In [3]:
```
#3/1/25
#printing hello world
print("Hello World")
```

```
Hello World
```

In [4]:
```
#Using backslash for continuation
total=1+2+3+4+5\
+6+7
print(total)
```

```
28
```

In [5]:
```
#multi line statement on a single line
x=5;y=10;z=x+y
print(z)
```

```
15
```

In [6]:
```
#type inference
variable = 30
print(type(variable))
variable = "Manaswini"
print(type(variable))
```

```
<class 'int'>
<class 'str'>
```

In [7]:
```
#4/1/25
#Type conversion
age = 25
print(type(age))
age_str = str(age)
print(age_str)
print(type(age_str))
```

```
<class 'int'>
25
<class 'str'>
```

In [8]:
```
age = '25'
print(type(int(age)))
height = 5.11
type(height)
int(height)
```

```
<class 'int'>
```

Out[8]:
```
5
```

In [9]:
```
##Dynamic Typing
var = 10
print(var,type(var))
var = "Hello"
print(var,type(var))
var = 3.14
print(var,type(var))
```

```
10 <class 'int'>
Hello <class 'str'>
3.14 <class 'float'>
```

In [10]:
```
##input function
age = input("Enter your age:")
print(age,type(age))
```

```
Enter your age:10
10 <class 'str'>
```

In [11]:
```
age = int(input("Enter your age:"))
print(age,type(age))
```

```
Enter your age:20
20 <class 'int'>
```

In [12]:
```
#Simple Calculator
input1 = float(input("Enter first value"))
input2 = float(input("Enter second value"))
sumval = input1+input2
difference = input1-input2
mult = input1*input2
div = input1/input2
print("Sum",sumval)
print("Difference",difference)
print("Multiplication",mult)
print("Division",div)
```

```
Enter first value12
Enter second value3
Sum 15.0
Difference 9.0
Multiplication 36.0
Division 4.0
```

In [13]:
```
##arithmetic operators
a=10
b=15
add_res = a+b
sub_res = a-b
mult_res = a*b
div_res = a/b
floor_div_res = a//b
modulo_res = a%b
exp_res = a**b
print(add_res)
print(sub_res)
print(mult_res)
print(div_res)
print(floor_div_res)
print(modulo_res)
print(exp_res)
```

```
25
-5
150
0.6666666666666666
0
10
1000000000000000
```

In [14]:
```
##Comparison operators
a=10
b=15
print(a==b)
print(a!=b)
print(a>b)
print(a<b)
print(a>=b)
print(a<=b)
```

```
False
True
False
True
False
True
```

In [15]:
```
##Logical operators
X = True
Y = False
res = X and Y
print(res)
res = X or Y
print(res)
res = not X
print(res)
```

```
False
True
False
```

In [16]:
```
#Conditional Statements
#5/1/25
#if statement
age = 20
if age>=18:
    print("You can vote")
```

```
You can vote
```

In [17]:
```
#else statement
if age>=18:
    print("You can vote")
else:
    print("You are a minor")
```

```
You can vote
```

In [18]:
```
#elif statement
age = 20
if age<13:
    print("You are a child")
elif age<18:
    print("You are a teenager")
else:
    print("You are an adult")
```

```
You are an adult
```

In [19]:
```
num = int(input("Enter the number"))
if num>=0:
    print("The number is positive")
    if num%2==0:
        print("The number is even")
    else:
        print("The number is odd")
else:
    print("The number is negative")
```

```
Enter the number3
The number is positive
The number is odd
```

In [20]:
```
#Leap year
year = int(input("Enter the year"))

if year%4==0:
    if year%100==0:
        if year%400==0:
            print(year,"is a leap year")
        else:
            print(year,"is not a leap year")
    else:
        print(year,"is a leap year")
else:
    print(year,"is not a leap year")
```

```
Enter the year1
1 is not a leap year
```

In [21]:
```
#simple calculator using if else
num1 = float(input("Enter first number"))
num2 = float(input("Enter second number"))
oper = input("Enter the operation needed : +,-,*,/,%,**")

if oper=="+":
    print("Addition",num1+num2)
elif oper=="-":
    print("Subtraction",num1-num2)
elif oper=="*":
    print("Multiplication",num1*num2)
elif oper=="/":
    if num2!=0:
        print("Division",num1/num2)
    else:
        print("Infinity")
elif oper=="%":
    print("Remainder",num1%num2)
elif oper=="**":
    print("Exponent",num1**num2)
else:
    print("Invalid operator")
```

```
Enter first number2
Enter second number3
Enter the operation needed : +,-,*,/,%,**+
Addition 5.0
```

In [22]:
```
#LOOPS
#6/1/25
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

In [23]:
```
for i in range(1,6):
    print(i)
```

```
1
2
3
4
5
```

In [24]:
```
for i in range(1,10,2):
    print(i)
```

```
1
3
5
7
9
```

In [25]:
```
for i in range(10,1,-1):
    print(i)
```

```
10
9
8
7
6
5
4
3
2
```

In [26]:
```
str = "Krish Naik"
for i in str:
    print(i)
```

```
K
r
i
s
h

N
a
i
k
```

In [27]:
```
#while loop
count=0
while count<5:
    print(count)
    count=count+1
```

```
0
1
2
3
4
```

In [28]:
```
#break
for i in range(10):
    if i==5:
        break
    print(i)
```

```
0
1
2
3
4
```

In [29]:
```
#continue
for i in range(10):
    if i%2==0:
        continue
    print(i)
```

```
1
3
5
7
9
```

In [30]:
```
#pass
for i in range(5):
    if i==3:
        pass
    print(i)
```

```
0
1
2
3
4
```

In [31]:
```
#nested loops
for i in range(3):
    for j in range(2):
        print(f"i:{i} and j:{j}")
```

```
i:0 and j:0
i:0 and j:1
i:1 and j:0
i:1 and j:1
i:2 and j:0
i:2 and j:1
```

In [32]:
```
#calculate the sum of first n natural numbers using while and for loop
val = int(input("Enter the value of n"))
sum = 0
for i in range(1,val+1):
    sum+=i
print("The total sum is:",sum)
```

