

**1.) Add two numbers.**

```
a=4
b=99
c=a+b
print(c)
```

 103**2.)subtract two numbers.**


```
a=99
b=48
c=a-b
print(c)
```

 51**3.)multiply two numbers.**


```
a=99
b=38
c=a*b
print(c)
```

 3762**4.)divide two numbers.**

```
a=40
b=35
c=a/b
print(c)
```

 1.1428571428571428**5.)add,multiply,subtract and divide two numbers.**

```
a=96
b=69
c=a+b
d=a-b
e=a*b
f=a/b
print(c,d,e,f)
```

 165 27 6624 1.391304347826087**6.)Convert hours into minutes.**

```
hours=40
minutes=hours*60
print(minutes)
```

 2400**7.)Convert minutes into hours.**

```
minutes=4800
hours=minutes/60
print(hours)
```

 80.0**8.)Convert dollars into Rs. Where 1 \$ = 48 Rs.**

```
dollar=9999
rs=dollar*48
```

```
print(rs)
```

```
↔ 479952
```

9.) Convert Rs. into dollars where 1 \$ = 48 Rs.

```
rs=9999999
dollar=rs/48
print(dollar)
```

```
↔ 208333.3125
```

10.) Convert dollars into pound where 1 \$ = 48 Rs. And 1 pound = 70 Rs.

```
dollar=8888
rs=dollar*48
pound=rs/70
print(pound)
```

```
↔ 6094.628571428571
```

11.) Convert grams into kg.

```
grams=99000
kg=grams/1000
print(kg)
```

```
↔ 99.0
```

12.) Convert kgs into grams.

```
kgs=8899
grams=kgs*1000
print(grams)
```

```
↔ 8899000
```

13.) Convert bytes into KB, MB and GB.

```
bytes=898989
kb=bytes/1024
mb=kb/1024
gb=mb/1024
print(kb,mb,gb)
```

```
↔ 877.9189453125 0.8573427200317383 0.0008372487500309944
```

14.) Convert celcius into Fahrenheit.  $F = (9/5 * C) + 32$

```
celcius=99
fahrenheit=(9/5*celcius)+32
print(fahrenheit)
```

```
↔ 210.20000000000002
```


15.) Convert Fahrenheit into celcius.  $C = 5/9 * (F - 32)$

```
fahrenheit=999
celcius=5/9*(fahrenheit-32)
print(celcius)
```

```
↔ 537.2222222222223
```


16.) Calculate interest where  $I = PRN/100$ .

```
p=100000
r=1000
n=1
i=p*r*n/100
print(i)
```

 1000000.0


17.) Calculate area & perimeter of a square.  $A = L^2$ ,  $P = 4L$

```
l=96
a=l**2
p=4*l
print(a,p)
```

 9216 384


18.) Calculate area & perimeter of a rectangle.  $A = L*B$ ,  $P = 2(L+B)$

```
l=92
b=89
a=l*b
p=2*(l+b)
print(a,p)
```

 8188 362


19.) Calculate area of a circle.  $A = \frac{22}{7} * R * R$

```
r=72
a=22/7*r*r
print(a)
```

 16292.571428571428


20.) Calculate area of a triangle.  $A = H*L/2$

```
h=85
l=49
a=h*l/2
print(a)
```

 2082.5

21.) Calculate net salary where net salary = gross salary + allowance – deduction. Allowances are 10% while deductions are 3% of gross salary.

```
gs=999999
a=gs*10/100
d=gs*3/100
ns=gs+a-d
print(ns)
```

 1069998.93

22.) Calculate net sales where net sales = gross sales – 10% discount of gross sales.

```
gs=999999
d=gs*10/100
ns=gs-d
print(ns)
```

 899999.1

23.) Calculate average of three subjects along with their total.

```
maths=98
chemistry=99
physics=97
total=maths+chemistry+physics
average=total/3
print(total,average)
```

 294 98.0

24.) Swap two values.

```
v1=9988
v2=8899
temp=v1
v1=v2
v2=temp
print(v1,v2)
```

↔ 8899 9988

### ***If-else conditionals.***

#### **1.) Print largest and smallest values out of two.**

```
num1= 32
num2 = 54
if (num1>num2):
    print("Num1 is largest values")
else:
    print("num1 is smallest")
if(num2>num1):
    print("num2 is largest values")
else:
    print("num2 is smallest")
```

↔ num1 is smallest  
num2 is largest values

#### **2.) Print largest and smallest values out of three.**

```
num1 = 99
num2 = 999
num3 = 9999
if(num1>num2 and num1 >num3):
    print("num1 is largest values")
else:
    print("num1 is smallest")
if(num2>num3 and num2>num1):
    print("num2 is largest values")
else:
    print("num2 is smallest")
if(num3>num1 and num3>num2):
    print("num3 is largest")
else:
    print("num3 is smallest")
```

