {num2} = {multiply(num1,num2)}")
            elif choice == '4':
                  print(f"\nResult: {num1} / {num2} = {divide(num1,num2)}")
      else:
            print("Invalid Choice")
```

Output :

1. ADDITION
2. SUBTRACTION
3. MULTIPLICATION
4. DIVISION
5. EXIT

Select an Operation :: 1

Enter the First Number : 12

Enter the Second Number :13

Result : 12 + 13 = 25

Select an Operation :: 2

Enter the First Number : 25

Enter the Second Number :15

Result : 25 - 15 = 10

Select an Operation :: 3

Enter the First Number : 12

Enter the Second Number :5

Result : 12 * 5 = 60

Select an Operation :: 4

Enter the First Number : 84

Enter the Second Number :12

Result : 84.0 / 12.0 = 7.0

Result :

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 14/11/2024

Aim :

Write a program to print the nth prime number.

Pseudocode :

1. Define a function prime(num) to check if a number is prime

    Return False if num <= 1

    Loop from 2 to sqrt(num), return False if divisible

    Return True

2. Define a function nth_prime(n)

    Initialize count = 0 and num = 2

    While True

    If prime(num), increment count

    If count == n, return num

    Increment num

3. Input n

4. If n <= 0, print error message

5. Else, print the nth prime number using nth_prime(n)

Source Code :

```
def prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
  return True
```

```
def nth_prime(n):
    count = 0
    num = 2
    while True:
        if prime(num):
            count += 1
            if count == n:
                return num
        num += 1

n = int(input("Enter the position :  "))
if n <= 0:
    print("Please enter a positive integer.")
else:
    print(f"The {n}th prime number is: {nth_prime(n)}")
```

Output :

Enter the position : 7

The 7^{th} prime Number is : 17

Result :

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 14/11/2024

Aim :

Write lambda functions to find the area of square, rectangle and triangle.

Pseudocode :

1. Define lambda functions for

   square = side * side

   rectangle = length * width

   triangle = 0.5 * base * height

2. While True

   Display menu for shape selection

   Input choice

   If choice is 4, exit

   Else, calculate area based on choice

   Print invalid choice for other inputs

3. Print result

Method :

Function	Description	Syntax
lambda()	A lambda function can take any number of arguments, but can only have one expression.	lambda arguments : expression

Source Code :

```
square = lambda side: side * side

rectangle = lambda length, width: length * width

triangle = lambda base, height: 0.5 * base * height
```

```
while True:
    print("\n2D SHAPES")
    print("1. Square")
    print("2. Rectangle")
    print("3. Triangle")
    print("4. Exit")

    choice = input("\nSelect any Shape to calculate its Area :: ")
    if choice == '1':
        side = float(input("\nEnter the side length of the square: "))
        print(f" The Area of the Square is: {square(side)}")

    elif choice == '2':
        length = float(input("\nEnter the length of the rectangle: "))
        width = float(input("Enter the width of the rectangle: "))
        print(f" The Area of the Rectangle is: {rectangle(length, width)}")

    elif choice == '3':
        base = float(input("\nEnter the base length of the triangle: "))
        height = float(input("Enter the height of the triangle: "))
        print(f" The Area of the Triangle is: {triangle(base, height)}")

    elif choice == '4':
        print("Exiting")
        break

    else:
        print("Invalid choice!")
```

Output :

2D SHAPES

1. SQUARE
2. RECTANGLE
3. TRIANGLE
4. EXIT

Select any Shapes to calculate its Area :: 1

Enter the side length of the Square : 8

The Area of the Square is : 64

Select any Shapes to calculate its Area :: 2

Enter the Length of the Rectangle : 8

Enter the Breadth of the Rectangle : 3

The Area of Rectangle is : 24.0

Select any Shapes to calculate its Area :: 3

Enter the base length of the Triangle : 7

Enter the height of the Triangle : 4

The Area of Triangle is : 14.0

Result :

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 14/11/2024

Aim :

Write a program to display powers of 2 using anonymous function.

[use map and lambda function]

Pseudocode :

1. Input number as a comma-separated list.

2. Use map and lambda to compute powers of 2 for each element in number.

3. Print the resulting list.

Method :

Function	Description	Syntax
map()	Executes a specified function for each item in an iterable. The item is sent to the function as a parameter.	map(function, iterables)

Source Code :