```
Enter the value of n5
The total sum is: 15
```

In [33]:
```
val = int(input("Enter the value of n"))
sum = 0
i = 0
while i<=val:
    sum+=i
    i=i+1
print("The total sum is:",sum)
```

```
Enter the value of n3
The total sum is: 6
```

In [34]:
```
#prime numbers between 1 and 100
for num in range(1,101):
    if num>1:
        for i in range(2,num):
            if num%i==0:
                break
        else:
            print(num)
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

In [35]:
```
#loop questions
#7/1/25
#print first 10 natural number

for i in range(1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [36]:
```
#print the pattern
for i in range(1,6):
    for j in range(1,i+1):
        print(j,end=" ")
    print(" ")
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

In [37]:
```
#calculate sum of all numbers from 1 to given number
num = int(input("Enter the number"))
sum=0
for i in range(1,num+1):
    sum+=i
print("The sum of number is",sum)
```

```
Enter the number1
The sum of number is 1
```

In [38]:
```
#multiplication table
num = int(input("Enter the value of table"))
for i in range(1,11):
    print(f"{num}*{i}={num*i}")
```

```
Enter the value of table2
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20
```

In [39]:
```
#find total number of digits in a number
num = int(input("Enter a number:"))
count=0
while num!=0:
    num=num//10
    count=count+1
print("The total number of digits in the number is:",count)
```

```
Enter a number:3
The total number of digits in the number is: 1
```

In [40]:
```
#print the reverse number pattern
for i in range(5,0,-1):
    for j in range(i,0,-1):
        print(j,end=" ")
    print(" ")
```

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

In [41]:
```
for i in range(-11,0,1):
    print(i)
```

```
-11
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
```

In [42]:
```
#print all prime numbers within a range
num = int(input("Enter the range:"))
for i in range(1,num+1):
    if num>1:
        for j in range(2,num):
            if num%j==0:
                break
        else:
            print(i)
```

```
Enter the range:4
```

In [43]:
```
#display fibonacci series upto 10 terms
a=0
b=1
print("Fibonacci series:")
for i in range(1,11):
    print(a,end=" ")
    c=a+b
    a=b
    b=c
```

```
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
```

In [44]:
```
num = int(input("Enter the number:"))
fact=1
for i in range(1,num+1):
    fact=fact*i
print("The factorial of a number is:",fact)
```

```
Enter the number:

---------------------------------------------------------------------------
---
ValueError                                Traceback (most recent call la
st)
Cell In[44], line 1
----> 1 num = int(input("Enter the number:"))
      2 fact=1
      3 for i in range(1,num+1):

ValueError: invalid literal for int() with base 10: ''
```

In [ ]:
```
#reverse number
num = int(input("Enter the number"))
rev_no=0
while num!=0:
    digit=num%10
    rev_no=rev_no*10+digit
    num=num//10
print("The reverse number is:",rev_no)
```

In [ ]:
```
#cube of all numbers
num = int(input("Enter the number"))
cube=0
for i in range(1,num+1):
    cube=cube+(i**3)
print("The cube of number is:",cube)
```

In [ ]:
```
#List
#8/1/25
lst = []
print(type(lst))
```

In [ ]:
```
mixed_lst = [1,"Hello",3.14,True]
print(mixed_lst)
```

In [ ]:
```
fruits = ["apple","banana","cherry","kiwi","orange"]
print(fruits[0])
print(fruits[-1])
print(fruits[1:])
```

In [ ]:
```
#modifying the list elements
fruits[1]="watermelon"
print(fruits)
```

In [ ]:
```
#List methods
fruits.append("orange")
print(fruits)
fruits.insert(1,"dragonfruit")
print(fruits)
fruits.remove("watermelon")
print(fruits)
popped_fruits=fruits.pop()
print(popped_fruits)
print(fruits)
index=fruits.index("apple")
print(index)
print(fruits.count("apple"))
fruits.sort()
print(fruits)
fruits.reverse()
fruits.clear()
print(fruits)
```

In [ ]:
```
#List Slicing
numbers = [1,2,3,4,5,6,7,8,9,10]
print(numbers[2:5])
print(numbers[:5])
print(numbers[::2])
print(numbers[::-1])
```

In [ ]:
```
#Iterating over list
fruits = ["apple","banana","cherry","kiwi","orange"]
for fruit in fruits:
    print(fruit)
```

In [ ]:
```
#Getting an index
for index,fruit in enumerate(fruits):
    print(index,fruit)
```

In [ ]:
```
#List Comprehension
lst = []
for x in range(10):
    lst.append(x**2)
print(lst)
```

In [ ]:
```
square = [x**2 for x in range(10)]
print(square)
```

In [ ]:
```
even_square = [x**2 for x in range(10) if x%2==0]
print(even_square)
```

In [ ]:
```
lst1 = [1,2,3,4]
lst2 = ['a','b','c','d']
pair = [[i,j] for i in lst1 for j in lst2]
print(pair)
```

In [ ]:
```
#Inbuilt function with list comprehension
words=["hello","world","python"]
lengths = [len(i) for i in words]
print(lengths)
```

In [ ]:
```
a = [1,2,3,4,5]
result = ["Even" if n%2==0 else "Odd" for n in a]
print(result)
```

In [ ]:
```
#Sets
#9/1/25
#Create a set
my_set = {1,2,3,4,5}
print(my_set)
print(type(my_set))
```

In [ ]:
```
my_empty_set=set()
print(type(my_empty_set))
```

In [ ]:
```
my_set = set([1,2,3,4,5,6,6])
print(my_set)
```

In [ ]:
```
#Basic Set Operations
#Adding elements
my_set.add(7)
print(my_set)
```

In [ ]:
```
#Removing elements
my_set.remove(3)
print(my_set)
```

In [ ]:
```
my_set.discard(11)
print(my_set)
```

In [ ]:
```
removed_element=my_set.pop()
print(removed_element)
print(my_set)
```

