**Experiment No : 1**

**Date : 25/09/2023**

**Aim:**

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

displays a message “Greetings!!! First name Last name”.

**Pseudocode:**

Read first name.

Read last name.

Print Greetings!!! First name Last name

**Method :**

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

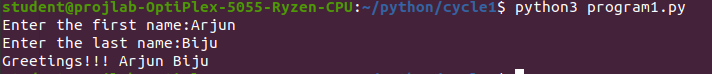
**Source Code:**

name1 = input("Enter first name: ")

name2 = input("Enter last name: ")

print("Greetings!!!", name1, name2)

**Output:**

****

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

**Experiment No : 2**

**Date : 25/09/2023**

**Aim:**

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

**Pseudocode :**

Initialize n1 as any integer value.

Initialize n2 as any float value.

Initialize n3 as any complex number.

Print n1 and its type.

Print n2 and its type.

Print n3 and its type.

**Method :**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| type() | Returns the type of the specified object | type(object) |

**Source Code:**

a = 10

b = 5.5

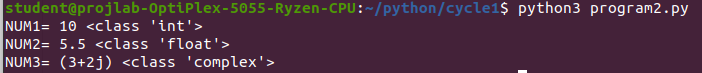
c = 5+10j

print("Num1 =",a, type(a))

print("Num2 =",b, type(b))

print("Num3 =",c, type(c))

**Output:**

****

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

**Experiment No : 3**

**Date : 25/09/2023**

**Aim:**

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

**Pseudocode :**

Read the radius r of the circle.

Area=π\*r\*r

Print Area

**Source Code :**

r = input("enter radius of circle: ")

area = 3.14\*float(r)\*float(r)

print("Area of Circle :",area)

**Output:**

****

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

**Experiment No : 4**

**Date : 25/09/2023**

**Aim:**

Write a program to calculate the volum of a sphere by reading inputs from the user.

**Pseudocode :**

Read the radius r of the sphere.

Volume =4\*π\*r\*r

Print Volume

**Source Code :**

r =float( input("Enter the Radius: ") )

Volume = 4\*(3.14\*(r\*r))

print("Volume of sphere :",volume)

**Output:**

****

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

**Experiment No : 5**

**Date : 25/09/2023**

**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 :**

Input basic pay bp.

HRA=10% of bp

TA=5% of bp

Salary=bp + HRA + TA

Print Salary

**Source Code :**

bp = input("Enter the basic pay: ")

hra = 0.1\*float(bp)

ta = 0.05\*float(bp)

salary = float(bp)+hra+ta

print("\nSalary of the Employee =",salary)

**Output:**

****

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

**Experiment No : 6**

**Date : 25/09/2023**

**Aim:**

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

**Pseudocode :**

Input numbers n1, n2.

Print n1+n2

Print n1-n2

Print n1\*n2

Print n1/n2

**Source Code :**

a = int(input("Enter number1: "))

b = int(input("Enter number2: "))

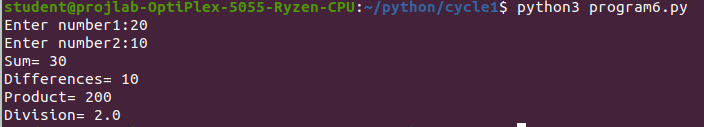
print("Sum = ",a+b)

print(“Differences=”, a-b)

print(“Product=”,a\*b)

print(“Divison=”,a/b)

**Output:**

****

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

**Experiment No : 7**

**Date : 07/10/2023**

**Aim:**

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

**Pseudocode :**

Input string s.

Input no of times of repetition r.

Print s\*r.

**Source Code :**

s = input("Enter a string : ")

r = int(input("Enter the no.of copies needed: "))

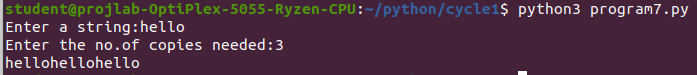
if(n<0):

Print(“please enter a non negative integer”)

else:

print(s\*r)

**Output:**

****

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

**Experiment No : 8**

**Date : 07/10/2023**

**Aim:**

Write a Python program to find biggest of three numbers entered.

**Pseudocode :**

Input three numbers n1,n2,n3.

Check n1<n2 and n3< n2

Print n2 is greater.

Check n1<n3

Print n3 is greater.

Else

Print n1 is greater.

**Source Code :**

num1=int(input("enter first number:"))

num2=int(input("enter second number:"))

num3=int(input("enter third number:"))

if num1>=num2 and num1>=num3:

greatest=num1

elif num2>=num1 and num2>=num3:

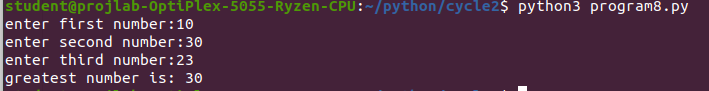
greatest=num2

else:

greatest=num3

print("greatest number is:",greatest)

**Output:**

****

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

**Experiment No : 9**

**Date : 07/10/2023**

**Aim:**

Write a Python program to accept an integer n and compute n+nn+nnn

**Pseudocode :**

Input n.

sum=int(n)+int(n\*2)+int(n\*3)

Print sum

**Source Code :**

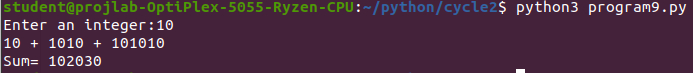
n = input("Enter an integer : ")

sum=int(n)+int(n\*2)+int(n\*3)

Print(n,”+”,n\*2,”+”,n\*3)

print("Sum = ",sum)

**Output:**

****

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

**Experiment No : 10**

**Date : 07/10/2023**

**Aim:**

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

**Pseudocode :**

Input string str.

Length of str, n=len(str)

Print rearranged string sliced using index.

str[n-1]+str[1:n-1]+str[0]

**Method:**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| Slicing | Returns a range of characters  using slice syntax | Str[a:b]  {a->start index  b->end index} |
| len() | Returns length of a string | len(x) |

**Source Code :**

input\_string=input("enter a string:")

if(len(input\_string)>=2):

new\_string=input\_string[-1]+input\_string[1:-1]+input\_string[0]

print("Modified string:",new\_string)

else:

print("please enter astring with atleast two characters")

**Output:**

****

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

**Experiment No : 11**

**Date : 07/10/2023**

**Aim:**

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

**Pseudocode :**

nitialize list x=[]

Input number of elements in the list, n.

For i=0 to n-1:

Input integer a

If a<=100:

Append a to x

Else:

Append “over” to x

End If

Next i

Print x

**Method:**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| for | A for loop is used for  iterating over a sequence | for i in sequence:  statement(s) |
| range | Returns a sequence of  numbers | range(n)  {default:0} |
| append | Used to add an item to the  end of the list | list\_name.append(“str”) |

**Source Code :**

x=[]

n=int(input("Enter no.of elements in list : "))

for i in range(0,n):

a = int(input("Enter element : "))

if a <= 100:

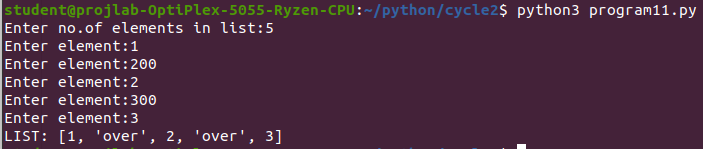
x.append(a)

else:

x.append("over")

print("LIST : ", x)

**Output:**

****

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

**Experiment No : 12**

**Date : 07/10/2023**

**Aim:**

Write a Python program to Store a list of first names. Count the occurrences of ‘a’ within the list.

**Pseudocode :**

Initialize list x=[], temp=0

Input number of names in list, n.

For i=0 to n:

Input string(name) a

Spilt a and store its Ist name to a itself.

Append a to x.

Count “a” in a and add it to temp.