```
number = list(map(int,input("Enter the numbers : ").split(",")))
power = list(map(lambda x: 2 ** x, number))
print(f" The Powers of 2 for the Entered Numbers: {power}")
```

Output :

Enter the Numbers : 2, 4, 7, 9

The Powrs of 2 for the entered numbers :

[4, 16, 128, 512]

Result :

The program is successfully executed and the output is verified.

Experiment No : 6

Date : 14/11/2024

Aim :

Write a program to display multiples of 3 using anonymous function.

[use filter and lambda function)

Pseudocode :

1. Input number as a comma-separated list.
2. Use filter and lambda to find multiples of 3 in number.
3. Print the resulting list.

Method :

Function	Description	Syntax
filter()	Used to create a new iterable containing only the elements that satisfy a condition.	filter(function, iterable)

Source Code :

```
number = list(map(int,input("Enter the  Numbers:").split(",")))
multiples = list(filter(lambda x: x % 3 == 0, number))
print(f"The multiples of 3 from the list: {multiples}")
```

Output :

Enter the Numbers : 2, 3, 7, 12, 36, 45

The multiples of 3 from the list :

[3, 12, 36, 45]

Result :

The program is successfully executed and the output is verified.

Experiment No : 7

Date : 18/11/2024

Aim :

Write a program to sum the series 1/1! + 4/2! + 27/3! + ….. + nth term.

[Use a function to find the factorial of a number].

Pseudocode :

1. Define a function factorial(num)

   Compute factorial iteratively

2. Define a function series(n)

   Initialize sum = 0

   Loop from i = 1 to n

   Calculate term = (i ** i) / factorial(i)

   Add term to sum

   Return sum

3. Input n

4. Print result of series(n)

Source Code :

```
def factorial(num):
    if num == 0 or num == 1:
        return 1
    else:
        fact = 1
        for i in range(2, num + 1):
            fact *= i
        return fact


def series(n):
    sum = 0
```

```
 for i in range(1, n + 1):
        term = (i ** i) / factorial(i)
        sum += term
    return sum

n = int(input("Enter the number of terms: "))
print(f"The sum of the series up to {n} terms is: {series(n)}")
```

Output :

Enter the number of terms : 5

The Sum of the series up to 5 terms is : 44.20833

Result :

The program is successfully executed and the output is verified.

Experiment No : 8

Date : 18/11/2024

Aim :

Write a function called compare which takes two strings S1 and S2 and an integer n as arguments. The function should return True if the first n characters of both the strings are the same else the function should return False.

Pseudocode :

1. Define a function compare(S1, S2, n)

   Return True if the first n characters of S1 and S2 match

   Else, return False

2. Input S1, S2, and n

3. Use compare(S1, S2, n) and print appropriate message.

Source Code :

```
def compare(S1, S2, n):
    if S1[:n] == S2[:n]:
        return True
    else:
        return False
S1 = input("Enter the String1:")
S2 = input("Enter the String2:")
n = int(input("Enter the number of characters to Compare: "))
result = compare(S1, S2, n)
if result:
    print(f"The first {n} characters of both the strings are Same.")
else:
    print(f"The first {n} characters of both the strings are not Same.")
```

Output :

Enter the String1: python3.7

Enter the String2: python3.8

Enter the Number of characters to Compare : 8

The first 8 character of both the strings are not Same.

Enter the Number of characters to Compare : 4

The first 4 character of both the strings are Same.

Result :

The program is successfully executed and the output is verified.

Experiment No : 9

Date : 18/11/2024

Aim :

Write a program to add variable length integer arguments passed to the function.

[Also demo the use of docstrings]

Pseudocode :

1. Define a function add(*args):

   Return sum of args

2. Print the sum of a set of numbers using add

3. Print the docstring of the add function.

Method :

Function	Description	Syntax
docstring	Provide a convenient way of associating documentation with python modules, methods and classes.	“”” triple double quotes “”””

Source Code :

```
def add(*args):
    """
    This is a program to add variable length integer arguments passed to the function.
    args- It is used pass any number of arguments to a function
    """
  return sum(args)
print("The Sum of First 10 numbers:", add(1,2,3,4,5,6,7,8,9,10))
print("\nThe Docstring is:")
print(add.__doc__)
```

Output :

The Sum of First 10 numers : 55

The Docstring is :

This is a program to add variable length integer arguments passed to the Function.

args – It is used to pass any number of arguments to a function

Result :

The program is successfully executed and the output is verified.

Experiment No : 10

Date : 18/11/2024

Aim :

Write a program using functions to implement these formulae for permutations and combinations.

The Number of permutations of n objects taken r at a time: p(n, r) = n!/(n − r)!.

The Number of combinations of n objects taken r at a time is: c(n, r) = n!/(r! ∗ (n − r)!)

Pseudocode :

1. Define a function factorial(num) to compute factorial

2. Define permutations(n, r) as factorial(n) / factorial(n - r)

3. Define combinations(n, r) as factorial(n) / (factorial(r) * factorial(n - r))

4. Input n and r

5. If r > n, print error message

6. Else, print permutations(n, r) and combinations(n, r)

Source Code :

