# Course Code: UGCA-1917 Course Name: Python Laboratory

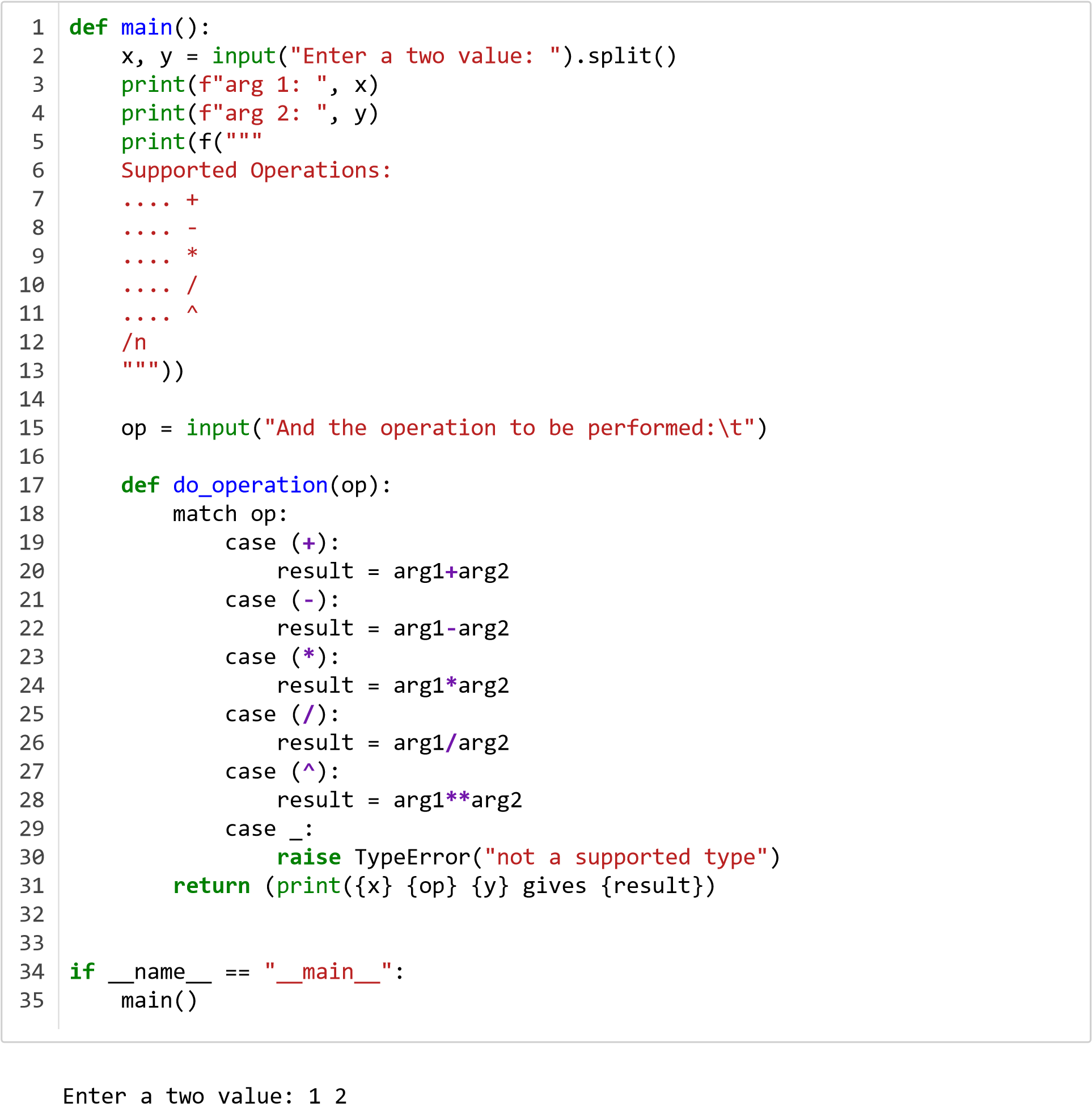
**Submitted to :**

**Submitted by :**

**BCA 3**

1. Compute sum, subtraction, multiplication, division and exponent of given variables input by the user.
2. Compute area of following shapes: circle, rectangle, triangle, square, trapezoid and parallelogram.
3. Compute volume of following 3D shapes: cube, cylinder, cone and sphere.
4. Compute and print roots of quadratic equation ax2+bx+c=0, where the values of a, b, and c are input by the user.
5. Print numbers up to N which are not divisible by 3, 6, 9,, e.g., 1, 2, 4, 5, 7,….
6. Write a program to determine whether a triangle is isosceles or not?
7. Print multiplication table of a number input by the user.
8. Compute sum of natural numbers from one to n number.
9. Print Fibonacci series up to n numbers e.g. 0 1 1 2 3 5 8 13…..n
10. Compute factorial of a given number.
11. Count occurrence of a digit 5 in a given integer number input by the user.
12. Print Geometric and Harmonic means of a series input by the user.
13. Evaluate the following expressions: a. x-x2/2!+x3/3!- x4/4!+… xn/n!
14. b. x-x3/3!+x5/5!- x7/7!+… xn/n!
15. Print all possible combinations of 4, 5, and 6.
16. Determine prime numbers within a specific range.
17. Count number of persons of age above 60 and below 90.
18. Compute transpose of a matrix.
19. Perform following operations on two matrices.
20. 1) Addition 2) Subtraction 3) Multiplication
21. Count occurrence of vowels.
22. Count total number of vowels in a word.
23. Determine whether a string is palindrome or not.
24. Perform following operations on a list of numbers: 1) Insert an element 2) delete an element 3) sort the list 4) delete entire list
25. Display word after Sorting in alphabetical order.
26. Perform sequential search on a list of given numbers.
27. Perform sequential search on ordered list of given numbers.
28. Maintain practical note book as per their serial numbers in library using Python dictionary.
29. Perform following operations on dictionary 1) Insert 2) delete 3) change
30. Check whether a number is in a given range using functions.
31. Write a Python function that accepts a string and calculates number of upper case letters and lower case letters available in that string.
32. To find the Max of three numbers using functions.
33. Multiply all the numbers in a list using functions.
34. Solve the Fibonacci sequence using recursion.
35. Get the factorial of a non-negative integer using recursion.
36. Write a program to create a module of factorial in Python.
37. Design a Python class named Rectangle, constructed by a length & width, also design a method which will compute the area of a rectangle.
38. Design a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.
39. Design a Python class to reverse a string ‘word by word’.
40. Write a Python program to read an entire text file.
41. Design a Python program to read first n lines of a text file.
42. Construct a Python program to write and append text to a file and display the text.