In [ ]:
```
#clear all the elements
#my_set.clear()
#print(my_set)
```

In [ ]:
```
#Set membership test
my_set = {1,2,3,4,5}
print(3 in my_set)
print(11 in my_set)
```

In [ ]:
```
#Mathematical Operations
set1 = {1,2,3,4,5,6}
set2 = {4,5,6,7,8,9}
union_set = set1.union(set2)
print(union_set)
intersection_set = set1.intersection(set2)
print(intersection_set)
print(set1.difference(set2))
```

In [ ]:
```
#set1.intersection_update(set2)
#print(set1)
```

In [ ]:
```
#Symmetric Difference
set1.symmetric_difference(set2)
```

In [ ]:
```
#Set Methods
set1 = {1,2,3}
set2 = {3,4,5}
print(set1.issubset(set2))
print(set1.issuperset(set2))
```

In [ ]:
```
#to remove duplicates
lst = [1,2,2,3,4,5,6,6,7]
print(set(lst))
```

In [ ]:
```
#Counting unique words
text = "In this tutorial say hii In"
words = text.split()
print(words)
unique_words = set(words)
print(unique_words)
print(len(unique_words))
```

In [ ]:
```
#Dictionary
#10/1/25
#Creating a Dictionary
empty_dict = {}
print(type(empty_dict))
```

In [ ]:
```
student = {"name":"Manaswini","age":22,"grade":"A"}
print(student)
```

In [ ]:
```
#Accessing dictionary elements
print(student['grade'])
print(student['age'])
```

In [ ]:
```
#Accessing using get method
print(student.get('grade'))
print(student.get('last_name'))
print(student.get('last_name',"Not Available"))
```

In [ ]:
```
#Modifying Dictionary Elements
student['age']=23
print(student)
student['address']="India"
print(student)
#del student['grade']
#print(student)
```

In [ ]:
```
#Dictionary Methods
#Getting all keys , values and key-value pairs
keys = student.keys()
print(keys)
values = student.values()
print(values)
items = student.items()
print(items)
```

In [ ]:
```
#Copy
student_copy = student
print(student)
print(student_copy)
```

In [ ]:
```
student['name']="Subham"
print(student)
print(student_copy)
```

In [ ]:
```
#Shallow copy
student_copy1 = student.copy()
print(student_copy1)
print(student)
```

In [ ]:
```
student['name']="Manaswini"
print(student_copy1)
print(student)
```

In [ ]:
```
#Iterating over key and value
for keys in student.keys():
    print(keys)
```

In [ ]:
```
for values in student.values():
    print(values)
```

In [ ]:
```
for key,value in student.items():
    print(f"{key}:{value}")
```

In [ ]:
```
#Nested Dictionaries
students = {
    "student1":{"name":"Manaswini","age":22},
    "student2":{"name":"Subham","age":23}
}
print(students)
```

In [ ]:
```
#Access nested dictionaries
print(students["student2"]["name"])
```

In [ ]:
```
#Iterating over nested list
for student_id,student_info in students.items():
    print(f"{student_id}:{student_info}")
    for key,value in student_info.items():
         print(f"{key}:{value}")
```

In [ ]:
```
#Dictionary Comprehension
squares = {x:x**2 for x in range(10)}
print(squares)
```

In [ ]:
```
even_squares = {x:x**2 for x in range(10) if x%2==0}
print(even_squares)
```

In [ ]:
```
#Practical ex
numbers = [1,2,2,3,3,3,4,4,4,4]
frequency={}
for number in numbers:
    if number in frequency:
        frequency[number]+=1
    else:
        frequency[number]=1
print(frequency)
```

In [ ]:
```
#Merge 2 dictionaries
dict1 = {"a":1,"b":2}
dict2 = {"c":3,"d":4}
merged_dict={**dict1,**dict2}
print(merged_dict)
```

In [ ]:
```
#Tuples
11/1/25
#creating a tuple
empty_tuple=()
print(empty_tuple)
print(type(empty_tuple))
```

In [ ]:
```
tpl = tuple()
print(type(tpl))
```

In [ ]:
```
numbers=tuple([1,2,3,4,5,6])
print(numbers)
```

In [ ]:
```
mixed_tuple=(1,"Hello World",3.14,True)
print(mixed_tuple)
```

In [ ]:
```
#Accessing Tuple Elements
print(numbers[0])
print(numbers[-1])
print(numbers[0:4])
print(numbers[::-1])
```

In [ ]:
```
#Tuple Operation
concat_tuple = numbers+mixed_tuple
print(concat_tuple)
```

In [ ]:
```
print(mixed_tuple*3)
print(numbers*3)
```

In [ ]:
```
#Immutable nature of tuples
tup = (1,2,3,4,5,6,7)
#tup[1]="Krish"
print(tup)
```

In [ ]:
```
#Tuple Methods
print(numbers.count(1))
print(numbers.index(3))
```

In [ ]:
```
#Packing and unpacking tuple
packed_tuple=1,"Hello",3.14
print(packed_tuple)
```

In [ ]:
```
#Unpacking a tuple
a,b,c=packed_tuple
print(a)
print(b)
print(c)
```

In [ ]:
```
#Unpacking with *
numbers = (1,2,3,4,5,6)
first,*middle,last=numbers
print(first)
print(middle)
print(last)
```

In [ ]:
```
#Nested Tuple
#Nested List
lst = [[1,2,3,4],[6,7,8,9],["Hello",3.14,"c"]]
print(lst[0][2])
print(lst[0][0:3])
```

In [ ]:
```
tupe = [[1,2,3,4],[6,7,8,9],(1,"Hello","c")]
print(tupe[2][0:3])
```

In [ ]:
```
nested_tuple=((1,2,3),("a","b","c"),(True,False))
print(nested_tuple[0])
print(nested_tuple[1][2])
```