```
def factorial(num):
    if num == 0 or num == 1:
        return 1
    else:
        fact = 1
        for i in range(2, num + 1):
            fact *= i
        return fact
def permutations(n, r):
    return factorial(n) // factorial(n - r)
def combinations(n, r):
    return factorial(n) // (factorial(r) * factorial(n - r))
```

```
n = int(input("Enter the value of n :"))
r = int(input("Enter the value of r :"))

if r > n:
    print("Invalid input: r should be less than or equal to n.")
else:
    print(f"The number of permutations P({n}, {r}) is: {permutations(n, r)}")
    print(f"The number of combinations C({n}, {r}) is: {combinations(n, r)}")
```

Output :

Enter the value of n : 6

Enter the value of r : 2

The number of Permutations P(6, 2) is : 30

The number of Combinations C(6,2) is : 15

Result :

The program is successfully executed and the output is verified.

LAB CYCLE – 5

Experiment No : 1

Date : 09/12/2024

Aim :

Write a program to determine whether a given year is a leap year.

[Use Calendar Module]

Pseudocode :

1. Import the `calendar` module.

2. Input year.

3. If calendar.isleap(year)

   Print `year` is a leap year.

4. Else

   Print `year` is not a leap year.

Method :

Function	Description	Syntax
Isleap()	Used to get value True, if the year is a leap year, otherwise gives False.	Calendar.isleap(year)

Source Code :

```
import calendar

year=int(input(" Enter a Year:"))

if calendar.isleap(year):

  print(f"{year} is a Leap Year")
```

```
else:
  print(f"{year} is not a Leap Year")
```

Output :

Enter a Year : 2024

2024 is a Leap year.

Enter a Year : 2025

2025 is not a Leap year.

Result :

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 09/12/2024

Aim :

Write a python script to display

a) Current date and time

b) Current Year

c) Month of the year

d) Week number of the year

e) Weekday of the week

f) Day of year

g) Day of the month

h) Day of week [Use time and datetime Module]

Pseudocode :

1. Import `time` and `datetime` modules.

2. Get the current date and time as current_datetime

3. Print

   Current date and time.

   Current year.

   Month of the year.

   Week number of the year.

   Weekday of the week.

   Day of the year.

   Day of the month.

   Day of the week.

Method :

Function	Description	Syntax
now()	Return the current local date and time.	datetime.now()
strftime()	Used to convert date and time object to their string representation.	datetime_object.strftime(format)

Source Code :

```
import time

from datetime import datetime

current_datetime = datetime.now()

print("a) Current date and time:", current_datetime.strftime("%d-%m-%Y %H:%M:%S"))

print("b) Current Year:", current_datetime.year)

print("c) Month of the year:", current_datetime.strftime("%B"))

print("d) Week number of the year:", current_datetime.strftime("%U"))

print("e) Weekday of the week:", current_datetime.strftime("%A"))

print("f) Day of the year:", current_datetime.strftime("%j"))

print("g) Day of the month:", current_datetime.strftime("%d"))

print("h) Day of the week:", current_datetime.strftime("%w"))
```

Output :

a) Current date and time : 09 - 12 – 2024 10 : 32 : 54
b) Current Year : 2024
c) Month of the Year : December
d) Week number of the year : 49
e) Weekday of the week : Monday
f) Day of the Year : 344
g) Day of the Month : 09
h) Day of the Week : 6

Result :

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 09/12/2024

Aim :

Write a python program to print yesterday, today and tomorrow.

Pseudocode :

1. Import `datetime` and `timedelta`.

2. Get today's date as `today`.

3. Calculate `yesterday` as `today - 1 day`.

4. Calculate `tomorrow` as `today + 1 day`.

5. Print `yesterday`, `today`, and `tomorrow`.

Method :

Function	Description	Syntax
timedelta()	Used for calculating differences in dates and also used for date manipulation in python.	timedelta(days=0,week=0, hour=0..)

Source Code :

```
from datetime import datetime, timedelta
today = datetime.now()
yesterday = today - timedelta(days=1)
tomorrow = today + timedelta(days=1)
print("Yesterday:", yesterday.strftime("%d-%m-%Y"))
print("Today: ", today.strftime("%d-%m-%Y"))
print("Tomorrow: ", tomorrow.strftime("%d-%m-%Y"))
```

Output :

Yesterday : 08 - 12 – 2024

Today : 09 - 12 – 2024

Tomorrow : 10 - 12 – 2024

Result :

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 09/12/2024

Aim :

Write a function in file palindrome.py to check whether a string is Palindrome or not. Import the module to find the longest palindromic substring in a given string by checking every possible substring and verifying if it is a palindrome.

Pseudocode :

Palindrome.py

1. Define is_palindrome(s)

   Convert `s` to lowercase.

   Return `True` if `s` equals its reverse, else, `False`.

2. Define palindrome_length(s)

   Initialize `longest` as an empty string.

   Loop over all substrings of `s`:

    If a substring is palindromic and longer than `longest`, update `longest`.

   Return longest.

Mainpalindrome.py

1. Import functions from `Palindrome.py`.
2. Input `s`.
3. If `is_palindrome(s)`

   Print `s` is palindromic.

4. Else

   Print `s` is not palindromic.

   Print the longest palindromic substring using `palindrome_length`.

Source Code :

Palindrome.py