# 1) Compute sum, subtraction, multiplication, division and exponent of given variables



arg 1: 1 arg 2: 2

Supported Operations:

.... +

.... -

.... \*

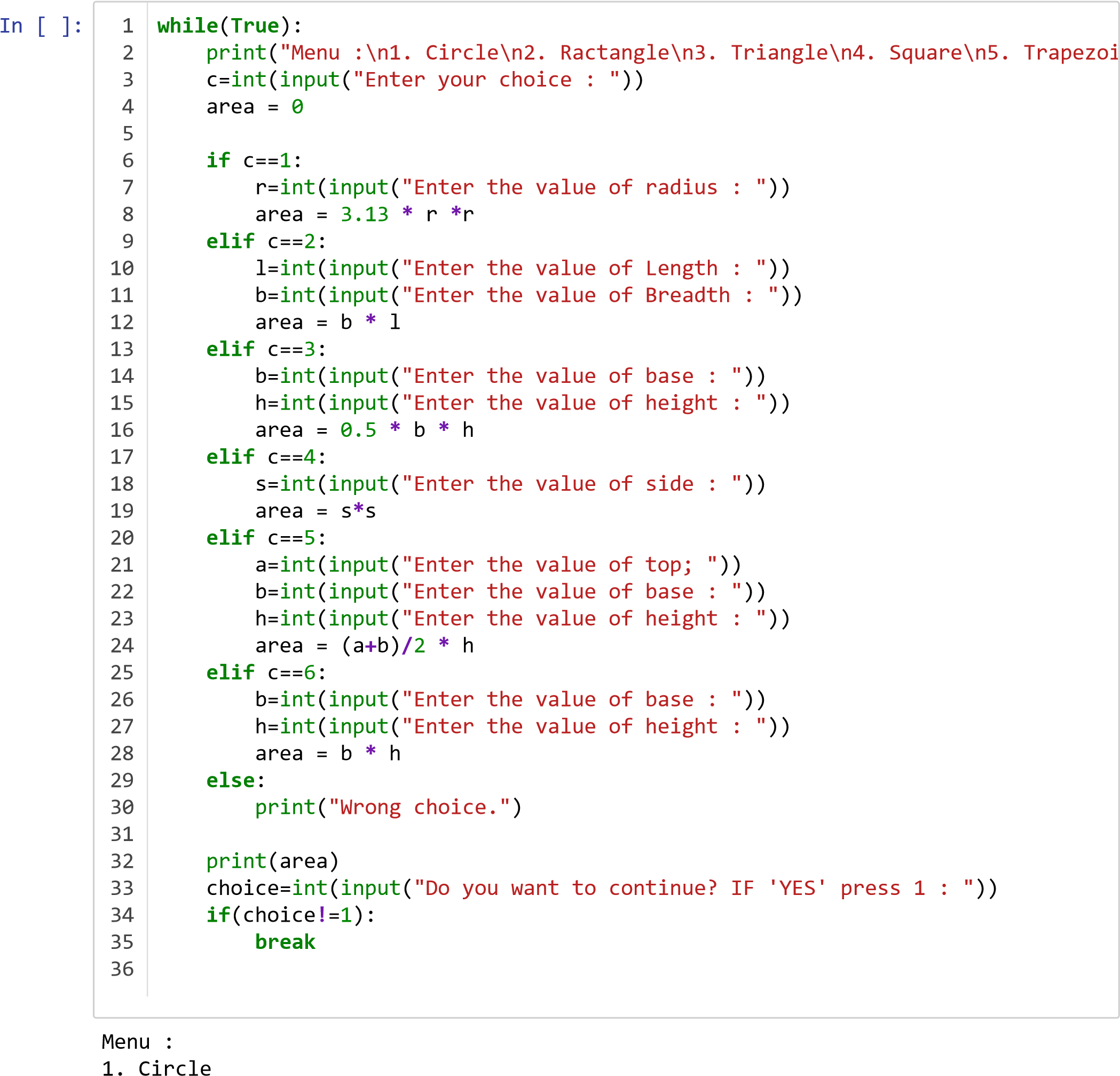
.... /

.... ^

And the operation to be performed: \*

1. \* 2 gives 2

**2) Compute area of following shapes: circle, rectangle, triangle, square, trapezoid and parallelogram.**



1. Ractangle
2. Triangle
3. Square
4. Trapezoid

6.Parallelogram

Enter your choice : 2

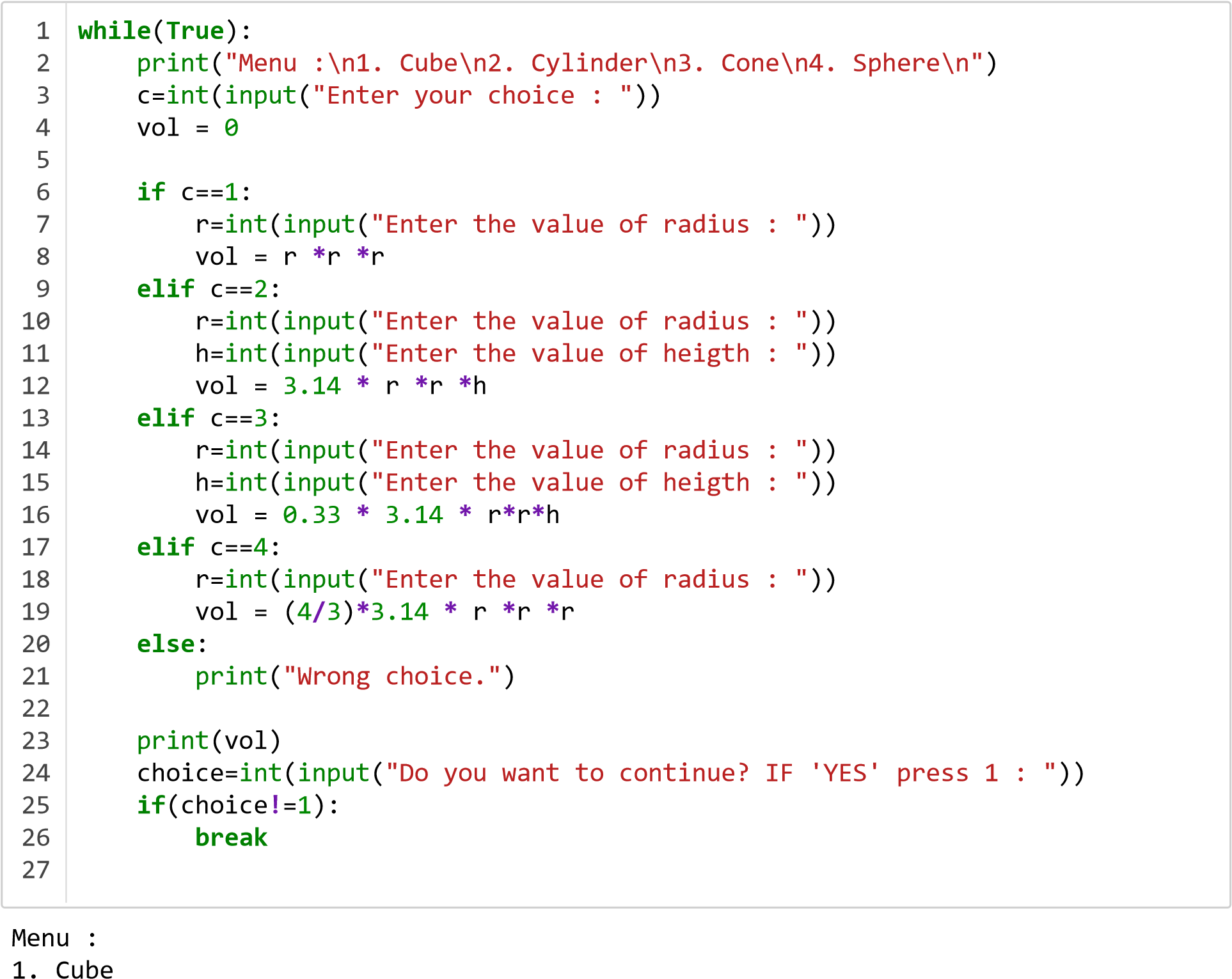
Enter the value of Length : 1

Enter the value of Breadth : 2

2

Do you want to continue? IF 'YES' press 1 : 2

**3) Compute volume of following 3D shapes: cube, cylinder, cone and sphere.**



1. Cylinder
2. Cone
3. Sphere

Enter your choice : 2

Enter the value of radius : 4

Enter the value of heigth : 2

100.48

Do you want to continue? IF 'YES' press 1 : 2

# 4) Print roots of quad. eq. inputted

In

[ ]:

Input a: 1

Input b: 2

**import**

cmath

a

=

int

(

input

(

"Input a: "

))

b

=

int

(

input

(

"Input b: "

))

c

=

int

(

input

(

"Input c: "

))

d

= (

b

**\*\***

2

)

**-**

(

4

**\***

a

**\***

c

)

solution1