Next i

Print x

Print temp

**Method:**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| count | Return the no of times a  specified value appears in the  string. | string.count(value) |

**Source Code :**

x=[]

temp=0

n=int(input("ENTER THE NO.OF NAMES:"))

for i in range(0, n):

a = input("ENTER THE NAME:")

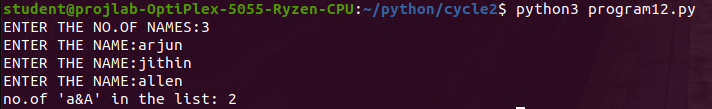
x.append(a)

temp += x[i].count("a")

print("The list is: ", x)

print("no.of 'a&A' in the list: ", temp))

**Output:**

****

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

**Experiment No : 13**

**Date : 12/10/2023**

**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 :**

Initialize list L1[], L2[]

Input number of elements in list1, n1

For i=0 to n1:

Input number,a

Append a to L1

Next i

Input number of elements in list2, n2

For i=0 to n1:

Input number,a

Append a to L2

Next i

Print L1, L2

If length of L1=Length of L2:

Print L1=L2

Else:

Print L1!=L2

End If

If Sum of L1=Sum of L2:

Print sum(L1)=sum(L2)

Else:

Print sum(L1)!=sum(L2)

End If

Print element that occur in both list

For element in L1:

If element in L2:

Print(element)

Else:

Print none

End If

Next element

**Source Code :**

List1=[]

List2=[]

common=[]

n1=int(input("ENTER THE NO.OF VALUES IN LIST1"))

n2=int(input("ENTER THE NO.OF VALUES IN LIST2"))

print("ENTER THE INTEGER VALUES IN 1ST LIST")

i=0

for i in range(n1):

num=int(input(""))

List1.append(num)

print("ENTER THE INTEGER VALUES IN 2ND LIST")

i=0

for i in range(n2):

num=int(input(""))

List2.append(num)

if len(List1)==len(List2):

print("BOTH LIST HAVE SAME LENGTH.")

else:

print("BOTH LIST HAVE DIFFERENT LENGTH.")

if sum(List1)==sum(List2):

print("SUM OF BOTH LIST ARE SAME")

else:

print("SUM OF BOTH LIST ARE DIFFERENT")

for value1 in List1:

for value2 in List2:

if value1==value2:

common.append(value1)

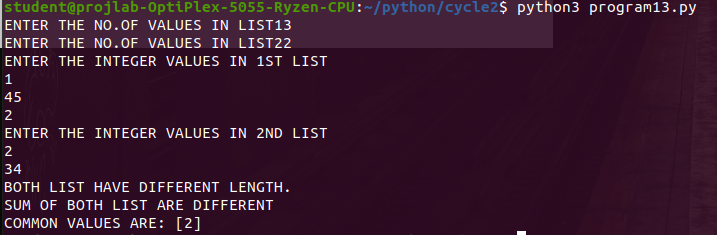
if common:

print("COMMON VALUES ARE:",common)

else:

print("NO COMMON VAULES")

**Output:**

****

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

**Experiment No : 14**

**Date : 12/10/2023**

**Aim:**

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

**Pseudocode :**

Input a string str

Initialize dictionary dict

Spilt str and store it as list str itself

For x in str:

If x in dict:

Add value by 1 of that key

Else:

Assign 1 to key( i.e., x )

End If

Next x

Print dict

**Method**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| spilt | Used to spilt a string into list  {default separator:whitespace} | string.spilt() |
| Dictionary | Used to store data value in  key:value pairs. Items can be  referred using keyname. | dict={key:value} |

**Source Code :**

input\_line=input("ENTER THE LINE OF TEXT:")

words=input\_line.split()

word\_count={}

for word in words:

word=word.strip('.,!?').lower()

word\_count[word]=word\_count.get(word,0)+1

print("WORD OCCURRENCS:",word\_count)

**Output:**

****

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

**Experiment No : 15**

**Date : 12/10/2023**

**Aim:**

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

**Pseudocode :**

Input string str

Initialize a as 0th index of str

Replace all occurrence of a as ‘$’ from 1st index of str

Print new string by concatenating a and replaced new string

**Source Code :**

string1=input("ENTER THE STRING:")

first=string1[0]

modified\_string=first

for char in string1[1:]:

if char==first:

modified\_string+="$"

else:

modified\_string+=char

print("modified string=",modified\_string)

**Output:**

****

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

**Experiment No : 16**

**Date : 12/10/2023**

**Aim:**

Create a single string separated with space from two strings by swapping the character

at position 1.

**Pseudocode :**

Input string s1,s2

Initialize new\_str by slicing and concatenating s1 and s2

Print new\_str

**Source Code :**

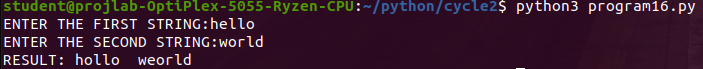
str1=input("ENTER THE FIRST STRING:")

str2=input("ENTER THE SECOND STRING:")

str3=str1[0]+str2[1]+str1[2:]+" "+str2[0]+str1[1]+str2[1:]

print("RESULT:",str3)

**Output:**

****

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

**Experiment No : 17**

**Date : 12/10/2023**

**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 :**

Input string s1, s2

Convert s1 and s2 as list using split

Find unique= set(s1) - set(s2)

Print unique

**Source Code :**

color\_list1=input("ENTER the colors:").split()

color\_list2=input("ENTER the colors:").split()

unique\_color=[]

for color in color\_list1:

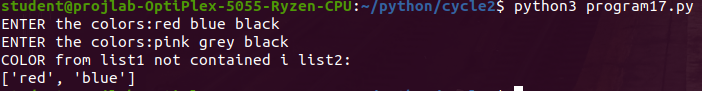
if color not in color\_list2:

unique\_color.append(color)

print("COLOR from list1 not contained i list2:")

print(unique\_color)

**Output:**

****

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

**Experiment No : 18**

**Date : 12/10/2023**

**Aim:**

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

**Pseudocode :**

Input string s

s is split with ‘,’ and stored in s as list

Print first element s[0]

Print last element s[-1]

**Source Code :**

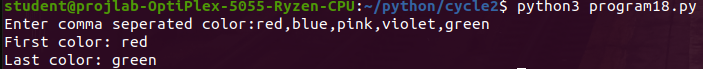
color=input("Enter comma seperated color:")

color\_list=color.split(',')

print("First color:",color\_list[0].strip())

print("Last color:",color\_list[-1].strip())

**Output:**

****

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

**Experiment No : 19**

**Date : 12/10/2023**

**Aim:**

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

**Pseudocode :**

Input list of integers into s

Spilt s

Print list

For x in s:

If (x%2!=0):

l=x

End If

Next x

Print list l

**Method**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| List comprehension | It offers a shorter syntax  when you want to create a  new list based on the values  of an existing list | newlist=[expression for item in iterable if  condition==True] |

**Source Code :**

numbers=[]

n=int(input("Enter the no.of values:"))

i=0

for i in range(n):

val=int(input("Enter the number:"))

numbers.append(val)

odd\_numbers=[]

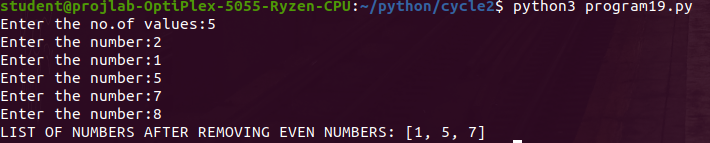
for num in numbers:

if num%2!=0:

odd\_numbers.append(num)

print("LIST OF NUMBERS AFTER REMOVING EVEN NUMBERS:",odd\_numbers)

**Output:**

****

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

**Experiment No : 20**

**Date : 26/10/2023**

**Aim:**

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

**Pseudocode :**

Input string str

Initialize dictionary dict

keys=List of keys of dict

For x in str:

If(x in keys):

dict(x)=dict(x)+1

Else:

dict(x)=1

End If

Next x

Print(dict)