```
def is_palindrome(s):
    s = s.lower()
    return s==s[::-1]
def palindrome_length(s):
    n=len(s)
    if n==0:
        return s
    longest=""
    for i in range(n):
        for j in range(i+1,n+1):
            substr=s[i:j]
            if is_palindrome(substr) and len(substr)>len(longest):
                longest=substr
    return longest
```

mainpalindrome.py

```
import palindrome as p
s= input(" Enter the String: ")
if p.is_palindrome(s):
    print(f"{s} is Palindromic")
else:
    print(f"{s} is not palindromic")
    print(f"Longest palindromic substring in {s} is \n{p.palindrome_length(s)}")
```

Output :

Enter the String : racecarbanana

racecarbanana is not palindromic

Longest palindromic substring in racecarbanana is racecar.

Result :

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 09/12/2024

Aim :

Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that find the area and perimeter of figures by different importing statements. (Include selective import of modules and import * statements)

Pseudocode :

Rectangle.py

1. Define area(l, w) as l * w

2. Define perimeter(l, w) as 2 * (l + w)

Circle.py

1. Define area(r) as $\pi * r^2$

2. Define `circumference(r) as $2 * \pi * r$

Cuboid.py

1. Define area(l, w, h) as 2 * (l * w + w * h + l * h)

2. Define volume(l, w, h) as l * w * h

Sphere.py

1. Define area(r) as $4 * \pi * r^2$

2. Define volume(r) as $(4/3) * \pi * r^3$.

MainGraphics.py

1. Import all modules from `Graphics` and `Graphics3D`.

2. Display a menu for selecting shapes

   For Rectangle, input length and width; calculate and print area and perimeter.

   For Circle, input radius; calculate and print area.

For Cuboid, input dimensions; calculate and print area and volume.

For Sphere, input radius; calculate and print area and volume.

3. Exit on selection of choice 5.

Source Code :

mainGraphics.py

```
from Graphics import rectangle as r
from Graphics import circle as c
from Graphics.graphics3D  cuboid import *
from Graphics.graphics3D import sphere as s
while(True):
choice=int(input("\n\t[][][]2D & 3D
  SHAPES[][][]\n\n\n1.Rectangle\n2.Circle\n\n\n3.cuboid\n4.Sphere\n5.Exit\n\tSelect
  any Shape ::"))
 if(choice==1):
    l=int(input("\nEnter the Length of the Rectangle: "))
    w=int(input("\nEnter the breadth of the Rectangle: "))
    print("\nThe Area of Rectangle =  ",r.area(l,w))
    print("\nThe Perimeter of Rectangle = : ",r.perimeter(l,w))
elif(choice==2):
    r=int(input("\nEnter the Radius of Circle: "))
    print("\nThe Area of Circle = ",c.area(r))
elif(choice==3):
   l=int(input("\nEnter the Length of Cuboid: "))
   w=int(input("\nEnter the breadth of Cuboid: "))
   h=int(input("\nEnter the height of cuboid: "))
   print("\nThe Area of Cuboid = ",area(l,w,h))
   print("\nThe Volume of Cuboid: ",volume(l,w,h))
elif(choice==4):
    r=int(input("\nEnter the Radius of Sphere: "))
    print("\nThe Area of sphere = ",s.area(r))
    print("\nThe Volume of sphere = ",s.volume(r))
```

```
 elif(choice==5):
     break;
 elif(choice not in [1,2,3,4,5]):
  print("Invalid choice")
```

rectangle.py

```
def area(l,w):
   return l*w
def perimeter(l,w):
   return 2*(l+w)
```

circle.py

```
import math as m
def area(r):
   return m.pi*r*r
def circumference(r):
   return 2*m.pi*r
```

cuboid.py

```
def area(l,w,h):
  return 2*(l*w + w*h + l*h)
def volume(l,w,h):
  return 2*(l*w*h)
```

sphere.py