↔ num1 is smallest  
num2 is smallest  
num3 is largest

#### **3.) Check whether a given number is odd or even**

```
num = 399
if(num%2==0):
    print("Even")
else:
    print("odd")
```

↔ odd

#### **4.) Check whether a given number is divisible by 10 or not.**

```
num = 459
if(num%10==0):
    print("it is divisiable by 10")
else:
    print("not it is not divisiable by 10")
```

↔ not it is not divisiable by 10

#### **5.) Accept age of a person. If age is less than 18, print minor otherwise Major.**

```
age = int(input("Enter the age of person:"))
if(age<18):
    print("minor")
```

```
else:
    print("major")
```

```
↵ Enter the age of person:19
major
```

6.) Accept a number from the user. And print number of digits in it.

```
number = input("Enter the numbers:")
print(len(number))
```

```
↵ Enter the numbers:999
3
```

7.) Accept a year value from the user. Check whether it is a leap year or not.

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

```
↵ Enter the year:2007
it is not leap year
```

8.) Check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard. A triangle is valid if the sum of all the three angles is equal to 180 degrees.

```
side1 = int(input("Enter the 1st side:"))
side2 = int(input("Enter the 2nd side:"))
side3 = int(input("Enter the 3rd side:"))
if(side1 + side2 + side3 == 180):
    print("triangle is valid")
else:
    print("triangle is invalid")
```

```
↵ Enter the 1st side:80
Enter the 2nd side:50
Enter the 3rd side:50
triangle is valid
```

9.) Print absolute value of a given number.

```
num = -9090
print(abs(num))
```

```
↵ 9090
```

10.) Given the length and breadth of a rectangle, write a program to find whether the area of the rectangle is greater than its perimeter.

```
length = 890
breath = 89
area = length * breath
perimeter = 2*(length + breath)
if(area > perimeter):
    print('area is greater than perimeter')
else:
    print('area is smaller than perimeter')
```

```
↵ area is greater than perimeter
```

11.) Given three points (x1,y1), (x2,y2) and (x3,y3), check if all the three points fall on one straight line

```
x1 = 99
x2 = 70
x3 = 96
y1 = 35
y2 = 44
y3 = 72
a = (x1*(y2-y3))+(x2*(y3-y1))+(x3*(y1-y2))
if (a == 0):
    print('all the points fall on one straight line')
```

```
else:
    print("all the points dont fall on one stright line")
```

→ all the points dont fall on one stright line

12.) Given the coordinates (x,y) of center of a circle and its radius, determine whether a point lies inside the circle, on the circle or outside the circle. (Hint: Use sqrt(), pow())

```
import math
circle_m = 99
circle_n = 198
radius = 99
point_m = 99
point_n = 99
distance = math.sqrt(math.pow(point_m - circle_m, 2) +
math.pow(point_n - circle_n, 2))
if distance < radius:
    print(f"The point ({point_m}, {point_n}) is inside the circle.")
elif distance == radius:
    print(f"The point ({point_m}, {point_n}) is on the circle.")
else:
    print(f"The point ({point_m}, {point_n}) is outside the circle.")
```

→ The point (99, 99) is on the circle.

13.) Convert number 0 to 19 to its equivalent words. E.g. 0 → zero, 19 → nineteen.

```
numbers = 19
numbers_word = [
    "zero", "one", "two", "three", "four", "five", "six", "seven",
    "eight", "nine",
    "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen",
    "sixteen",
    "seventeen", "eighteen", "nineteen"
]
if 0 <= numbers <= 19:
    print(f"The number {numbers} is written as'{numbers_word[numbers]}'.")
else:
    print("Number is out of range (0-19).")
```

→ The number 19 is written as'nineteen'.

14.) Accept marks of three subjects. Print total and average along with whether a candidate has passed or fail. If student secures <= 39 marks in any subject, consider him as fail. Also assigned a subject wise grade based on the following table: bold text

```
sub1 = int(input("Enter the maths marks:"))
sub2 = int(input("Enter the chemistry marks:"))
sub3 = int(input("Enter the physics marks:"))
sum = ( sub1 + sub2 + sub3 ) / 3
if(sub1 <= 39 or sub2 <=39 or sub3<=39):
    print("Fail")
elif(sum>= 40 and sum <=44 ):
    print("P")
elif(sum>= 45 and sum <=49 ):
    print("C")
elif(sum>= 50 and sum <=54 ):
    print("B")
elif(sum>= 55 and sum <=59 ):
    print("B+")
elif(sum>= 60 and sum <= 69 ):
    print("A")
elif(sum>= 70 and sum <= 79):
    print("A+")
elif(sum>= 80 and sum <=100 ):
    print("O")
else:
    print("good")
```