**Source Code :**

input\_string = input("Enter a string: ")

char\_frequency = {}

for char in input\_string:

if char in char\_frequency:

char\_frequency[char] += 1

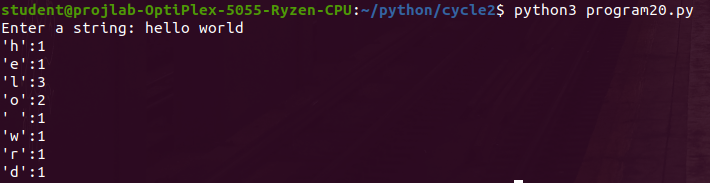
else:

char\_frequency[char] = 1

for char, frequency in char\_frequency.items():

print(f"'{char}':{frequency}")

**Output:**

****

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

**Experiment No : 21**

**Date : 26/10/2023**

**Aim:**

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

**Pseudocode :**

Input string str

If str[-3]=’ing’:

Print str+’ly’

Else:

Print str+’ing’

End If

**Source Code :**

str=input("ENTER THE STRING:")

if str.endswith("ing"):

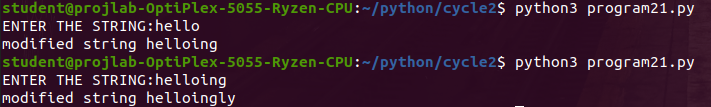
str=str+"ly"

else:

str=str+"ing"

print("modified string",str)

**Output:**

****

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

**Experiment No : 22**

**Date : 26/10/2023**

**Aim:**

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

**Pseudocode :**

List[]

Read the words into a list splited by space.

For word in list

If length (word)>longest\_word

longest\_word = length(word)

Print longest\_word

**Source Code :**

list=input("ENTER THE WORDS SEPERATED BY SPACE").split()

long\_len=0

for word in list:

if len(word)>long\_len:

long\_len=len(word)

print("LENGTH OF LONGEST WORD IS :",long\_len)

**Output:**

****

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

**Experiment No : 23**

**Date : 26/10/2023**

**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 :**

Input str

For (x in split(str)):

l=int(x)

Next x

Print l

For (x in l):

If (x>=0):

pos=x

End If

Next x

Print positive numbers, pos

For (x in l):

sq=x2

Next x

Print squares of x, sq

For ch in str:

m=ch

Next x

Print m

For (ch in m):

If (ch=a,e,i,o,u,A,E,I,O,U):

vow=ch

End if

Next ch

Print vowels, vow

For x in m:

ordi=ord(x)

Print ordinal value, ordi

**Method:**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| ord() | Returns the number representing  the unicode code of a specified  character | ord(character) |

**Source Code :**

list\_num=[]

n=int(input("ENTER THE LIMIT:"))

i=0

for i in range(n):

num=int(input("ENTER THE VALUES:"))

list\_num.append(num)

positive=[num1 for num1 in list\_num if num1>0]

N=int(input("ENTER THE LIMIT:"))

squered=[num2 \*\*2 for num2 in range(1,N+1)]

print("THE LIST OF NUMBER:",list\_num)

print("+ve numbers=",positive)

print("^2 of numbers=",squered)

word=input("ENTER THE STRING")

print(word)

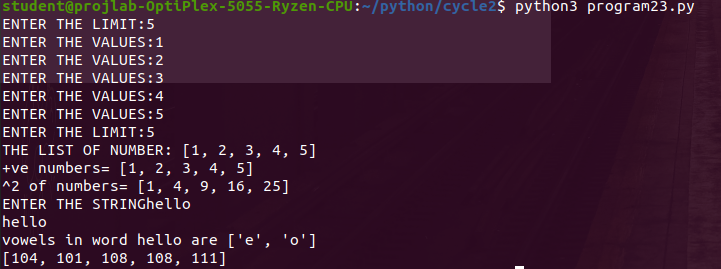
vowels=[char for char in word if char.lower() in'aeiou']

print("vowels in word",word,"are",vowels)

orginal\_val=[ord(char)for char in word]

print(orginal\_val)

**Output:**

****

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

**Experiment No : 24**

**Date : 26/10/2023**

**Aim:**

Sort dictionary in ascending and descending order

**Pseudocode :**

Initialize dictionary dict

Input number of elements num

For i in range(num):

Input key k

Input value v

Update dict

Next i

Print dict

Sort dict by key in ascending order using sort function

Sort dict by key in descending order by reverse=true

Sort dict by value in ascending and descending order

Print sorted dictionaries

**Method:**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| items | Returns a view object (contains  key-value pairs of dictionary as  tuple in list) | dictionary.items() |
| sorted | Returns a sorted list of specified  iterable object. [Iterable(Req) =>  Seq to sort; key(optional)=>fn to  execute to decide order] | sorted(iterable, key=key,  reverse=reverse) |

**Source Code :**

dictionary={}

N=int(input("Enter the no.of values:"))

i=0

for i in range(N):

new\_key=input("Enter the key.it should be a alphabet")

new\_val=int(input("Enter the value number:"))

dictionary[new\_key]=new\_val

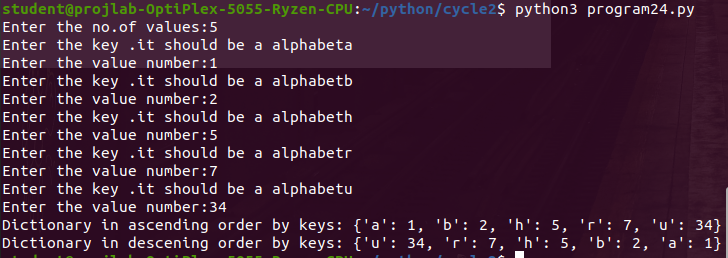
ascend\_dict=dict(sorted(dictionary.items()))

print("Dictionary in ascending order by keys:",ascend\_dict)

descend\_dict=dict(sorted(dictionary.items(),reverse=True))

print("Dictionary in descening order by keys:",descend\_dict)

**Output:**

****

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

**Experiment No : 25**

**Date : 26/10/2023**

**Aim:**

Merge two dictionaries

**Pseudocode :**

Initialize dictionaries dict1, dict2

Print “Dictionary 1”

Input number of elements num

For i in range (num):

Input key k

Input value v corresponding to k

Next i

Print “Dictionary 2”

Merged\_dict=dict1.copy()

Merged\_dict.update(dict2)

Print("Merged dictionary:",merged\_dict)

**Source Code :**

dict1={}

dict2={}

n1=int(input("ENTER THE LIMIT OF FIRST DICTIONARY:"))

i=0

for i in range(n1):

new\_key=input("ENTER THE ALPHABET KEY")

new\_value=input("ENTER THE VALUE INTEGER")

dict1[new\_key]=new\_value

n2=int(input("ENTER THE LIMIT OF SECOND DICTIONARY:"))

i=0

for i in range(n2):

new\_key=input("ENTER THE ALPHABET KEY")

new\_value=input("ENTER THE VALUE INTEGER")

dict2[new\_key]=new\_value

merged\_dict=dict1.copy()

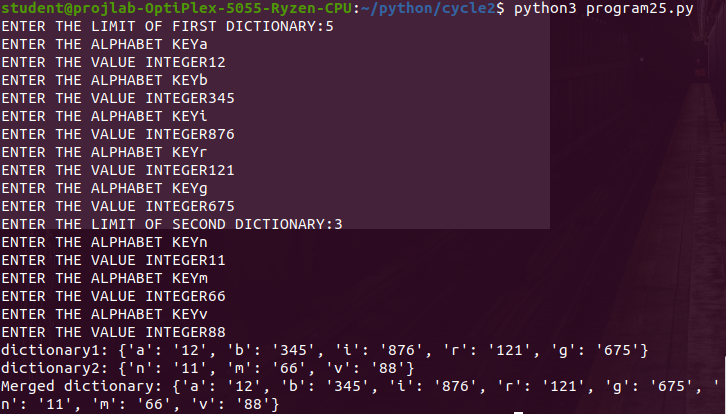
merged\_dict.update(dict2)

print("dictionary1:",dict1)

print("dictionary2:",dict2)

print("Merged dictionary:",merged\_dict)