```
import math as m
def area(r):
   return 4*m.pi*r*r
def volume(r):
  return (4*m.pi*(r**3))/3
```

Output :

2D & 3D SHAPES

1. Rectangle
2. Circle
3. Cuboid
4. Sphere
5. Exit

Select any Shape :: 1

Enter the Length of the Rectangle : 8

Enter the Breadth of the Rectangle : 4

The Area of Rectangle = 32

The Perimeter of Rectangle = 24

Select any Shape :: 2

Enter the Radius of Circle : 7

The Area of Circle = 153.93804

Select any Shape :: 3

Enter the Length of Cuboid : 9

Enter the Breadth of Cuboid : 5

Enter the Height of Cuboid : 6

The Area of Cuboid = 258

The Volume of Cuboid = 540

Select any Shape :: 4

Enter the Radius of Sphere : 6

The Area of Sphere = 452.3898

The Volume of Sphere = 904.778

Result :

The program is successfully executed and the output is verified.

LAB CYCLE – 6

Experiment No : 1

Date : 19/12/2024

Aim :

Define a class to represent a bank account. Include the following details like name of the depositor, account number, type of account, balance amount in the account. Write methods to assign initial values, to deposit an amount, withdraw an amount after checking the balance, to display details such as name, account number, account type and balance.

Pseudocode :

1. Define class BankAccount with

   Attributes: name, account_number, account_type, balance

   Methods:

   __init__ to initialize attributes.

   deposit(amount) : Add `amount` to `balance` if positive.

   withdraw(amount) : Subtract `amount` from `balance` if it has sufficient funds.

   display_details() : Display all account details.

2. Input account details and create an instance of `BankAccount`.

3. Display a menu:

   If `1`, prompt for deposit amount and call `deposit()`.

   If `2`, prompt for withdrawal amount and call `withdraw()`.

   If `3`, call `display_details()`.

   If `4`, exit.

Source Code :

```
class BankAccount:
    def __init__(self, name, account_number, account_type, balance=0):
        self.name = name
        self.account_number = account_number
```

```
        self.account_type = account_type
        self.balance = balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited: {amount}\nAvailable Balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")
    def withdraw(self, amount):
        if amount > 0:
            if self.balance >= amount:
                self.balance -= amount
                print(f"Debited: {amount}\nAvailable Balance: {self.balance}")
            else:
                print("Insufficient balance for withdrawal.")
        else:
            print("Withdrawal amount must be positive.")
    def display_details(self):
        print("\n*** Bank Account Details ***")
        print(f"Name: {self.name}")
        print(f"Account Number: {self.account_number}")
        print(f"Account Type: {self.account_type}")
        print(f"Balance Amount: {self.balance}")
name = input("Enter your Name: ")
account_number = input("Enter your Account number: ")
account_type = input("Enter your Account type [Savings / Current]: ")
balance = float(input("Enter your Initial Balance : ") or 0)
account = BankAccount(name=name, account_number=account_number,
account_type=account_type, balance=balance)
```

```
while True:
    print("\n*** Menu ***")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Display Account Details")
    print("4. Exit")
    try:
        choice = int(input("Choose an option :: "))
    except ValueError:
        print("Invalid input. Please enter a number between 1 and 4.")
        continue
    if choice == 1:
        try:
            deposit_amount = float(input("Enter the amount to deposit: "))
            account.deposit(deposit_amount)
        except ValueError:
            print("Invalid amount. Please enter a valid number.")
    elif choice == 2:
        try:
            withdraw_amount = float(input("Enter the amount to withdraw: "))
            account.withdraw(withdraw_amount)
        except ValueError:
            print("Invalid amount. Please enter a valid number.")
    elif choice == 3:
        account.display_details()
    elif choice == 4:
        print("Thank you for using the Bank Account System. Goodbye!")
        break
    else:
        print("Invalid choice. Please select a number between 1 and 4.")
```

Output :

Enter your Name : Arun

Enter your Account Number : 110053

Enter your Account Type [Savings / Current] : Savings

Enter your Initial Balance : 280

*** MENU ***

1. Deposit
2. Withdraw
3. Display Account Details
4. Exit

Choose an Option :: 1

Enter the Amount to Deposit : 53500

Deposited : 53500.0

Available Balance : 53780.0

Choose an Option :: 2

Enter the Amount to Withdraw : 50

Debited : 50.0

Available Balance : 53730.0

Choose an Option :: 3

*** Bank Account Details ***

Name : Arun

Account Number : 110053

Account Type : Savings

Balance Amount : 53730.0

Choose an Option :: 4

Thank You !

Result :

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 19/12/2024

Aim :

Create a class Publisher with attributes publisher id and publisher name. Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no_of_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.

Pseudocode :

1. Define class Publisher

   Attributes: publisher_id, publisher_name.

   Method: display_publisher_details().

2. Define class Book that inherits Publisher.

   Attributes: title, author.

   Method: display_book_details().

3. Define class Python that inherits Book

   Attributes: price, no_of_pages.

   Method: display_python_details().

4. Input details for a Python book.

5. Create an instance of `Python`.

6. Call display_python_details().

Source Code :