= (

**-**

b

**+**

cmath

.

sqrt

(

d

))

**/**

(

2

**\***

a

)

solution2

= (

**-**

b

**-**

cmath

.

sqrt

(

d

))

**/**

(

2

**\***

a

)

print

(

"The solutions to the quadratic "

"equation are {} and {}"

.

format

(

solutio

1

2

3

4

5

6

7

8

9

10

11

Input c: 3

The solutions to the quadratic equation are (-1+1.4142135623730951j) and (-1-1.414 2135623730951j)

1. **) N numbers not divisble by 3,6,9...**

In

[ ]:

Enter : 10

3

N

=

int

(

input

(

"Enter : "

))

**for**

i

**in**

range

(

1

,

N

**+**

1

):

**if**

((

i

**%**

3

)==

0

)

**or**

((

i

**%**

6

)==

0

)

**or**

((

i

**%**

9

)==

0

):

print

(

i

,

"\n"

)

1

2

3

4

5

6

9

1. **) Triangle - isosceles or not?**

In

]:

side A : 1

side b : 2

side1

=

int

(

input

(

"side A : "

))

side2

=

int

(

input

(

"side b : "

))

side3

=

int

(

input

(

"side c : "

))

**if**

side1

==

side2

**or**

side2

==

side3

**or**

side1

==

side3

:

print

(

"This is an isosceles triangle!"

)

**else**

print

(

"Not an Isosceles "

)

1

2

3

4

5

6

7

side c : 1 This is an isosceles triangle!