**Output:**

****

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

**Experiment No : 26**

**Date : 26/10/2023**

**Aim:**

Write a program to find the factorial of a number

**Pseudocode :**

Input number num

Initialize f=1

For i in 1 to n+1:

f=f\*i

Next i

Print fact

**Source Code :**

num=int(input("Enter the number:"))

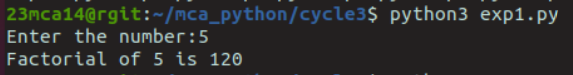
fact=1;

for i in range(1,num+1):

fact=fact\*i

print("Factorial of",num,"is",fact)

**Output:**



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

**Experiment No : 27**

**Date : 26/10/2023**

**Aim:**

Write a program to generate Fibonacci series of N terms

**Pseudocode :**

Input number x

Initialize n1=0, n2=1, count=0

If x<=0:

Print “Enter positive no”

Else if x==1:

Print n1 as Fibonacci sequence

Else:

While count<x:

Print n1

nt=n1+n2

n1=n2

n2=nt

count=count+1

End if

**Source Code :**

n=int(input("ENTER THE NUMBER OF TERMS"))

first\_term=0

second\_term=1

i=0

while(i<=n):

print(first\_term,end=",")

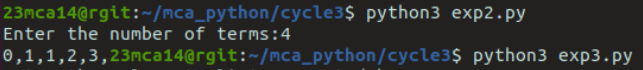
nth=first\_term+second\_term

first\_term=second\_term

second\_term=nth

i=i+1

**Output:**



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

**Experiment No : 28**

**Date : 26/10/2023**

**Aim:**

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

**Pseudocode :**

Input a list of comma separated integers str

For (x in split(str)):

l=int(x)

Next x

Print l

Initialize sum=0

For i in l:

sum = sum + i

Next i

Print sum

**Source Code :**

num\_list=input("Enter the values to list separated by space:").split()

num\_list1=[int(num) for num in num\_list]

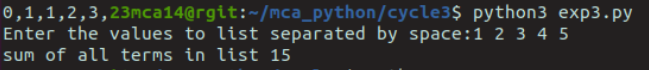
total\_sum=0

for num in num\_list1:

total\_sum+=num

print("sum of all terms in list",total\_sum)

**Output:**



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

**Experiment No : 29**

**Date : 06/11/2023**

**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 :**

Initialize an empty list ‘list’

Input limit l

If l>=1000 and l<10000:

For i=1000 to l:

If (floor(i^0.5)^2)=i:

If all(int(b)%2=0 for b in str(i)):

Insert i to list

End If End If

Next i

Else: Print “Invalid range”

End If

Print list

**Source Code :**

start=int(input("Enter the starting number : "))

end=int(input("Enter the ending number : "))

fourdigit=[]

if start<1000 or start>9999 or end<1000 or end>9999:

print("Please enter a four digit number !! " )

else:

for num in range(start,end+1):

tmp=num

if num%2==0 and int(num\*\*0.5)\*\*2==num:

num=int(num/10)

if num%2==0:

num=int(num/10)

if num%2==0:

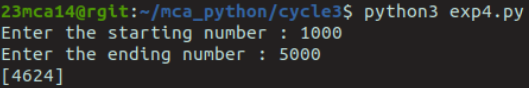
num=int(num/10)

if num%2==0:

fourdigit.append(tmp)

print(fourdigit)

**Output:**



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

**Experiment No : 30**

**Date : 06/11/2023**

**Aim:**

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

**Pseudocode :**

Input number num

For i=1 to 10:

Print ‘i’\*’num’=’num\*i’

Next i

**Source Code :**

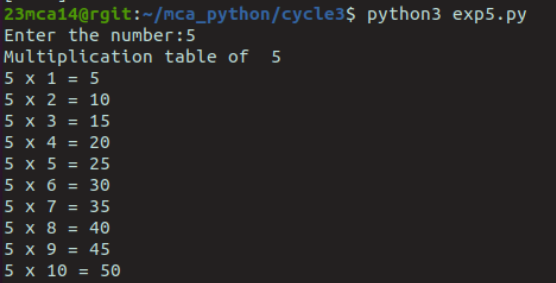
n=int(input("Enter the number:"))

print("Multiplication table of ",n)

for i in range(1,11):

print(n,"x",i,"=",n\*i)

**Output:**



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

**Experiment No : 31**

**Date : 06/11/2023**

**Aim:**

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

**Pseudocode :**

Input num

For i=1 to num:

For j=1 to i:

Print(j\*i, end=” ”)

Print()

Next j

**Source Code :**

n=int(input("Enter the number:"))

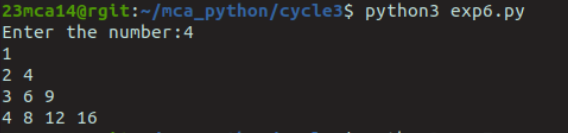
for i in range(1,n+1):

for j in range(i):

print(i\*(j+1),end=" ")

print()

**Output:**



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

**Experiment No : 32**

**Date : 06/11/2023**

**Aim:**

Write a program to generate all factors of a number [use while loop].

**Pseudocode :**

Input num

For i=1 to num:

If num%i==0:

Print i

End if

Next i

**Source Code :**

num=int(input("Enter the number:"))

divisor=1

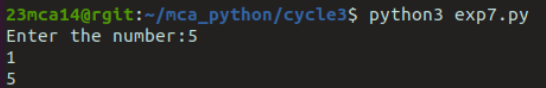
while divisor <=num:

if num%divisor==0:

print(divisor)

divisor+=1

**Output:**



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

**Experiment No : 33**

**Date : 06/11/2023**

**Aim:**

Write a program to print reverse of a number [use while loop].

**Pseudocode :**

Input number num

Initialize rev=0

while num!=0:

remainder=num%10

rev=rev\*10+remainder

num=num//10

End While

Print rev

**Source Code :**

num=int(input("Enter the number:"))

reverse\_num=0

while num>0:

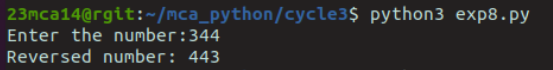
digit=num%10

reverse\_num=reverse\_num\*10+digit

num=num//10

print("Reversed number:",reverse\_num)

**Output:**



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

**Experiment No : 34**

**Date : 06/11/2023**

**Aim:**

Write a program to find whether the given number is an Armstrong or not [use while loop].

**Pseudocode :**

Input num

Store num to x

Initialize sum=0

While num>0:

remainder=num%10

sum=sum+(remainder)^3

num=num//10

End while

If sum==x:

Print x is Armstrong number

Else:

Print x is not Armstrong number

End if

**Source Code :**

num=int(input("Enter the number:"))

num\_cpy=num

num\_dig=0

while num\_cpy>0:

num\_cpy //=10

num\_dig+=1

num\_cpy=num

amstrong\_sum=0

while num\_cpy >0:

digit=num\_cpy%10

amstrong\_sum+=digit\*\*num\_dig

num\_cpy //=10

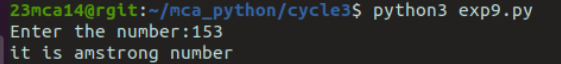
if amstrong\_sum==num:

print("it is amstrong number")

else:

print("it is not amstrong")

**Output:**



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

**Experiment No : 35**

**Date : 06/11/2023**

**Aim:**

Display star pattern using nested loop

**Pseudocode :**

Input number of rows n

For i=0 to n-1:

For j=0 to i:

Print \*,” ”

Print “ ”

Next j

Next i

For i=n to 1:

For j=0 to i-2:

Print \*,” ”

Print “ ”

Next j

i--

Next I

**Source Code :**

num=5

for i in range(num):

for j in range(i+1):

print("\*",end=" ")

print()

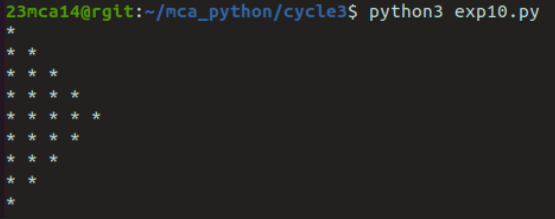
for i in range(num-1,0,-1):

for j in range(i):

print("\*",end=" ")

print()

**Output:**



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

**Experiment No : 36**

**Date : 06/11/2023**

**Aim:**

Write a program using functions to calculate the simple interest. Suppose the customer is a senior citizen. He is being offered a 12 percent rate of interest, for all other customers, the rate of interest is 10 percent.