```
class Publisher:
    def __init__(self, publisher_id, publisher_name):
        self.publisher_id = publisher_id
        self.publisher_name = publisher_name
    def display_publisher_details(self):
```

```
        print(f"Publisher ID: {self.publisher_id}")
        print(f"Publisher Name: {self.publisher_name}")
class Book(Publisher):
    def __init__(self, publisher_id, publisher_name, title, author):
        super().__init__(publisher_id, publisher_name)
        self.title = title
        self.author = author
    def display_book_details(self):
        print(f"\nBook Title: {self.title}")
        print(f"Author: {self.author}")
        self.display_publisher_details()
class Python(Book):
    def __init__(self, publisher_id, publisher_name, title, author, price, no_of_pages):
        super().__init__(publisher_id, publisher_name, title, author)
        self.price = price
        self.no_of_pages = no_of_pages
    def display_python_details(self):
        print(f"\nPrice: ${self.price}")
        print(f"Number of Pages: {self.no_of_pages}")
        self.display_book_details()
publisher_id = input("Enter Publisher ID: ")
publisher_name = input("Enter Publisher Name: ")
title = input("Enter Book Title: ")
author = input("Enter Author Name: ")
price = float(input("Enter Price of the Book: $"))
no_of_pages = int(input("Enter the Number of Pages: "))
python_book = Python(
    publisher_id=publisher_id,
    publisher_name=publisher_name,
    title=title,
    author=author,
```

```
  price=price,
   no_of_pages=no_of_pages
)
python_book.display_python_details()
```

Output :

```
Enter Publisher ID : 1053
Enter Publisher Name : Penguin
Enter Book Title : The Blue Umbrella
Enter Author Name : Ruskin Bond
Enter Price of the Book : $10
Enter the Number of Pages : 87
About the Book :
Book Title : The Blue Umbrella
Author : Ruskin Bond
Publisher ID : 1053
Publisher Name : Penguin
Price : $10
Number of Pages : 87
```

Result :

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 19/12/2024

Aim :

Write a program that has an abstract class Polygon. Derive two classes Rectangle and Triangle from Polygon and write methods to get the details of their dimensions and hence calculate the area.

Pseudocode :

1. Import ABC and abstractmethod from abc module.
2. Define abstract class Polygon with method calculate_area.
3. Define class Rectangle that inherits Polygon

   Attributes: length, width.

   Method calculate_area(): Return length * width.
4. Define class Triangle that inherits Polygon

   Attributes: base , height.

   Method calculate_area() : Return 0.5 * base * height.
5. Input dimensions for a rectangle and triangle.
6. Create instances of Rectangle and Triangle.
7. Call calculate_area() for both shapes and print the results.

Source Code :

```
from abc import ABC, abstractmethod
class Polygon(ABC):
    @abstractmethod
    def calculate_area(self):
        pass
class Rectangle(Polygon):
    def __init__(self, length, width):
        self.length = length
```

```
    self.width = width
  def calculate_area(self):
      area = self.length * self.width
      return area
class Triangle(Polygon):
  def __init__(self, base, height):
      self.base = base
      self.height = height
  def calculate_area(self):
      area = 0.5 * self.base * self.height
      return area
print("Enter Dimensions for Rectangle:")
length = float(input("Enter length: "))
width = float(input("Enter width: "))
rectangle = Rectangle(length, width)
rectangle_area = rectangle.calculate_area()
print(f"Area of Rectangle :: {rectangle_area}")
print("\nEnter Dimensions for Triangle:")
base = float(input("Enter base: "))
height = float(input("Enter height: "))
triangle = Triangle(base, height)
triangle_area = triangle.calculate_area()
print(f"Area of Triangle :: {triangle_area}")
```

Output :

```
Enter Dimensions for Rectangle :
Enter Lengh : 8
Enter Width : 5
Area of Rectangle :: 40.0
```

Enter Dimensions for Triangle:

Enter Base :7

Enter Height : 8

Area of Triangle :: 28.0

Result :

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 19/12/2024

Aim :

Create a Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.

Pseudocode :

1. Define class Rectangle:

   Attributes: length, breadth.

   Methods:

   area(): Return length * breadth.

   perimeter(): Return 2 * (length + breadth).

   compare_area(other) : Compare area with another rectangle.

2. Input dimensions for two rectangles.

3. Create two Rectangle instances.

4. Call area(), perimeter(), and compare_area() methods for comparison.

Source Code :

```
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth
    def area(self):
        return self.length * self.breadth
    def perimeter(self):
        return 2 * (self.length + self.breadth)
    def compare_area(self, other):
        if self.area() > other.area():
            return "First Rectangle has a larger area."
```