In [ ]:
```
#Iterating Over Nested loops
for i in nested_tuple:
    for j in i:
        print(j,end=" ")
    print()
```

In [ ]:
```
#Real world examples using a list
#EX) Manage a to do list
#12/1/25
to_do_lst = ["Buy Groceries","Clean the house","Pay bills"]
#Adding a task
to_do_lst.append("Schedule a meeting")
to_do_lst.append("Go for a run")
#Removing a completed a task
to_do_lst.remove("Clean the house")
#check if a task is present
if "Pay bills" in to_do_lst:
    print("Don't forget to pay utility bills")
print("To do list remaianing")
for task in to_do_lst:
    print(task)
```

In [ ]:
```
#Organizing Student Grades
#Create a list to store and calculate average grades for students
grades=[85,92,78,90,88]
grades.append(95)
average_grade=sum(grades)/len(grades)
print(f"Average Grade:{average_grade:.2f}")
highest_grade=max(grades)
lowest_grade=min(grades)
print(f"Highest Grade is:{highest_grade}")
print(f"Lowest Grade is:{lowest_grade}")
```

In [ ]:
```
#Use a list to collect and analyze user feedback
feedback = ["Great Service!","Very Satisfied","Could be better","Excellent
#Adding new feedback
feedback.append("Not happy with the service!")
#Counting specific feedback
positive_count=sum(1 for comment in feedback if "great" in comment.lower()
print(f"Positive feedback count:{positive_count}")
#Printing all feedback
print("User Feedback")
for comment in feedback:
    print(f"-{comment}")
```

In [ ]:
```
#Functions in Python
#13/1/25
#Syntax

#def function_name(parameter):
    #""" """
    #function body
    #return expression
```

In [ ]:
```
def even_or_odd(num):
    """This function finds even or odd"""
    if num%2==0:
        print("Even")
    else:
        print("Odd")
even_or_odd(5)
```

In [ ]:
```
#function with multiple parameter
def add(a,b):
    return a+b
result = add(5,3)
print(result)
```

In [ ]:
```
##default parameter
def greet(name="Guest"):
    print(f"Hii {name}")
greet()
```

In [ ]:
```
#Positional Arguments
def print_numbers(*args):
    for number in args:
        print(number)
print_numbers(1,2,3,4,5,"Krish")
```

In [ ]:
```
#Keyword Arguments
def print_details(**kwargs):
    for key,value in kwargs.items():
        print(f"{key}:{value}")
print_details(name="Manaswini",age=22)
```

In [ ]:
```
#Combining both
def print_values(*args,**kwargs):
    for number in args:
        print(number)
    for key,value in kwargs.items():
        print(f"{key}:{value}")
print_values(1,2,3,4,name="Manaswini",age=22)
```

In [ ]:
```
#Return multiple statements
def multiply(a,b):
    return a*b,a
ans = multiply(2,3)
print(ans)
```

In [ ]:
```
#More questions on function
#14/1/25
#1)Temperature Conversion
def celToFahr(temp):
    fahr=((temp*(9/5))+32)
    return fahr
print("The teperature in fahrenheit is :",celToFahr(50))
```

In [ ]:
```
def fahrToCel(temp):
    cel=((temp-32)*(5/9))
    return cel
print("The teperature in celcius is :",fahrToCel(40))
```

In [ ]:
```python
#Temperature conversion in one function
def convert_temp(temp,unit):
    """This function is used for temperature conversion"""
    if unit=='F':
        return (temp-32)*5/9
    elif unit=='C':
        return temp * 9/5 +32
    else:
        return None
print("The teperature in fahrenheit is :",convert_temp(50,'F'))
print("The teperature in celcius is :",convert_temp(50,'C'))
```

In [ ]:
```python
#Password Strength Checker
def is_strong_password(password):
    """This function checks if a password strong or not"""
    if len(password)<8:
        return False
    if not any(char.isdigit() for char in password):
        return False
    if not any(char.islower() for char in password):
        return False
    if not any(char.isupper() for char in password):
        return False
    if not any(char in '!@#$%^&*()_+' for char in password):
        return False
    return True
print(is_strong_password("WeakPwd"))
print(is_strong_password("Str0ngPwd!"))
```

In [ ]:
```python
#Calculate the total cost of items in a shopping cart
#15/1/25
def calculate_total_cost(cart):
    total_cost=0
    for item in cart:
        total_cost+=item['price']*item['quantity']
    return total_cost
cart=[
    {'name':'Apple','price':0.5,'quantity':4},
    {'name':'Banana','price':0.3,'quantity':6},
    {'name':'Orange','price':0.7,'quantity':3}
]
total_cost=calculate_total_cost(cart)
print(total_cost)
```

In [ ]:
```
#Check if a string is pallindrome
def is_Pallindrome(str):
    str=str.lower().replace(" ","")
    return str==str[::-1]
print(is_Pallindrome("MADAM"))
print(is_Pallindrome("Hello"))
```

In [ ]:
```
#Find factorial of a number
def find_Factorial(num):
    if num==0:
        return 1
    else:
        return num*find_Factorial(num-1)
fact=find_Factorial(3)
print("The factorial is:",fact)
```

In [ ]:
```
#Email verification
import re
def is_valid_email(email):
    pattern=r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    return re.match(pattern,email) is not None
print(is_valid_email("test@gmail.com"))
```

In [ ]:
```
#Lambda function
#16/1/25
addition=lambda a,b:a+b
type(addition)
print(addition(5,10))
```

In [ ]:
```
#print even number
even_no = lambda num:num%2==0
even_no(10)
```

In [ ]:
```
addition = lambda x,y,z:x+y+z
print(addition(10,11,12))
```

In [ ]:
```
#Map function
numbers = [1,2,3,4,5]
list(map(lambda x:x**2,numbers))
```

In [ ]:
```
#Map Function
#17/1/25
numbers = [1,2,3,4,5,6,7,8]
print(list(map(lambda x:x**2,numbers)))
```