**Pseudocode :**

Function simpleInterest (p,r,age):

if age>=60:

return (p\*t\*12)/100

else:

return (p\*t\*10)/100

End Function

Input name

Input age

Input principal amount, p

Input no.of years, t

Input age

End If

si (p,r,age)

**Source Code :**

def simpleInterest(p,t,age):

if age>=60:

return (p\*t\*12)/100

else:

return (p\*t\*10)/100

p=float(input("Enter the principal amount:"))

t=int(input("Entr the TimePeriod:"))

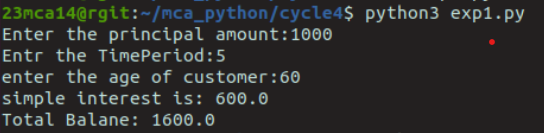
age=int(input("enter the age of customer:"))

sim=simpleInterest(p,t,age)

print("simple interest is:",sim)

print("Total Balane:",p+sim)

**Output:**



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

**Experiment No : 37**

**Date : 16/11/2023**

**Aim:**

Write a program using functions and return statements to check whether a number is even or odd.

**Pseudocode :**

Function evenodd(num):

If num%2=0:

Return even

Else:

Return odd

End If

End Function

Input num

Print evenodd(num)

**Source Code :**

def oddOrEven(num):

if (num==0):

return " zero"

elif (num%2==0):

return "even"

else:

return "odd"

num=int(input("Enter the number:"))

val=oddOrEven(num)

print(val)

**Output:**



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

**Experiment No : 38**

**Date : 16/11/2023**

**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 :**

Function compare (s1,s2,n):

If s1[:n]=s2[:n]:

Return True

Else:

Return False

End If

End Function

Input strings s1, s2

Input comparison limit, n

Print compare(s1,s2,n)

**Source Code :**

def compare(s1,s2,n):

for x in range(0,n):

if s1[x]==s2[x]:

return "true"

else:

return "false"

s1=input("Enter first string:")

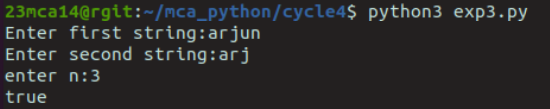
s2=input("Enter second string:")

n=int(input("enter n:"))

res=compare(s1,s2,n)

print(res)

**Output:**



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

**Experiment No : 39**

**Date : 16/11/2023**

**Aim:**

Write a program to print the Fibonacci series using recursion

**Pseudocode :**

Function fibo(n):

If n<=1:

Return n

Else:

Return (fibo(n-1) + fibo(n-2))

End If

End Function

Input no. of terms, nterms

If nterms <= 0:

Print "Enter positive no."

Else:

Print "Fibonacci Sequence"

For i = 0 to nterms:

Print Fibo(i)

Next i

End If

**Source Code :**

def recur\_fibo(n):

if n <= 1:

return n

else:

return(recur\_fibo(n-1) + recur\_fibo(n-2))

nterms=int(input("Enter the number of terms:"))

if nterms <= 0:

print("Plese enter a positive integer")

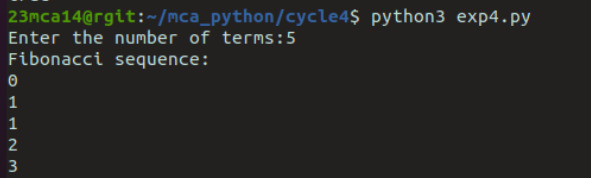
else:

print("Fibonacci sequence:")

for i in range(nterms):

print(recur\_fibo(i))

**Output:**



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

**Experiment No : 40**

**Date : 16/11/2023**

**Aim:**

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

**Pseudocode :**

Function add (\*args):

sum=0

For i in args:

sum = sum + i

Next i

Print sum

Input integers

add (num1, num2, num3)

add (num1, num2, num3, num4, num5)

**Source Code :**

def add\_integers(\*args):

"""

Adds variable-length integer arguments.

Parameters:

\*args (int): Variable number of integer arguments.

Returns:

int: Sum of the integer arguments.

"""

total = 0

for num in args:

if isinstance(num, int):

total += num

else:

raise TypeError("Arguments must be integers.")

return total

user\_input = input("Enter integers separated by spaces: ")

user\_numbers = [int(num) for num in user\_input.split()]

try:

result = add\_integers(\*user\_numbers)

print(f"Result: {result}")

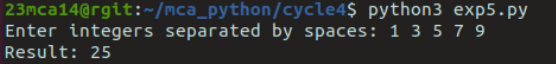
except TypeError as e:

print(f"Error: {e}")

except ValueError:

print("Error: Please enter valid integers.")

**Output:**



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

**Experiment No : 41**

**Date : 16/11/2023**

**Aim:**

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

**Pseudocode :**

Initialize ar1 as lambda x:x\*x

Initialize ar2 as lambda x,y:x\*y

Initialize ar3 as lambda x,y:0.5\*x\*y

Input length of square, a

Print ar1(a)

Input length of rectangle, l

Input breadth of rectangle, b

Print ar2(l, b)

Input base of triangle, b

Input height of triangle, h

Print ar3(b, h)

**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 a : a\*a

rectangle = lambda a,b : a\*b

triangle = lambda b,h : 0.5\*(b\*h)

a=int(input("Enter the lenth of sides of square:"))

l=int(input("Enter the lenth rectangle:"))

b=int(input("Enter the breadth of rectangle:"))

base=float(input("Enter the base of Triangle:"))

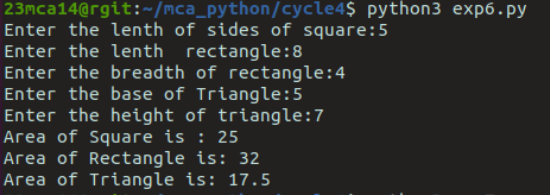
h=float(input("Enter the height of triangle:"))

print("Area of Square is :",square(a))

print("Area of Rectangle is:",rectangle(l,b))

print("Area of Triangle is:",triangle(base,h))

**output:**



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

**Experiment No : 42**

**Date : 23/11/2023**

**Aim:**

Write a program to display powers of 2 using anonymous function

**Pseudocode :**

Input no. of terms, n

Store map (lambda x:2\*\*x, range(n)) as list into result

Print n

For i = 0 to n:

Print result[i]

Next i

**Source Code :**

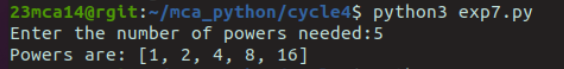
display\_powers\_of\_2 = lambda n:list(map(lambda x: 2\*\*x, range(n)))

num\_of\_powers =int(input("Enter the number of powers needed:"))

result = display\_powers\_of\_2(num\_of\_powers)

print(f"Powers are:",result)

**Output:**



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

**Experiment No : 43**

**Date : 23/11/2023**

**Aim:**

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

**Pseudocode :**

Function fact(n):

If n=1:

Return 1

Else:

Return n\*fact(n-1)

End If

End Function

Input number of terms, n

Initialize result as 0

For i = 1 to n:

f = fact(i)

result = result + (Power(i,i)/f)

Next i

Print result

**Source Code :**

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

def series\_sum(n):

result = 0

for i in range(1, n + 1):

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

result += term

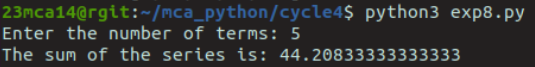
return result

n = int(input("Enter the number of terms: "))

result = series\_sum(n)

print(f"The sum of the series is: {result}")

**Output:**



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

**Experiment No : 44**

**Date : 23/11/2023**

**Aim:**

Write a program to determine whether a given year is a leap year [ Use Calendar Module].