→ Enter the maths marks:87  
Enter the chemistry marks:78  
Enter the physics marks:99  
0

loops:-

**1) Print all alphabets in upper case and in lower case.**

```
print("Upper case alphabets:")
for l in range(65, 91): # ASCII values for A to Z
    print(chr(l), end=" ")
print()

print("Lower case alphabets:")
for l in range(97, 123): # ASCII values for a to z
    print(chr(l), end=" ")
print()
```

```
↗ Upper case alphabets:
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Lower case alphabets:
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

**2) Print a multiplication table of a given number.**

```
def print_multiplication_table(number):
    print(f"Multiplication Table for {number}:")
    for m in range(1, 11):
        print(f"{number} x {m} = {number * m}")

number = int(input("Enter a number: "))
print_multiplication_table(number)
```

```
↗ Enter a number: 999
Multiplication Table for 999:
999 x 1 = 999
999 x 2 = 1998
999 x 3 = 2997
999 x 4 = 3996
999 x 5 = 4995
999 x 6 = 5994
999 x 7 = 6993
999 x 8 = 7992
999 x 9 = 8991
999 x 10 = 9990
```

**3) Count no. of alphabets and no. of digits in any given string.**

```
def count_digits_of_alphabets(input_string):
    alphabt_of_counte = 0
    digit_of_count = 0

    for char in input_string:
        if char.isalpha():
            alphabt_of_counte += 1
        elif char.isdigit():
            digit_of_count += 1

    return alphabt_of_counte, digit_of_count
input_of_string = input("Enter a string: ")
alphabets, digits = count_digits_of_alphabets(input_of_string)

print(f"Number of alphabets: {alphabets}")
print(f"Number of digits: {digits}")
```

```
↗ Enter a string: hello friends 9999
Number of alphabets: 12
Number of digits: 4
```

**4) Check whether a given number is prime, is perfect, is Armstrong, is palindrome, is automorphic.**

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

def is_perfect(num):
    if num <= 0:
        return False
    sum_of_divisors = 0
    for b in range(1, num):
```

```

    if num % b == 0:
        sum_of_divisors += b
    return sum_of_divisors == num

def is_armstrong(num):
    num_str = str(num)
    num_length = len(num_str)
    sum_of_powers = 0
    for digit in num_str:
        sum_of_powers += int(digit) ** num_length
    return sum_of_powers == num

def is_palindrome(num):
    num_str = str(num)
    return num_str == num_str[::-1]

def is_automorphic(num):
    num_str = str(num)
    square_str = str(num ** 2)
    return square_str.endswith(num_str)

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

print(f"Is {number} prime? {'Yes' if is_prime(number) else 'No'}")
print(f"Is {number} perfect? {'Yes' if is_perfect(number) else 'No'}")
print(f"Is {number} Armstrong? {'Yes' if is_armstrong(number) else 'No'}")
print(f"Is {number} palindrome? {'Yes' if is_palindrome(number) else 'No'}")
print(f"Is {number} automorphic? {'Yes' if is_automorphic(number) else 'No'}")

```

```

↗ Enter a number: 98989
Is 98989 prime? No
Is 98989 perfect? No
Is 98989 Armstrong? No
Is 98989 palindrome? Yes
Is 98989 automorphic? No

```

#### 5) Generate all Pythagorean Triplets with side length <= 30.

```

def generate_pythagorean_triplets(limit):
    triplets = []
    for a in range(1, limit + 1):
        for b in range(a, limit + 1):
            c = (a ** 2 + b ** 2) ** 0.5
            if c.is_integer() and c <= limit:
                triplets.append((a, b, int(c)))
    return triplets

limit = 30
triplets = generate_pythagorean_triplets(limit)

print(f"Pythagorean triplets with side lengths <= {limit}:")
for triplet in triplets:
    print(triplet)

```

```

↗ Pythagorean triplets with side lengths <= 30:
(3, 4, 5)
(5, 12, 13)
(6, 8, 10)
(7, 24, 25)
(8, 15, 17)
(9, 12, 15)
(10, 24, 26)
(12, 16, 20)
(15, 20, 25)
(18, 24, 30)
(20, 21, 29)

```

#### 6) Print 24 hours of day with suitable suffixes like AM, PM, Noon and Midnight.



```
def print_24_hours_with_suffixes():
    for hour in range(0, 24):
        if hour == 0:
            print("12 Midnight")
        elif hour < 12:
            print(f"{hour} AM")
        elif hour == 12:
            print("12 Noon")
        else:
            print(f"{hour - 12} PM")
```

```
print_24_hours_with_suffixes()
```

```
↩ 12 Midnight
1 AM
2 AM
3 AM
4 AM
5 AM
6 AM
7 AM
8 AM
9 AM
10 AM
11 AM
12 Noon
1 PM
2 PM
3 PM
4 PM
5 PM
6 PM
7 PM
8 PM
9 PM
10 PM
11 PM
```