In [ ]:
```
#map multiple iterables
numbers1 = [1,2,3]
numbers2 = [4,5,6]
added_numbers = list(map(lambda x,y:x+y,numbers1,numbers2))
print(added_numbers)
```

In [ ]:
```
#map to convert list of string to integers
str_no = ['1','2','3','4','5']
int_no = list(map(int,str_no))
print(int_no)
```

In [ ]:
```
#inbuilt function using map
words = ['apple','banana','orange','cherry']
upper_words = list(map(str.upper,words))
print(upper_words)
```

In [ ]:
```
def get_name(person):
    return person['name']
people = [
    {'name':'Krish','age':32},
    {'name':'Jack','age':33}
]
list(map(get_name,people))
```

In [ ]:
```
#Filter function
#18/1/25
def even(num):
    if num%2==0:
        return True
even(24)
```

In [ ]:
```
lst=[1,2,3,4,5,6,7,8,9]
list(filter(even,lst))
```

In [ ]:
```
#filter with lambda function
numbers = [1,2,3,4,5,6,7,8,9]
greater_than_five = list(filter(lambda x:x>5,numbers))
print(greater_than_five)
```

In [ ]:
```
#filter with multiple conditions
numbers = [1,2,3,4,5,6,7,8,9]
even_and_greater_than_five = list(filter(lambda x:x%2==0 and x>5,numbers))
print(even_and_greater_than_five )
```

In [ ]:
```
#filter to check if age is greater than 25 in dictionary
people = [
    {'name':'Krish','age':32},
    {'name':'Jack','age':33},
    {'name':'Manish','age':22}
]
def age_greater(person):
    return person['age']>25
list(filter(age_greater,people))
```

In [ ]:
```
#basic square
#21/1/25
def generate_square(n):
    lst=[]
    for i in range(n):
        lst.append('*'*n)
    return lst
result=generate_square(5)
print(result)
```

In [ ]:
```
#hollow square
def generate_hollow_square(n):
    lst=[]
    for i in range(n):
        if i==0 or i==n-1:
            lst.append('*' * n)
        else:
            lst.append('*' + ' '*(n-2) + '*')
    return lst

ans=generate_hollow_square(5)
print(ans)
```

In [ ]:
```
#Functions
#22/1/25
def celcius_to_fahrenheit(c):
    return (9/5*c)+32
fahr = celcius_to_fahrenheit(40)
print("Temperature in fahrenheit is:",fahr)
```

In [ ]:
```
def area_of_rectangle(length,breadth):
    return length*breadth
result = area_of_rectangle(10,5)
print("The area of rectangle is:",result)
```

In [ ]:
```
def calculate_distance(speed,time):
    return speed*time
distance = calculate_distance(5,10)
print("The distance covered is :",distance)
```

In [ ]:
```
import math
def calculate_lift_rounds(n,capacity):
    rounds = math.ceil(n/capacity)
    return rounds
result = calculate_lift_rounds(7,4)
print(result)
```

In [ ]:
```
def calculate_y(slope,intercept,x):
    return ((slope*x)+intercept)
ans = calculate_y(2,3,4)
print(ans)
```

In [ ]:
```
#Patterns
#24/1/25
def generate_rectangle(n,m):
    lst=[]
    for i in range(n):
        lst.append('*'*m)
    return lst
ans = generate_rectangle(3,2)
print(ans)
```

In [ ]:
```
def generate_triangle(n):
    lst=[]
    for i in range(1,n+1):
        lst.append('*'*i)
    return lst
ans = generate_triangle(3)
print(ans)
```

In [ ]:
```
def generate_inverted_triangle(n):
    lst=[]
    for i in range(n,0,-1):
        lst.append('*' * i)
    return lst
ans = generate_inverted_triangle(3)
print(ans)
```

In [ ]:
```
def generate_pyramid(n):
    pyramid=[]
    for i in range(1,n+1):
        stars='*'*(2*i-1)
        spaces=' '*(n-i)
        pyramid.append(spaces+stars+spaces)
    return pyramid
ans = generate_pyramid(3)
print(ans)
```

In [ ]:
```
def inverted_pyramid(n):
    pyramid=[]
    for i in range(1,n+1):
        stars='*'*(2*n-2*i+1)
        spaces=' '*(i-1)
        pyramid.append(spaces+stars+spaces)
    return pyramid
ans = inverted_pyramid(3)
print(ans)
```

In [ ]:
```
#26/1/25
def generate_number_triangle(n):
   return [str(i)*i for i in range(1,n+1)]
ans = generate_number_triangle(5)
print(ans)
```

In [ ]:
```
#Floyds triangle
def floyd(n):
    val=1
    for i in range(1,n+1):
        for j in range(1,i+1):
            print(val,end=" ")
            val+=1
        print(" ")
floyd(5)
```

In [ ]:
```python
#Diamond Pattern
def generate_diamond(n):
    result=[]
    for i in range(1,n+1):
        spaces= " "*(n-i)
        stars='*'*(2*i-1)
        result.append(spaces+stars+spaces)
    for i in range(n-1,0,-1):
        spaces= " "*(n-i)
        stars='*'*(2*i-1)
        result.append(spaces+stars+spaces)
    return result
ans = generate_diamond(3)
print(ans)
```

In [ ]:
```python
def generate_stars(n):
    result=[]
    for i in range(1,n+1):
        spaces=" "*(n-i)
        stars="*"*i
        result.append(spaces+stars)
    return result
ans = generate_stars(3)
print(ans)
```

In [ ]:
```python
#sandglass
def generate_sandglass(n):
    result=[]
    for i in range(1,n+1):
        stars= "*"*(2*(n-i)+1)
        spaces=' '*(i-1)
        result.append(spaces+stars+spaces)
    for i in range(n-1,0,-1):
        stars= "*"*(2*(n-i)+1)
        spaces=' '*(i-1)
        result.append(spaces+stars+spaces)
    return result
ans = generate_sandglass(3)
print(ans)
```