# 7) Multiplication table using input()

In [ ]:

num

=

int

(

input

(

"Input: "

))

**for**

i

**in**

range

(

0

,

num

**+**

1

):

print

(

num

,

'x\t'

,

i

,

'=\t'

,

num

**\***

i

)

1

2

3

|  |  |  |
| --- | --- | --- |
| Input: 10 | |  |
| 10 x | 0 = | 0 |
| 10 x | 1 = | 10 |
| 10 x | 2 = | 20 |
| 10 x | 3 = | 30 |
| 10 x | 4 = | 40 |
| 10 x | 5 = | 50 |
| 10 x | 6 = | 60 |
| 10 x | 7 = | 70 |
| 10 x | 8 = | 80 |
| 10 x | 9 = | 90 |
| 10 x | 10 = | 100 |

# 8) Sum of N natural numbers

Input : 10

S : 1

s

=

1

i

=

1

n

=

int

(

input

(

"Input : "

))

**for**

i

**in**

range

(

0

,

n

**+**

1

):

s

=

s

**+**

i

print

(

"S : "

,

s

)

print

(

"sum of first 10 natural numbers"

,

s

)

1

2

3

4

5

6

7

S : 2

S : 4

S : 7

S : 11

S : 16

S : 22

S : 29

S : 37

S : 46 S : 56 sum of first 10 natural numbers 56

# 9) Fibonacci N

In

[ ]:

Enter n : 7

8

1 1 2 3 5

0

n

=

int

(

input

(

"Enter n : "

))

f

=

0

*#first*

s

=

1

*#second*

**if**

(

n

==

1

):

print

(

f

)

**if**

(

n

==

2

):

print

(

f

,

s

,

end

=

" "

)

**for**

i

**in**

range

(

3

,

n

**+**

1

):

new

=

f

**+**

s

print

(

new

,

end

=

" "

)

f

=

s

s

=

new

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

# 10 ) Factorial

1 n = int(input("Enter n : "))

2

1. **if**( n == 0 **or** n == 1):
2. print(1)

5

1. fact = 1
2. **for** i **in** range(1,n**+**1):
3. fact \*= i

9

1. print("The factorial is : ",fact)
2. **%**time *# time ~1 ns*

Enter n : 100000

The factorial is : 28242294079603478742934215780245355184774949260912248505789

1808654297795090106301787255177141383116361071361173736196295147499618312391802

272607340909383242200555696886678403803773794449612683801478751119669063860449

# 11 ) Occurence of 5 in an int. via input()

In

[ ]:

Enter List Elements:3 4 5 555

a

=(

input

(

"Enter List Elements:"

)).

split

()

l

=

list

(

map

(

int

,

a

))

print

(

"Original List:"

,

l

)

d

=

dict

()

**for**

i

**in**

l

:

**if**

i

**in**

d

:

d

[

i

]+=

1

**else**

:

d

[

i

]=

1

print

(

"Printing count of each item:"

,

d

)

1

2

3

4

5

6

7

8

9

10

Original List: [3, 4, 5, 555]

Printing count of each item: {3: 1, 4: 1, 5: 1, 555: 1}

# 12) Geometric and Harmonic Mean

The Geometric Mean is: 16.916852032061655

The Harmoic Mean is: 14.453529428013562

**import**

statistics

multiply

=

1

values

= [

8

,

16

,

22

,

12

,

41

]

AMean

=

statistics

.

mean

(

values

)

**for**

i

**in**

values

:

multiply

= (

multiply

)

**\***

(

i

)

GMean

= (

multiply

)

**\*\***

(

1

**/**

n

)

HMean

= (

GMean

**\*\***

2

)

**/**

AMean

print

(

'The Geometric Mean is: '

**+**

str

(

GMean

))

print

(

'The Harmoic Mean is: '

**+**

str

(

HMean

))

1

2

3

4

5

6

7

8

9

10

11

12

13

# 13 ) Evaluate expressions

In

[1]:

**def**

factorial

(

n

):

**if**

n

**<**

=

1

:

**return**

1

**return**

n

**\***

factorial

(

n

**-**

1

)

1

2

3

4

1. **x-x2/2!+x3/3!- x4/4!+… xn/n!**

In

[3]:

Enter n: 7

**def**

sum\_series

(

x

,

n

):

sum

=

0

**for**

i

**in**

range

(

1

,

n

**+**

1

,

1

):

term

= ((

**-**

1

)

**\*\***

(

i

**+**

1

))

**\***

(

x

**\*\***

(

2

**\***

i

**-**

2

)

**/**

factorial

(

2

**\***

i

**-**

2

))

*#Calculating nth ter*

sum

+=

term

**return**

sum

**if**

\_\_name\_\_

==

"\_\_main\_\_"

:

n

=

int

(

input

(

"Enter n: "

))

x

=

int

(

input

(

"Enter x: "

))

sum

=

sum\_series

(

x

,

n

)

print

(

"\nSum of {} terms of series for x={}: {}"

.

format

(

n

,

x

,

sum

))

1

2

3

4

5

6

7

8

9

10

11

12

Enter x: 4

Sum of 7 terms of series for x=4: -0.6507594463150026

1. **x-x3/3!+x5/5!- x7/7!+… xn/n!**

[8]:

Enter n: 7

**def**

sum\_series

(

x

,

n

):

sum

=

0

**for**

i

**in**

range

(

1

,

n

**+**

1

,

1

):

term

= ((

**-**

1

)

**\*\***

(

i

**+**

1

))

**\***

(

x

**\*\***

(

2

**\***

i

**-**

1

)

**/**

factorial

(

2

**\***

i

**-**

1

))

*#Calculating nth ter*

sum

+=

term

**return**

sum

**if**

\_\_name\_\_

==

"\_\_main\_\_"

:

n

=

int

(

input

(

"Enter n: "

))

x

=

int

(

input

(

"Enter x: "

))

sum

=

sum\_series

(

x

,

n

)

print

(

"\nSum of {} terms of series for x={}: {}"

.

format

(

n

,

x

,

sum

))

1

2

3

4

5

6

7

8

9

10

11

12

Enter x: 4

Sum of 7 terms of series for x=4: -0.7560275115830666

# 14) Combinations 4,5,6

In [20]:

my\_list

=

list

(

map

(

int

,

input

(

"Enter the list of numbers (sep by space): "

).

spl

my\_list

1

2

Enter the list of numbers (sep by space): 4 5 6

Out[20]: [4, 5, 6] In [19]:

res

= [(

a

,

b

,

c

)