### 7) Print nCr and nPr.

```
def factorial(n):
    result = 1
    for y in range(1, n + 1):
        result *= y
    return result

def nCr(n, r):
    return factorial(n) // (factorial(r) * factorial(n - r))

def nPr(n, r):
    return factorial(n) // factorial(n - r)

2
n = int(input("Enter the value of n: "))
r = int(input("Enter the value of r: "))

print(f"{n}C{r} = {nCr(n, r)}")
print(f"{n}P{r} = {nPr(n, r)}")
```

```
↩ Enter the value of n: 5
Enter the value of r: 2
5C2 = 10
5P2 = 20
```

### 8) Print factorial of a given number.

```
def factorial(num):
    result = 1
    for u in range(1, num + 1):
        result *= u
    return result

number = int(input("Enter a number: "))
print(f"The factorial of {number} is {factorial(number)}")
```

```
↩ Enter a number: 26
The factorial of 26 is 403291461126605635584000000
```

### 9) Print N natural nos. in reverse.

```
def print_natural_numbers_reverse(N):
    for w in range(N, 0, -1):
        print(w, end=" ")
    print()

N = int(input("Enter the value of N: "))
print(f"The first {N} natural numbers in reverse order are:")
print_natural_numbers_reverse(N)
```

Enter the value of N: 48  
 The first 48 natural numbers in reverse order are:  
 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2

#### 10) Generate N numbers of Fibonacci series.

```
def generate_fibonacci_series(m):
    fibonacci_series = []
    a, b = 0, 1
    for _ in range(m):
        fibonacci_series.append(a)
        a, b = b, a + b
    return fibonacci_series

m = int(input("Enter the value of m: "))
fibonacci_series = generate_fibonacci_series(m)

print(f"The first {m} numbers of the Fibonacci series are:")
print(fibonacci_series)
```

Enter the value of m: 9  
 The first 9 numbers of the Fibonacci series are:  
 [0, 1, 1, 2, 3, 5, 8, 13, 21]

#### 11) Calculate sin(x); x is a radian value. The formula is as under: $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$ (hint: degrees can be converted into radians by $3.14159 / 180$ .)

```
import math

def factorial(m):
    result = 1
    for i in range(1, m + 1):
        result *= i
    return result

def sin(x, terms=10):
    sin_x = 0
    for m in range(terms):
        term = ((-1) ** m) * (x ** (2 * m + 1)) / factorial(2 * m + 1)
        sin_x += term
    return sin_x

angle_degrees = float(input("Enter the angle in degrees: "))
angle_radians = angle_degrees * (math.pi / 180)

print(f"sin({angle_degrees} degrees) = {sin(angle_radians)}")
```

Enter the angle in degrees: 90  
 sin(90.0 degrees) = 1.0

#### Write following programs considering list in mind:

1. Create a list of 5 odd integers using random nos. Similarly create a list of 4 even integers using random nos. Replace the third element of odd integers with a list of 4 even integers. Flatten, sort and print the list. Provide appropriate message at each stage.

```
import random

odd_integers = [random.choice(range(1, 101, 2)) for _ in range(5)]
print("List of 5 odd integers: ", odd_integers)

even_integers = [random.choice(range(0, 101, 2)) for _ in range(4)]
print("List of 4 even integers: ", even_integers)

odd_integers[2] = even_integers
print("Replaced the third element of odd integers with a list of 4 even integers: ", odd_integers)

flattened_of_list = []
```

```

for item in odd_integers:
    if isinstance(item, list):
        flattened_of_list.extend(item)
    else:
        flattened_of_list.append(item)
print("Flattened list: ", flattened_of_list)

sorted_list = sorted(flattened_of_list)
print("Sorted list: ", sorted_list)

```

```

→ List of 5 odd integers: [91, 3, 69, 87, 71]
List of 4 even integers: [16, 36, 70, 58]
Replaced the third element of odd integers with a list of 4 even integers: [91, 3, [16, 36, 70, 58], 87, 71]
Flattened list: [91, 3, 16, 36, 70, 58, 87, 71]
Sorted list: [3, 16, 36, 58, 70, 71, 87, 91]

```

2.) Generate 20 random integers and store them in a list. Accept a number from the user and print position of all occurrences of that number in the list.

```

import random

random_integers = [random.randint(1, 100) for _ in range(20)]
print("List of 20 random integers:", random_integers)

users_input = int(input("Enter a number to find its occurrences: "))

positions = [index for index, value in enumerate(random_integers) if value == users_input]

if positions:
    print(f"Number {users_input} found at positions: {positions}")
else:
    print(f"Number {users_input} not found in the list.")