```
        elif self.area() < other.area():
            return "Second Rectangle has a larger area."
        else:
            return "Both Rectangles have the same area."
print("Enter Dimensions for the first Rectangle:")
length1 = float(input("Enter Length: "))
breadth1 = float(input("Enter Breadth: "))
rect1 = Rectangle(length1, breadth1)
print("\nEnter Dimensions for the second Rectangle:")
length2 = float(input("Enter Length: "))
breadth2 = float(input("Enter Breadth: "))
rect2 = Rectangle(length2, breadth2)
print(f"\nFirst Rectangle:\n Area = {rect1.area()},  Perimeter = {rect1.perimeter()}")
print(f"Second Rectangle:\n Area = {rect2.area()},  Perimeter = {rect2.perimeter()}")
print("\nComparison of Areas:")
print(rect1.compare_area(rect2))
```

Output :

```
Enter Dimensions for the First Rectangle :
Enter Length : 12
Enter Breadth : 7
Enter Dimensions for the Second Rectangle :
Enter Length : 17
Enter Breadth : 9
First Rectangle :
  Area = 84.0,    Perimeter = 38.0
Second Rectangle :
  Area = 153.0,  Perimeter = 52.0
```

Comparison of Areas :

Second Rectangle has a Larger Area.

Result :

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 19/12/2024

Aim :

Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 times.

Pseudocode :

1. Define class Time with:

   Private attributes: _hour, _minute, _second.

   Overload __add__(other) to add two Time objects

   Convert both times to total seconds, add them, and convert back.

   display_time() : Print time in HH:MM:SS format.

2. Input details for two Time objects.

3. Create Time instances.

4. Add the two times using the overloaded `+` operator.

5. Call display_time() for both individual and summed times.

Source Code :

```
class Time:
    def __init__(self, hour, minute, second):
        self._hour = hour
        self._minute = minute
        self._second = second
    def __add__(self, other):
        total_seconds = (self._hour * 3600 + self._minute * 60 + self._second) + \
                        (other._hour * 3600 + other._minute * 60 + other._second)
        hours = total_seconds // 3600
        minutes = (total_seconds % 3600) // 60
        seconds = total_seconds % 60
```

```
            return Time(hours, minutes, seconds)
    def display_time(self):
        print(f"Time: {self._hour:02}:{self._minute:02}:{self._second:02}")
print("Enter details for the Time 1 [Hour, Minute, Second]:")
hour1 = int(input("Enter hour: "))
minute1 = int(input("Enter minute: "))
second1 = int(input("Enter second: "))
time1 = Time(hour1, minute1, second1)
print("\nEnter details for the Time 2 [Hour, Minute, Second]:")
hour2 = int(input("Enter hour: "))
minute2 = int(input("Enter minute: "))
second2 = int(input("Enter second: "))
time2 = Time(hour2, minute2, second2)
sum_time = time1 + time2
print("\nTime 1")
time1.display_time()
print("Time 2")
time2.display_time()
print("Sum of Time 1 and Time 2:")
sum_time.display_time()
```

Output :

```
Enter Details for the Time 1 [Hour, Minute, Second]:
Enter Hour : 5
Enter Minute : 45
Enter Second : 53
Enter Details for the Time 2 [Hour, Minute, Second]:
Enter Hour : 3
Enter Minute : 12
Enter Second : 10
```

Time 1

Time : 05 : 45 : 53

Time 2

Time : 03 : 12 : 10

Sum of Time 1 and Time 2 :

Time : 08 : 58 : 03

Result :

The program is successfully executed and the output is verified.

LAB CYCLE - 7

Experiment No : 1

Date : 26/12/2024

Aim :

Write a Python program to read a file line by line and store it into a list.

Pseudocode :

1. Set file_name to the file path Samplefile.txt

2. Open file_name in read mode.

3. Create a list lines_list where each line from the file is stripped of the whitespaces.

4. Print "Lines stored in the list".

5. Print `lines_list`.

Method :

Function	Description	Syntax
open()	Open a file and return it as a file object.	open(file, mode)

Source Code :

Samplefile.txt

Hello World !

This is Python Programming.

File Handling in Python.

Regular Expressions in python.

Pg1.py

```
file_name = "Samplefile.txt"
with open(file_name, 'r') as file:
lines_list = [line.strip() for line in file]
```