In [ ]:
```python
#Hollow Right Triangle
def hollow_triangle(n):
    for i in range(1,n+1):
        if i==1 or i==n:
            print('*' * i)
        else:
            print('*'+" "*(i-2)+'*')
hollow_triangle(5)
```

In [ ]:
```
def hollow_inverted(n):
    for i in range(n,0,-1):
        if i==n or i==1:
            print('*' * i)
        else:
            print('*'+" "*(i-2)+'*')
hollow_inverted(5)
```

In [ ]:
```
#pyramid numbers
def number_pyramid(n):
    for i in range(1,n+1):
        numbers = ' '.join(str(j) for j in range(1,i+1))
        print(numbers.center(2*n-1))
number_pyramid(4)
```

In [ ]:
```
#Inbuilt data structure
#Sum of List Elements
def sum_list(numbers):
    sum=0
    for i in numbers:
        sum+=i
    return sum
print(sum_list([1,2,3,4,5]))
```

In [ ]:
```
#Largest element in a list
def find_largest(numbers):
    if not numbers:
        return None
    max_num=numbers[0]
    for i in numbers:
        if i>max_num:
            max_num=i
    return max_num
print(find_largest([1,2,3,4,5]))
```

In [ ]:
```python
#Remove duplicates
def remove_duplicates(lst):
    unique_elements=[]
    for num in lst:
        is_duplicate=False
        for unique in unique_elements:
            if num==unique:
                is_duplicate=True
                break
        if not is_duplicate:
            unique_elements.append(num)
    return unique_elements
print(remove_duplicates([1, 2, 2, 3, 4, 4, 5]))
```

In [ ]:
```python
#Check if all elements are unique
def check_unique(lst):
    for i in range(len(lst)):
        for j in range(i+1,len(lst)):
            if lst[i]==lst[j]:
                return False
    return True
print(check_unique([2,3,1,4,4,5,6,7,7]))
```

In [ ]:
```python
#Reverse a list
def reverse_list(lst):
    reverse_lst=[]
    for i in range(len(lst),0,-1):
        reverse_lst.append(i)
    return reverse_lst
ans=reverse_list([1,2,3,4,5])
print(ans)
```

In [ ]:
```python
#Even Odd elements
lst=[1,2,3,4,5]
def even_odd(lst):
    even,odd=0,0
    for i in lst:
        if i%2==0:
            even=even+1
        else:
            odd=odd+1
    return (even,odd)
result = even_odd(lst)
print(result)
```

In [ ]:
```python
#Max difference between two elements
def max_diff(lst):
    diff=0
    for i in range(len(lst)-1):
        if abs(lst[i]-lst[i+1])>diff:
            diff=abs(lst[i]-lst[i+1])
    return diff
print(max_diff([1,7,3,10,5]))
```

In [ ]:
```python
#Merge two sorted list
def merge_lst(list1,list2):
    sorted_lst=[]
    i,j=0,0
    while i<len(list1) and j<len(list2):
        if list1[i]<list2[j]:
            sorted_lst.append(list1[i])
            i+=1
        else:
            sorted_lst.append(list2[j])
            j+=1
    while i<len(list1):
        sorted_lst.append(list1[i])
        i+=1

    while j<len(list2):
        sorted_lst.append(list2[j])
        j+=1
    return sorted_lst
result = merge_lst([1,3,5],[2,4,6])
print(result)
```

In [ ]:
```python
#Basic Questions
#30/1/25
def calculate_val(num1,num2):
    if num1*num2<=1000:
        return num1*num2
    else:
        return num1+num2
print(calculate_val(100,20))
```

In [ ]:
```python
print("Printing current and previous number sum in a range(10)")
for i in range(1,11):
    print("Current number",i,"Previous number",i-1,"Sum:",(i+(i-1)))
```

In [ ]:
```
def calc_str(str):
    stri=""
    print("Original string is:",str)
    print("String at even index is:")
    for i in range(0,len(str),2):
        stri+=str[i]
    return stri
ans = calc_str("PYnative")
print(ans)
```

In [ ]:
```
def rem_char(str,n):
    stri=""
    for i in range(n,len(str)):
        stri+=str[i]
    return stri
ans = rem_char("PYnative",4)
print(ans)
```

In [ ]:
```
def check_lst(list1):
    if list1[0]==list1[len(list1)-1]:
        return True
    return False
ans = check_lst([1,2,3,4,5])
print(ans)
```

In [ ]:
```
def display_lst(list1):
    for i in range(len(list1)):
        if list1[i]%5==0:
            print(list1[i])
ans = display_lst([5,20,15,3,2,4,25])
```

In [ ]:
```
def calc_str(s):
    count = s.count("Emma")
    print("Emma appeared", count, "times")
    return count

ans = calc_str("Emma is a good girl. Emma is beautiful")
print(ans)
```

In [ ]:
```
#31/1/25
def check_pallindrome(num):
    digit=0
    rev_no=0
    n=num
    while(n>0):
        digit=n%10
        rev_no=rev_no*10+digit
        n=n//10
    if rev_no==num:
        print("Pallindrome number")
    else:
        print("Not Pallindrome")
check_pallindrome(121)
```

In [ ]:
```
def merge_lst(list1,list2):
    lst=[]
    for i, j in zip(list1,list2):
        if i%2!=0:
            lst.append(i)
        if j%2==0:
            lst.append(j)
    return lst
ans=merge_lst([10, 20, 25, 30, 35],[40, 45, 60, 75, 90])
ans.sort()
print(ans)
```

In [ ]:
```
#2/1/25
val1 = int(input("Enter first value"))
val2 = int(input("Enter second value"))
print(val1*val2)
```

In [ ]:
```
print("Name","as","James",sep="**")
```

In [ ]:
```
#give the octal value of num
num=8
print('%o' % num)
```

In [ ]:
```
num = 458.541315
print('%.2f'% num)
```

In [ ]:
```
lst=[]
for i in range(5):
    lst.append(float(input("Enter values")))
print(lst)
```