```

```

→ List of 20 random integers: [96, 100, 30, 23, 98, 53, 65, 11, 40, 57, 65, 70, 89, 99, 43, 14, 30, 37, 20, 13]
Enter a number to find its occurrences: 98
Number 98 found at positions: [4]

```

3.) Generate 50 random numbers in the range 1 and 30. Remove all duplicate values from the list.

```

import random

random_num = [random.randint(1, 30) for _ in range(50)]
print("List of 50 random numbers:", random_num)

unique_num = list(set(random_num))
print("List after removing duplicates:", unique_num)

```

```

→ List of 50 random numbers: [4, 18, 19, 12, 17, 14, 22, 10, 24, 16, 16, 18, 23, 21, 7, 15, 23, 22, 20, 23, 4, 27, 5, 20, 21, 23, 30,
List after removing duplicates: [4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 30]

```

4.) Generate 30 random numbers and put them in a list. Create two more lists – one containing only +ve numbers and another with –ve nos.

```

import random

random_num = [random.randint(-50, 50) for _ in range(30)]
print("List of 30 random numbers:", random_num)

positive_num = [num for num in random_num if num > 0]
negative_num = [num for num in random_num if num < 0]

print("List of positive numbers:", positive_num)
print("List of negative numbers:", negative_num)

```

```

→ List of 30 random numbers: [50, 44, 35, -12, 50, 13, 47, 5, 44, -43, -12, -28, -3, 31, 19, -49, -15, -32, 38, 36, 7, -11, -25, -19,
List of positive numbers: [50, 44, 35, 50, 13, 47, 5, 44, 31, 19, 38, 36, 7, 34, 25, 6, 7, 50]
List of negative numbers: [-12, -43, -12, -28, -3, -49, -15, -32, -11, -25, -19, -40]

```

5.) A list contains 5 strings. Convert all these strings to uppercase.

```

list_of_strings = ["hello", "world", "python", "programming", "rocks"]

uppercase_strings = [string.upper() for string in list_of_strings]

print("List of uppercase strings:", uppercase_strings)

```

↩ List of uppercase strings: ['HELLO', 'WORLD', 'PYTHON', 'PROGRAMMING', 'ROCKS']

#### 6. Convert list of temperatures in Fahrenheit degrees to equivalent Celsius degrees.

```
fahrenheit_temp = [32, 68, 104, 50, 77]

celsius_temp = [(temp - 32) * 5/9 for temp in fahrenheit_temp]

print("List of temperatures in Celsius:", celsius_temp)
```

↩ List of temperatures in Celsius: [0.0, 20.0, 40.0, 10.0, 25.0]

#### 7. Write a menu-driven program to implement the stack data structure.

```
stacks = []

def push(item):
    stacks.append(item)
    print(f"Item {item} pushed onto stacks.")

def pop():
    if stacks:
        item = stacks.pop()
        print(f"Item {item} popped from stacks.")
        return item
    else:
        print("Stack is empty. Cannot pop item.")
        return None

def display():
    print("Stack contents:", stacks)

while True:
    print("\nMenu:")
    print("1. Push")
    print("2. Pop")
    print("3. Display stacks")
    print("4. Exit")

    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        item = input("Enter the item to push: ")
        push(item)
    elif choice == '2':
        pop()
    elif choice == '3':
        display()
    elif choice == '4':
        print("Exiting program.")
        break
    else:
        print("Invalid choice. Please enter a number between 1 and 4.")
```

↩

```
Menu:
1. Push
2. Pop
3. Display stacks
4. Exit
Enter your choice (1-4): 4
Exiting program.
```

#### 8. Write a menu-driven program to implement the Queue data structure.

```
queue = []

def enqueue(item):
    queue.append(item)
    print(f"Item {item} added to the queue.")

def dequeue():
    if queue:
        item = queue.pop(0)
        print(f"Item {item} removed from the queue.")
        return item
```

```

else:
    print("Queue is empty. Cannot dequeue item.")
    return None

def display():
    print("Queue contents:", queue)

while True:
    print("\nMenu:")
    print("1. Enqueue")
    print("2. Dequeue")
    print("3. Display queue")
    print("4. Exit")

    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        item = input("Enter the item to enqueue: ")
        enqueue(item)
    elif choice == '2':
        dequeue()
    elif choice == '3':
        display()
    elif choice == '4':
        print("Exiting program.")
        break
    else:
        print("Invalid choice. Please enter a number between 1 and 4.")

```



```

Menu:
1. Enqueue
2. Dequeue
3. Display queue
4. Exit
Enter your choice (1-4): 1
Enter the item to enqueue: 2
Item 2 added to the queue.

Menu:
1. Enqueue
2. Dequeue
3. Display queue
4. Exit
Enter your choice (1-4): 4
Exiting program.

```

9. Take two lists of numbers. Create third list of numbers for only those numbers from first list which are not there in 2 nd list (use list comprehension).

```

list1 = [98, 97, 86, 76, 80, 48, 56, 66, 82]
list2 = [66, 56, 82, 46, 48]

list3 = [num for num in list1 if num not in list2]

print("First list:", list1)
print("Second list:", list2)
print("Third list (numbers in first list not in second list):", list3)

```



```

First list: [98, 97, 86, 76, 80, 48, 56, 66, 82]
Second list: [66, 56, 82, 46, 48]
Third list (numbers in first list not in second list): [98, 97, 86, 76, 80]

```

Write following programs considering tuple in mind:

1. A list contains names of boys and girls as its elements. Boys' names are stored as tuples. Write a program to find out number of boys and girls in the list. (Hint: use isinstance(ele,tuple))

```

def count_boys_and_girls(names_of_list):
    boys_of_count = 0
    girls_of_count = 0

    for ele in names_of_list:
        if isinstance(ele, tuple):
            boys_of_count += 1
        else:
            girls_of_count += 1

```

```

    return boys_of_count, girls_of_count

names_of_list = [("John",), "Emma", ("Mike",), "Sophia", "Olivia", ("David",)]

boys, girls = count_boys_and_girls(names_of_list)

print(f"Number of boys: {boys}")
print(f"Number of girls: {girls}")

```

```

→ Number of boys: 3
   Number of girls: 3

```

2. A list contains tuples containing roll no., name and age of student. Write a python program to create three lists separately for roll no., name and age

```

students = [(1, "Johny", 20), (2, "Emmi", 29), (3, "Mikel", 37), (4, "Sophiana", 17), (5, "Olvi", 19)]

Roll_Numbers = []
Names = []
Ages = []

for student in students:
    roll_no, name, age = student
    Roll_Numbers.append(roll_no)
    Names.append(name)
    Ages.append(age)

print("Roll Numbers:", Roll_Numbers)
print("Names:", Names)
print("Ages:", Ages)

```

```

→ Roll Numbers: [1, 2, 3, 4, 5]
   Names: ['Johny', 'Emmi', 'Mikel', 'Sophiana', 'Olvi']
   Ages: [20, 29, 37, 17, 19]

```

3. Suppose a date is represented as a tuple (d, m, y). Create two date tuples and find the number of days between the two dates.

```

from datetime import datetime

date1 = ( 28, 2, 2025 )
date2 = ( 28, 2, 2024 )

date1_object = datetime(date1[2], date1[1], date1[0])
date2_object = datetime(date2[2], date2[1], date2[0])

days_difference = abs((date1_object - date2_object).days)

print(f"Number of days between two dates: {days_difference}")

```

```

→ Number of days between two dates: 366

```

4. Create a list of tuples containing a food item and its price. Sort the tuples in descending order by price.

```

# List of tuples containing food items and their prices
food_items = [("Ramen", 9.99), ("Burger", 6.79), ("Sushi", 14.69), ("Noodles", 7.49), ("Salad", 4.99)]

# Sort the list of tuples in descending order by price of food items
sorted_food_items = sorted(food_items, key=lambda item: item[1], reverse=True)

# Print the sorted list of food items
print("Sorted food items by price (descending):")
for item in sorted_food_items:
    print(f"{item[0]}: ${item[1]:.2f}")

```

```

→ Sorted food items by price (descending):
   Sushi: $14.69
   Ramen: $9.99
   Noodles: $7.49
   Burger: $6.79
   Salad: $4.99

```

5. Remove empty tuple(s) from the list of tuples.

```
# list of tuples with some empty tuples
tuples_list = [("Pizza", 8.99), (), ("Burger", 5.99), (), ("Sushi", 12.99), ("Pasta", 7.49), (), ("Salad", 4.99)]

# Remove empty tuples
filtered_tuples_list = [tup for tup in tuples_list if tup]

# Print the filtered list
print("List after removing empty tuples:")
print(filtered_tuples_list)
```

```
→ List after removing empty tuples:
[('Pizza', 8.99), ('Burger', 5.99), ('Sushi', 12.99), ('Pasta', 7.49), ('Salad', 4.99)]
```

### 6. Modify an element of a tuple.

```
main_tuple = (12, 23, 34, 45, 56)

index_to_modify = 2
new_value = 105

modified_tuple = main_tuple[:index_to_modify] + (new_value,) + main_tuple[index_to_modify + 1:]

print("Original tuple:", main_tuple)
print("Modified tuple:", modified_tuple)
```

```
→ Original tuple: (12, 23, 34, 45, 56)
Modified tuple: (12, 23, 105, 45, 56)
```

### 7. Delete an element of a tuple.

```
main_tuple = (14, 25, 36, 47, 59)

index_to_delete = 2

modified_tuple = main_tuple[:index_to_delete] + main_tuple[index_to_delete + 1:]

print("Original tuple:", main_tuple)
print("Modified tuple:", modified_tuple)
```

```
→ Original tuple: (14, 25, 36, 47, 59)
Modified tuple: (14, 25, 47, 59)
```

Write following programs considering sets in mind:

### 1. Write a program that converts words present in a list into uppercase and stores them in a set.

```
word_lists = input("Enter words with space: ").split()
uppercase_setting = {word.upper() for word in word_lists}
print(uppercase_setting)
```

```
→ Enter words with space: hello world
{'HELLO', 'WORLD'}
```

### 2. Write a program to create a set containing 10 random numbers in the range 15 to 45. Count how many of these numbers are less than 30. Delete all numbers that are greater than 35.

```
import random
numbers = {random.randint(15, 45) for _ in range(10)}
print(f"Numbers: {numbers}")
print(f"Count < 30: {sum(1 for num in numbers if num < 30)}")
numbers = {num for num in numbers if num <= 35}
print(f"After removal: {numbers}")
```

```
→ Numbers: {32, 35, 37, 39, 19, 24, 25, 26, 31}
Count < 30: 4
After removal: {32, 35, 19, 24, 25, 26, 31}
```

### 3. Create an empty set. Write a program that adds five new names to this set, modifies one existing name and deletes two names from it.

```
names = set()

names.update(['sasuke', 'jinwoo', 'naruto', 'kakashi', 'jiraya'])

names.remove('sasuke')
names.add('minato')

names.discard('jiraya')
names.discard('kakashi')

print(names)
```

```
↩ {'jinwoo', 'naruto', 'minato'}
```

4. A set contains names which begin either with A or with B. Write a program to separate out the names into two sets, one containing names beginning with A and another with B.

```
names = {'akatsuki', 'bob', 'aris', 'beerus', 'boruto'}

a_name = {name for name in names if name.startswith('a')}
b_name = {name for name in names if name.startswith('b')}

print("A Names:", a_name)
print("B Names:", b_name)
```

```
↩ A Names: {'aris', 'akatsuki'}
  B Names: {'boruto', 'beerus', 'bob'}
```

Write following programs considering Dictionary in mind

1. Write a program to create three dictionaries and concatenate them to create fourth dictionary.

```
Dict1 = {"A": 101, "B": 110}
Dict2 = {"C": 999, "D": 111}
Dict3 = {"E": 520, "F": 456}

Dict4 = {**Dict1, **Dict2, **Dict3}

print(Dict4)
```

```
↩ {'A': 101, 'B': 110, 'C': 999, 'D': 111, 'E': 520, 'F': 456}
```

2. Write a program to check whether a dictionary is empty or not

```
My_Dictionary = {"9999"}

if not My_Dictionary:
    print("My dictionary is empty.")
else:
    print("My dictionary is not empty.")
```

```
↩ My dictionary is not empty.
```

3. Create a dictionary with dept no, employee roll no. and salary. Find out department wise min and maximum of salary.

```
Data = [
    {"dept_no": 9, "emp_roll": 1001, "salary": 90000},
    {"dept_no": 9, "emp_roll": 1002, "salary": 80000},
    {"dept_no": 10, "emp_roll": 2001, "salary": 10000},
    {"dept_no": 10, "emp_roll": 2002, "salary": 70000}
]

from collections import defaultdict

Dept_Salaries = defaultdict(list)

for record in Data:
    Dept_Salaries[record["dept_no"]].append(record["salary"])
```



```

max_min_salary = {
    dept: {"min": min(salaries), "max": max(salaries)}
    for dept, salaries in Dept_Salaries.items()
}

print(max_min_salary)

```

```

↩ 9: {'min': 80000, 'max': 90000}, 10: {'min': 10000, 'max': 70000}}

```

4. Write a program that reads a string from the keyboard and creates dictionary containing frequency of each character occurring in the string.

```

String = input("Enter a string: ")

Char_Frequency = {}

for char in String:
    Char_Frequency[char] = Char_Frequency.get(char, 0) + 1

print(Char_Frequency)

```

```

↩ Enter a string: hello guys welcome
{'h': 1, 'e': 3, 'l': 3, 'o': 2, ' ': 2, 'g': 1, 'u': 1, 'y': 1, 's': 1, 'w': 1, 'c': 1, 'm': 1}

```

5. Create two dictionaries – one containing grocery items and their prices and another containing grocery items and quantity purchased. By using the values from these two dictionaries compute the total bill.

```

prices = {"ramen": 200, "cheese ball": 500, "sting": 900}

quantities = {"ramen": 10, "cheese ball": 20, "sting": 99}

total_bill = sum(prices[item] * quantities.get(item, 0) for item in prices)

print("Total Bill:", total_bill)

```

```

↩ Total Bill: 101100

```

Write following programs considering functions or recursive functions in mind:

1. Write a program that defines a function count\_lower\_upper() that accepts a string and calculates the number of uppercase and lowercase alphabets in it. It should return these values as a dictionary. Call this function for some sample string.

```

def count_lower_uppers(s):
    result = {"uppercase": 0, "lowercase": 0}
    for char in s:
        if char.isupper():
            result["uppercase"] += 1
        elif char.islower():
            result["lowercase"] += 1
    return result

sample_string = "Hello World!"
print(count_lower_uppers(sample_string))