**Pseudocode :**

Import calendar module

Input year

If calendar.isleap(year):

Print ‘year’ is a leap year

Else:

Print ‘year’ is not a leap year

End If

**Method**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| Calendar | Built-in module in Python which  allows you to perform date, month,  and calendar-related operations. | import calendar |
| Isleap | Year to be tested leap or not. | isleap(year) |

**Source Code :**

import calendar

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

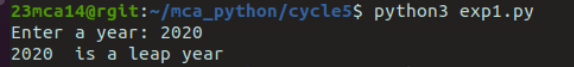
if calendar.isleap(year):

print(year," is a leap year")

else:

print(year," is not a leap year")

**Output:**



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

**Experiment No : 45**

**Date : 23/11/2023**

**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 :**

Import time module

Import datetime module

Print Current date and time: datetime.datetime.now()

Print Current year: tdy.strftime("%Y")

Print Month of the year: tdy.strftime("%B")

Print Week number of the year: tdy.strftime("%W")

Print Weekday of the week: tdy.strftime("%w")

Print Day of year: tdy.strftime("%j")

Print Day of the month: tdy.strftime("%d")

**Source Code :**

import time

import datetime

current\_time = datetime.datetime.now()

print("a) Current date and time:", current\_time)

current\_year = current\_time.year

print("b) Current Year:", current\_year)

month = current\_time.strftime("%B")

print("c) Month of the year:", month)

week\_number = current\_time.strftime("%U")

print("d) Week number of the year:", week\_number)

weekday = current\_time.strftime("%A")

print("e) Weekday of the week:", weekday)

day\_of\_year = current\_time.strftime("%j")

print("f) Day of year:", day\_of\_year)

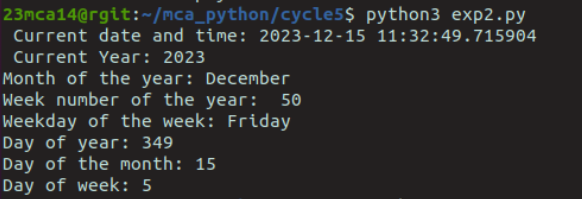
day\_of\_month = current\_time.strftime("%d")

print("g) Day of the month:", day\_of\_month)

day\_of\_week = current\_time.strftime("%w")

print("h) Day of week (0 - Sunday, 1 - Monday, ...):", day\_of\_week)

**Output:**



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

**Experiment No : 46**

**Date : 30/11/2023**

**Aim:**

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

**Pseudocode :**

Import datetime module

Initialize today as datetime.date.today()

Store today - datetime.timedelta(days = 1) into yesterday

Store today + datetime.timedelta(days = 1) into tomorrow

Print yesterday

Print today

Print tomorrow

**Method**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| Timedelta | It is built in function in datetime module which is used for calculating differences in dates and also can be used for date  manipulations in Python. | datetime.timedelta(days=0,  seconds=0,microseconds=0,  milliseconds=0, minutes=0,  hours=0, weeks=0) |

**Source Code :**

import datetime

today = datetime.date.today()

yesterday = today - datetime.timedelta(days=1)

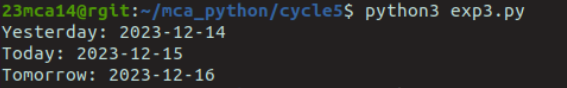
tomorrow = today + datetime.timedelta(days=1)

print("Yesterday:", yesterday)

print("Today:", today)

print("Tomorrow:", tomorrow)

**Output:**



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

**Experiment No : 47**

**Date : 30/11/2023**

**Aim:**

Write a function in file Armstrong.py to check whether a number is an Armstrong number. Import the module to generate Armstrong numbers between two limits.

**Pseudocode :**

Function Amstrong(num):

Sum=0 and temp=num

While temp!=0

rem=temp%10

sum=sum+(rem\*\*3)

temp=temp/10

If sum equals to num

Print num is amstrong number

Else

Print num is not amstrong number

End If

End Function

Read lower and upper limits

For i from lower limit to upper limit+1

If Amstrong(i) equals true

Print i is amstrong

End If

Next i

**Source Code :**

from armstrong import generate\_armstrong\_numbers

start\_limit = int(input("Enter the start limit: "))

end\_limit = int(input("Enter the end limit: "))

armstrong\_numbers = generate\_armstrong\_numbers(start\_limit, end\_limit)

print("Armstrong numbers between", start\_limit," and", end\_limit,":", armstrong\_numbers)

**Armstrong.py**

def is\_armstrong\_number(num):

order = len(str(num))

sum\_of\_digits = sum(int(digit) \*\* order for digit in str(num))

return num == sum\_of\_digits

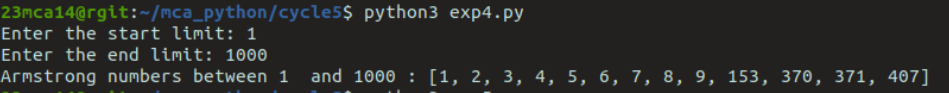
def generate\_armstrong\_numbers(start, end):

armstrong\_numbers = [num for num in range(start, end + 1) if is\_armstrong\_number(num)]

return armstrong\_numbers else:

return False

**Output:**



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

**Experiment No : 48**

**Date : 08/12/2023**

**Aim:**

Create a package graphics with modules rectangle, circle and sub-packagethreeDgraphics with modules cuboid and sphere. Include methods to findarea and perimeter of respective figures in each module. Write programsthat find the area and perimeter of figures by different importing statements.

(Include selective import of modules and import \* statements)

**Pseudocode :**

Read length and width

Print rectangle.perimeter(length,width)

Print rectangle.area(length,width)

Read radius

Print circle.perimeter(radius)

Print circle.area(radius)

Read length, width and height

Print cuboid.area(length,width,height)

Print cuboid.perimeter (length,breadth,height)

Read radius

Print sphere.area(radius)

Print sphere.perimeter(radius)

Create package graphics with modules circle and rectangle

**circle.py**

Function area(r):

return 3.14\*r\*r

Function perimeter(r):

return 2\*3.14\*r

**rectangle.py**

Function area(l,w):

return l\*w

Function perimeter(l,w):

return 2\*(l+w)

Create package threeDgraphics with modules cuboid and sphere

**cuboid.py**

Function area(l,w,h):

return ((2\*(l\*w))+(2\*(l\*h))+(2\*(w\*h)))

Function perimeter(l,b,h):

return 4\*(l+b+h)

**sphere.py**

Function area(r):

return 4\*3.14\*r\*r

Function perimeter(r):

return (2\*3.14\*r)

**Source Code :**

from rectangle import area as rectArea

from rectangle import perimeter as rectPerimeter

from circle import perimeter as circlePerimeter

from circle import area as circleArea

from ThreeD.cuboid import \*

from ThreeD.sphere import \*

length=int(input("Enter the length of rectangle:"))

breadth=int(input("Enter the breadth of rectangle:"))

radius=int(input("Enter radius of circle:"))

l=int(input("Enter the length of cuboid:"))

w=int(input("Enter the width of cuboid:"))

h=int(input("Enter the height of cuboid:"))

r=int(input("Enter radius of sphere:"))

a1=rectArea(length,breadth)

a2=circleArea(radius)

p1=rectPerimeter(length,breadth)

p2=circlePerimeter(radius)

sa1=surface\_area(l,w,h)

v1=volu\_me(l,w,h)

sa2=surfaceArea(r)

v2=volume(r)

print("Area of raectangle",a1)

print("Perimeter of rectangle:",p1)

print("Area of circle",a2)

print("Perimeter of circle:",p2)

print("Surface Area of cuboid",sa1)

print("volume of cuboid",v1)

print("Surface Area of sphere",sa2)

print("volume of sphere",v2)

**rectangle.py**

def area(length, width):

return length \* width

def perimeter(length,width):

return 2 \* (length + width)

**circle.py**

def area(radius):

return 3.14 \* radius \*\* 2

def perimeter(radius):

return 2 \* 3.14 \* radius

**cuboid.py**

def surface\_area(length, width, height):

return 2 \* ((length \* width) +( width \* height) +( height \* length))

def volu\_me(length, width, height):

return length \* width \* height

**sphere.py**

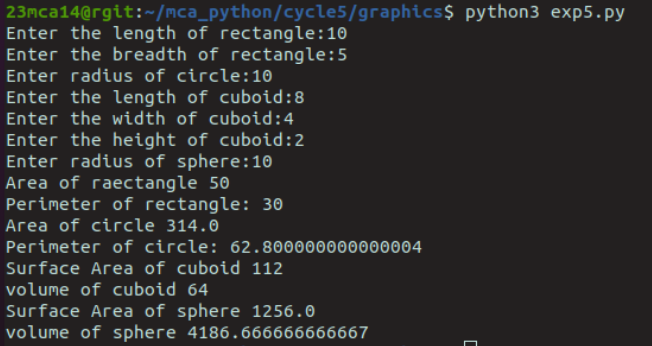
def surfaceArea(radius):

return 4 \*3.14 \* radius \*\* 2

def volume(radius):

return (4 / 3) \*3.14 \* radius \*\* 3

**Output:**



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

**Experiment No : 49**

**Date : 08/12/2023**

**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 depositan amount , withdraw an amount after checking the balance, to display details such as name, account number, account type and balance.

**Pseudocode :**

create a class to represent bank account with instance variable

name, account no, type and balance

Create method deposit()

Update balance=balance+amount

Create method withdrawal ()

If(balance>=amount)

Update balance=balance-amount

Create method display ()

Print name, account number, type and balance

Create object obj of class account

Call the methods

Obj.deposit(amount)

Obj.withdraw(amount)

Obj.display()

**Source Code:**

class BankAccount:

def \_\_init\_\_(self, name, account\_number, account\_type, initial\_balance=0):

self.name = name

self.account\_number = account\_number

self.account\_type = account\_type

self.balance = initial\_balance

def deposit(self, amount):

if amount > 0:

self.balance += amount

print(f"Deposit of ${amount} successful.")

else:

print("Deposit amount should be greater than zero.")

def withdraw(self, amount):

if amount > 0:

if self.balance >= amount:

self.balance -= amount

print(f"Withdrawal of ${amount} successful.")

else:

print("Insufficient funds.")

else:

print("Withdrawal amount should be greater than zero.")

def display\_details(self):

print(f"Account Details\nName: {self.name}\nAccount Number: {self.account\_number}\nAccount Type: {self.account\_type}\nBalance: ${self.balance}")

name = input("Enter depositor's name: ")

account\_number = input("Enter account number: ")

account\_type = input("Enter account type: ")

initial\_balance = float(input("Enter initial balance: "))

account = BankAccount(name, account\_number, account\_type, initial\_balance)

account.display\_details()

deposit\_amount = float(input("Enter amount to deposit: "))

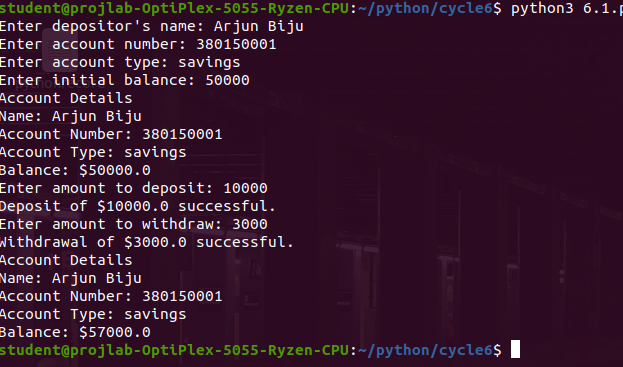
account.deposit(deposit\_amount)

withdraw\_amount = float(input("Enter amount to withdraw: "))

account.withdraw(withdraw\_amount)

account.display\_details()

**Output:**



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

**Experiment No : 50**

**Date : 08/12/2023**

**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 :**

Create a class publisher with instance variable

Publisher\_id and Publisher\_name

Create method display()

Print publisher\_id and publisher\_name

Create class book and inherit class publisher with instance variable

Title and Author

Create method display()

Print title and author

Create a class python by inheriting class book with instance

Variable, Price and No of pages

Create method display()

Print price and no of pages

Create object obj of class python

Obj.display()

**Source Code:**

class Publisher:

def \_\_init\_\_(self, publisher\_id, publisher\_name):

self.publisher\_id = publisher\_id

self.publisher\_name = 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\_info(self):

print(f"Title: {self.title}\nAuthor: {self.author}\nPublisher ID: {self.publisher\_id}\nPublisher Name: {self.publisher\_name}")

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\_info(self):

super().display\_info()

print(f"Price: ${self.price}\nNumber of Pages: {self.no\_of\_pages}")

publisher\_id = input("Enter Publisher ID: ")

publisher\_name = input("Enter Publisher Name: ")

title = input("Enter Title of the book: ")

author = input("Enter Author of the book: ")

price = float(input("Enter Price of the book: "))

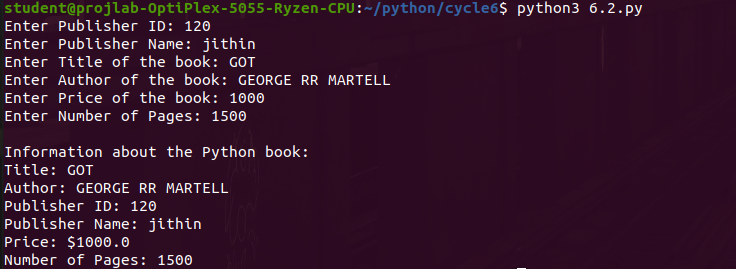
no\_of\_pages = int(input("Enter Number of Pages: "))

python\_book = Python(publisher\_id, publisher\_name, title, author, price, no\_of\_pages)

print("\nInformation about the Python book:")

python\_book.display\_info()

**Output:**



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

**Experiment No : 51**

**Date : 15/12/2023**

**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 :**

Create an abstract class polygon

Create abstract method area()

Pass

Create clays Triangle by inheriting polygon with instance

Variable hand b.

Create method area ()

Print h &b

Create class Rectangle by inheriting polygon with instance variable

Create method area ()

Print word

Read choice.

If choice equals 1,

Read base and height of triangle

Create object obj of Triangle

Obj. Area ()

Else if choice equals 2

Read length and width of rectangle

create object obj of Rectangle obj. Area ()

**Source Code:**

from abc import ABC, abstractmethod

class Polygon(ABC):

def \_\_init\_\_(self, num\_sides):

self.num\_sides = num\_sides

@abstractmethod

def get\_dimensions(self):

pass

@abstractmethod

def calculate\_area(self):

pass

class Rectangle(Polygon):

def \_\_init\_\_(self):

super().\_\_init\_\_(4)

self.length = 0

self.breadth = 0

def get\_dimensions(self):

self.length = float(input("Enter length of the rectangle: "))

self.breadth = float(input("Enter breadth of the rectangle: "))

def calculate\_area(self):

return self.length \* self.breadth

class Triangle(Polygon):

def \_\_init\_\_(self):

super().\_\_init\_\_(3)

self.base = 0

self.height = 0

def get\_dimensions(self):

self.base = float(input("Enter base length of the triangle: "))

self.height = float(input("Enter height of the triangle: "))

def calculate\_area(self):

return 0.5 \* self.base \* self.height

rectangle = Rectangle()

triangle = Triangle()

rectangle.get\_dimensions()

triangle.get\_dimensions()

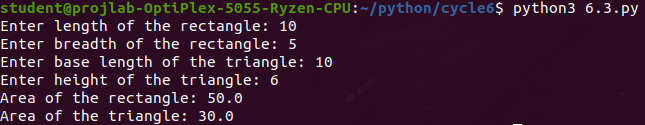
area\_rectangle = rectangle.calculate\_area()

area\_triangle = triangle.calculate\_area()

print(f"Area of the rectangle: {area\_rectangle}")

print(f"Area of the triangle: {area\_triangle}")

**Output:**

****

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

**Experiment No : 52**

**Date : 15/12/2023**

**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 :**

Create class named Rectangle with instance variables

breadth and length.