In [ ]:
```
str1,str2,str3=input("Enter the names").split()
print("Name1:",str1)
print("Name2:",str2)
print("Name3:",str3)
```

In [ ]:
```
totalMoney = 1000
quantity = 3
price = 450
print(f"I have {totalMoney} dollars so I can buy {quantity} football for
```

In [46]:
```
i=0
while i<=10:
    print(i)
    i+=1
```

```
0
1
2
3
4
5
6
7
8
9
10
```

In [48]:
```
val=int(input("Enter the value"))
sum_val=0
for i in range(1,val+1):
        sum_val+=i
print(sum_val)
```

```
Enter the value10
55
```

In [49]:
```
val=int(input("Enter the value"))
for i in range(1,11):
    print(f"{val} * {i} = {val*i}")
```

```
Enter the value5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

In [50]:
```
numbers = [12, 75, 150, 180, 145, 525, 50]
for i in numbers:
    print(i)
```

```
12
75
150
180
145
525
50
```

In [1]:
```
val = int(input("Enter the number"))
count=0
while(val!=0):
    val=val//10
    count+=1
print("The number of digits is :",count)
```

```
Enter the number542
The number of digits is : 3
```

In [6]:
```
numbers = [12, 75, 150, 180, 145, 525, 50]
reverse_lst=reversed(numbers)
for i in reverse_lst:
    print(i)
```

```
50
525
145
180
150
75
12
```

In [13]:
```
for i in range(-10,0,1):
    print(i)
```

```
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
```

In [1]:
```
#4/2/25
#printing prime numbers in a range
start = int(input("Enter the first number"))
end = int(input("Enter the second number"))
for i in range(start,end+1):
    if i>1:
        is_prime=True
        for j in range(2,int(i ** 0.5)+1):
            if i%j==0:
                is_prime=False
                break
        if is_prime:
            print(i)
```

```
Enter the first number2
Enter the second number10
2
3
5
7
```

In [3]:
```
#Fibonacci series
a=0
b=1
print(a)
print(b)
for i in range(3,11):
    c=a+b
    print(c)
    a=b
    b=c
```

```
0
1
1
2
3
5
8
13
21
34
```

In [4]:
```
#Factorial of a number
n=int(input("Enter the number:"))
fact=1
for i in range(1,n+1):
    fact*=i
print(fact)
```

```
Enter the number:5
120
```

In [5]:
```
#Reverse an integer
n=int(input("Enter the number"))
digit=0
rev_no=0
num=n
while(num!=0):
    digit=num%10
    rev_no=rev_no*10+digit
    num=num//10
print(rev_no)
```

```
Enter the number76542
24567
```

In [7]:
```
my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
n=len(my_list)
for i in range(1,n,2):
    print(my_list[i])
```

```
20
40
60
80
100
```

In [8]:
```
n=int(input("Enter the number"))
for i in range(1,n+1):
    print(f"The cube of {i} is : {i ** 3}")
```

```
Enter the number6
The cube of 1 is : 1
The cube of 2 is : 8
The cube of 3 is : 27
The cube of 4 is : 64
The cube of 5 is : 125
The cube of 6 is : 216
```

In [10]:
```
n=int(input("Enter the number"))
start=2
sum_seq=0
for i in range(0,n):
    sum_seq+=start
    start=start*10+2
print("Sum of above sequence is",sum_seq)
```

```
Enter the number5
Sum of above sequence is 24690
```

In [2]:
```
#6/2/25
def create_func(name,age):
    print("Name:",name,"Age:",age)
create_func("Manaswini",22)
```

```
Name: Manaswini Age: 22
```

In [4]:
```
#Variable length of arguments
def func1(*args):
    for i in args:
        print(i)
func1(22,24,25)
```

```
22
24
25
```

In [5]:
```
#Return multiple values from a function
def calculation(val1,val2):
    return val1+val2,val1-val2
calculation(5,3)
```

Out[5]: (8, 2)

In [6]:
```
def show_employee(name="Manaswini",salary=9000):
    print(f"Name:{name} Salary:{salary}")
show_employee("Ben",12000)
show_employee("Joseph")
```

```
Name:Ben Salary:12000
Name:Joseph Salary:9000
```

In [7]:
```
#Create an outer function and inner function
def outer_func(a,b):
    def addition(a,b):
        return a+b
    add=addition(a,b)
    return add+5
outer_func(5,10)
```

Out[7]: 20

In [9]:
```
def even_num():
    even_lst=[i for i in range(4,30) if i%2==0]
    return even_lst
result=even_num()
print(result)
```

```
[4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

In [11]:
```
def find_max(lst):
    max_val=0
    for i in lst:
        if max_val<i:
            max_val=i
    return max_val
ans=find_max([1,2,4,5,23,4,5,8])
print(ans)
```

```
23
```

In [2]:
```
#7/2/25
def sum_even_nos(n):
    sum=0
    for i in range(2,(2*n)+1,2):
        sum+=i
    return sum
sum_even_nos(3)
```

Out[2]: 12

In [4]:
```
def is_even(n):
    if n%2==0:
        return True
    else:
        return False
is_even(7)
```

Out[4]: False

In [10]:
```
def is_prime(n):
    if n<=1:
        return False
    for i in range(2,int(n**(0.5)+1)):
        if n%i==0:
            return False
    return True
is_prime(13)
```

Out[10]: True

In [16]:
```
def is_perfectSquare(num):
    if num<1:
        return False
    sqrt_num=int(num ** 0.5)

    return sqrt_num*sqrt_num==num

is_perfectSquare(10)
```

Out[16]: False

In [17]:
```
def gcd(n,m):
    if n<m:
        for i in range(n,0,-1):
            if n%i==0 and m%i==0:
                return i
    else:
         for i in range(m,0,-1):
            if n%i==0 and m%i==0:
                return i
    return 1

gcd(12,18)
```

Out[17]: 6

In [2]:
```
#9/2/25
str1="James"
n=len(str1)
print(str1[0]+str1[n//2]+str1[n-1])
```

```
Jms
```