**for**

a

**in**

my\_list

**for**

b

**in**

my\_list

**for**

c

**in**

my\_list

]

res

1

2

[(4, 4, 4), (4, 4, 5),

(4, 4, 6), (4, 5, 4),

(4, 5, 5), (4, 5, 6),

(4, 6, 4), (4, 6, 5),

(4, 6, 6), (5, 4, 4),

(5, 4, 5), (5, 4, 6),

(5, 5, 4), (5, 5, 5),

(5, 5, 6), (5, 6, 4),

(5, 6, 5), (5, 6, 6),

(6, 4, 4), (6, 4, 5),

(6, 4, 6), (6, 5, 4),

(6, 5, 5), (6, 5, 6),

(6, 6, 4), (6, 6, 5),

(6, 6, 6)]

# 15) Prime numbers within range

In

[ ]:

Enter start of the range : 2

Enter end of the range : 10

s

=

eval

(

input

(

'Enter start of the range : '

))

e

=

eval

(

input

(

'Enter end of the range : '

))

**for**

i

**in**

range

(

s

,

e

**+**

1

):

**if**

i

**>**

1

:

**for**

n

**in**

range

(

2

, (

i

**//**

2

**+**

1

)):

**if**

(

i

**%**

n

)==

0

:

**break**

**else**

:

print

(

i

,

end

=

' '

)

1

2

3

4

5

6

7

8

9

2 3 5 7

# 16) Count people aged between 60 and 90

In

[ ]:

Enter ages (seperated by spaces): 56 78 97 67

Employees in age group 60 - 90: 2

ageRange

= {(

60

,

90

):

0

}

ages

=

map

(

int

,

input

(

"Enter ages (seperated by spaces): "

).

split

())

**for**

age

**in**

ages

:

**if**

(

60

**<**

=

age

**<**

=

90

):

ageRange

[(

60

,

90

)]

+=

1

**for**

k

,

v

**in**

ageRange

.

items

():

lower

,

upper

=

k

print

(

f"Employees in age group {lower} - {upper}: {v}"

)

1

2

3

4

5

6

7

8

9

10

# 17 ) Transpose Matrix

In

[ ]:

7]

, 4,

[1

8]

, 5,

[2

X

= [[

1

,

2

,

3

]

,

[

4

,

5

,

6

,

]

[

7

,

8

,

9

]]

result

= [[

0

,

0

,

0

]

,

[

0

,

0

,

0

]

,

[

0

,

0

,

0

]]

**for**

i

**in**

range

(

len

(

X

)):

**for**

j

**in**

range

(

len

(

X

[

0

])):

result

[

j

][

i

]

=

X

[

i

][

j

]

**for**

r

**in**

result

:

print

(

r

)

1

2

3

4

5

6

7

8

9

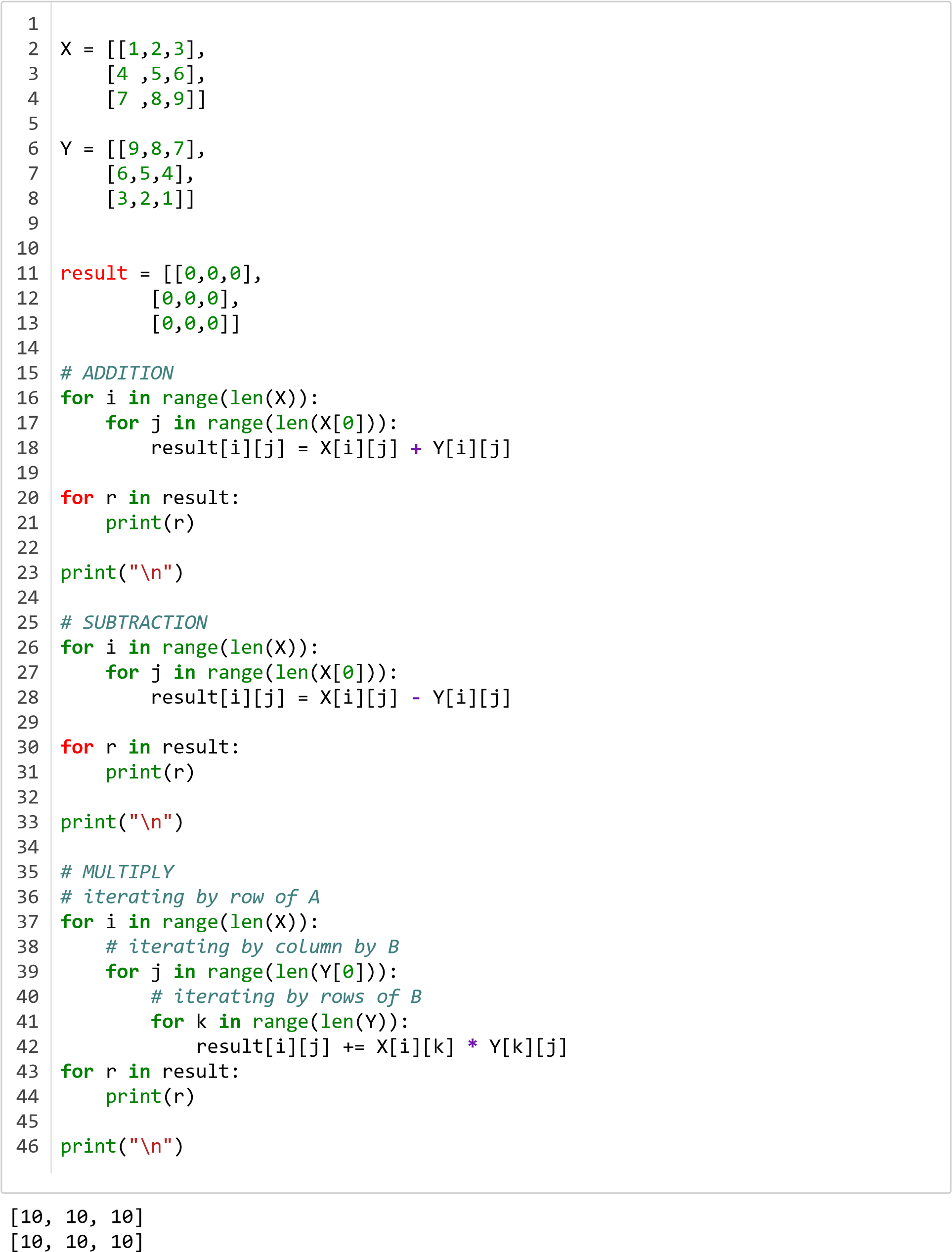
10

11

12

[3, 6, 9]