```

```

↩ {'uppercase': 2, 'lowercase': 8}

```

3. Write a program that defines a function create\_array() to create and return a 3D array whose dimensions are passed to the function. Also initialize each element of this array to a value passed to the function. e.g. create\_array(3,4,5,n) where first three arguments are 3D array dimensions and 4th value is for initialing each value of the 3D array.

```

def creates_array(dim1, dim2, dim3, value):
    return [[[value for _ in range(dim3)] for _ in range(dim2)] for _ in range(dim1)]

arrays = creates_array(2, 2, 2, 7)
print(arrays)

```

```

↩ [[[7, 7], [7, 7]], [[7, 7], [7, 7]]]

```

4. Write a program that defines a function `sum_avg()` to accept marks of five subjects and calculates total and average. It should return directly both values.

```
def sum_and_avg(marks):
    Total = sum(marks)
    Average = Total / len(marks)
    return Total, Average

marks = [99, 90, 91, 84, 69]
Total, Average = sum_and_avg(marks)
print("Total:", Total)
print("Average:", Average)
```

```
→ Total: 433
   Average: 86.6
```

5. Pangram is a sentence that uses every letter of the alphabet. Write a program to check whether a given string is pangram or not through a user-defined function `ispangram()`. Test the function with "The quick brown fox jumps over the lazy dog" or "Crazy Fredrick bought many very exquisite opal jewels". Hint: use `set()` to convert the string into a set of characters present in the string and use `<=` to check whether alphaset is a subset of the given string

```
import string

def ispangram(sentence):
    Alphaset = set(string.ascii_lowercase)
    return Alphaset <= set(sentence.lower())

test_1 = "Those Who do not know pain will never understand true peace"
test_2 = "Crazy Fredrick bought many very exquisite opal jewels"

print(ispangram(test_1))
print(ispangram(test_2))
```

```
→ False
   True
```

6. Write a function to create and return a list containing tuples of the form  $(x, x^2, x^3)$  for all  $x$  between 1 and given ending value (both inclusive).

```
def generates_tuples(end_value):
    return [(x, x**2, x**3) for x in range(1, end_value + 1)]

result = generates_tuples(4)
print(result)
```

```
→ [(1, 1, 1), (2, 4, 8), (3, 9, 27), (4, 16, 64)]
```

7. A palindrome is a word or phrase that reads the same in both directions. Write a program that defines a function `ispalindrome()` which checks whether a given string is a palindrome or not. Ignore spaces and case mismatch while checking for palindrome.

```
def ispalindrome(str):
    str = str.replace(" ", "").lower()
    return str == str[::-1]

test_str1 = "a man is powerful"
test_str2 = "sage o egas"

print(ispalindrome(test_str1))
print(ispalindrome(test_str2))
```

```
→ False
   True
```

8. Write a program that defines a function `convert()` that receives a string containing a sequence of whitespace separated words and returns a string after removing all duplicate words and sorting them alphanumerically. Hint: use `set()`, `list()`, `sorted()`, `join()`.

```
def convert(s):
    return " ".join(sorted(set(s.split())))
```

```
input_string = "naruto sasuke naruto akatsuki sasuke boruto"
output_string = convert(input_string)
print(output_string)
```

```
akatsuki boruto naruto sasuke
```

9. Write a program that defines a function `count_alpha_digits()` that accepts a string and calculates the number of alphabets and digits in it. It should return these values as a dictionary.

```
def count_alphabets_digits(str):
    result = {"alphabet": 0, "digit": 0}
    for char in str:
        if char.isalpha():
            result["alphabet"] += 1
        elif char.isdigit():
            result["digit"] += 1
    return result

samp_string = "hello1234prince6!"
print(count_alphabets_digits(samp_string))
```

```
{'alphabet': 11, 'digit': 5}
```

10. Write a program that defines a function called `frequency()` which computes the frequency of words present in a string passed to it. The frequencies should be returned in sorted order of words in the string.

```
def frequency(str):
    words = str.split()
    frequency_dict = {}
    for word in words:
        frequency_dict[word] = frequency_dict.get(word, 0) + 1
    return dict(sorted(frequency_dict.items()))

text = "gato uzumaki gato konoichi uchiha "
result = frequency(text)
print(result)
```

```
{'gato': 2, 'konoichi': 1, 'uchiha': 1, 'uzumaki': 1}
```

11. Write a function `create_list()` that creates and returns a list which is an intersection of two lists passed to it.

```
def create_list(list_1, list_2):
    return [item for item in list_1 if item in list_2]

list_1 = [11, 22, 33, 44, 55]
list_2 = [44, 55, 67, 79, 18]
result = create_list(list_1, list_2)
print(result)
```

```
[44, 55]
```

Write following programs considering functions or recursive functions in mind:

1. If a positive integer is entered through the keyword, write a recursive function to obtain the prime factors of the number.

```
def prime_factors(n, divisor=2):
    if n <= 1:
        return []
    if n % divisor == 0:
        return [divisor] + prime_factors(n // divisor, divisor)
    else:
        return prime_factors(n, divisor + 1)

number = int(input("Enter a positive integer: "))
result = prime_factors(number)
print("Prime factors:", result)
```