Create method area ()

return length \* breadth

Create method Perimeter()

return 2\* length breadth

Create method \_\_gt\_\_ (Self, other)

if (self.area() > other.area()):

return True

else

return False

Read length and breadth for first rectangle

Create object obj1 of rectangle

Read length and breadth for second rectangle

Create object obj2 of rectangle

If (obj1> obj2 )

Print obj1, is greater

Else

Print obj2 is greater.

**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)

length1 = float(input("Enter length of first rectangle: "))

breadth1 = float(input("Enter breadth of first rectangle: "))

length2 = float(input("Enter length of second rectangle: "))

breadth2 = float(input("Enter breadth of second rectangle: "))

rectangle1 = Rectangle(length1, breadth1)

rectangle2 = Rectangle(length2, breadth2)

area1 = rectangle1.area()

area2 = rectangle2.area()

if area1 > area2:

print("Area of first rectangle is greater.")

elif area2 > area1:

print("Area of second rectangle is greater.")

else:

print("Both rectangles have the same area.")

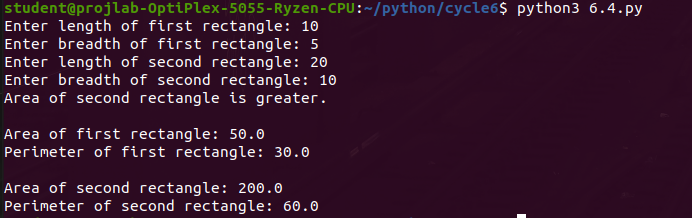
print(f"\nArea of first rectangle: {area1}")

print(f"Perimeter of first rectangle: {rectangle1.perimeter()}")

print(f"\nArea of second rectangle: {area2}")

print(f"Perimeter of second rectangle: {rectangle2.perimeter()}")

**Output:**

****

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

**Experiment No : 53**

**Date : 15/12/2023**

**Aim:**

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

**Pseudocode :**

Create class named Time with instance variables

Hour, Minute and Seconds

Create method \_\_add\_\_(self,other)

Time1=self.hour+self.minute+self.seconds

Return time1+(other.hour+other.minute+other.seconds)

Read first time as hour minute and second

Create object of Time t1

Read second time as hour minute and second

Create object of Time t2

New\_time = t1+t2

print New\_time

**Source Code:**

class Time:

def \_\_init\_\_(self):

self.\_\_hour = 0

self.\_\_minute = 0

self.\_\_second = 0

def set\_time(self, hour, minute, second):

self.\_\_hour = hour

self.\_\_minute = minute

self.\_\_second = second

def \_\_add\_\_(self, other):

total\_seconds\_self = self.\_\_hour \* 3600 + self.\_\_minute \* 60 + self.\_\_second

total\_seconds\_other = other.\_\_hour \* 3600 + other.\_\_minute \* 60 + other.\_\_second

total\_seconds\_sum = total\_seconds\_self + total\_seconds\_other

hours = total\_seconds\_sum // 3600

minutes = (total\_seconds\_sum % 3600) // 60

seconds = total\_seconds\_sum % 60

result = Time()

result.set\_time(hours, minutes, seconds)

return result

def display\_time(self):

print(f"Time: {self.\_\_hour}:{self.\_\_minute}:{self.\_\_second}")

hour1 = int(input("Enter hours for time 1: "))

minute1 = int(input("Enter minutes for time 1: "))

second1 = int(input("Enter seconds for time 1: "))

hour2 = int(input("Enter hours for time 2: "))

minute2 = int(input("Enter minutes for time 2: "))

second2 = int(input("Enter seconds for time 2: "))

time1 = Time()

time1.set\_time(hour1, minute1, second1)

time2 = Time()

time2.set\_time(hour2, minute2, second2)

result\_time = time1 + time2

print("\nTime 1:")

time1.display\_time()

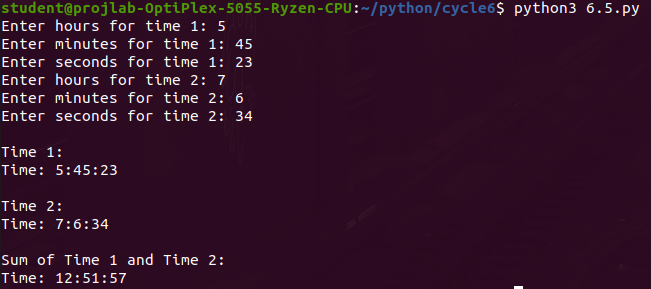
print("\nTime 2:")

time2.display\_time()

print("\nSum of Time 1 and Time 2:")

result\_time.display\_time()

**Output:**



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

**Experiment No : 54**

**Date : 15/12/2023**

**Aim:**

Write a program that validates name and age as entered by the user to determine whether the person can cast a vote or not.

**Pseudocode :**

Create class VoteError by inheriting Exception

Pass

Create class InvalidError by inheriting Exception

Pass

Read age and Name

If age <= 0 then

Invoke InvalidError

Else if age<18 then

Invoke VoteError

Else

Print Eligible for voting

Except InvalidError

Print “Enter a valid age”

Except VoteError

Print “Not eligible for voting”

**Source Code:**

def validate\_name(name):

return name.isalpha() or ' ' in name

def validate\_age(age):

try:

age = int(age)

if age >= 18:

return True

else:

return False

except ValueError:

return False

name = input("Enter your name: ")

age = input("Enter your age: ")

if validate\_name(name) and validate\_age(age):

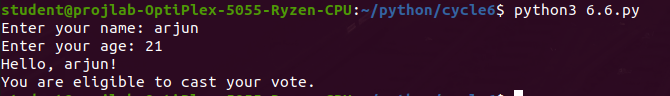
print(f"Hello, {name}!")

print("You are eligible to cast your vote.")

else:

print("Sorry, you are not eligible to cast your vote.")

**Output:**

****

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

**Experiment No : 55**

**Date : 04/01/2024**

**Aim:**

Write a program that prompts the user to enter a number. If the number is positive or zero print it, otherwise raise a ‘ValueError’ Exception.

**Pseudocode :**

Read value

If value < 0 then

Invoke ValueError(“This is a negative number”)

Else

Print value

Except ValueError as e

Print e

**Source Code:**

try:

number = float(input("Enter a number: "))

if number >= 0:

print(f"The entered number is: {number}")

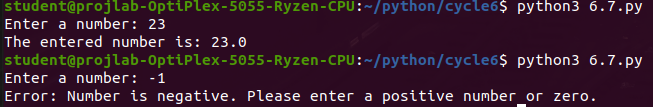
else:

raise ValueError("Number is negative")

except ValueError as e:

print(f"Error: {e}. Please enter a positive number or zero.")

**Output:**

****

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

**Experiment No : 56**

**Date : 04/01/2024**

**Aim:**

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

**Pseudocode :**

Open file with read access

Lines=[]

For every line in file

Add line to Lines

Print Lines

**Source Code:**

file = open("File1.txt","r")

lines=[]

for line in file:

lines.append(line)

print(lines)

file.close()

**Output:**

****

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

**Experiment No : 57**

**Date : 04/01/2024**

**Aim:**

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

**Pseudocode :**

Open file1 with read access

Open file2 with write access

Read all lines of file1 and store to Lines

For i from 0 to length of Lines

If i%2!=0

Write Lines[i] to file2

**Source Code:**

file = open("File1.txt","r")

file2 = open("File2.txt","w")

line = file.readlines()

for i in range(len(line)):

if(i%2!=0):

file2.write(line[i])

file.close()

file2.close()

print (” successfully copied”)

**Output:**

****

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