# 18) Matrix Operation



[10, 10, 10]

[-8, -6, -4]

[-2, 0, 2]

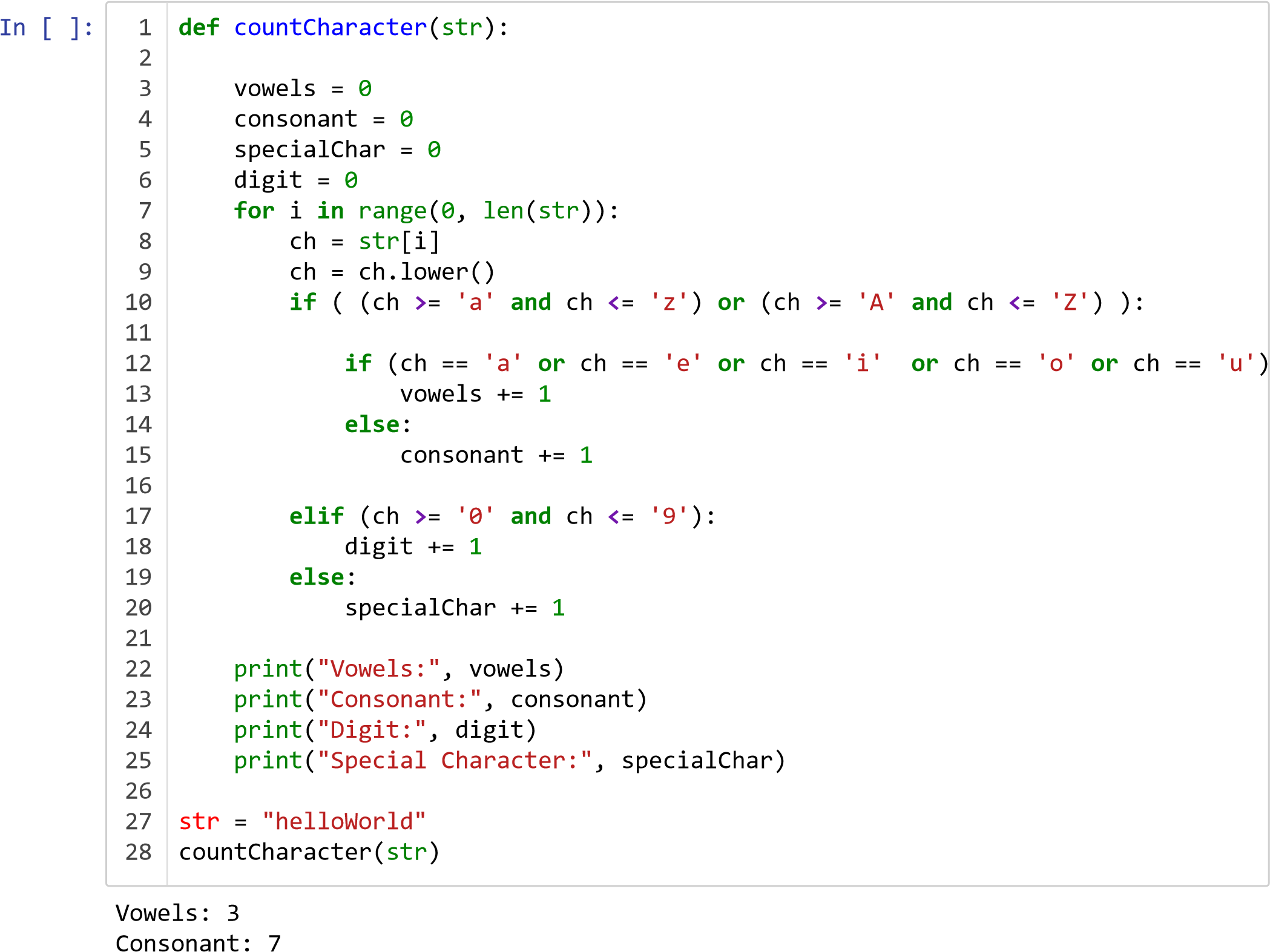
[4, 6, 8]

[22, 18, 14]

[82, 69, 56]

[142, 120, 98]

# 19) Vowels



Digit: 0

Special Character: 0

1. **) palindrome or not?**

In

[ ]:

Input: hello hello

The 'hello hello' is not palindrome.

Pal

=

**lambda**

s

:

s

==

s

[::

**-**

1

]

S

=

input

(

"Input: "

)

print

(

"The '{}' is{}palindrome."

.

format

(

S

,

' '

**if**

(

Pal

(

S

))

**else**

' not '

))

1

2

3

4

1. **) list of numbers:**

1) Insert an element 2) delete an element 3) sort the list 4) delete

Initial stack

[

'a', 'b', 'c'

]

stack

= []

stack

.

append

(

'a'

)

stack

.

append

(

'b'

)

stack

.

append

(

'c'

)

print

(

'Initial stack'

)

print

(

stack

)

print

(

'\nElements popped from stack:'

)

print

(

stack

.

pop

())

print

(

stack

.

pop

())

print

(

stack

.

pop

())

print

(

'\nStack after elements are popped:'

)

print

(

stack

)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

Elements popped from stack:

c b a

Stack after elements are popped: []

In [ ]:

ME

= [

"Lesson"

,

"was "

,

"Short?"

]

print

(

ME

)

1

2

['Lesson', 'was ', 'Short?']

**delete**

In [ ]:

ME2

= [

"Lesson"

,

"Felt"

,

"Long?"

]

print

(

ME2

)

1

2

In

[ ]:

In

[ ]:

[

'Lesson', 'Felt', 'Long?'

]

**del**

ME2

ME2

1

1

**---------------------------------------------------------------------------** **NameError** Traceback (most recent call last)

**<ipython-input-5-161024c480b6>** in <module>**()**

**----> 1** ME2

**NameError**: name 'ME2' is not defined

In [ ]:

ME

.

remove

(

"Lesson"

)

print

(

ME

)

1

2

['was ', 'Short?', 'Nope!']

In [ ]:

ME

.

clear

()

print

(

ME

)

1

2

In

[ ]:

In

[ ]:

[]

Out[21]:

[

'a', 'e', 'o', 'u'

]

*#Sort*

vowels

.

sort

()

vowels

1

2

1

In

[ ]:

6]

, 2, 3, 4, 5,

[1

*# Combine*

test1

= [

1

,

2

,

3

]

test2

= [

4

,

5

,

6

]

test3

= [

**\***

test1

,

**\***

test2

]

print

(

test3

)

1

2

3

4

5

6

# 23 ) sort words in alphabetical order

In

[ ]:

*#25*

str

=

input

(

"Enter a string: "

)

words

= [

word

.

lower

()

**for**

word

**in**

str

.

split

()]

words

.

sort

()

print

(

"The sorted words are:"

)

**for**

word

**in**

words

:

print

(

word

)

1

2

3

4

5

6

7

8

9

Enter a string: Hello who is this typing for the lab file

The sorted words are:

file for hello is lab the this typing who

# 24 ) Linear Search

Element found at index: 3

**def**

Linear

(

array

,

n

,

x

):

**for**

i

**in**

range

(

0

,

n

):

**if**

(

array

[

i

]

==

x

):

**return**

i

**return**

**-**

1

array

= [

2

,

4

,

0

,

1

,

9

]

x

=

1

n

=

len

(

array

)

result

=

Linear

(

array

,

n

,

x

)

**if**

(

result

==

**-**

1

):

print

(

"Element not found"

)

**else**

:

print

(

"Element found at index: "

,

result

)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

# 25 Linear search - sorted

In

[ ]:

9]

, 1, 2, 4,

[0

Element found at index: 1

**def**

LinearOr

(

array

,

n

,

x

):

**for**

i

**in**

range

(

0

,

n

):

**if**

(

array

[

i

]

==

x

):

**return**

i

**return**

**-**

1

array

= [

2

,

4

,

0

,

1

,

9

]

array

.

sort

()

print

(

array

)

x

=

1

n

=

len

(

array

)

result

=

LinearOr

(

array

,

n

,

x

)

**if**

(

result

==

**-**

1

):

print

(

"Element not found"

)

**else**

:

print

(

"Element found at index: "

,

result

)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

# 26 ) Maintain practical note book as per their serial numbers - dict

In

[13]:

Enter number of books:

5

n

=

int

(

input

(

"Enter number of books: \n"

))

d

={}

**for**

i

**in**

range

(

1

,

n

**+**

1

):

key

=

i

value

=

input

(

"Enter Name :"

)

d

.

update

({

key

:

value

})

print

(

d

)

1

2

3

4

5

6

7

8

Enter Name :Deep Learning with CUDA

Enter Name :Spacy

Enter Name :AutoNLP

Enter Name :CS Manual

Enter Name :The AWK and Sed

{1: 'Deep Learning with CUDA', 2: 'Spacy', 3: 'AutoNLP', 4: 'CS Manual', 5: 'The A WK and Sed'}

# 27 ) Dictionary

1) Insert 2) delete 3) change

In

[ ]:

dict1

={

'Name'

:

'Me'

,

"Age"

:

1000

,

"Country"

:

"India"

}

print

(

dict1

)

*#insert*

dict1

[

'Gender'

]=

'Male'

print

(

dict1

)

dict1

[

'Initial'

]=

'A'

print

(

dict1

)

*# delete item*

dict1

.

pop

(

'Country'

)

print

(

dict1

)

1

2

3

4

5

6

7

8

9

10

11

12

{'Name': 'Me', 'Age': 1000, 'Country': 'India'}

{'Name': 'Me', 'Age': 1000, 'Country': 'India', 'Gender': 'Male'}

{'Name': 'Me', 'Age': 1000, 'Country': 'India', 'Gender': 'Male', 'Initial': 'A'}

{'Name': 'Me', 'Age': 1000, 'Gender': 'Male', 'Initial': 'A'}

In

[ ]:

In

[ ]:

*#delete entire*

**del**

dict1

dict1

*# It will give error*

1

2

1

**---------------------------------------------------------------------------**

**NameError** Traceback (most recent call last)

**<ipython-input-57-e36219336d90>** in <module>**()**

**----> 1** dict1

**NameError**: name 'dict1' is not defined

**28) Number is in a given range using functions.**

5

**def**

count

(

list1

,

l

,

r

):

c

=

0

**for**

x

**in**

list1

:

**if**

x

**>**

=

l

**and**

x

**<**

=

r

:

c

+=

1

**return**

c

list1

= [

1

,

2

,

3

,

4

,

5

,

6

,

7

,

8

]

l

=

4

r

=

8

print

(

count

(

list1

,

l

,

r

))

1

2

3

4

5

6

7

8

9

10

11

# 29 ) Function that accepts a string and calculates upper and lower case

In

[ ]:

Please enter the sentence : Hello World

Upper : 2

str1

=

input

(

'Please enter the sentence : '

)

upper

=

lower

=

0

**for**

i

**in**

range

(

len

(

str1

)):

**if**

(

str1

[

i

].

isupper

()):

upper

+=

1

**elif**

(

str1

[

i

].

islower

()):

lower

+=

1

print

(

f"Upper :\t{upper} \nLower :\t{lower}"

)

1

2

3

4

5

6

7

8

Lower : 8

# 30 ) Max of three

In [ ]:

a

=(

input

(

"Enter List Elements:"

)).

split

()

l

=

list

(

map

(

int

,

a

))

1

2

Enter List Elements:2 5 6

In [ ]:

print

(

max

(

l

))

1

6

**31 ) Multiply all the numbers in a list using functions.**

In

[ ]:

Output: 10000

total

=

1

A

= [

1

,

10

,

10

,

10

,

10

]

**for**

i

**in**

range

(

0

,

len

(

A

)):

total

=

total

**\***

A

[

i

]

print

(

"Output: "

,

total

)

1

2

3

4

5

6

7

# 32 ) Fibonacci sequence - 33 Factorial

using recursion.

In

[ ]:

Enter a number: 4

3

th term of fibonacci series:

4

*# Fibonacci*

**def**

fib

(

n

):

*#Recursive*

**if**

n

**<**

=

0

:

**return**

0

**if**

n

==

1

:

**return**

1

**return**

fib

(

n

**-**

1

)

**+**

fib

(

n

**-**

2

)

*# Factorial*

**def**

factorial

(

n

):

**if**

n

**<**

=

1

:

**return**

1

**return**

n

**\***

factorial

(

n

**-**

1

)

num

=

int

(

input

(

"Enter a number: "

))

print

(

"{}th term of fibonacci series: "

.

format

(

num

)

,

fib

(

num

))

print

(

"Factorial of {}: "

.

format

(

num

,

)

factorial

(

num

))

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

Factorial of 4: 24

1. **) create a module of factorial in Python.**

In

[ ]:

*#36*

*#Importing factorial function created previously*

**from**

facto

**import**

factorial

print

(

factorial

(

4

))

1

2

3

4

5

24

Out[45]: '\nfacto.py\n\ndef factorial(n):\n if n <= 1:\n return 1\n return n\*f actorial(n-1)\n\nnum = int(input("Enter a number: "))\nprint("{}th term of fibona cci series: ".format(num), fib(num))\nprint("Factorial of {}: ".format(num), facto rial(num))'

1. **) class named Rectangle, constructed by a length& width, also design a method which will compute the area of a rectangle.**

Enter the Length of the Rectangle : 6

Enter the Breadth of the Rectangle : 7

*# 35*

**class**

Rectangle

(

Shape

):

length

=

0.0

breadth

=

0.0

**def**

\_\_init\_\_

(

self

,

l

,

br

,

co

):

self

.

length

=

l

self

.

breadth

=

br

**def**

Calc\_Area

(

self

):

Area

= (

self

.

length

)

**\***

(

self

.

breadth

)

**return**

Area

a

=

eval

(

input

(

" Enter the Length of the Rectangle : "

))

b

=

eval

(

input

(

" Enter the Breadth of the Rectangle : "

))

ob1

=

Rectangle

(

a

,

b

,

c

)

print

(

"Area of the Rectangle : "

,

ob1

.

Calc\_Area

())

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

Area of the Rectangle : 42

1. **) Class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.**

In

[ ]:

**class**

Circle

:

radius

=

0.0

**def**

\_\_init\_\_

(

self

,

r

):

self

.

radius

=

r

**def**

Get\_Radius

(

self

):

x

=

self

.

radius

**return**

x

**def**

Calc\_Area

(

self

):

Area

= (

3.14

)

**\***

(

self

.

radius

)

**\***

(

self

.

radius

)

**return**

Area

**def**

Calc\_Peri

(

self

):

Peri

= (

2

)

**\***

(

3.14

)

**\***

(

self

.

radius

)

**return**

Peri

a

=

eval

(

input

(

'Enter Radius of your Choice : '

))

ob1

=

Circle

(

a

)

print

(

"The Area of the Circle Is : "

,

ob1

.

Calc\_Area

())

print

(

"The Area of the Circle Is : "

,

ob1

.

Calc\_Peri

())

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

Enter Radius of your Choice : 3

The Area of the Circle Is : 28.259999999999998

The Area of the Circle Is : 18.84

# 37 ) Class to reverse a string word by word

In

[21]:

wow it worked

>>>> :

The string is : wow it worked

**class**

reverser

():

**def**

reverse\_string

(

self

,

my\_string

):

**if**

len

(

my\_string

==

)

0

:

**return**

my\_string

**else**

:

**return**

self

.

reverse\_string

(

my\_string

[

1

:])

**+**

my\_string

[

0

]

obj

=

reverser

()

my\_str

=

str

(

input

(

">>>> : "

))

print

(

"The string is :"

,

my\_str

)

print

(

"The reversed string is :"

)

print

(

obj

.

reverse\_string

(

my\_str

))

1

2

3

4

5

6

7

8

9

10

11

12

The reversed string is : dekrow ti wow

# 38 ) Read a file

In

[ ]:

Enter file name : h

dlroWolleH

x

=

input

(

"Enter file name : "

)

f

=

open

(

x

,

"w"

)

f

.

write

(

"HelloWorld"

)

f

=

open

(

x

,

"r"

)

**for**

i

**in**

f

:

print

(

i

[

: :

**-**

1

])

1

2

3

4

5

6

1. **) Read first n lines of a text file.**

In

[10]:

Enter number of lines : 3

[

]

'\n', " It's me\n", ' Yup me\n'

f

=

open

(

"h.txt"

,

"w"

)

f

.

write

(

"\n It's me\n"

)

f

.

write

(

" Yup me\n"

)

f

.

write

(

" OK\n"

)

f

.

write

(

" Bye\n"

)

f

.

close

()

N

=

int

(

input

(

"Enter number of lines : "

))

fptr

=

open

(

"h.txt"

,

"r"

)

line

= (

fptr

.

readlines

()[

0

:

N

])

print

(

line

)

1

2

3

4

5

6

7

8

9

10

11