```
print("Lines stored in the list:")
print(lines_list)
```

Output :

Lines Stored in the List :

[‘Hello World !’, ‘This is Python Programming’, ‘File Handling in Python’, ‘Regular Expressions in Python.’]

Result :

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 26/12/2024

Aim :

Python program to copy odd lines of one file to other.

Pseudocode :

1. Set source_file to the file path Samplefile.txt

2. Set target_file to the file path odd_file.txt.

3. Open source_file in read mode and target_file in write mode.

4. For each line in source_file

   Check the line number:

   If the line number is odd, write the line to target_file.

5. Print a message indicating odd lines were copied.

Method :

Function	Description	Syntax
write()	Write a string to a file.	file.write(data)

Source Code :

Samplefile.txt

Hello World !

This is Python Programming.

File Handling in Python.

Regular Expressions in python.

Pg2.py

```
source_file = "Samplefile.txt"
target_file = "odd_file.txt"
with open(source_file, 'r') as source, open(target_file, 'w') as target:
for line_number, line in enumerate(source, start=1):
if line_number % 2 != 0:
target.write(line)
print(f"Odd lines have been copied from {source_file} to {target_file}.")
reader = csv.reader(csv_file)
for row in reader:
print(row)
```

Output :

Odd lines have been copied from samplefile.txt to odd_file.txt.

Odd_file.txt

Hello World !

File Handling in Python.

Result :

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 26/12/2024

Aim :

Write a Python program to read each row from a given csv file and print a list of strings.

Pseudocode :

1. Import csv module.
2. Set file_name to the file path details.csv.
3. Open file_name in read mode.
4. Read the CSV file using csv.reader.
5. For each row in the file

   Print row as a list of strings.

Method :

Function	Description	Syntax
csv.reader()	Reads the rows of a CSV file.	csv.reader(file)

Source Code :

Details.csv

```
Name,Age,District
Arun,20,Thrissur
Aravind,21,Kollam
Shamil,20,Malappuram
```

Pg3.py

```
import csv
file_name = "details.csv"
```

```
with open(file_name, 'r') as csv_file:
reader = csv.reader(csv_file)
for row in reader:
print(row)
```

Output :

[‘Name’, ‘Age’, ‘District’]

[‘Arun’, ‘20’, ‘Thrissur’]

[‘Aravind’, ‘21’, ‘Kollam’]

[‘Shamil’, ‘20’, ‘Malappuram’]

Result :

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 26/12/2024

Aim :

Write a Python program to read specific columns of a given CSV file and print the content of the columns.

Pseudocode :

1. Import csv` module.
2. Set file_name to the file path details.csv.
3. Set columns_to_read to a list of indices.
4. Open file_name in read mode.
5. Read the CSV file using csv.reader.
6. For each row in the file

   Extract the columns specified in `columns_to_read`.

   Print the extracted columns.

Source Code :

Details.csv

```
Name,Age,District
Arun,20,Thrissur
Aravind,21,Kollam
Shamil,20,Malappuram
```

Pg4.py

```
import csv
file_name = "details.csv"
columns_to_read = [0, 2]
with open(file_name, 'r') as csv_file:
reader = csv.reader(csv_file)
```

```
for row in reader:
selected_columns = [row[i] for i in columns_to_read]
print(selected_columns)
```

Output :

['Name', 'District']

['Arun', 'Thrissur']

['Aravind', 'Kollam']

['Shamil', 'Malappuram']

Result :

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 26/12/2024

Aim :

Write a Python program to write a Python dictionary to a csv file. After writing the CSV file, read the CSV file and display the content.

Pseudocode :

1. Import csv module.
2. Create a list data of dictionaries containing sample data.
3. Set file_name to the output file path output.csv.
4. Open file_name in write mode:
   Use csv.DictWriter to write the dictionary to the CSV file.
   Write the header row and then the data rows.
5. Print a message indicating the dictionary has been written.
6. Open file_name in read mode:
   Use csv.reader to read the CSV file.
   For each row, print the row contents.

Source Code :

```
import csv
data = [
{"Name": "Ajay", "Age": 21, "District": "Kannur"},
{"Name": "Sujith", "Age": 21, "District": "Palakkad"},
{"Name": "George", "Age": 22, "District": "Alappuzha"}
]
file_name = "output.csv"
with open(file_name, 'w', newline='') as csv_file:
writer = csv.DictWriter(csv_file, fieldnames=data[0].keys())
writer.writeheader()
writer.writerows(data)
```

```
print(f"Dictionary written to {file_name}.")
print("\nReading and displaying the CSV file content:")
with open(file_name, 'r') as csv_file:
reader = csv.reader(csv_file)
for row in reader:
print(row)
```

Output :

Dictionay has been written to output.csv

Reading and Displaying the CSV file content :

[‘Name’, ‘Age’, ‘District’]

[‘Ajay’, ‘21’, ‘Kannur’]

[‘Sujith’, ‘21’, ‘Palakkad’]

[‘George’, ‘22’, ‘Alappuzha’]

Result :

The program is successfully executed and the output is verified.