In [4]:
```
str1="JhonDipPeta"
n=len(str1)
print(str1[n//2-1]+str1[n//2]+str1[n//2+1])
```

```
Dip
```

In [8]:
```
def insert_string(s1,s2):
    middle_index=len(s1)//2
    return s1[:middle_index]+s2+s1[middle_index:]
insert_string("Ault","Kelly")
```

Out[8]: 'AuKellylt'

In [10]:
```
str1="PyNaTive"
lower=[]
upper=[]
for char in str1:
    if char.islower():
        lower.append(char)
    else:
        upper.append(char)
sorted_str=''.join(lower+upper)
print(sorted_str)
```

```
yaivePNT
```

In [2]:
```
#Count all digits,chars,symbols
str1 = "P@#yn26at^&i5ve"
digits,chars,symbols=0,0,0
for i in str1:
    if i.isalpha():
        chars+=1
    elif i.isdigit():
        digits+=1
    else:
        symbols+=1
print("Characters:",chars,"Digits:",digits,"Symbols:",symbols)
```

```
Characters: 8 Digits: 3 Symbols: 4
```

In [3]:
```
s1="Abc"
s2="Xyz"
s1_len=len(s1)
s2_len=len(s2)
length=s1_len if s1_len>s2_len else s2_len
result=""
s2=s2[::-1]
for i in range(length):
    if i<s1_len:
        result+=s1[i]
    if i<s2_len:
        result+=s2[i]
print(result)
```

```
AzbycX
```

In [10]:
```
def find_ans(s1 = "Yn",s2 = "PYnative"):
    for i in s1:
        if s1 in s2:
            return True
    return False
find_ans()
```

Out[10]:
```
True
```

In [13]:
```
str1 = "Welcome to USA. usa awesome, isn't it?"
str2 = "USA"
count=str1.lower().count(str2.lower())
print(count)
```

```
2
```

In [15]:
```
str1 = "PYnative29@#8496"
sum_val,count=0,0
for i in str1:
    if i.isdigit():
        sum_val+=int(i)
        count+=1
print("Sum:",sum_val)
print("Average",sum_val/count)
```

```
Sum: 38
Average 6.333333333333333
```

In [16]:
```
str1="Apple"
char_count={}
for char in str1:
    if char in char_count:
        char_count[char]+=1
    else:
        char_count[char]=1
print(char_count)
```

```
{'A': 1, 'p': 2, 'l': 1, 'e': 1}
```

In [19]:
```
str1="PYnative"
print(str1[-1::-1])
```

```
evitanYP
```

In [20]:
```
str1 = "Emma is a data scientist who knows Python. Emma works at google."
substring = "Emma"
last_index = str1.rfind(substring)
print(f"Last occurrence of {substring} starts at index {last_index}")
```

```
Last occurrence of Emma starts at index 43
```

In [1]:
```
#10/2/25
def reverse_string(s):
    return s[-1::-1]
reverse_string("hello")
```

Out[1]:
```
'olleh'
```

In [4]:
```
def count_vowels(s):
    count=0
    for char in s:
        if (char=='a' or char=='e' or char=='i' or char=='o' or char=='u'
            count+=1
    return count
count_vowels("Python Programming")
```

Out[4]: 4

In [8]:
```
def are_equal_strings(s,t):
    if s==t:
        return True
    return False
are_equal_strings("hello","helo")
```

Out[8]: False

In [20]:
```
def is_palindrome(s):
    s=''.join(c.lower() for c in s if c.isalnum())
    return s == s[::-1]
is_palindrome("A man a plan a canal Panama")
```

Out[20]: True

In [22]:
```
def count_words(s):
    count=0
    in_word=False
    for char in s:
        if char.isspace():
            in_word=False
        elif not in_word:
            count+=1
            in_word=True
    return count
count_words("Hello, World!")
```

Out[22]: 2

In [3]:
```
#11/2/25
#remove duplicates
def remove_duplicates(s):
    seen = set()
    result=[]
    for ch in s:
        if ch not in seen:
            seen.add(ch)
            result.append(ch)
    return "".join(result)
remove_duplicates("programming")
```

Out[3]: 'progamin'

In [5]:
```
def count_consonants(s):
    vowels={'a','e','i','o','u','A','E','I','O','U'}
    count=0
    for ch in s:
        if ch.isalpha() and ch not in vowels:
            count+=1
    return count
count_consonants("Hello, World")
```

Out[5]: 7

In [10]:
```
from collections import Counter

def is_anagram(s, t):
    return Counter(s) == Counter(t)
is_anagram("rat", "car")
```

Out[10]: False

In [1]:
```
def is_subsequence(s,t):
    i,j=0,0
    while i<len(s) and j<len(t):
        if s[i]==t[j]:
            j+=1
        i+=1
    return j==len(t)
is_subsequence("abcde","ace")
```

Out[1]: True

In [3]:
```
def longest_word(s):
    max_length=0
    current_length=0

    for char in s:
        if char!=' ':
            current_length+=1
        else:
            if current_length>max_length:
                max_length=current_length
            current_length=0
    if current_length>max_length:
        max_length=current_length
    return max_length
s=longest_word("The quick brown fox jumps over the lazy dog")
print(s)
```

```
5
```

In [5]:
```
def check_substring(s,t):
    if t in s:
        return True
    else:
        return False
check_substring("hello world","word")
```

Out[5]:
```
False
```

In [8]:
```
def decimal_to_binary(n):
    rem=""
    while(n!=0):
        rem = str(n%2) + rem
        n=n//2
    return int(rem) if rem else 0
decimal_to_binary(14)
```

Out[8]:
```
1110
```

In [11]:
```
def binary_to_decimal(bn):
    dec,i=0,0
    while bn>0:
        r=bn%10
        exp=r*(2**i)
        dec=dec+exp
        bn=bn//10
        i+=1
    return dec
binary_to_decimal(101)
```

Out[11]:
```
5
```

In [ ]: