Milind Mali / Data Scientist

#1.basic excercise for begineers

#2.python input and output exercises

#3.files and exception

#4.python loop exercises

#5.python function exercises

#6.python string exercises

#7.python data structure exercises

#8.python list exercises

#9.python dictionary exercises

#10 python set exercises

#11.python tuple exercises

#12.python date and time exercises

#13.python OOP exercises

#14.python JSON exercises

#15.python Numpy exercises

#16.python pandas exercises

#17.python matplotlib exercises

#18.python random data generation exercises

1. Basic Exercise for Beginners

Excercise 1: Swap the number without using third varible

In [103]:

```
x=int(input("enter the first number "))
y=int(input("enter the Second number "))

print(x,y)

x=x+y
y=x-y
x=x-y

print(x,y)
```

```
enter the first number 4
enter the Second number 5
4 5
5 4
```

Exercise 2: Write a Program to extract each digit from an integer in the reverse order.

For example, If the given int is 7536, the output shall be "6 3 5 7", with a space separating the digits.

In [9]:

```
n=input("enter the number: ")

y=n[::-1]

k=" ".join(y)

print(k)
```

```
enter the number: 1234
4 3 2 1
```

Excercise 3: Write a program that will give you the sum of 3 digits

In [35]:

```
x=int(input("enter three digit number"))

a=x%10   # we will get last digit

num=x//10  # integer-divison here we will get first two digit

b=num%10   # here we will get last digit of two digit number

c=num//10 # here will get first digit of two digit number

print(a+b+c)
```

```
enter three digit number567
18
```

Excercise 4: Write a program that will reverse a four digit number.Also it checks whether the reverse

In [37]:

```
x=int(input("enter the four digit number "))

a=x%10  # to get last number

num_1=x//10 # to get first three numbers

b=num_1%10   # this way i will get 2nd last number

num_2=num_1//10  # here will get first two digit number

c=num_2%10  # we will get 2 nd number

d=num_2//10  # here we will get 1st digit

#formula for reverse

rev=a*1000+b*100+c*10+d

print(rev)

#now let check whether both number are equal or not

if x==rev:
    print(True)
else:
    print(False)
```

```
enter the four digit number 4567
7654
False
```

Excercise 5: Write a program to find the euclidean distance between two coordinates.

In [40]:

```
import math
```

In [41]:

```
x1=float(input("x1: "))
y1=float(input("y1: "))
x2=float(input("x2: "))
y2=float(input("y2: "))

x=[x1,y1]
y=[x2,y2]

print("="*100)

print("Eucledian distance for given co-ordinate will be",round(math.dist(x,y),2))
```

```
x1: 4
y1: 5
x2: 6
y2: 8
====================================================================================================
==========================
Eucledian distance for given co-ordinate will be 3.61
```

Excercise 6: Write a program that will tell whether the given number is divisible by 3 & 6.

In [49]:

```
num=int(input("enter the number "))

if num%3==0 and num%6==0:
    print("the number is divisible by 3 and 6")
else:
    print("the number is not divisible by 3 and 6")
```

```
enter the number 45
the number is not divisible by 3 and 6
```

Excercise 7: Write a program that will take three digits from the user and add the square of each digit.

In [53]:

```
n=int(input("Enter three digit number: "))

#123

a=n%10  #3

num=n//10  #12

b=num%10  #2

c=num//10  #1

output_value=(a**2)+(b**2)+(c**2)

print(output_value)
```

```
Enter three digit number: 345
50
```

Excercise 8: Write a program that will check whether the number is armstrong number or not.

An Armstrong number is one whose sum of digits raised to the power three equals the number itself. 371, for example, is an Armstrong number because 3**3 + 7**3 + 1**3 = 371.

In [54]:

```
n=int(input("Enter three digit number: "))

a=n%10  #3

num=n//10  #12

b=num%10  #2

c=num//10  #1

if (a**3 + b**3 + c**3)==n:
    print("the number is armstrong number")
else:
    print("the number is not armstrong number")
```

```
Enter three digit number: 121
the number is not armstrong number
```

Excercise 9:Write a program that will take user input of (4 digits number) and check whether the number is narcissist number or not.

In [55]:

```
n=int(input("Enter Four digit number: "))

#1234

a=n%10  #4
num=n//10  #123
b=num%10  #3
num_1=num//10  #12
c=num_1%10 #2
d=num_1//10  #1


if (a**4 + b**4 + c**4+d**4)==n:
    print("the number is narcissist number")
else:
    print("the number is not narcissist number")
```

```
Enter Four digit number: 1234
the number is not narcissist number
```

2. python input and output exercise

Exercise 1: Accept numbers from a user

In [125]:

```
name=input("enter the name: ")
```

```
enter the name: milind
```

Exercise 2: Display three string “Name”, “Is”, “James” as “Name ** Is ** James”

In [126]:

```
print("name","is","james",sep="**")
```

```
name**is**james
```

Exercise 3: Convert Decimal number to octal using print() output formatting

The octal number of decimal number 8 is 10

In [129]:

```
number=12

print(oct(number)[-2:])
```

```
14
```

Exercise 4: Display float number with 2 decimal places using print()

In [137]:

```
num=56.87547

y=float(round(num,2))

print(y)
```

```
56.88
```

Excercise 5:Print all factors of a given number provided by the user.

In [81]:

```
n=int(input("enter the number: "))

for i in range(1,n+1):
    if n%i==0:
        print(i,end=" ")
```

```
enter the number: 45
1 3 5 9 15 45
```

Exercise 6: Accept a list of 5 float numbers as an input from the user

In [144]:

```
l1=[]

while len(l1)<5:
    n=float(input("enter the number: "))
    l1.append(n)

print(l1)
```

```
enter the number: 67.65
enter the number: 98.98
enter the number: 655.65
enter the number: 78.9997
enter the number: 65.545
[67.65, 98.98, 655.65, 78.9997, 65.545]
```

Type *Markdown* and LaTeX: α^2

Exercise 7: Accept any three string from one input() call

In [196]:

```
inp=input("enter the three names with keeping space in between: ")

sep=inp.split()

#print(sep)

name1=sep[0]
name2=sep[1]
name3=sep[2]

print(name1)
print(name2)
print(name3)
```

```
enter the three names with keeping space in between: govind damodar madhav
eti
govind
damodar
madhaveti
```

Exercise 8: Format variables using a string.format() method.

Write a program to use string.format() method to format the following three variables as per the expected output

Given:

```
totalMoney = 1000
quantity = 3
price = 450
```

Expected Output:

```
I have 1000 dollars so I can buy 3 football for 450.00 dollars.
```

In [204]:

```
totalmoney=1200
quantity=4
price=450

print("i have {} dollars so i can buy {} football for {} dollars".format(totalmoney,quant
```

```
i have 1200 dollars so i can buy 4 football for 450 dollars
```

Excercise 9: Write a program to find the simple interest when the value of principle,rate of interest and time period is given.

In [45]:

```
P =float(input("principle amount: "))
R =float(input("Rate of interest: "))
T =int(input("for the time period: "))

Simple_interest= (P*R*T)/100

print(Simple_interest)

total_due=P+Simple_interest

print("Total due amount will need to pay wll be ",total_due)
```

```
principle amount: 67000
Rate of interest: 12
for the time period: 5
40200.0
Total due amount will need to pay wll be  107200.0
```

Excercise 10: Write a program to find the volume of the cylinder. Also find the cost when ,when the cost of 1litre milk is 40Rs.

In [47]:

```
rad=float(input("enter the radius of cylinder in cm "))
ht=float(input("enter the height of cylinder in cm "))

vol=3.142*(rad**2)*ht
litr=vol/1000
cost=litr*40

print("Volume of cylinder will be ",vol)
print("How much milk we can carry in this cylinder ",litr, 'ltr')
print("this cost of that milk will be",cost)
```

```
enter the radius of cylinder in cm 56
enter the height of cylinder in cm 5
Volume of cylinder will be  49266.56
How much milk we can carry in this cylinder  49.26656 ltr
this cost of that milk will be 1970.6624
```

3.files and exceptions

what are the files and what are the exceptions

Your programs can read information in from files, and they can write data to files.

Reading from files allows you to work with a wide variety of information; writing to files allows users to pick up where they left off the next time they run your program.

You can write text to files, and you can store Python structures such as lists in data files.

Exceptions are special objects that help your programs respond to errors in appropriate ways. For example if

Exercise 1: Check file is empty or not,Write a program to check if the given file is empty or not

In [205]:

```
import os
size=os.stat("test file.txt").st_size

if size==0:
    print("file is empty")
else:
    print("file is not empty")
```

```
file is not empty
```

Excercise 2:read content from the file

In [2]:

```
filename="siddharth.txt"

with open(filename) as f_obj:
    content=f_obj.read()

print(content)
```

```
i'm aspiring data scientist
i love to work with data
i love making the games
i love python
```

Excercise 3:# read line by line

In [3]:

```
filename="siddharth.txt"

with open (filename) as f_obj:
    for line in f_obj:
        print(line.rstrip())
```

```
i'm aspiring data scientist
i love to work with data
i love making the games
i love python
```

In [4]:

```
#storing line to the list
filename="siddharth.txt"

with open(filename) as f_obj:
    lines=f_obj.readlines()

for line in lines:
    print(line.rstrip())
```

```
i'm aspiring data scientist
i love to work with data
i love making the games
i love python
```

Excercise 4:now we will learn about writing to the files

passing 'w' argument to open() tells python you want to write to the file

be carefull: this will erase the contents of the file if it already exists

passing the 'a' arguement tell python you want to append to the end of an existing file

In [5]:

```
filename="programming.txt"

with open(filename,'w') as f:
    f.write("i love machine learning as well")
```

Excercise 5: writing multiple line to empty file

In [7]:

```
with open(filename,'w') as f:
    f.write("Deep learning is awesome.\n")
    f.write("AI is future\n")
```

In [8]:

```
with open(filename,'a') as f:
    f.write("i love to work with data science projects \n")
    f.write("i love making the application")
```

```
### syntax for opening the file using path

filename="< file path >"

with open(filename) as f:
    lines=f.readlines()
```

Exercise 6: Write all content of a given file into a new file by skipping line number 5

Create a test.txt file and add the below content to it.

Given test.txt file:

```
line1
line2
line3
line4
line5
line6
line7
```

Expected Output: `new_file.txt`

```
line1
line2
line3
line4
line6
line7
```

In [9]:

```
#read test file
with open ("test file.txt",'r') as fp:
    #read all the lines from the file
    lines=fp.readlines()

#open new file in write mode

with open ("test file new.txt","w") as fp:
    count=0
    #iterate each line from test file
    for line in lines:
        if count==4:
            count=+1
            continue
        else:
            fp.write(line)
        count+=1
```

Exercise 7: Read line number 4 from the following file

In [10]:

```
with open("test file.txt","r") as fp:
    lines=fp.readlines()
    print(lines[3])
```

```
HR analytics power bi project
```

The Try and Except block

- when you think error may occur .you can write try and except block to handle the exception that might be raised
- the try block tells python to try runnins some code and except block tells python what to do if the code result in a particular kind or error

Excercise 8: Let see handling zero division exceptional error

In [10]:

```
print(5/0)
```

```
---------------------------------------------------------------------------
-
ZeroDivisionError                         Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_13444\1152173066.py in <module>
----> 1 print(5/0)

ZeroDivisionError: division by zero
```

In [11]:

```
try:
    print(5/0)
except ZeroDivisionError:
    print("you cant divide by zero")
```

```
you cant divide by zero
```

In [11]:

```
filename="gautam.txt"

try:
    with open(filename) as f:
        content=f.read()
    print(content)
except FileNotFoundError:
    print(f"Dear user {filename} may be this file dont exist in directory!")
```

```
Dear user gautam.txt may be this file dont exist in directory!
```

the else block

- the try block should only contain the code that may cause an error
- any code that depends on try block running succesfully should be placed in the else block

Excercise 9:using else block

In [13]:

```
x=int(input("enter the number: "))
y=int(input("enter the number: "))

try:
    result=x/y
except ZeroDivisionError:
    print("hey you...denominater cant be zero")
else:
    print(result)
```

```
enter the number: 5
enter the number: 0
hey you...denominater cant be zero
```

preventing crashes from user input

- without except block in following example , the programe would crash if user try to divide by zero
- as written it will handle the error gracefully and keep running

In [14]:

```
# create simple calculater

print("enter the numbers: ")
print("enter the 'q' to exit")

while(True):
    x=input("enter the number:")
    if x=='q':
        break
    y=input("enter the number:")
    if y=='q':
        break

    try:
        result=int(x)/int(y)
    except ZeroDivisionError:
        print("bhidu zero se divide nahi kar sakate")
    else:
        print(result)
```

```
enter the numbers:
enter the 'q' to exit
enter the number:3
enter the number:4
0.75
enter the number:q
```

Excercise 10: how does finally block work in python

In [1]:

```
# no matter what if your exception cought or not your finally code always executed

try:
    print(5/2)
except ZeroDivisionError:
    print("you cant divide by zero")
finally:
    print("Hip Hip hurray!!")
```

```
2.5
Hip Hip hurray!!
```

4. Python if else, for loop,while loop and range() Exercises with Solutions

Exercise 1: Print the sum of the current number and the previous number

Write a program to iterate the first 10 numbers and in each iteration, print the sum of the current and previous number.

Expected result: 0,1,3,5,7,9,11,13,15,17

In [10]:

```
prev_num=0

for i in range(0,11):
    x=i+prev_num
    prev_num=i

    print(x,end=" ")
```

```
0 1 3 5 7 9 11 13 15 17 19
```

Excercise 2: Write a python program to search a given number from a list

In [89]:

```
l1=[4,5,6,2,3,9,1,4,5,6,3]

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

for i in l1:
    if i==n:
        print("Number exist")
        break
else:
    print("number dont exist")
```

```
enter the number: 5
Number exist
```

Exercise 3:Write a program to check if the given number is a palindrome number.

A palindrome number is a number that is same after reverse. For example 545, is the palindrome numbers

In [14]:

```
n=input("enter the number: ")

rev_number=n[::-1]

print(rev_number)

if n==rev_number:
    print("the given number is palindrom")
else:
    print("the given number is not palindrom")
```

```
enter the number: 343
343
the given number is palindrom
```

Exercise 4: Create a new list from a two list using the following condition. Given a two list of numbers, write a program to create a new list such that the new list should contain odd numbers from the first list and even numbers from the second list.

In [100]:

```
l1=[1,2,3,4,5]
l2=[6,7,8,9,10]

l3=[]

for i in l1:
    if i%2!=0:
        l3.append(i)

for i in l2:
    if i%2==0:
        l3.append(i)

print(l3)
```

```
[1, 3, 5, 6, 8, 10]
```

Exercise 5: Calculate income tax for the given income by adhering to the below rules

first 10k--> 0%

second 10 --> 10%

remaining-->20%

Expected Output:

For example, suppose the taxable income is 45000 the income tax payable is

100000% + *1000*10% + 25000*20% = $6000.

In [96]:

```
n=int(input("enter the number: "))

if n<=10000:
    print(n)
else:
    a=n-10000
    b=a-10000
    c=b*0.2
    d=c+1000

    print(d)
```

```
enter the number: 45000
6000.0
```

In [15]:

```
# Another way of doing the same

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

a=10000
b=10000

c=n-(a+b)

tax=0

if n<=a:
    tax=0
elif n>a and n<20000:
    tax=b*0.1
else:
    tax=(b*0.1)+(c*0.2)

print(tax)
```

```
enter the number: 45000
6000.0
```

Exercise 6: Print multiplication table form 1 to 10

```
1  2 3 4 5 6 7 8 9 10
2  4 6 8 10 12 14 16 18 20
3  6 9 12 15 18 21 24 27 30
4  8 12 16 20 24 28 32 36 40
5  10 15 20 25 30 35 40 45 50
6  12 18 24 30 36 42 48 54 60
7  14 21 28 35 42 49 56 63 70
8  16 24 32 40 48 56 64 72 80
9  18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

In [101]:

```
for i in range(1,11):
    for j in range(1,11):
        print(i*j,end=" ")
    print("\n")
```

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

2 4 6 8 10 12 14 16 18 20

3 6 9 12 15 18 21 24 27 30

4 8 12 16 20 24 28 32 36 40

5 10 15 20 25 30 35 40 45 50

6 12 18 24 30 36 42 48 54 60

7 14 21 28 35 42 49 56 63 70

8 16 24 32 40 48 56 64 72 80

9 18 27 36 45 54 63 72 81 90

10 20 30 40 50 60 70 80 90 100
```

Excercise 7: Print all the armstrong numbers in the range of 100 to 1000

In [69]:

```
armstrong_list=[]

for i in range(100,1000):
    a=i%10  # 123-->3
    num=i//10 #123-->12
    b=num%10  #--->2
    c=num//10  #-->1

    if (a**3)+(b**3)+(c**3)==i:
        armstrong_list.append(i)

print(armstrong_list)
```

```
[153, 370, 371, 407]
```

Excercise 8: The current population of a town is 10000. The population of the town is increasing at the rate of 10% per year. You have to write a program to find out the population at the end of each of the last 10 years. For eg current population is 10000 so the output should be like this:

*** 10th year - 10000**

*** 9th year - 9000**

*** 8th year - 8100 and so on**

In [71]:

```
p=10000

for i in range(1,10):
    p=p-0.1*p
    print(round(p),end=" ")
```

```
9000 8100 7290 6561 5905 5314 4783 4305 3874
```

Excercise 9: Write a program to print all the unique combinations of two digits from 1 to 4 for ex (1,2), (2,3).....

In [72]:

```
# since we have 4 digit there unique combination will be 16

for i in range(1,5):
    for j in range(1,5):
        if i != j:
            print(i,j)
```

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

Excercise 10:Write a program to print whether a given number is prime number or not

In [68]:

```
x=int(input("enter the number: "))

new=[]

if x==1:
    print("1 is not prime number")
else:
    for i in range(1,x+1):
        if x%i==0:
            new.append(i)
            #print(new)

    if len(new) > 2:
        print("it is not a prime number")
    else:
        print("it is prime number ")
```

```
enter the number: 45
it is not a prime number
```

Excercise 11: User will provide 2 numbers you have to find the HCF of those 2 numbers

In [73]:

```
x=int(input("enter the number: "))
y=int(input("enter the number: "))

x_div=[]
y_div=[]

for i in range(1,x+1):
    if x%i==0:
        x_div.append(i)

for i in range(1,y+1):
    if y%i==0:
        y_div.append(i)

comman_list=[]

for i in x_div:
    if i in y_div:
        comman_list.append(i)

print("HCF of given two number is ",max(comman_list))
```

```
enter the number: 34
enter the number: 56
HCF of given two number is  2
```

Excercise 12: User will provide 2 numbers you have to find the by LCM of those 2 numbers

In [2]:

```
a=int(input("enter the first number: "))
b=int(input("enter the second number: "))

if a>b:
    greater=a
else:
    greater=b

while(True):
    if (greater%a==0)  and (greater%b==0):
        LCM=greater
        break
    greater+=1

print(LCM)
```

```
enter the first number: 3
enter the second number: 4
12
```

Excercise 13: Write a program that take a user input of three angles and will find out whether it can form a triangle or not.

In [42]:

```
# since sum of all the angle of trianle is 180 deg

a=float(input("1st angle of triangle: "))
b=float(input("2nd angle of triangle: "))
c=float(input("3rd angle of triangle: "))

if (a+b+c)==180 and a!=0 and b!=0 and c!=0:
    print("yes it can form a triangle")
else:
    print("No it cant form a triangle")
```

```
1st angle of triangle: 90
2nd angle of triangle: 90
3rd angle of triangle: 90
No it cant form a triangle
```

Excercise 14:Write a program that will determine weather when the value of temperature and humidity is provided by the user.

```
TEMPERATURE(C)      HUMIDITY(%)      WEATHER

     >= 30          >=90             Hot and Humid
     >= 30          < 90             Hot
     <30            >= 90            Cool and Humid
     <30            <90              Cool
```

In [51]:

```
temp=int(input("enter the value of temp "))
humidity=int(input("enter the value of humidity "))

if temp>=30 and humidity>=90:
    print("Hot and Humid")
elif temp>=30 and humidity<90:
    print("Hot")
elif temp<30 and humidity>=90:
    print("cool and humid")
else:
    print("cool")
```

```
enter the value of temp 45
enter the value of humidity 56
Hot
```

Excercise 15:Write a program that will take user input of cost price and selling price and determines whether its a loss or a profit

In [43]:

```
cost=int(input("what is cost of product: "))
sell=int(input("what is sell price of product: "))

if (sell-cost)>0:
    amount=sell-cost
    print("it is profit by ",amount)
else:
    amount=sell-cost
    print("it is loss by ",amount)
```

```
what is cost of product: 67
what is sell price of product: 89
it is profit by  22
```

Excercise 16-Write a program that will give you the in hand salary after deduction of

HRA(10%),

DA(5%),

PF(3%),

and tax

(if salary is between 5-10 lakh–10%),

(11-20lakh–20%),

(20< _ – 30%)

(0-1lakh print k).

In [56]:

```
salary=int(input("enter the salary amount: "))

if salary in range(500000,1000000):
    salary=salary-salary*0.1
elif salary in range(1100000,2000000):
    salary=salary-salary*0.2
elif salary> 2000000:
    salary=salary-salary*0.3
else:
    print("no tax")

print("Salary after tax cutting",salary)

HRA=salary*0.10   # 63000
DA=salary*0.05   #31500
PF=salary*0.03   #18900

remain=salary-(HRA+DA+PF)

#print("in hand salary after HRA/DA/PF cutting",remain)

if remain in range(1000,99999):
    print("in hand salary after tax/HRA/DA/PF cutting",remain/1000,"k")
elif remain in range(100000,9999999):
    print("in hand salary after tax/HRA/DA/PF cutting",remain/100000,"lakh")
else:
    print("in hand salary after tax/HRA/DA/PF cutting",remain/10000000,"Cr")
```

```
enter the salary amount: 456789
no tax
Salary after tax cutting 456789
in hand salary after tax/HRA/DA/PF cutting 0.037456698 Cr
```

Excercise 17: Write a menu driven program -

1.cm to ft

2.kl to miles

3.usd to inr

4.exit

In [57]:

```
user_input=int(input("""Enter 1 to 'cm' to 'ft'
Enter 2 to 'km' to 'miles'
Enter 3 to  'USD' to 'INR'
Enter 4 to exit """))

if user_input==1:
    var1=int(input("enter the value in cm"))
    output=var1/30.48
    print(output,'ft')

elif user_input==2:
    var2=int(input("enter the value in km"))
    output=var2/1.609
    print(output,'miles')

elif user_input==3:
    var3=int(input("enter the value in usd"))
    output=var3*81.80
    print(output,"Inr")

else:
    print("Exit")
```

```
Enter 1 to 'cm' to 'ft'
Enter 2 to 'km' to 'miles'
Enter 3 to  'USD' to 'INR'
Enter 4 to exit 2
enter the value in km34
21.13113735239279 miles
```

Excercise 18:Write a program that will tell the number of dogs and chicken are there when the user will provide the value of total heads and legs.

In [59]:

```
# number of head-->     1                          1
# number of legs-->     4                          2

head=int(input("enter the number of heads: "))
legs=int(input("enter the number of legs: "))

if head/legs==0.25:
    number_of_dogs=legs/4
    print("there are ",round(number_of_dogs),'number of dogs')
elif head/legs==0.5:
    number_of_chicken=legs/2
    print("there are ",round(number_of_chicken),'number of chicken')
else:
    print("enter correct numbers")
```

```
enter the number of heads: 2
enter the number of legs: 4
there are  2 number of chicken
```

Excercise 19: Write a program to find the sum of first n numbers, where n will be provided by the user. Eg if the user provides n=10 the output should be 55.

In [17]:

```
user_input=int(input("enter the number: "))

x=0

for i in range(1,user_input+1):
    x=x+i
print(x)
```

```
enter the number: 10
55
```

Excercise 20: Write a program that can multiply 2 numbers provided by the user without using the * operator

In [62]:

```
x=int(input("enter the first number: "))  #2
y=int(input("enter the seconde number "))   #3

result=0

for i in range(0,y):
    result=result+x

print(result)
```

```
enter the first number: 3
enter the seconde number 4
12
```

Excercise 21:Write a program that can find the factorial of a given number provided by the user.

In [64]:

```
n=int(input("enter the number : "))

factorial=1

if n < 0 :
    print("factorial dont exist for negative number")
elif n==0:
    print("for zero factorial is always one")
else:
    for i in range(1,n+1):
        factorial=factorial*i
    print(factorial)
```

```
enter the number : 4
24
```

Excercise 22:print following pattern

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

In [4]:

```
n=int(input("enter the number: "))

for i in range(0,n+1):
    for j in range(0,i):
        print(i,end=" ")
    print("\n")
```

```
enter the number: 5


1

2 2

3 3 3

4 4 4 4

5 5 5 5 5
```

Exercise 23: Print First 10 natural numbers using while loop

In [5]:

```
while i <10:
    for i in range(1,11):
        print(i,end=" ")
```

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

Exercise 24: Write a program to count the total number of digits in a number using a while loop.

For example, the number is 75869, so the output should be 5.

In [246]:

```
n=input("enter the number: ")

x=list(n)

count=0

while count<len(x):
    for i in x:
        count+=1

print(count)
```

```
enter the number: 66464
5
```

Excercise 25: Write a program that keeps on accepting a number from the user until the

user enters Zero. Display the sum and average of all the numbers.

In [29]:

```
n=int(input("enter the number: "))

total=0
avg=0
count=0

while True:
    if n != 0:
        total=total+n
        count +=1
        avg=total/count
        n=int(input("print another number: "))

    else:
        print("Thank you")
        break

print("your sum is:",total)
print("your avg is:",round(avg,2))
```

```
enter the number: 56
print another number: 0
Thank you
your sum is: 56
your avg is: 56.0
```

Excercise 26: Write a program to print the first 25 odd numbers

In [66]:

```
# get me first 25 odd number

# i dont know how many time loop will run--> while loop
flag=0
i=1

odd_list=[]

while True:
    if i%2 != 0:
        odd_list.append(i)
        flag=flag+1
    if flag==25:
        break
    i=i+1

print(odd_list)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 3
9, 41, 43, 45, 47, 49]
```

Exercise 27: Print the following pattern

In [8]:

```
n=int(input('enter the number: '))

#1,2,3,4,5

for i in range(1,n+1):
    for j in range(1,i+1):
        print(j,end=" ")

    print("\n")
```

```
enter the number: 5
1

1 2

1 2 3

1 2 3 4

1 2 3 4 5
```

Excercise 28

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

In [9]:

```
n=int(input("enter the number: "))

num=1

for i in range(0,n):
    for j in range(0,i+1):
        print(num,end=" ")
        num=num+1
    print()
```

```
enter the number: 5
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Excercise 29: Print Pascal Triangle

In [27]:

```
n=int(input("enter the number: "))

list1=[]

for i in range(0,n):
    temp=[]
    for j in range(0,i+1):
        if j==0 or j==i:
            temp.append(1)
        else:
            temp.append(list1[i-1][j]+list1[i-1][j-1])
    list1.append(temp)

#print(list1)

for i in range(n):
    for j in range(0,n-i-1):
        print(" ",end="")

    for j in range(0,i+1):
        print(list1[i][j],end=" ")

    print()
```

```
enter the number: 5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

Exercise 30: Write a program to print multiplication table of a given number For example, num = 2 so the output should be 2,4,6,8....20.

In [233]:

```
n=int(input("enter the number: "))

for i in range(1,11):
    x=n*i
    print(x,end=" ")
```

```
enter the number: 16
16 32 48 64 80 96 112 128 144 160
```

Exercise 31: Write a program to display only those numbers from a list that satisfy the following conditions

The number must be divisible by five

If the number is greater than 150, then skip it and move to the next number

If the number is greater than 500, then stop the loop

In [236]:

```
num=[12,75,150,180,145,525,50]

for i in num:
    if i>500:
        break
    elif i>150:
        continue
    else:
        if i%5==0:
            print(i)
```

```
75
150
145
```

Exercise 32: Print the following pattern

Write a program to use for loop to print the following reverse number pattern

In [263]:

```
n=int(input('enter the number: '))

for i in range(n,0,-1):
    for j in range(i,0,-1):
        print(j,end=" ")

    print("\n")
```

```
enter the number: 5
5 4 3 2 1

4 3 2 1

3 2 1

2 1

1
```

Exercise 33: Print list in reverse order using a loop

In [265]:

```
l1=[1,2,3,4,5]

for i in l1[::-1]:
    print(i,end=" ")
```

```
5 4 3 2 1
```

Exercise 34: Display numbers from -10 to -1 using for loop

In [273]:

```
for i in range(-10,0):
    print(i,end=" ")
```

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

Exercise 35: Use else block to display a message “Done” after successful execution of for loop

For example, the following loop will execute without any error.

In [277]:

```
for i in range(0,5):
    print(i)
else:
    print("done !!")
```

```
0
1
2
3
4
done !!
```

Exercise 36: Write a program to display all prime numbers within a range

Note: A Prime Number is a number which only divisible by one and number itself.but 1 is not prime number

In [322]:

```
def prime(n):

    if n<=1:
        return False
    else:
        l1=[]

        for i in range(1,n+1):
            if n%i==0:
                l1.append(i)

        #print(l1)

        if len(l1)<=2:
            return True
        else:
            return False

inp1=int(input("enter the number : "))
inp2=int(input("enter the number : "))


for i in range(inp1,inp2+1):
    if prime(i)==True:
        print(i,end=" ")
```

```
enter the number : 25
enter the number : 50
29 31 37 41 43 47
```

Exercise 37: Display Fibonacci series up to 10 terms

The Fibonacci Sequence is a series of numbers. The next number is found by adding up the two numbers before it. The first two numbers are 0 and 1.

For example, 0, 1, 1, 2, 3, 5, 8, 13, 21. The next number in this series above is 13+21 = 34.

In [20]:

```
n=int(input("enter the number: "))

num1=0
num2=1

print(num1,end=" ")
print(num2,end=" ")

for i in range(0,n):

    num3=num1+num2
    num1=num2
    num2=num3


    print(num3,end=" ")
```

```
enter the number: 9
0 1 1 2 3 5 8 13 21 34 55
```

Exercise 38: Write a program to use the loop to find the factorial of a given number.

The factorial (symbol: !) means to multiply all whole numbers from the chosen number down to 1.

In [18]:

```
n=int(input("enter the number: "))

fact=1

for i in range(1,n+1):
    fact=fact*i

print(fact)
```

```
enter the number: 5
120
```

Exercise 39: Reverse a given integer number

Given:

76542

Expected output:

24567

In [36]:

```
n=int(input("enter the number: "))

x=str(n)

x=list(x)[::-1]

z=int("".join(x))

print(z)
#print(type(z))
```

```
enter the number: 76542
24567
```

Exercise 40: Use a loop to display elements from a given list present at odd index positions

my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

In [42]:

```
my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

my_list[1::2]   #using list slicing
```

Out[42]:

```
[20, 40, 60, 80, 100]
```

Exercise 41: Write a program to print the cube of all numbers from 1 to a given number

Given:

input_number = 6

In [44]:

```
n=6
for i in range(1,7):
    x=pow(i,3)
    #print(x)
    print("current number is {} and the cube is {}".format(i,x))
```

```
current number is 1 and the cube is 1
current number is 2 and the cube is 8
current number is 3 and the cube is 27
current number is 4 and the cube is 64
current number is 5 and the cube is 125
current number is 6 and the cube is 216
```

Exercise 42: Write a program to calculate the sum of series up to n term. For example, if n =5 the series will become 2 + 22 + 222 + 2222 + 22222 = 24690

In [53]:

```
n=5

x=2
result=0

for i in range(1,6):
    y=str(x)*i
    z=int("".join(y))
    #print(z)
    result=result+z

#type(z)
print(result)
```

```
24690
```

Exercise 43: Print downward Half-Pyramid Pattern with Star (asterisk)

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

In [119]:

```
n=int(input("enter the number : "))

for i in range(n+1,1,-1):
    for j in range(1,i):
        print("*",end=" ")
    print("\n")
```

```
enter the number : 5
* * * * *

* * * *

* * *

* *

*
```

Exercise 44: print right half side of the pyramid

In [25]:

```
n=int(input("enter the number: "))

for i in range(1,n+1):
    for j in range(1,i+1):
        print("*",end=" ")

    print("\r")
```

```
enter the number: 5
*
* *
* * *
* * * *
* * * * *
```

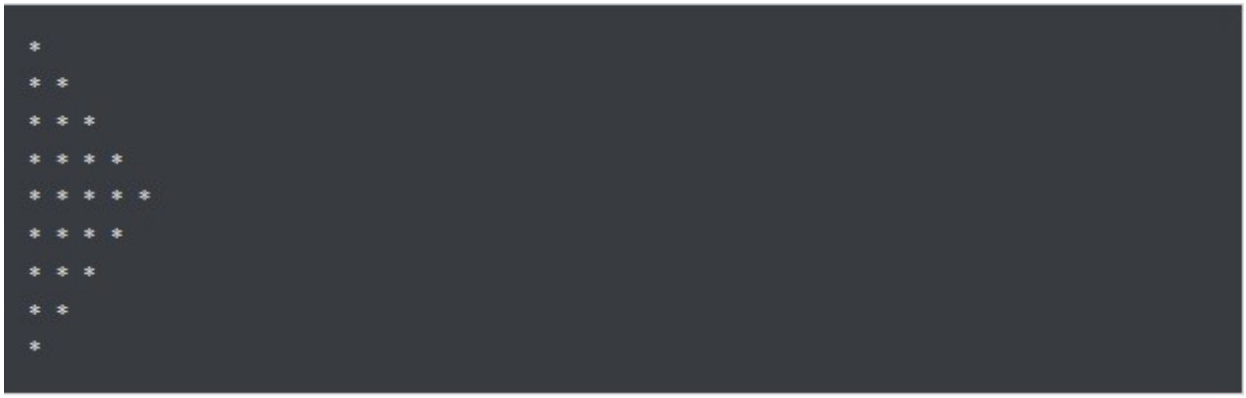
Excercise 45

In [26]:

```
#n=int(input("enter the number: "))
n=5

for i in range(0,n):

    for j in range(0,n-i):
        print(end=" ")

    for k in range(0,i+1):
        print("*",end=" ")

    print()
```

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

Exercise 46: Write a program to print the following start pattern using the for loop



In [71]:

```
n=int(input("enter the number: "))

for i in range(1,n+1):
    for j in range(1,i+1):
        print("*",end=" ")
    print("\n")

for i in range(n-1,0,-1):
    for j in range(0,i):
        print("*",end=" ")
    print("\n")
```

```
enter the number: 5
*

* *

* * *

* * * *

* * * * *

* * * *

* * *

* *

*
```

5. Python Functions Exercise

Exercise 1: Write a function called exponent(base, exp) that returns an int value of base raises to the power of exp.

Note here exp is a non-negative integer, and the base is an integer.

In [102]:

```
def exponent(base,exp):
    result=pow(base,exp)
    print(result)

exponent(10,2)
```

```
100
```

Exercise 2: Write a program to create a function that takes two arguments, name and age, and print their value.

In [98]:

```
def biodata(name,age):
    print("Name of the person is {} and age is {}".format(name,age))

biodata("Milind",26)
```

```
Name of the person is Milind and age is 26
```

Exercise 3: Write a function to return True if the first and last number of a given list is same. If numbers are different then return False.

In [97]:

```
l1=[1,2,3,4,5,1]

num1=l1[0]
num2=l1[-1]

def xyz(list):
    if num1==num2:
        return True
    else:
        return False

xyz(l1)
```

Out[97]:

```
True
```

Exercise 4:Given two integer numbers return their product only if the product is equal to or lower than 1000, else return their sum.

In [14]:

```
num1=int(input("enter the number: " ))
num2=int(input("enter the number: " ))


def mul_sum_int(num1,num2):
    #calculate product of two number
    mul=num1*num2
    total=num1+num2
    #checking if product is less than 1000
    if mul<=1000:
        return mul
    else:
        return total

print(mul_sum_int(num1,num2))
```

```
enter the number: 34
enter the number: 34
68
```

Exercise 5: Create a function with variable length of arguments

Write a program to create function func1() to accept a variable length of arguments and print their value.

Note: Create a function in such a way that we can pass any number of arguments to this function, and the function should process them and display each argument's value.

In [81]:

```
def inputs(*num):
    return num
```

In [82]:

```
inputs(45,67,78)
```

Out[82]:

```
(45, 67, 78)
```

In [76]:

```
inputs(34,56)
```

```
(34, 56)
```

Excercies 6:collecting arbitrary number of arguments

In [10]:

```
def make_pizza(size,*toppings):
    print(f"making {size} pizza")
    print("toppings: ")
    for i in toppings:
        print(i)
```

In [11]:

```
make_pizza("small","butter")
```

```
making small pizza
toppings:
butter
```

In [12]:

```
make_pizza("medium","amli","paeri","dijvoe")
```

```
making medium pizza
toppings:
amli
paeri
dijvoe
```

Excercise 7: collecting arbitrary number of keyword arguments

In [12]:

```
def build_profile(fname,lname,**userinfo):
    # build dictionary
    profile={"first Name":fname,"Last Name":lname}

    for key,value in userinfo.items():
        profile["key"]=value

    return profile
```

In [13]:

```
build_profile("milind","mali")
```

Out[13]:

```
{'first Name': 'milind', 'Last Name': 'mali'}
```

In [14]:

```
build_profile("milind","mali",age=26,Loc="Pune")
```

Out[14]:

```
{'first Name': 'milind', 'Last Name': 'mali', 'key': 'Pune'}
```

Exercise 8: Return multiple values from a function

Write a program to create function calculation() such that it can accept two variables and calculate addition and subtraction. Also, it must return both addition and subtraction in a single return call.

In [79]:

```
def add_sub(n1,n2):
    x1=n1+n2
    x2=n1-n2

    return x1,x2
```

In [80]:

```
add_sub(40,10)
```

Out[80]:

```
(50, 30)
```

Exercise 9: Create a function with a default argument

Write a program to create a function show_employee() using the following conditions.

It should accept the employee's name and salary and display both.

If the salary is missing in the function call then assign default value 9000 to salary

In [83]:

```
def show_employee(name,salary="9000"):
    print("name of employee is {} and his\her salary is {}".format(name,salary))
```

In [85]:

```
show_employee("milind",78686)
```

```
name of employee is milind and his\her salary is 78686
```

In [86]:

```
show_employee("Vishal")
```

```
name of employee is Vishal and his\her salary is 9000
```

Exercise 10: Create an inner function to calculate the addition in the following way

Create an outer function that will accept two parameters, a and b

Create an inner function inside an outer function that will calculate the addition of a and b

At last, an outer function will add 5 into addition and return it

In [107]:

```
def outerfunc(n1,n2):

    def innerfunc(n1,n2):
        return n1+n2

    add=innerfunc(n1,n2)

    return add+5
```

In [108]:

```
outerfunc(5,10)
```

Out[108]:

```
20
```

Exercise 11: Create a recursive function

A] Write a program to create a recursive function to calculate the sum of numbers from 0 to 10.

A recursive function is a function that calls itself again and again.

Expected Output:

55

In [134]:

```
def addition(num):
    if num==0:
        return 0
    else:
        return num+addition(num-1)
```

In [135]:

```
addition(10)
```

Out[135]:

55

B] Write a program to create a recursive function to calculate the factorial of numbers from 0 to 10.

A recursive function is a function that calls itself again and again.

Expected Output:

120

In [132]:

```
def factorial(num):
    if (num==1):
        return 1
    else:
        return num*factorial(num-1)
```

In [133]:

```
factorial(5)
```

Out[133]:

120

Exercise 12: Assign a different name to function and call it through the new name

Below is the function display_student(name, age). Assign a new name show_student(name, age) to it and call it using the new name.

In [141]:

```
def display_student(name,age):
    print("name of student {} and his age is {}".format(name,age))

show_student=display_student
```

In [143]:

```
show_student("Mihir",29)
```

```
name of student Mihir and his age is 29
```

Exercise 13: Generate a Python list of all the even numbers between 4 to 30

In [145]:

```
for i in range(4,31):
    if i%2==0:
        print(i,end=" ")
```

```
4 6 8 10 12 14 16 18 20 22 24 26 28 30
```

In [146]:

```
# another way
list(range(4,31,2))
```

Out[146]:

```
[4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]
```

Exercise 14: Find the largest item from a given list

In [19]:

```
x = [4, 6, 8, 24, 12, 2,65]

y=0

for i in x:
    if i > y:
        y=i

print(y)
```

```
65
```

In [20]:

```
print(max(x))
```

```
65
```

Excercise 15: first 25 prime number

In [77]:

```
# first we will write code for getting prime number

a=23

new=[]

for i in range(1,a+1):
    if a%i==0:
        new.append(i)

if len(new)>2:
    print("it's not a prime number")
else:
    print("it's prime number")
```

```
it's prime number
```

In [78]:

```
# to get first 25 numbers

def is_prime(x):
    new=[]
    if x<=1:
        return false
    else:
        for i in range(1,x+1):
            if x%i==0:
                new.append(i)
    #print(new)
    if len(new)<=2:
        return True
    else:
        return False

#now will write the main code
count=0
num=2

while count<25:
    if is_prime(num)==True:
        print(num,end=" ")
        count+=1

    num+=1
```

```
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
```

Excercise 16: Print the first 20 numbers of a Fibonacci series

In [79]:

```
n=int(input("enter the number: "))


def fibonacci(n):

    a=0
    b=1
    count=0

    if n<=0:
        print("You can enter only Positive integer values")
    elif n==1:
        print(a)
    else:
        while count<n:
            print(a,end=' ')
            c=a+b
            a=b
            b=c

            count+=1

fibonacci(n)
```

```
enter the number: 12
0 1 1 2 3 5 8 13 21 34 55 89
```

Excercise 17: Write the logic for strong number strong number in python

#145-->1!+4!+5!

In [7]:

```
n=145

total=0

def fact(n):
    result=1
    for i in range(1,n+1):
        result*=i
    return result

for i in str(n):
    x=fact(int(i))
    total+=x

if n==total:
    print("It is strong number")
else:
    print("it is not a strong number")
```

```
It is strong number
```

Excercise 18: write logic to know whethe given number is perfect number or not

In [ ]:

```
from functools import reduce

n=28

l1=[]

for i in range(1,n):
    if n%i==0:
        l1.append(i)

result=reduce(lambda x,y:x+y,l1)

if n==result:
    print("perfect number")
else:
    print("Not perfect number")
```

6. Python string Excercise with solutions

Exercise 1: Return the count of a given substring from a string

Write a program to find how many times substring “radha” appears in the given string

"radha is most beutiful,radha is queen of vraj"

In [95]:

```
sentence="radha is most beutiful,radha is queen of vraj,radha is most beloved to govind"

x=sentence.count("radha")

print(x)
```

```
3
```

Exercise 2: Print characters from a string that are present at an even index number

Write a program to accept a string from the user and display characters that are present at an even index number.

For example, str = "pynative" so you should display ‘p’, ‘n’, ‘t’, ‘v’.

In [93]:

```
string=input("enter the text: ")

x=list(string)

print(string)

for i in x[0::2]:
    print(i,end=" ")
```

```
enter the text: milind
milind
m l n
```

Exercise 3: Write a program to remove characters from a string starting from zero up to n and return a new string.

For example:

remove_chars("pynative", 4) so output must be tive. Here we need to remove first four characters from a string. remove_chars("pynative", 2) so output must be native. Here we need to remove first two characters from a string. Note: n must be less than the length of the string.

In [94]:

```
def remove (word,n):

    x=len(word)
    p=list(word)

    for i in p:
        if n<=x:
            z=word[n:]
    print(z)


remove("pynative",2)
```

```
native
```

Exercise 4 1A: Create a string made of the first, middle and last character

Write a program to create a new string made of an input string's first, middle, and last character.

In [186]:

```
name="james"

a=name[0]
print(a)

c=name[-1]
print(c)

l=len(name)
x=int(l//2)

b=name[x]
print(b)

"".join([a,b,c])
```

```
j
s
m
```

Out[186]:

```
'jms'
```

Exercise 4B: Write a program to create a new string made of the middle three characters of an input string.

ex.JaSonAy-->Son

JhonDipPeta -->Dip

In [193]:

```
name="JaSonAy"

l=len(name)

if l%2==0:
    print("not possible")
else:
    c=l//2
    x=name[c-1]
    y=name[c]
    z=name[c+1]

    result="".join([x,y,z])

    print(result)
```

```
Son
```

Excercise 5 :Count the frequency of a particular character in a provided string. Eg 'hello how are you' is the string, the frequency of h in this string is 2.

In [83]:

```
a=input("enter the text: ")
b=input("enter the character: ")

count=0

for i in a:
    if i in b:
        count=count+1

print("frequency of searched character is ",count,"times")
```

```
enter the text: milind mali
enter the character: i
frequency of searched character is  3 times
```

Excercise 6:Find the index position of a particular character in another string.

In [84]:

```
a=input("enter the text: ")
b=input("enter the character: ")

print(a.index(b))
```

```
enter the text: milind mali
enter the character: d
5
```

Excercise 7: Write a program which can remove a particular character from a string.

In [85]:

```
a=input("enter the string: ")
b=input('enter the character you  want to remove: ')

a=a.replace(b,"")

print(a)
```

```
enter the string: milind
enter the character you  want to remove: i
mlnd
```

Exercise 8: Append new string in the middle of a given string

Given two strings, s1 and s2. Write a program to create a new string s3 by appending s2 in the middle of s1.

s1 = "Ault"

s2 = "Kelly"

expected-->AuKellylt

In [206]:

```
s1="Ault"

s2="Kelly"

l=len(s1)
mid=l//2

f=s1[:mid]
s=s1[mid:]

s3=f+s2+s

print(s3)
```

```
AuKellylt
```

Exercise 9: Create a new string made of the first, middle, and last characters of each input string

Given two strings, s1 and s2, write a program to return a new string made of s1 and s2's first, middle, and last characters.

s1 = "America"

s2 = "Japan"

Expected output-->AJrpan

In [212]:

```
def unite(s1,s2):
    l1=len(s1)//2
    a=s1[0]
    b=s1[l1]
    c=s1[-1]

    l2=len(s2)//2
    x=s2[0]
    y=s2[l2]
    z=s2[-1]

    result="".join([a,x,b,y,c,z])

    return result
```

In [213]:

```
unite("america","japan")
```

Out[213]:

```
'ajrpan'
```

Exercise 10: Arrange string characters such that lowercase letters should come first

Given string contains a combination of the lower and upper case letters. Write a program to arrange the characters of a string so that all lowercase letters should come first.

given-->PyNaTive

In [223]:

```
string="PyNaTive"

cap=list(string.upper())
sma=list(string.lower())

new=[]

l1=list(string)

for i in l1:
    if i in sma:
        new.append(i)

for i in l1:
    if i in cap:
        new.append(i)

result="".join(new)

print(result)
```

```
yaivePNT
```

In [21]:

```
# anther way of doing the same thing
given="PyNaTive"

x=given.upper()

l1=[]

for i in given:
    if i not in x:
        l1.append(i)
for i in given:
    if i in x:
        l1.append(i)

result="".join(l1)

print(result)
```

```
yaivePNT
```

Exercise 11: Count all letters, digits, and special symbols from a given string

given-->str1 = "P@#yn26at^&i5ve"

expected-->Total counts of chars, digits, and symbols

Chars = 8 Digits = 3 Symbol = 4

In [231]:

```
str1 = "P@#yn26at^&i5ve"

l1=list(str1)

char=0
digit=0
special_char=0

for i in l1:
    if i.isalpha()==True:
        char+=1
    elif i.isdigit()==True:
        digit+=1
    else:
        special_char+=1

print("char",char)
print("digit",digit)
print("special_char",special_char)
```

```
char 8
digit 3
special_char 4
```

Exercise 12: Create a mixed String using the following rules

Given two strings, s1 and s2. Write a program to create a new string s3 made of the first char of s1, then the last char of s2, Next, the second char of s1 and second last char of s2, and so on. Any leftover chars go at the end of the result.

In [235]:

```
s1="ABC"
s2="xyz"

a=s1[0]
b=s1[len(s1)//2]
c=s1[-1]

x=s2[0]
y=s2[len(s2)//2]
z=s2[-1]

result="".join([a,z,b,y,c,x])
print(result)
```

```
AzByCx
```

Exercise 13: String characters balance Test

Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character's position doesn't matter.

In [239]:

```
s1 = "Yn"
s2 = "PYnative"

count=0

for i in s1:
    if i in s2:
        count+=1

if len(s1)==count:
    print("s1 and s2 are balanced")
else:
    print("s1 and s2 are not balanced")
```

```
s1 and s2 are balanced
```

Exercise 14: Find all occurrences of a substring in a given string by ignoring the case

Write a program to find all occurrences of “USA” in a given string ignoring the case.

str1 = "Welcome to USA. usa awesome, isn't it?"

expected ans --> USA:-->2

In [242]:

```
str1 = "Welcome to USA. usa awesome, isn't it?"

str2=str1.upper()
print(str2)

str2.count("USA")
```

```
WELCOME TO USA. USA AWESOME, ISN'T IT?
```

Out[242]:

```
2
```

Excercise 15 Write a python program to convert a string to title case without using the title()

In [87]:

```
a=input("enter the title: ")

b=a.split()

r=''

#print(b)

for i in b:
    r=r+i.capitalize()+" "

print(r)
```

```
enter the title: milind mali
Milind Mali
```

Exercise 16: Calculate the sum and average of the digits present in a string

Given a string s1, write a program to return the sum and average of the digits that appear in the string, ignoring all other characters.

In [254]:

```
str1 = "PYnative29@#8496"

str2=list(str1)

total=0
counter=0

for i in str2:
    if i.isdigit()==True:
        total=total+int(i)
        counter+=1

print("sum of digits in the given string is ",total)
print("avg of digits in the given string is ",round(total/counter,2))
```

```
sum of digits in the given string is  38
avg of digits in the given string is  6.33
```

Exercise 17: Reverse a given string

str1 = "PYnative"

expected-->evitanYP

In [255]:

```
str1 = "PYnative"

str1[::-1]
```

Out[255]:

```
'evitanYP'
```

Exercise 18: Find the last position of a given substring

Write a program to find the last position of a substring “Emma” in a given string.

str1 = "Milind is a data scientist who knows Python. Milind works at google."

expected-->Last occurrence of Emma starts at index 43

In [256]:

```
str1 = "Milind is a data scientist who knows Python. Milind works at google."

str1.rfind("Milind",2)
```

Out[256]:

```
45
```

Exercise 19: Split a string on hyphens

Write a program to split a given string on hyphens and display each substring.

str1 = Emma-is-a-data-scientist

Expected-->Displaying each substring

Emma

is

a

data

scientist

In [258]:

```
str1 = "Emma-is-a-data-scientist"

str2=str1.split("-")

for i in str2:
    print(i)
```

```
Emma
is
a
data
scientist
```

Exercise 20: Remove empty strings from a list of strings

```
Original list of sting
['Emma', 'Jon', '', 'Kelly', None, 'Eric', '']

After removing empty strings
['Emma', 'Jon', 'Kelly', 'Eric']
```

In [260]:

```
str_list = ["Emma", "Jon", "", "Kelly", None, "Eric", ""]

for i in str_list:
    if i == "" or i==None:
        str_list.remove(i)

print(str_list)
```

```
['Emma', 'Jon', 'Kelly', 'Eric']
```

Exercise 21: Remove special symbols / punctuation from a string

expected-->"Jon is developer musician"

In [269]:

```
sentence = "/*Jon is @developer & musician"

import re

clean_sentence=re.sub('[^A-Za-z0-9\s]+',"",sentence)

print(clean_sentence)
```

```
Jon is developer  musician
```

Exercise 22: Removal all characters from a string except integers

str1 = 'I am 25 years and 10 months old'

expected-->2510

In [284]:

```
str1 = 'I am 26 years and 10 months old'

str2=str1.split()

new=[]

for i in str2:
    if i.isdigit()==True:
        new.append(i)

print("".join(new))
```

```
2610
```

Exercise 23: Find words with both alphabets and numbers

Write a program to find words with both alphabets and numbers from an input string.

In [1]:

```
#isalnum()

str1 = "Emma253 is Data scientist50000 and AI Expert"

str2=str1.split()

new=[]

for i in str2:
    for j in i:
        if j.isdigit()==True:
            if i not in new:
                new.append(i)

print(new[:])
```

```
['Emma253', 'scientist50000']
```

Exercise 24: Replace each special symbol with # in the following string

str1 = '/*Jon is @developer & musician!!'

Expected-->##Jon is #developer # musician##

In [324]:

```
str1 = '/*Jon is @developer & musician!!'

import string

for i in string.punctuation:
    if i in str1:
        str1=str1.replace(i,"#")

print(str1)
```

```
##Jon is #developer # musician##
```

Exercise:25 extract the email service provider name

In [1]:

```
emaillist=["KSR@datavizion.com","mymail@yahoo.com","milindmali@google.com","snehal@health
```

In [2]:

```
for i in emaillist:
    i=i.replace("@",".").split(".")
    print(i[1])
```

```
datavizion
yahoo
google
healthcare
```

Exercies 26: extract all the emailid for the given string

In [3]:

```
string="Hi my name is Govind Das and my mail id is milindmali108@gmail.com and my org mai
```

In [4]:

```
new=list(string.split())

for i in new:
    if ".com" in i:
        print(i)
```

```
milindmali108@gmail.com
milind@google.com
```

Excercise 27: write progrme to count the number of vowels in the string

In [3]:

```
string="Milind Dattatray Mali"

vowel="AEIOUaeiou"

count=0

for i in string:
    if i in vowel:
        count+=1

print(count)
```

```
7
```

7. Python Data Structure Exercise for Beginners

Exercise 1: Create a list by picking an odd-index items from the first list and even index items from the second

Given two lists, l1 and l2, write a program to create a third list l3 by picking an odd-index element from the list l1 and even index elements from the list l2.

In [335]:

```
l1 = [3, 6, 9, 12, 15, 18, 21]
l2 = [4, 8, 12, 16, 20, 24, 28]

l3=[]

for i in l1[1::2]:
    l3.append(i)
for i in l2[0::2]:
    l3.append(i)

print(l3)
```

```
[6, 12, 18, 4, 12, 20, 28]
```

Exercise 2: Remove and add item in a list

Write a program to remove the item present at index 4 and add it to the 2nd position and at the end of the list.

sample_list = [34, 54, 67, 89, 11, 43, 94]

List After removing element at index 4 [34, 54, 67, 89, 43, 94]

List after Adding element at index 2 [34, 54, 11, 67, 89, 43, 94]

List after Adding element at last [34, 54, 11, 67, 89, 43, 94, 11]

In [351]:

```
list1 = [34, 54, 67, 89, 11, 43, 94]

list1.pop(4)
list1.insert(2,11)
list1.append(11)

print(list1)
```

```
[34, 54, 11, 67, 89, 43, 94, 11]
```

Exercise 3: Slice list into 3 equal chunks and reverse each chunk

In [6]:

```
l1 = [11,49,8,23,14,12,78,45,89]

n=3

output=[l1[i:i+n][::-1] for i in range(0,len(l1),n)]

print(output)
```

```
[[8, 49, 11], [12, 14, 23], [89, 45, 78]]
```

Exercise 4: Count the occurrence of each element from a list

Write a program to iterate a given list and count the occurrence of each element and create a dictionary to show the count of each element.

sample_list = [11, 45, 8, 11, 23, 45, 23, 45, 89]

Expected Output:--> Printing count of each item {11: 2, 45: 3, 8: 1, 23: 2, 89: 1}

In [7]:

```
list1 = [11, 45, 8, 11, 23, 45, 23, 45, 89]

count=dict()

for i in list1:
    if i in count:
        count[i]+=1
    else:
        count[i]=1

print(count)
```

```
{11: 2, 45: 3, 8: 1, 23: 2, 89: 1}
```

Exercise 5: Create a Python set such that it shows the element from both lists in a pair

first_list = [2, 3, 4, 5, 6, 7, 8]

second_list = [4, 9, 16, 25, 36, 49, 64]

Result is {(6, 36), (8, 64), (4, 16), (5, 25), (3, 9), (7, 49), (2, 4)}

In [10]:

```
first_list = [2, 3, 4, 5, 6, 7, 8]

second_list = [4, 9, 16, 25, 36, 49, 64]

result=zip(first_list,second_list)

result_set=set(result)

print(result_set)
```

```
{(7, 49), (2, 4), (4, 16), (8, 64), (6, 36), (3, 9), (5, 25)}
```

Exercise 6: Find the intersection (common) of two sets and remove those elements from the first set

first_set = {23, 42, 65, 57, 78, 83, 29}

second_set = {57, 83, 29, 67, 73, 43, 48}

Intersection is {57, 83, 29}

First Set after removing common element {65, 42, 78, 23}

In [20]:

```
first_set = {23, 42, 65, 57, 78, 83, 29}

second_set = {57, 83, 29, 67, 73, 43, 48}

intersection=first_set.intersection(second_set)

print(intersection)

for i in intersection:
    first_set.remove(i)

print(first_set)
```

```
{57, 83, 29}
{65, 23, 42, 78}
```

Exercise 6: Checks if one set is a subset or superset of another set. If found, delete all elements from that set

first_set = {27, 43, 34}

second_set = {34, 93, 22, 27, 43, 53, 48}

expected output:

First set is subset of second set - True

Second set is subset of First set - False

First set is Super set of second set - False

Second set is Super set of First set - True

First Set set{}

Second Set {67, 73, 43, 48, 83, 57, 29}

In [31]:

```
first_set = {27, 43, 34}

second_set = {34, 93, 22, 27, 43, 53, 48}

print("first set is subset of second set:\n",first_set.issubset(second_set))

print("second set is subset of first set:\n",second_set.issubset(first_set))

print("first set is superset of second set:\n",first_set.issuperset(second_set))

print("second set is superset of first set:\n",second_set.issuperset(first_set))

if first_set.issubset(second_set):
    first_set.clear()

if second_set.issubset(first_set):
    second_set.clear()

print("first set:\n",first_set)
print("second set:\n",second_set)
```

```
first set is subset of second set:
 True
second set is subset of first set:
 False
first set is superset of second set:
 False
second set is superset of first set:
 True
first set:
 set()
second set:
 {48, 34, 53, 22, 27, 43, 93}
```

Exercise 7: Iterate a given list and check if a given element exists as a key's value in a dictionary. If not, delete it from the list

roll_number = [47, 64, 69, 37, 76, 83, 95, 97]

sample_dict = {'Jhon':47, 'Emma':69, 'Kelly':76, 'Jason':97}

Expected result:--> After removing unwanted elements from list [47, 69, 76, 97]

In [36]:

```
roll_number = [47, 64, 69, 37, 76, 83, 95, 97]

sample_dict = {'Jhon':47, 'Emma':69, 'Kelly':76, 'Jason':97}

#will modify the original list

roll_number[:]=[item for item in roll_number if item in sample_dict.values()]

print(roll_number)
```

```
[47, 69, 76, 97]
```

Exercise 8: Get all values from the dictionary and add them to a list but don't add duplicates

speed = {'jan': 47, 'feb': 52, 'march': 47, 'April': 44, 'May': 52, 'June': 53, 'july': 54, 'Aug': 44, 'Sept': 54}

expected_output: [47, 52, 44, 53, 54]

In [37]:

```
speed = {'jan': 47, 'feb': 52, 'march': 47, 'April': 44, 'May': 52, 'June': 53, 'july': 5

speed_list=[]

for value in speed.values():
    if value not in speed_list:
        speed_list.append(value)

print(speed_list)
```

```
[47, 52, 44, 53, 54]
```

Exercise 9: Remove duplicates from a list and create a tuple and find the minimum and maximum number

sample_list = [87, 45, 41, 65, 94, 41, 99, 94]

Expected-->

unique items [87, 45, 41, 65, 99]

tuple (87, 45, 41, 65, 99)

min: 41

max: 99

In [43]:

```
sample_list = [87, 45, 41, 65, 94, 41, 99, 94]

list1=set(sample_list)  # removed all the duplicates

tuple1=tuple(list1)

print("Unique element in the sample list:\n",tuple1)

print("Maximum number in the sample list:\n",max(tuple1))
print("Minimun number in the sample list:\n",min(tuple1))
```

```
Unique element in the sample list:
 (65, 99, 41, 45, 87, 94)
Maximum number in the sample list:
 99
Minimun number in the sample list:
 41
```

8. Python List exercise with solutions

Excercise: 1 Write a python program to find the max item from a list without using the max function

In [88]:

```
l1=[4,6,2,8,1]

l1.sort(reverse=True)

print(l1[0])
```

```
8
```

Excercise 2 Find the reverse of a number provided by the user(any number of digit)

In [82]:

```
n=int(input("enter the number: "))

digit_list=list(map(int,str(n)))

digit_list=digit_list[::-1]

num=int("".join(map(str,digit_list)))

print(num)
```

```
enter the number: 1234
4321
```

Excercise 3 Take a number from the user and find the number of digits in it.

In [80]:

```
n=int(input("enter the number: "))

digit_list=list(map(int,str(n)))

print(digit_list)

print(len(digit_list))
```

```
enter the number: 34457
[3, 4, 4, 5, 7]
5
```

Exercise 4: Write a python program to remove all the duplicates from a list

In [31]:

```
l1=[1,2,3,3,3,4,4,5,6,7,8,9,9]

l2=[]

for i in l1:
    if i not in l2:
        l2.append(i)

print(l2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Exercise 5: Reverse a list in Python

In [49]:

```
list1 = [100, 200, 300, 400, 500]

print(list1[::-1])

#or you can also use "reverse()" function
```

```
[500, 400, 300, 200, 100]
```

Exercise 6: Concatenate two lists index-wise

Write a program to add two lists index-wise. Create a new list that contains the 0th index item from both the list, then the 1st index item, and so on till the last element. any leftover items will get added at the end of the new list.

list1 = ["M", "na", "i", "Ke"]

list2 = ["y", "me", "s", "lly"]

expected result:

['My', 'name', 'is', 'Kelly']

In [55]:

```
list1 = ["M", "na", "i", "Ke"]

list2 = ["y", "me", "s", "lly"]

list3=[i+j for i,j in zip(list1,list2)]

print(list3)
```

```
['My', 'name', 'is', 'Kelly']
```

Exercise 7: Turn every item of a list into its square

Given a list of numbers. write a program to turn every item of a list into its square.

In [58]:

```
numbers = [1, 2, 3, 4, 5, 6, 7]

new=[]

for i in numbers:
    x=pow(i,2)
    new.append(x)

print(new)
```

```
[1, 4, 9, 16, 25, 36, 49]
```

In [59]:

```
#same using list comprehenssion

result=[pow(x,2) for x in numbers]
print(result)
```

```
[1, 4, 9, 16, 25, 36, 49]
```

Exercise 8: Concatenate two lists in the following order

list1 = ["Hello ", "take "]

list2 = ["Dear", "Sir"]

Expected result:

['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']

In [60]:

```
list1 = ["Hello ", "take "]

list2 = ["Dear", "Sir"]

for i in list1:
    for j in list2:
        print(i+j)
```

```
Hello Dear
Hello Sir
take Dear
take Sir
```

In [63]:

```
#using list comprehension
result=[i+j for i in list1 for j in list2]
print(result)
```

```
['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']
```

Exercise 9: Iterate both lists simultaneously

Given a two Python list. Write a program to iterate both lists simultaneously and display items from list1 in original order and items from list2 in reverse order.

list1 = [10, 20, 30, 40]

list2 = [100, 200, 300, 400]

Expected-->

10 400

20 300

30 200

40 100

In [64]:

```
list1 = [10, 20, 30, 40]

list2 = [100, 200, 300, 400]

for x,y in zip(list1,list2[::-1]):
    print(x,y)
```

```
10 400
20 300
30 200
40 100
```

Exercise 10: Remove empty strings from the list of strings

list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]

expected:

["Mike", "Emma", "Kelly", "Brad"]

In [76]:

```
list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]

result=list(filter(lambda x: x!="",list1))

print(result)
```

```
['Mike', 'Emma', 'Kelly', 'Brad']
```

Exercise 11: Add new item to list after a specified item

Write a program to add item 7000 after 6000 in the following Python List

list1 = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]

Expected output:

[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]

In [78]:

```
list1 = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]

list1[2][2].append(7000)

print(list1)
```

```
[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]
```

Exercise 12: Extend nested list by adding the sublist

You have given a nested list. Write a program to extend it by adding the sublist ["h", "i", "j"] in such a way that it will look like the following list.

list1 = ["a", "b", ["c", ["d", "e", ["f", "g"], "k"], "l"], "m", "n"]

*sub list to add sub_list = ["h", "i", "j"]

Expected result:

['a', 'b', ['c', ['d', 'e', ['f', 'g', 'h', 'i', 'j'], 'k'], 'l'], 'm', 'n']

In [79]:

```
list1 = ["a", "b", ["c", ["d", "e", ["f", "g"], "k"], "l"], "m", "n"]

sub_list = ["h", "i", "j"]

list1[2][1][2].extend(sub_list)

print(list1)
```

```
['a', 'b', ['c', ['d', 'e', ['f', 'g', 'h', 'i', 'j'], 'k'], 'l'], 'm',
'n']
```

Exercise 13: Replace list's item with new value if found

You have given a Python list. Write a program to find value 20 in the list, and if it is present, replace it with 200. Only update the first occurrence of an item.

list1 = [5, 10, 15, 20, 25, 50, 20]

Expected output: [5, 10, 15, 200, 25, 50, 20]

In [81]:

```
list1 = [5, 10, 15, 20, 25, 50, 20]

x=list1.index(20)

list1[x]=200

print(list1)
```

```
[5, 10, 15, 200, 25, 50, 20]
```

Exercise 14: Remove all occurrences of a specific item from a list.

Given a Python list, write a program to remove all occurrences of item 20.

list1 = [5, 20, 15, 20, 25, 50, 20]

Expected output:

[5, 15, 25, 50]

In [83]:

```
list1 = [5, 20, 15, 20, 25, 50, 20]


while 20 in list1:
    list1.remove(20)

print(list1)
```

```
[5, 15, 25, 50]
```

Excercise 15 :Write a program that can perform union and intersection on 2 given list.

In [90]:

```
l1=[1,2,3,4,5]
l2=[4,5,6,7,8]

uni=[]
inter=[]

for i in l1:
    if i in l2:
        inter.append(i)

print(inter)

for i in l1:
    if i not in uni:
        uni.append(i)

for i in l2:
    if  i not in uni:
        uni.append(i)

print(uni)
```

```
[4, 5]
[1, 2, 3, 4, 5, 6, 7, 8]
```

Excercise 16: break down the list into equal number of chucks

In [2]:

```
l1=[1,2,3,4,5,6,7,8,9,10,11,12]

chunk=3

output=[l1[i:i+chunk] for i in range(0,len(l1),chunk)]

print(output)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```

Excercise 17: logic for quickly swap the first and last number of the list

In [4]:

```
l1=[1,2,3,4,5]

temp=l1[-1]

l1[-1]=l1[0]

l1[0]=temp

print(l1)
```

```
[5, 2, 3, 4, 1]
```

Excercise 18: what is easiest way to shuffle the list

In [8]:

```
from random import shuffle

list1=["Milind","Kanchan","Rohit","Shashi"]

shuffle(list1)

print(list1)
```

```
['Rohit', 'Milind', 'Shashi', 'Kanchan']
```

Excercise 19 create lambda function for the sum of all the elements in the list

In [16]:

```

l1=[5,8,10,20,50,100]

from functools import reduce

result=reduce(lambda x,y:x+y, l1)

print(result)
```

```
193
```

9. Python Dictionary Exercise with Solutions

Excercise 1:Get the only keys of dictionary

In [17]:

```
dicti={"name":"Milind","Age":34}
```

In [18]:

```
# to get list of all the keys
dicti.keys()
```

Out[18]:

```
dict_keys(['name', 'Age'])
```

Exercise 2: Convert two lists into a dictionary

Below are the two lists. Write a Python program to convert them into a dictionary in a way that item from list1 is the key and item from list2 is the value

keys = ['Ten', 'Twenty', 'Thirty']

values = [10, 20, 30]

Expected output:

{'Ten': 10, 'Twenty': 20, 'Thirty': 30}

In [85]:

```
keys = ['Ten', 'Twenty', 'Thirty']

values = [10, 20, 30]

my_dict=zip(keys,values)

result=dict(my_dict)

print(result)
```

```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30}
```

Exercise 3: Merge two Python dictionaries into one

dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}

dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}

Expected output

{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Fourty': 40, 'Fifty': 50}

In [86]:

```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}

dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}

dict3={**dict1,**dict2}

print(dict3)
```

```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
```

In [92]:

```
# another way of achieving same is:
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}

dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}

dict3=dict1.copy()

dict3.update(dict2)

print(dict3)
```

```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
```

Exercise 4: Print the value of key 'history' from the below dict

expected output: 80

sampleDict = { "class": { "student": { "name": "Mike", "marks": { "physics": 70, "history": 80 } } } }

In [107]:

```
sampleDict = {"class": {"student": {"name": "Mike","marks": {"physics": 70,"history": 80}

sampleDict["class"]["student"]["marks"]["history"]
```

Out[107]:

```
80
```

Exercise 5: Initialize dictionary with default values

In Python, we can initialize the keys with the same values.

employees = ['Kelly', 'Emma']

defaults = {"designation": 'Developer', "salary": 8000}

expected output: {'Kelly': {'designation': 'Developer', 'salary': 8000}, 'Emma': {'designation': 'Developer', 'salary': 8000}}

In [116]:

```
employees = ['Kelly', 'Emma']

defaults = {"designation": 'Developer', "salary": 8000}

result=dict.fromkeys(employees,defaults)

print(result)
print("="*100)
print("Details of kelly: ",result["Kelly"])
```

```
{'Kelly': {'designation': 'Developer', 'salary': 8000}, 'Emma': {'designat
ion': 'Developer', 'salary': 8000}}
====================================================================================
==========================
Details of kelly:  {'designation': 'Developer', 'salary': 8000}
```

Exercise 6: Create a dictionary by extracting the keys from a given dictionary

Write a Python program to create a new dictionary by extracting the mentioned keys from the below dictionary.

sample_dict = { "name": "Kelly", "age": 25, "salary": 8000, "city": "New york"}

*key to extract keys = ["name", "salary"]

expected output:

{'name': 'Kelly', 'salary': 8000}

In [120]:

```
sample_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}

keys=["name","salary"]

result=dict()

for k in keys:
    result.update({k:sample_dict[k]})

print(result)
```

```
{'name': 'Kelly', 'salary': 8000}
```

In [124]:

```
#using dictionary comprehension
sample_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}

keys=["name","salary"]

new_dict={k:sample_dict[k] for k in keys}

print(new_dict)
```

```
{'name': 'Kelly', 'salary': 8000}
```

Exercise 7: Check if a value exists in a dictionary

We know how to check if the key exists in a dictionary. Sometimes it is required to check if the given value is present.

Write a Python program to check if value 200 exists in the following dictionary.

given:

sample_dict = {'a': 100, 'b': 200, 'c': 300}

expected:

200 present in a dict

In [125]:

```
sample_dict = {'a': 100, 'b': 200, 'c': 300}

if 200 in sample_dict.values():
    print("200 is present in given dict")
```

```
200 is present in given dict
```

Exercise 8: Rename key of a dictionary

Write a program to rename a key city to a location in the following dictionary.

sample_dict = { "name": "Kelly", "age":25, "salary": 8000, "city": "New york" }

expected:

```
{'name': 'Kelly', 'age': 25, 'salary': 8000, 'location': 'New york'}
```

In [126]:

```
sample_dict = {
  "name": "Kelly",
  "age":25,
  "salary": 8000,
  "city": "New york"
}

sample_dict["location"]=sample_dict.pop("city")

print(sample_dict)
```

```
{'name': 'Kelly', 'age': 25, 'salary': 8000, 'location': 'New york'}
```

Exercise 9: Get the key of a minimum value from the following dictionary

sample_dict = { 'Physics': 82, 'Math': 65, 'history': 75 }

Expected:

math

In [2]:

```
sample_dict = {
  'Physics': 82,
  'Math': 65,
  'history': 75
}

#list1=[]

min(sample_dict,key=sample_dict.get)
```

Out[2]:

```
'Math'
```

Exercise 10: Change value of a key in a nested dictionary

Write a Python program to change Brad's salary to 8500 in the following dictionary.

sample_dict = { 'emp1': {'name': 'Jhon', 'salary': 7500}, 'emp2': {'name': 'Emma', 'salary': 8000}, 'emp3': {'name': 'Brad', 'salary': 500} }

expected output:

{ 'emp1': {'name': 'Jhon', 'salary': 7500}, 'emp2': {'name': 'Emma', 'salary': 8000}, 'emp3': {'name': 'Brad', 'salary': 8500} }

In [130]:

```
sample_dict = {
    'emp1': {'name': 'Jhon', 'salary': 7500},
    'emp2': {'name': 'Emma', 'salary': 8000},
    'emp3': {'name': 'Brad', 'salary': 500}
}

sample_dict["emp3"]["salary"]=8500

print(sample_dict)
```

```
{'emp1': {'name': 'Jhon', 'salary': 7500}, 'emp2': {'name': 'Emma', 'salar
y': 8000}, 'emp3': {'name': 'Brad', 'salary': 8500}}
```

10. Python Set Exercise with Solutions

Exercise 1: Add a list of elements to a set

Given a Python list, Write a program to add all its elements into a given set.

sample_set = {"Yellow", "Orange", "Black"}

sample_list = ["Blue", "Green", "Red"]

In [135]:

```
sample_set = {"Yellow", "Orange", "Black"}

sample_list = ["Blue", "Green", "Red"]

sample_set.update(sample_list)

print(sample_set)
```

```
{'Blue', 'Green', 'Black', 'Red', 'Orange', 'Yellow'}
```

Exercise 2: Return a new set of identical items from two sets

set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

Expected output: {40, 50, 30}

In [136]:

```
set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

set3=set1.intersection(set2)

print(set3)
```

```
{40, 50, 30}
```

Exercise 3: Get Only unique items from two sets

Write a Python program to return a new set with unique items from both sets by removing duplicates.

set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

Expected: {70, 40, 10, 50, 20, 60, 30}

In [137]:

```
set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

set3=set1.union(set2)

print(set3)
```

```
{70, 40, 10, 50, 20, 60, 30}
```

Exercise 4: Update the first set with items that don’t exist in the second set

Given two Python sets, write a Python program to update the first set with items that exist only in the first set and not in the second set.

set1 = {10, 20, 30}

set2 = {20, 40, 50}

expected result:

set1 {10, 30}

In [141]:

```
set1 = {10, 20, 30}

set2 = {20, 40, 50}

set1.difference_update(set2)

print(set1)
```

```
{10, 30}
```

Exercise 5: Remove items from the set at once

Write a Python program to remove items 10, 20, 30 from the following set at once.

set1 = {10, 20, 30, 40, 50}

expected:

```
    {40, 50}
```

In [143]:

```
set1 = {10, 20, 30, 40, 50}

set2={10,20,30}

set1.difference_update(set2)

print(set1)
```

```
{50, 40}
```

Exercise 6: Return a set of elements present in Set A or B, but not both

set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

Expected Output:

{20, 70, 10, 60}

In [145]:

```
set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

set1.symmetric_difference(set2)
```

Out[145]:

```
{10, 20, 60, 70}
```

Exercise 6: Check if two sets have any elements in common. If yes, display the common elements

set1 = {10, 20, 30, 40, 50}

set2 = {60, 70, 80, 90, 10}

expected:

Two sets have items in common {10}

In [29]:

```
set1 = {10 ,20, 30, 40, 50}

set2 = {60, 70, 80, 90, 10}

if set1.isdisjoint(set2):
    print("above two set dont have any common element")
else:
    print(set1.intersection(set2))
```

```
{10}
```

Exercise 7: Update set1 by adding items from set2, except common items

set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

Expected;

{70, 10, 20, 60}

In [152]:

```
set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

set1.symmetric_difference_update(set2)

print(set1)
```

```
{20, 70, 10, 60}
```

Exercise 8: Remove items from set1 that are common to both set1 and set2

set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

Expected output:

{40, 50, 30}

In [156]:

```
set1 = {10, 20, 30, 40, 50}

set2 = {30, 40, 50, 60, 70}

set1.intersection_update(set2)

set1
```

Out[156]:

```
{30, 40, 50}
```

11. Python Tuple Exercise with Solutions

Exercise 1: Reverse the tuple

```
tuple1 = (10, 20, 30, 40, 50)
```

In [157]:

```
tuple1 = (10, 20, 30, 40, 50)

tuple1[::-1]
```

Out[157]:

```
(50, 40, 30, 20, 10)
```

Exercise 2: Access value 20 from the tuple

The given tuple is a nested tuple. write a Python program to print the value 20.

tuple1 = ("Orange", [10, 20, 30], (5, 15, 25))

In [158]:

```
tuple1 = ("Orange", [10, 20, 30], (5, 15, 25))

tuple1[1][1]
```

Out[158]:

```
20
```

Exercise 3: Create a tuple with single item 50

In [159]:

```
tuple1=(50,)

print(tuple1)
```

```
(50,)
```

Exercise 4: Unpack the tuple into 4 variables

Write a program to unpack the following tuple into four variables and display each variable.

tuple1 = (10, 20, 30, 40)

- your code

print(a) # should print 10

print(b) # should print 20

print(c) # should print 30

print(d) # should print 40

In [160]:

```
tuple1 = (10, 20, 30, 40)

a,b,c,d=tuple1

print(a)
print(b)
print(c)
print(d)
```

```
10
20
30
40
```

Exercise 5: Swap two tuples in Python

tuple1 = (11, 22)

tuple2 = (99, 88)

In [163]:

```
tuple1 = (11, 22)

tuple2 = (99, 88)

tuple1,tuple2=tuple2,tuple1

print("tuple1",tuple1)
print("tuple2",tuple2)
```

```
tuple1 (99, 88)
tuple2 (11, 22)
```

Exercise 6: Copy specific elements from one tuple to a new tuple

Write a program to copy elements 44 and 55 from the following tuple into a new tuple.

tuple1 = (11, 22, 33, 44, 55, 66)

Expected

tuple2: (44, 55)

In [164]:

```
tuple1 = (11, 22, 33, 44, 55, 66)

tuple2=tuple1[3:5]

print(tuple2)
```

```
(44, 55)
```

Exercise 7: Modify the tuple

Given is a nested tuple. Write a program to modify the first item (22) of a list inside a following tuple to 222

tuple1 = (11, [22, 33], 44, 55)

Expected

tuple1: (11, [222, 33], 44, 55)

In [167]:

```
tuple1 = (11, [22, 33], 44, 55)

tuple1[1][0]=222

tuple1
```

Out[167]:

```
(11, [222, 33], 44, 55)
```

Exercise 8: Sort a tuple of tuples by 2nd item

```
tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))
```

Expected:

```
(('c', 11), ('a', 23), ('d', 29), ('b', 37))
```

In [175]:

```
tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))

tuple1=sorted(list(tuple1),key=lambda x:x[1])

tuple(tuple1)
```

Out[175]:

```
(('c', 11), ('a', 23), ('d', 29), ('b', 37))
```

Exercise 9: Counts the number of occurrences of item 50 from a tuple

```
tuple1 = (50, 10, 60, 70, 50)
```

In [176]:

```
tuple1 = (50, 10, 60, 70, 50)

tuple1.count(50)
```

Out[176]:

```
2
```

Exercise 10: Check if all items in the tuple are the same

tuple1 = (45, 45, 45, 45)

In [199]:

```
t = (45, 45, 45, 45)

#to check whether all the item are same

all_same=all(i==t[0] for i in t)

if all_same:
    print("yes all the item are same in the given tuple")
else:
    print("No all the item are not same in the given table")
```

```
yes all the item are same in the given tuple
```

12. Python Date and Time Exercise with Solutions

Exercise 1: Print current date and time in Python

In [201]:

```
import datetime

# for date and time
print(datetime.datetime.now())

# for time only
print(datetime.datetime.now().time())
```

```
2023-05-24 15:12:07.167241
15:12:07.167241
```

Exercise 2: Convert string into a datetime object

For example, You received the following date in string format. Please convert it into Python's DateTime object

date_string = "Feb 25 2020 4:20PM"

In [6]:

```
from datetime import datetime

date_string="Feb 25 2020 4:20PM"

date_time_obj=datetime.strptime(date_string,"%b %d %Y %H:%M%p")

print(date_time_obj)
```

```
2020-02-25 04:20:00
```

Exercise 3: Subtract a week (7 days) from a given date in Python

```
given_date = datetime(2020, 2, 25)
```

expected date:

```
2020-02-18
```

In [212]:

```
given_date = datetime(2020, 2, 25)

from datetime import datetime ,timedelta

print(given_date)

days_to_substract=7

res_date=given_date-timedelta(days_to_substract)

print(res_date)
```

```
2020-02-25 00:00:00
2020-02-18 00:00:00
```

Exercise 4: Print a date in a the following format

given_date = datetime(2020, 2, 25)

expected:

Tuesday 25 February 2020

In [217]:

```
from datetime import datetime

given_date = datetime(2020, 2, 25)

given_date.strftime("%A %d %b %Y")
```

Out[217]:

```
'Tuesday 25 Feb 2020'
```

Exercise 5: Find the day of the week of a given date

Given:

given_date = datetime(2020, 7, 26)

Expected output:

Sunday

In [218]:

```
given_date = datetime(2020, 7, 26)

given_date.strftime("%A")
```

Out[218]:

```
'Sunday'
```

Exercise 6: Add a week (7 days) and 12 hours to a given date

Given:

#2020-03-22 10:00:00

given_date = datetime(2020, 3, 22, 10, 0, 0)

Expected output:

2020-03-29 22:00:00

In [221]:

```
given_date = datetime(2020, 3, 22, 10, 0, 0)

days_to_add=7

res_date=given_date+timedelta(days_to_add,12)

print("new date will be: \n",res_date)
```

```
new date will be:
 2020-03-29 10:00:12
```

Exercise 7: Print current time in milliseconds

In [222]:

```
import time

milliseconds=int(round(time.time()*1000))

print(milliseconds)
```

```
1684923177395
```

Exercise 8: Convert the following datetime into a string

Given:

given_date = datetime(2020, 2, 25)

Expected output:

"2020-02-25 00:00:00"

In [227]:

```
given_date = datetime(2020, 2, 25)

string_date=given_date.strftime("%Y-%m-%d %H:%M:%S")

string_date
```

Out[227]:

```
'2020-02-25 00:00:00'
```

Exercise 9: Calculate the date 4 months from the current date

Given:

#2020-02-25

given_date = datetime(2020, 2, 25).date()

Expected output:

2020-06-25

In [233]:

```
# 2020-02-25

from dateutil.relativedelta import relativedelta

given_date = datetime(2020, 2, 25).date()

months_to_add=4

result_date=given_date+relativedelta(months= + months_to_add)

print(result_date)
```

```
2020-06-25
```

Exercise 10: Calculate number of days between two given dates

Given:

#2020-02-25

date_1 = datetime(2020, 2, 25)

#2020-09-17

date_2 = datetime(2020, 9, 17)

Expected output:

205 days

In [237]:

```
# 2020-02-25
date_1 = datetime(2020, 2, 25).date()

# 2020-09-17
date_2 = datetime(2020, 9, 17).date()

delta=None

if date_1>date_2:
    delta=date_1-date_2

else:
    delta=date_2-date_1

print("Difference of the days will be: \n",delta.days)
```

```
Difference of the days will be:
 205
```

13.Python Object-Oriented Programming (OOP) Exercise: Classes and Objects Exercises

OOP Exercise 1: Create a Class with instance attributes

Write a Python program to create a Vehicle class with max_speed and mileage instance attributes.

In [249]:

```
class Vehicle:
    def __init__ (self,max_speed,mileage):
        self.max_speed=max_speed
        self.mileage=mileage

maruti=Vehicle(120,15)

print(maruti.max_speed,maruti.mileage)
```

```
120 15
```

OOP Exercise 2: Create a Vehicle class without any variables and methods

In [250]:

```
class Vehicle:
    pass
```

OOP Exercise 3: How to Modify attributes in class

In [21]:

```
class Car():
    """A simple attempt to create a car"""
    def __init__(self,make,model,year):  #__init__: means this will run automatically
        """initialize car attribute"""
        self.make=make
        self.model=model
        self.year=year

        #fuel capacity and level in gallons
        self.fuel_capacity=15
        self.fuel_level=0

    def fill_tank(self):
        """fill gas tank to capacity"""
        self.fuel_level=self.fuel_capacity
        print("Fuel tank is full")

    def drive(self):
        """stimulate driving"""
        print("The car is moving")
```

In [22]:

```
# let create object of Car class

my_car=Car("Audi","G3",2022)
```

In [23]:

```
# accessing attribute values

print(my_car.make)
print(my_car.model)
print(my_car.year)
```

```
Audi
G3
2022
```

In [24]:

```
# calling methods
my_car.drive()
```

```
The car is moving
```

In [25]:

```
my_car.fill_tank()
```

```
Fuel tank is full
```

Modifying attributes

In [26]:

```
my_car.fuel_level=5
```

In [27]:

```
my_car.fuel_level
```

Out[27]:

```
5
```

In [28]:

```
#Writing a method to update an attributes values
```

In some programming languages, such as Python, it is possible to add methods to a class from outside the class definition. This is known as "monkey patching" or "dynamic class modification.

In [29]:

```
def update_fuel_level(self,new_level):
    """Update the fuel level"""
    if new_level<=self.fuel_capacity:
        self.fuel_level=new_level
    else:
        print("The tank can't hold that much")
```

In [30]:

```
# adding above function to original class as method from outside
Car.update_fuel_level=update_fuel_level
```

In [34]:

```
my_car.update_fuel_level(12)
```

In [35]:

```
my_car.fuel_level
```

Out[35]:

```
12
```

In [ ]:

```
# Writing a method to increament an attributes values
```

In [36]:

```
def add_fuel(self,amount):
    """Add fuel to tank"""
    if (self.fuel_level+amount <= self.fuel_capacity):
        self.fuel_level+=amount
        print("Added fuel")
    else:
        print("the tank cant hold that much")
```

In [37]:

```
Car.add_fuel=add_fuel
```

In [38]:

```
my_car.add_fuel(10)
```

```
the tank cant hold that much
```

In [39]:

```
my_car.add_fuel(2)
```

```
Added fuel
```

In [40]:

```
my_car.fuel_level
```

Out[40]:

```
14
```

OOP Exercise 4: Create a child class Bus that will inherit all of the variables and methods of the Vehicle class

In [252]:

```
class Vehicle:
    def __init__ (self,max_speed,mileage):
        self.max_speed=max_speed
        self.mileage=mileage

class Bus(Vehicle):
    pass

School_bus=Bus(70,10)
print(School_bus.max_speed,School_bus.mileage)
```

```
70 10
```

OOP Exercise 4: Class Inheritance

Given:

Create a Bus class that inherits from the Vehicle class. Give the capacity argument of Bus.seating_capacity() a default value of 50.

In [5]:

```
class Vehicle:

    def __init__(self, name, max_speed, mileage):
        self.name = name
        self.max_speed = max_speed
        self.mileage = mileage

    def sitting_capacity(self,capacity):
        return f"sitting capacity of bus {self.name} is {capacity} passanger"
```

In [6]:

```
class Bus(Vehicle):
    #assign defualt value to capacity 50
    def sitting_capacity(self,capacity=50):
        return super().sitting_capacity(capacity=50)
```

In [14]:

```
School_bus=Bus("Volvo",120,23)
```

In [12]:

```
School_bus.sitting_capacity()
```

Out[12]:

```
'sitting capacity of bus Volvo is 50 passanger'
```

OOP Exercise 5: Define a property that must have the same value for every class instance (object)

Define a class attribute"color" with a default value white. I.e., Every Vehicle should be white.

In [34]:

```
class Vehicle:

    Color="White"

    def __init__(self, name, max_speed, mileage):
        self.name = name
        self.max_speed = max_speed
        self.mileage = mileage

class Bus(Vehicle):
    pass

class Car(Vehicle):
    pass
```

In [37]:

```
School_bus=Bus("Volvo",120,23)
print(School_bus.name,School_bus.Color,School_bus.max_speed,School_bus.mileage)
```

```
Volvo White 120 23
```

In [39]:

```
Maruti=Car("Swift",90,41)
print(Maruti.name,Maruti.Color,Maruti.max_speed,Maruti.mileage)
```

```
Swift White 90 41
```

OOP Exercise 6: Class Inheritance

Given:

Create a Bus child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity * 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare.

Note: The bus seating capacity is 50. so the final fare amount should be 5500. You need to override the fare() method of a Vehicle class in Bus class.

In [46]:

```
class Vehicle:
    def __init__(self, name, mileage, capacity):
        self.name = name
        self.mileage = mileage
        self.capacity = capacity

    def fare(self):
        return self.capacity * 100

class Bus(Vehicle):
    def fare(self):
        amount=super().fare()
        amount+=amount*0.1
        return amount

School_bus = Bus("School Volvo", 12, 50)
print("Total Bus fare is:", School_bus.fare())
```

```
Total Bus fare is: 5500.0
```

OOP Exercise 7:Write a program to determine which class a given Bus object belongs to.

In [47]:

```
class Vehicle:
    def __init__(self, name, mileage, capacity):
        self.name = name
        self.mileage = mileage
        self.capacity = capacity

class Bus(Vehicle):
    pass

School_bus = Bus("School Volvo", 12, 50)
```

In [48]:

```
type(School_bus)
```

Out[48]:

```
__main__.Bus
```

In [49]:

```
# so given object is belong to class bus
```

OOP Exercise 8: Determine if School_bus is also an instance of the Vehicle class

In [50]:

```
class Vehicle:
    def __init__(self, name, mileage, capacity):
        self.name = name
        self.mileage = mileage
        self.capacity = capacity

class Bus(Vehicle):
    pass

School_bus = Bus("School Volvo", 12, 50)
```

In [52]:

```
isinstance(School_bus,Vehicle)
```

Out[52]:

```
True
```

14. Python JSON(Java Script Object Notation) Exercise

Exercise 1: Convert the following dictionary into JSON format

data = {"key1" : "value1", "key2" : "value2"}

Expected Output:

data = {"key1" : "value1", "key2" : "value2"}

In [59]:

```
import json

data = {"key1" : "value1", "key2" : "value2"}

jsondata=json.dumps(data)   #dict converted to string

print(jsondata)
```

```
{"key1": "value1", "key2": "value2"}
```

Exercise 2: Access the value of key2 from the following JSON

import json

sampleJson = """{"key1": "value1", "key2": "value2"}"""

#write code to print the value of key2

Expected Output:

value2

In [60]:

```
sampleJson ="""{"key1": "value1", "key2": "value2"}"""

data=json.loads(sampleJson)   # converted to dictionary

data["key2"]
```

Out[60]:

```
'value2'
```

Exercise 3: PrettyPrint following JSON data

PrettyPrint following JSON data with indent level 2 and key-value separators should be (",", " = ").

sampleJson = {"key1": "value1", "key2": "value2"}

Expected Output:

{

"key1" = "value1",

"key2" = "value2",

"key3" = "value3"

}

In [75]:

```
sampleJson = {"key1" : "value1", "key2" : "value2", "key3" : "value3"}

PrettyPrintedJson = json.dumps(sampleJson,indent=2,separators=(",","="))

print(PrettyPrintedJson)
```

```
{
  "key1"="value1",
  "key2"="value2",
  "key3"="value3"
}
```

Exercise 4: Sort JSON keys in and write them into a file

Sort following JSON data alphabetical order of keys

sampleJson = {"id" : 1, "name" : "value2", "age" : 29}

Expected Output:

```
{

    "age": 29,

    "id": 1,

    "name": "value2"

}
```

In [78]:

```
sampleJson = {"id" : 1, "name" : "value2", "age" : 29}

with open ("sampleJson.json","w") as write_file:
    json.dump(sampleJson,write_file,indent=4,sort_keys=True)

print("Done writing data into Json file")
```

```
Done writing data into Json file
```

Exercise 5: Access the nested key ‘salary’ from the following JSON

```
import json

sampleJson = """{
   "company":{
```

In [84]:

```
import json

sampleJson = """{
   "company":{
      "employee":{
         "name":"emma",
         "payble":{
            "salary":7000,
            "bonus":800
         }
      }
   }
}"""

data=json.loads(sampleJson)

print(data["company"]["employee"]["payble"]["salary"])
```

```
7000
```

Exercise 6: Convert the following Vehicle Object into JSON

```
import json

class Vehicle:
    def __init__(self, name, engine, price):
        self.name = name
        self.engine = engine
        self.price = price

vehicle = Vehicle("Toyota Rav4", "2.5L", 32000)

# Convert it into JSON format
```

In [85]:

```
import json

class Vehicle:
    def __init__(self, name, engine, price):
        self.name = name
        self.engine = engine
        self.price = price

vehicle = Vehicle("Toyota Rav4", "2.5L", 32000)
```

In [86]:

```
from json import JSONEncoder

class Vehicle:
    def __init__(self, name, engine, price):
        self.name = name
        self.engine = engine
        self.price = price

class VehicleEncoder(JSONEncoder):
    def default(self,o):
        return o.__dict__

vehicle = Vehicle("Toyota Rav4", "2.5L", 32000)

print("Encode Vehicle Object into JSON")
vehicleJson = json.dumps(vehicle, indent=4, cls=VehicleEncoder)
print(vehicleJson)
```

```
Encode Vehicle Object into JSON
{
    "name": "Toyota Rav4",
    "engine": "2.5L",
    "price": 32000
}
```

Exercise 7: Convert the following JSON into Vehicle Object

{ "name": "Toyota Rav4", "engine": "2.5L", "price": 32000 }

For example, we should be able to access Vehicle Object using the dot operator like this.

vehicleObj.name, vehicleObj.engine, vehicleObj.price

In [87]:

```
import json

class Vehicle:
    def __init__(self, name, engine, price):
        self.name = name
        self.engine = engine
        self.price = price

def vehicleDecoder(obj):
        return Vehicle(obj['name'], obj['engine'], obj['price'])

vehicleObj = json.loads('{ "name": "Toyota Rav4", "engine": "2.5L", "price": 32000 }',
           object_hook=vehicleDecoder)

print("Type of decoded object from JSON Data")
print(type(vehicleObj))
print("Vehicle Details")
print(vehicleObj.name, vehicleObj.engine, vehicleObj.price)
```

```
Type of decoded object from JSON Data
<class '__main__.Vehicle'>
Vehicle Details
Toyota Rav4 2.5L 32000
```

Exercise 8: Check whether following json is valid or invalid. If Invalid correct it

```
{
   "company":{
      "employee":{
         "name":"emma",
         "payble":{
            "salary":7000
            "bonus":800
         }
      }
   }
}
```

In [90]:

```
echo { "company":{ "employee":{ "name":"emma", "payble":{ "salary":7000, "bonus":800} } }
```

```
{
    "company": {
        "employee": {
            "name": "emma",
            "payble": {
                "salary": 7000,
                "bonus": 800
            }
        }
    }
}
```

Exercise 9: Parse the following JSON to get all the values of a key 'name' within an array

```
[
   {
      "id":1,
      "name":"name1",
      "color":[
         "red",
         "green"
      ]
   },
   {
      "id":2,
      "name":"name2",
      "color":[
         "pink",
         "yellow"
      ]
   }
]
```

In [91]:

```
import json

sampleJson = """[
   {
      "id":1,
      "name":"name1",
      "color":[
         "red",
         "green"
      ]
   },
   {
      "id":2,
      "name":"name2",
      "color":[
         "pink",
         "yellow"
      ]
   }
]"""

data = []
try:
    data = json.loads(sampleJson)
except Exception as e:
    print(e)

dataList = [item.get('name') for item in data]
print(dataList)
```

```
['name1', 'name2']
```

15. Python NumPy Exercise

Excercise 1:the max item of each row of a matrix.

In [92]:

```
import numpy as np

x=[[1,2,3],[4,5,6],[7,8,9]]

y=np.asarray(x)

print(y)

y.max()
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Out[92]:

```
9
```

Exercise 2: Create a 4X2 integer array and Prints its attributes

Note: The element must be a type of unsigned int16. And print the following Attributes: –

- The shape of an array.
- Array dimensions.
- The Length of each element of the array in bytes.

In [99]:

```
import numpy as np


firstArray = np.empty([4,2], dtype = np.uint16)

print(firstArray)

print("shape of the array is: ",firstArray.shape)
print("dimension of the array is: ",firstArray.ndim)
print("itemsize of each element in array is:",firstArray.itemsize)
```

```
[[41232 40805]
 [17953 58303]
 [ 1887  4407]
 [31432 20031]]
shape of the array is:  (4, 2)
dimension of the array is:  2
itemsize of each element in array is: 2
```

Exercise 3: Create a 5X2 integer array from a range between 100 to 200 such that the difference between each element is 10

In [104]:

```
my_array=np.arange(100,200,10)

my_array=my_array.reshape(5,2)

print(my_array)
```

```
[[100 110]
 [120 130]
 [140 150]
 [160 170]
 [180 190]]
```

Exercise 4: Following is the provided numPy array. Return array of items by taking the third column from all rows

sampleArray = numpy.array([[11 ,22, 33], [44, 55, 66], [77, 88, 99]])

In [112]:

```
sampleArray = np.array([[11 ,22, 33], [44, 55, 66], [77, 88, 99]])

#print(sampleArray)

new_array=sampleArray[...,2]

print(new_array)
```

```
[33 66 99]
```

Exercise 5: Return array of odd rows and even columns from below numpy array

sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24], [27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])

In [2]:

```
import numpy as np
```

In [3]:

```
sampleArray = np.array([[3 ,6, 9, 12], [15 ,18, 21, 24],
[27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])

print(sampleArray)
```

```
[[ 3  6  9 12]
 [15 18 21 24]
 [27 30 33 36]
 [39 42 45 48]
 [51 54 57 60]]
```

In [4]:

```
new_array=sampleArray[1::2,::2]
print(new_array)
```

```
[[15 21]
 [39 45]]
```

Exercise 6: Create a result array by adding the following two NumPy arrays. Next, modify the result array by calculating the square of each element

arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]])

arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])

In [122]:

```
arrayOne = np.array([[5, 6, 9], [21 ,18, 27]])

arrayTwo = np.array([[15 ,33, 24], [4 ,7, 1]])

#print(arrayOne)

#print(arrayTwo)

result_array=arrayOne+arrayTwo

print(result_array)

final_output=pow(result_array,2)

print(final_output)
```

```
[[20 39 33]
 [25 25 28]]
[[ 400 1521 1089]
 [ 625  625  784]]
```

Exercise 7: Split the array into four equal-sized sub-arrays

Note: Create an 8X3 integer array from a range between 10 to 34 such that the difference between each element is 1 and then Split the array into four equal-sized sub-arrays.

In [129]:

```
new_array=np.arange(10,34,1)
new_array=new_array.reshape(8,3)
print(new_array)

# now to divide array into 4 equal division

subarray=np.split(new_array,4)

print("="*100)

print(subarray)
```

```
[[10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]
 [22 23 24]
 [25 26 27]
 [28 29 30]
 [31 32 33]]
====================================================================================
==========================
[array([[10, 11, 12],
       [13, 14, 15]]), array([[16, 17, 18],
       [19, 20, 21]]), array([[22, 23, 24],
       [25, 26, 27]]), array([[28, 29, 30],
       [31, 32, 33]])]
```

Exercise 8: Sort following NumPy array

Case 1: Sort array by the second row

Case 2: Sort the array by the second column

sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])

In [132]:

```
sampleArray = np.array([[34,43,73],[82,22,12],[53,94,66]])

print(sampleArray)

#sorting original array by second row
sortArrayByRow = sampleArray[:,sampleArray[1,:].argsort()]
print("Sorting Original array by secoond row")
print(sortArrayByRow)

print("Sorting Original array by second column")
sortArrayByColumn = sampleArray[sampleArray[:,1].argsort()]
print(sortArrayByColumn)
```

```
[[34 43 73]
 [82 22 12]
 [53 94 66]]
Sorting Original array by secoond row
[[73 43 34]
 [12 22 82]
 [66 94 53]]
Sorting Original array by second column
[[82 22 12]
 [34 43 73]
 [53 94 66]]
```

Exercise 9: Print max from axis 0 and min from axis 1 from the following 2-D array.

sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])

In [134]:

```
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)

minOfAxisOne = numpy.amin(sampleArray, 1)
print("Printing amin Of Axis 1")
print(minOfAxisOne)

maxOfAxisOne = numpy.amax(sampleArray, 0)
print("Printing amax Of Axis 0")
print(maxOfAxisOne)
```

```
Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]
Printing amin Of Axis 1
[34 12 53]
Printing amax Of Axis 0
[82 94 73]
```

Exercise 10: Delete the second column from a given array and insert the following new column in its place.

sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])

newColumn = numpy.array([[10,10,10]])

In [135]:

```
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)

print("Array after deleting column 2 on axis 1")
sampleArray = numpy.delete(sampleArray , 1, axis = 1)
print (sampleArray)

arr = numpy.array([[10,10,10]])

print("Array after inserting column 2 on axis 1")
sampleArray = numpy.insert(sampleArray , 1, arr, axis = 1)
print (sampleArray)
```

```
Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]
Array after deleting column 2 on axis 1
[[34 73]
 [82 12]
 [53 66]]
Array after inserting column 2 on axis 1
[[34 10 73]
 [82 10 12]
 [53 10 66]]
```

16. Python Pandas Exercise

Excercise 1: basic operation of pandas

In [5]:

```
import pandas as pd
```

In [6]:

```
df=pd.read_csv("iris_data.csv")
```

In [7]:

```
df.head()
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [8]:

```
df=df.drop("Id",axis=1)
```

In [9]:

```
df.head()
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [10]:

```
# rename the column name
df=df.rename(columns={"SepalLengthCm":"sepal_length","SepalWidthCm":"sepal_width","PetalL
```

In [13]:

```
df["species"]=df["species"].str.replace("Iris-","")
```

In [14]:

```
df.head()
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [15]:

```
#shape of the table
df.shape
```

Out[15]:

```
(150, 5)
```

In [16]:

```
# how many different species are there?
df["species"].nunique()
```

Out[16]:

```
3
```

In [17]:

```
#how many time each species is repeated in given dataset
df["species"].value_counts()
```

Out[17]:

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

In [18]:

```
#how many rows are there whose sepal length is greater than its mean
(df["sepal_length"]>df["sepal_length"].mean()).sum()
# 70 rows are there whose sepal length is greater than avg sepal length
```

Out[18]:

```
70
```

In [19]:

```
#extract all the rows whose petal width is greater than 1.5 and  petal length is lesser t
df[(df["petal_width"]>1.5) & (df["petal_length"]<5)]
```

Out[19]:

	sepal_length	sepal_width	petal_length	petal_width	species
56	6.3	3.3	4.7	1.6	versicolor
70	5.9	3.2	4.8	1.8	versicolor
85	6.0	3.4	4.5	1.6	versicolor
106	4.9	2.5	4.5	1.7	virginica
121	5.6	2.8	4.9	2.0	virginica
123	6.3	2.7	4.9	1.8	virginica
126	6.2	2.8	4.8	1.8	virginica
127	6.1	3.0	4.9	1.8	virginica
138	6.0	3.0	4.8	1.8	virginica

In [20]:

```
#which species has highest sepal length

df["sepal_length"].max()
```

Out[20]:

```
7.9
```

In [21]:

```
df[df["sepal_length"]==df["sepal_length"].max()]
```

Out[21]:

	sepal_length	sepal_width	petal_length	petal_width	species
131	7.9	3.8	6.4	2.0	virginica

In [22]:

```
#extract mean median of sepal lenth,sepal width ,petal lenth , petal width for all the sp
```

In [23]:

```
df.groupby("species").agg(["mean","median"])
```

Out[23]:

	sepal_length		sepal_width		petal_length		petal_width	
	mean	median	mean	median	mean	median	mean	median
species								
setosa	5.006	5.0	3.418	3.4	1.464	1.50	0.244	0.2
versicolor	5.936	5.9	2.770	2.8	4.260	4.35	1.326	1.3
virginica	6.588	6.5	2.974	3.0	5.552	5.55	2.026	2.0

Excercise 200

In [105]:

```
import pandas as pd
```

In [106]:

```
df = pd.read_csv("Automobile_data.csv", na_values={
'price':["?","n.a"],
'stroke':["?","n.a"],
'horsepower':["?","n.a"],
'peak-rpm':["?","n.a"],
'average-mileage':["?","n.a"]})

df
```

Out[106]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	1
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	1
2	2	alfa-romero	hatchback	94.5	171.2	ohcv	six	154	19	1
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	1
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	1
...	...	...	...	...	...	...	...	...	...	
56	81	volkswagen	sedan	97.3	171.7	ohc	four	85	27	
57	82	volkswagen	sedan	97.3	171.7	ohc	four	52	37	
58	86	volkswagen	sedan	97.3	171.7	ohc	four	100	26	
59	87	volvo	sedan	104.3	188.8	ohc	four	114	23	1
60	88	volvo	wagon	104.3	188.8	ohc	four	114	23	1

61 rows × 10 columns

Exercise 2: From the given dataset print the first and last five rows

In [107]:

```
df.head()
```

Out[107]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	p
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	134
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	165
2	2	alfa-romero	hatchback	94.5	171.2	ohcv	six	154	19	165
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	139
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	174

In [186]:

```
missing_value=["?","n.a",np.NaN]
df=pd.read_csv("Automobile_data.csv",na_values=missing_value)
```

In [187]:

```
df.isnull().sum()
```

Out[187]:

```
index              0
company            0
body-style         0
wheel-base         0
length             0
engine-type        0
num-of-cylinders   0
horsepower         0
average-mileage    0
price              3
dtype: int64
```

In [189]:

```
df.isnull().sum()
```

Out[189]:

```
index               0
company             0
body-style          0
wheel-base          0
length              0
engine-type         0
num-of-cylinders    0
horsepower          0
average-mileage     0
price               3
dtype: int64
```

In [190]:

```
df=df.drop("index",axis=1)
```

Exercise 3: Find the most expensive car company name

Print most expensive car's company name and price.

In [191]:

```
df["price"].max()
```

Out[191]:

```
45400.0
```

In [192]:

```
most_exp_car=df[["company","price"]][df.price==df["price"].max()]
most_exp_car
```

Out[192]:

	company	price
35	mercedes-benz	45400.0

Exercise 3: Print All Toyota Cars details

In [201]:

```
df.groupby("company").get_group("toyota")
```

Out[201]:

	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
48	toyota	hatchback	95.7	158.7	ohc	four	62	35	5348.0
49	toyota	hatchback	95.7	158.7	ohc	four	62	31	6338.0
50	toyota	hatchback	95.7	158.7	ohc	four	62	31	6488.0
51	toyota	wagon	95.7	169.7	ohc	four	62	31	6918.0
52	toyota	wagon	95.7	169.7	ohc	four	62	27	7898.0
53	toyota	wagon	95.7	169.7	ohc	four	62	27	8778.0
54	toyota	wagon	104.5	187.8	dohc	six	156	19	15750.0

Exercise 4: Count total cars per company

In [202]:

```
df["company"].value_counts()
```

Out[202]:

```
toyota           7
bmw              6
mazda            5
nissan           5
audi             4
mercedes-benz    4
mitsubishi       4
volkswagen       4
alfa-romero      3
chevrolet        3
honda            3
isuzu            3
jaguar           3
porsche          3
dodge            2
volvo            2
Name: company, dtype: int64
```

Exercise 5: Find each company's Higesht price car

In [211]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [220]:

```
df.groupby("company")[["company","price"]].max()
```

Out[220]:

company	company	price
alfa-romero	alfa-romero	16500.0
audi	audi	18920.0
bmw	bmw	41315.0
chevrolet	chevrolet	6575.0
dodge	dodge	6377.0
honda	honda	12945.0
isuzu	isuzu	6785.0
jaguar	jaguar	36000.0
mazda	mazda	18344.0
mercedes-benz	mercedes-benz	45400.0
mitsubishi	mitsubishi	8189.0
nissan	nissan	13499.0
porsche	porsche	37028.0
toyota	toyota	15750.0
volkswagen	volkswagen	9995.0
volvo	volvo	13415.0

Exercise 6: Find the average mileage of each car making company

In [221]:

```
df.groupby("company")[["company","average-mileage"]].mean()
```

Out[221]:

company	average-mileage
alfa-romero	20.333333
audi	20.000000
bmw	19.000000
chevrolet	41.000000
dodge	31.000000
honda	26.333333
isuzu	33.333333
jaguar	14.333333
mazda	28.000000
mercedes-benz	18.000000
mitsubishi	29.500000
nissan	31.400000
porsche	17.000000
toyota	28.714286
volkswagen	31.750000
volvo	23.000000

Exercise 7: Sort all cars by Price column

In [223]:

```
df.sort_values("price",ascending=False)
```

Out[223]:

	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
35	mercedes-benz	hardtop	112.0	199.2	ohcv	eight	184	14	45400.0
11	bmw	sedan	103.5	193.8	ohc	six	182	16	41315.0
34	mercedes-benz	sedan	120.9	208.1	ohcv	eight	184	14	40960.0
46	porsche	convertible	89.5	168.9	ohcf	six	207	17	37028.0
12	bmw	sedan	110.0	197.0	ohc	six	182	15	36880.0
...	...	...	...	...	...	...	...	...	...
27	mazda	hatchback	93.1	159.1	ohc	four	68	30	5195.0
13	chevrolet	hatchback	88.4	141.1	l	three	48	47	5151.0
22	isuzu	sedan	94.5	155.9	ohc	four	70	38	NaN
23	isuzu	sedan	94.5	155.9	ohc	four	70	38	NaN
47	porsche	hatchback	98.4	175.7	dohcv	eight	288	17	NaN

61 rows × 9 columns

Exercise 8: Concatenate two data frames using the following conditions

In [224]:

```
GermanCars = {'Company': ['Ford', 'Mercedes', 'BMV', 'Audi'], 'Price': [23845, 171995, 13
japaneseCars = {'Company': ['Toyota', 'Honda', 'Nissan', 'Mitsubishi '], 'Price': [29995,
```

In [225]:

```
germanCar=pd.DataFrame.from_dict(GermanCars)
```

In [226]:

```
germanCar
```

Out[226]:

	Company	Price
0	Ford	23845
1	Mercedes	171995
2	BMV	135925
3	Audi	71400

In [227]:

```
japaneseCar=pd.DataFrame.from_dict(japaneseCars)
japaneseCar
```

Out[227]:

	Company	Price
0	Toyota	29995
1	Honda	23600
2	Nissan	61500
3	Mitsubishi	58900

In [229]:

```
carsDF=pd.concat([germanCar,japaneseCar],keys=["germany","Japan"])
carsDF
```

Out[229]:

		Company	Price
germany	0	Ford	23845
	1	Mercedes	171995
	2	BMV	135925
	3	Audi	71400
Japan	0	Toyota	29995
	1	Honda	23600
	2	Nissan	61500
	3	Mitsubishi	58900

Exercise 9: Merge two data frames using the following condition

In [230]:

```
Car_Price = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'Price': [23845, 17995, 13592
car_Horsepower = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'horsepower': [141, 80,
```

In [232]:

```
carPrice=pd.DataFrame.from_dict(Car_Price)
carPrice
```

Out[232]:

	Company	Price
0	Toyota	23845
1	Honda	17995
2	BMV	135925
3	Audi	71400

In [233]:

```
carHorsepower=pd.DataFrame.from_dict(car_Horsepower)
carHorsepower
```

Out[233]:

	Company	horsepower
0	Toyota	141
1	Honda	80
2	BMV	182
3	Audi	160

In [234]:

```
carsDF=pd.merge(carPrice,carHorsepower,on="Company")
carsDF
```

Out[234]:

	Company	Price	horsepower
0	Toyota	23845	141
1	Honda	17995	80
2	BMV	135925	182
3	Audi	71400	160

17. Python Matplotlib Exercise

Exercise 1: Read Total profit of all months and show it using a line plot

Total profit data provided for each month. Generated line plot must include the following properties: –

X label name = Month Number

Y label name = Total profit

In [6]:

```
import pandas as pd
import matplotlib.pyplot as plt

sales_data=pd.read_csv("company_sales_data.csv")
sales_data.head()
```

Out[6]:

	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	tota
0	1	2500	1500	5200	9200	1200	1500	
1	2	2630	1200	5100	6100	2100	1200	
2	3	2140	1340	4550	9550	3550	1340	
3	4	3400	1130	5870	8870	1870	1130	
4	5	3600	1740	4560	7760	1560	1740	

In [21]:

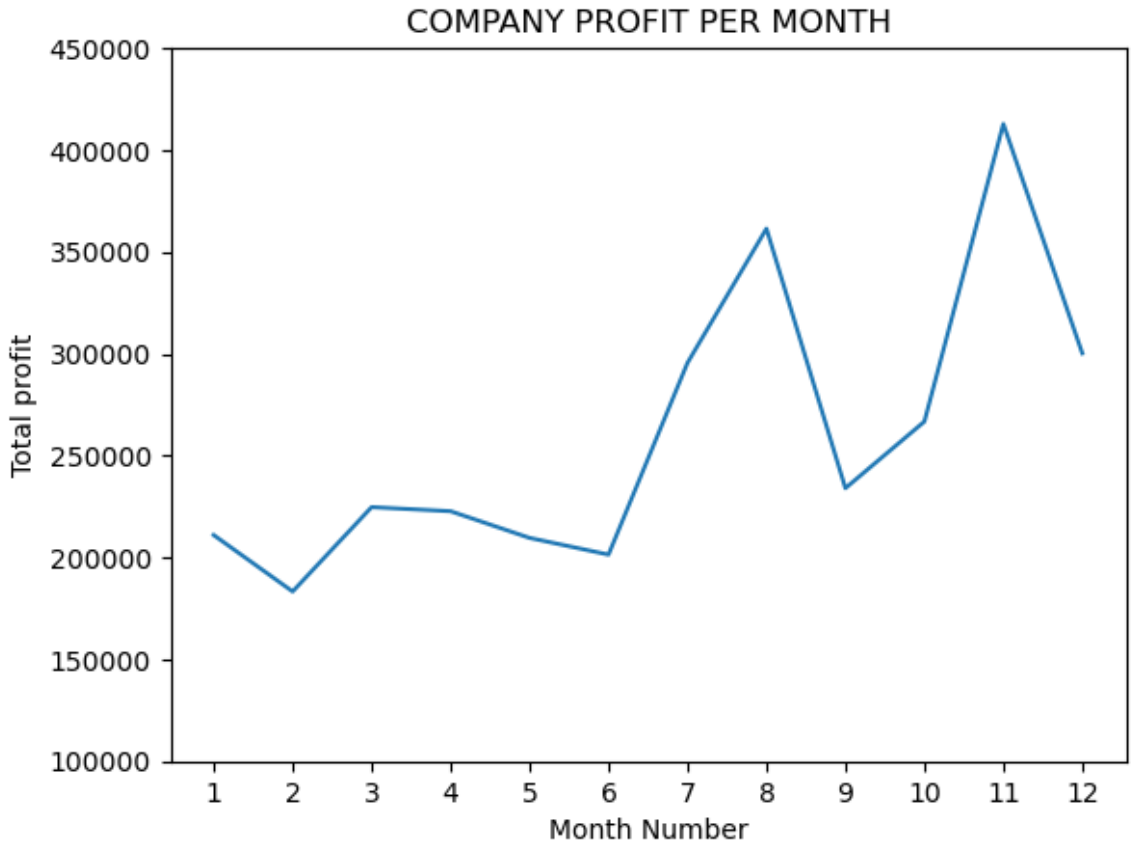
```
import numpy as np
```

In [33]:

```
monthList=sales_data["month_number"].tolist()
profitList=np.arange(100000,500000,50000).tolist()
```

In [36]:

```
plt.plot("month_number","total_profit",data=sales_data)
plt.xlabel("Month Number")
plt.xticks(monthList)
plt.yticks(profitList)
plt.ylabel("Total profit")
plt.title("COMPANY PROFIT PER MONTH")
plt.show()
```



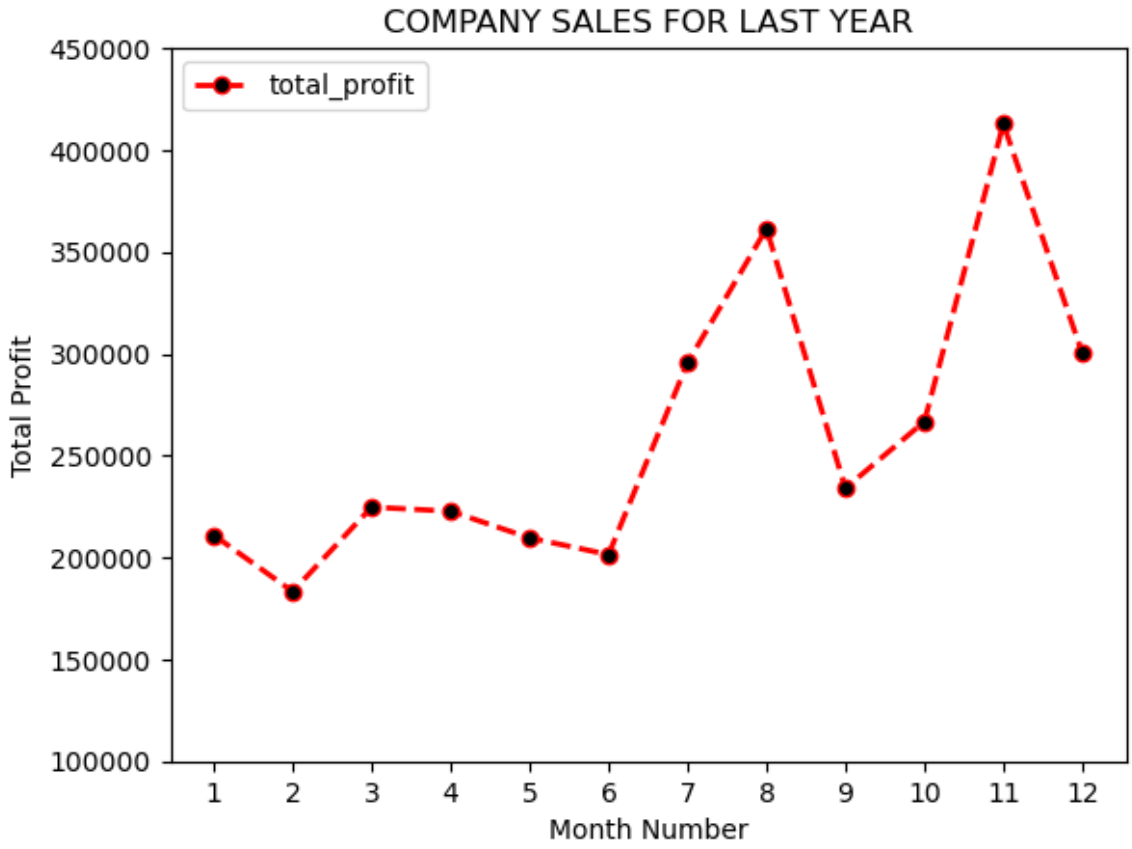
Exercise 2: Get total profit of all months and show line plot with the following Style properties

Generated line plot must include following Style properties: –

- Line Style dotted and Line-color should be red
- Show legend at the lower right location.
- X label name = Month Number
- Y label name = Sold units number
- Add a circle marker.
- Line marker color as read
- Line width should be 3

In [58]:

```
plt.plot("month_number","total_profit",data=sales_data,color="red", marker='o',markerface
linewidth=2, markersize=6)
plt.xlabel("Month Number")
plt.xticks(monthList)
plt.ylabel("Total Profit")
plt.yticks(profitList)
plt.legend(loc="upper left")
plt.title("COMPANY SALES FOR LAST YEAR")
plt.show()
```



Exercise 3: Read all product sales data and show it using a multiline plot

Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).

The graph should look like this.

In [59]:

```
sales_data.head()
```

Out[59]:

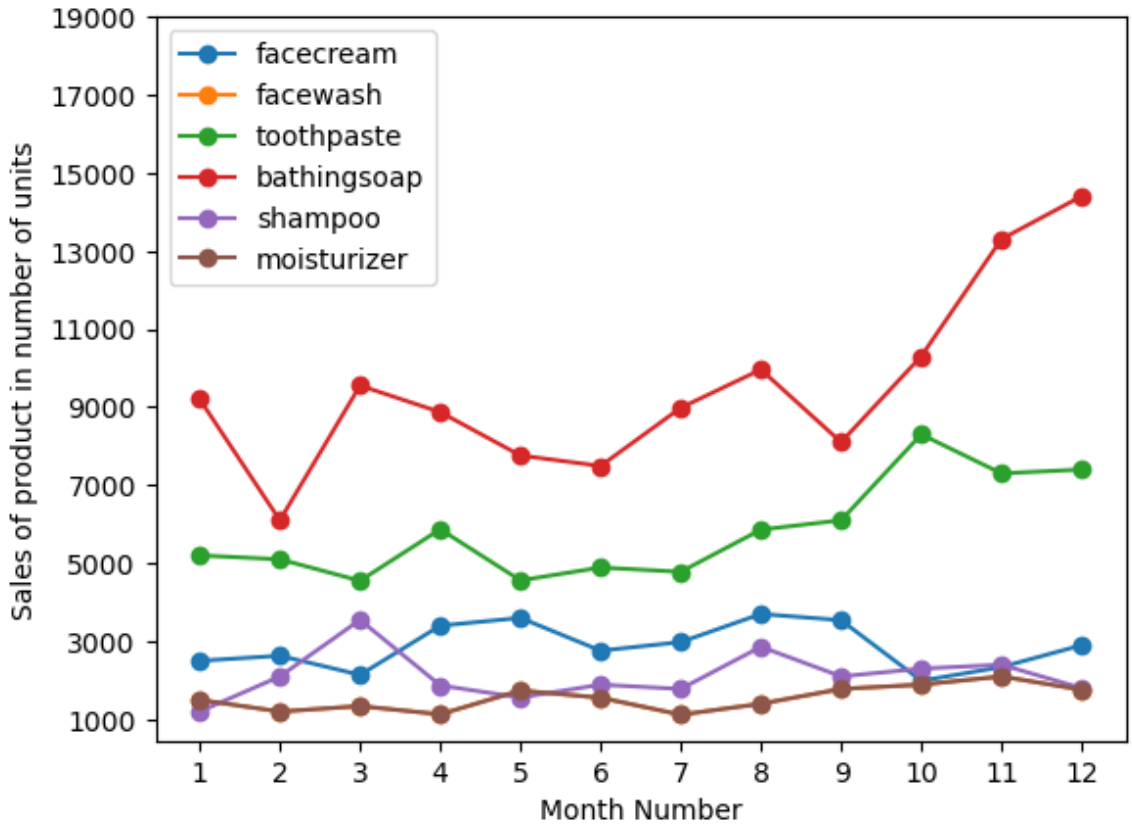
	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	tota
0	1	2500	1500	5200	9200	1200	1500	
1	2	2630	1200	5100	6100	2100	1200	
2	3	2140	1340	4550	9550	3550	1340	
3	4	3400	1130	5870	8870	1870	1130	
4	5	3600	1740	4560	7760	1560	1740	

In [91]:

```
quantityList=np.arange(1000,20000,2000)
```

In [92]:

```
plt.plot("month_number","facecream",data=sales_data,marker="o")
plt.plot("month_number","facewash",data=sales_data,marker="o")
plt.plot("month_number","toothpaste",data=sales_data,marker="o")
plt.plot("month_number","bathingsoap",data=sales_data,marker="o")
plt.plot("month_number","shampoo",data=sales_data,marker="o")
plt.plot("month_number","moisturizer",data=sales_data,marker="o")

plt.xticks(monthList)
plt.xlabel("Month Number")
plt.yticks(quantityList)
plt.ylabel("Sales of product in number of units")
plt.legend(loc="upper left")
plt.show()
```



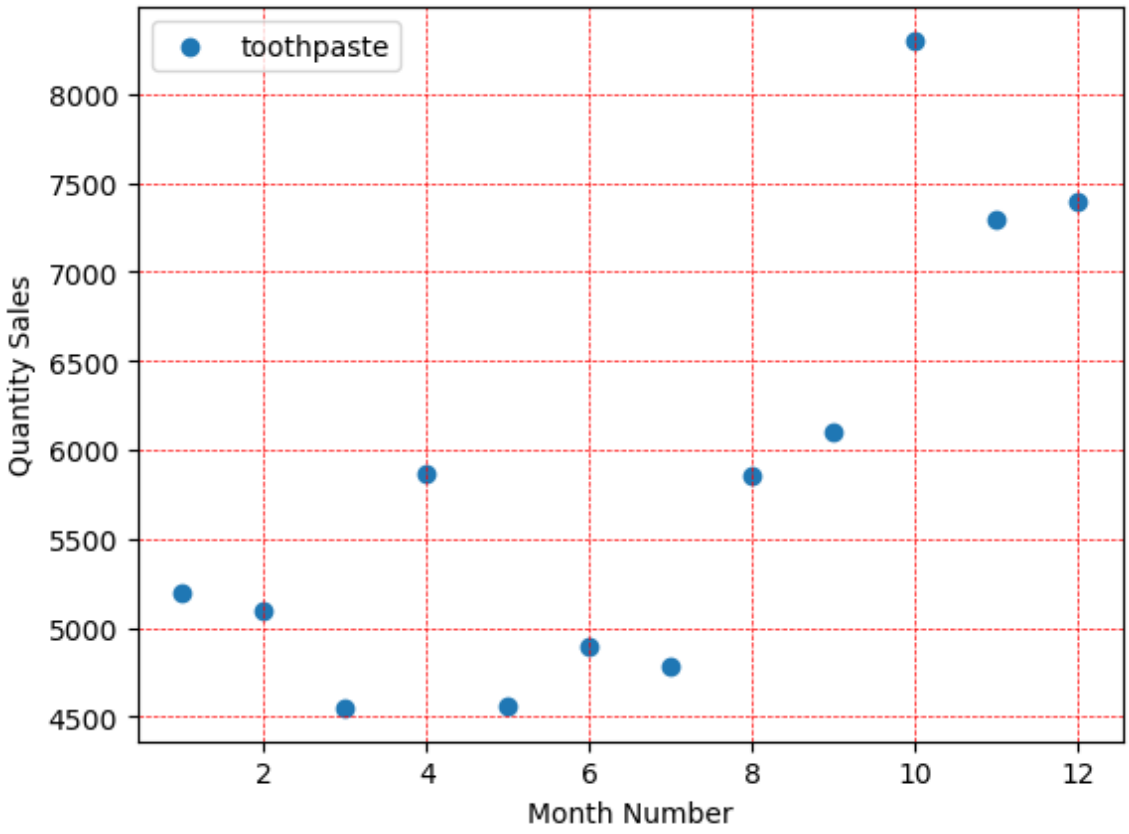
#since column values for facewash and moisturizer are same line plot is overlapping each other

Exercise 4: Read toothpaste sales data of each month and show it using a scatter plot

Also, add a grid in the plot. gridline style should “–“.

In [103]:

```
plt.scatter("month_number","toothpaste",data=sales_data)
plt.legend(loc="upper left")
plt.grid(color='r', linestyle='dashed', linewidth=0.5)
plt.xlabel("Month Number")
plt.ylabel("Quantity Sales")
plt.show()
```

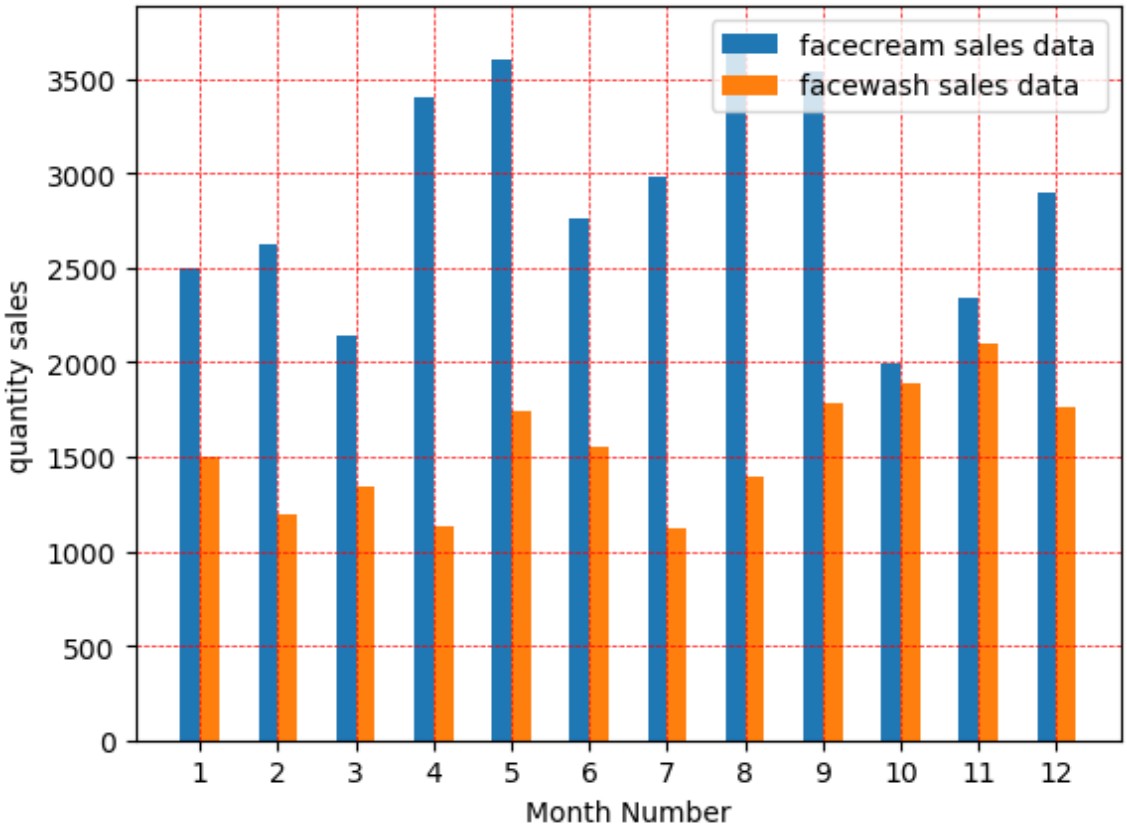


Exercise 5: Read face cream and facewash product sales data and show it using the bar chart

The bar chart should display the number of units sold per month for each product. Add a separate bar for each product in the same chart.

In [129]:

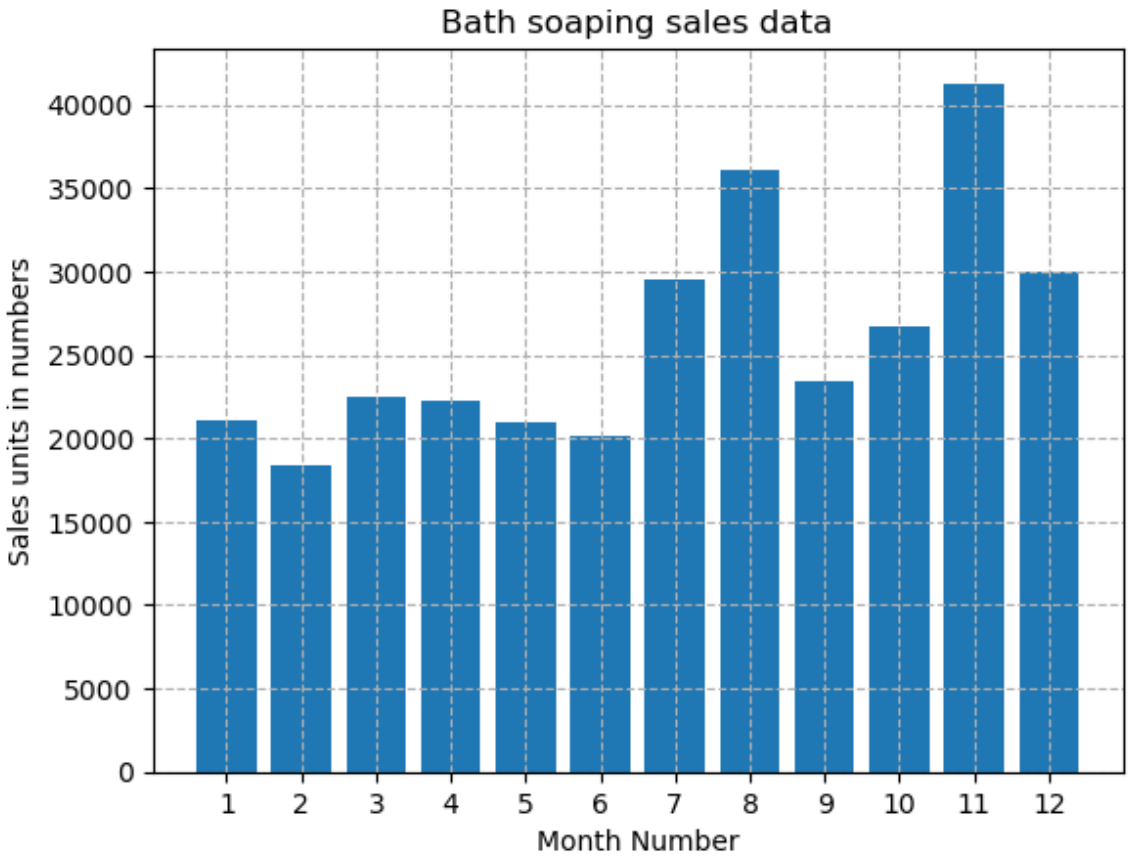
```
plt.bar([a-0.25 for a in monthList],"facecream",data=sales_data,width=0.25,align="edge",l
plt.bar([a+0.25 for a in monthList],"facewash",data=sales_data,width=-0.25,align="edge",l

plt.xticks(monthList)
plt.xlabel("Month Number")
plt.ylabel("quantity sales")
plt.legend()
plt.grid(color='r', linestyle='dashed', linewidth=0.5)
plt.show()
```



Exercise 6: Read sales data of bathing soap of all months and show it using a bar chart. Save this plot to your hard disk

In [139]:

```
plt.bar("month_number","total_units",data=sales_data)
plt.grid(True,linestyle="dashed")
plt.xticks(monthList)
plt.xlabel("Month Number")
plt.ylabel("Sales units in numbers")
plt.title("Bath soaping sales data")
plt.show()
```



Exercise 7: Read the total profit of each month and show it using the histogram to see the most common profit ranges

In [144]:

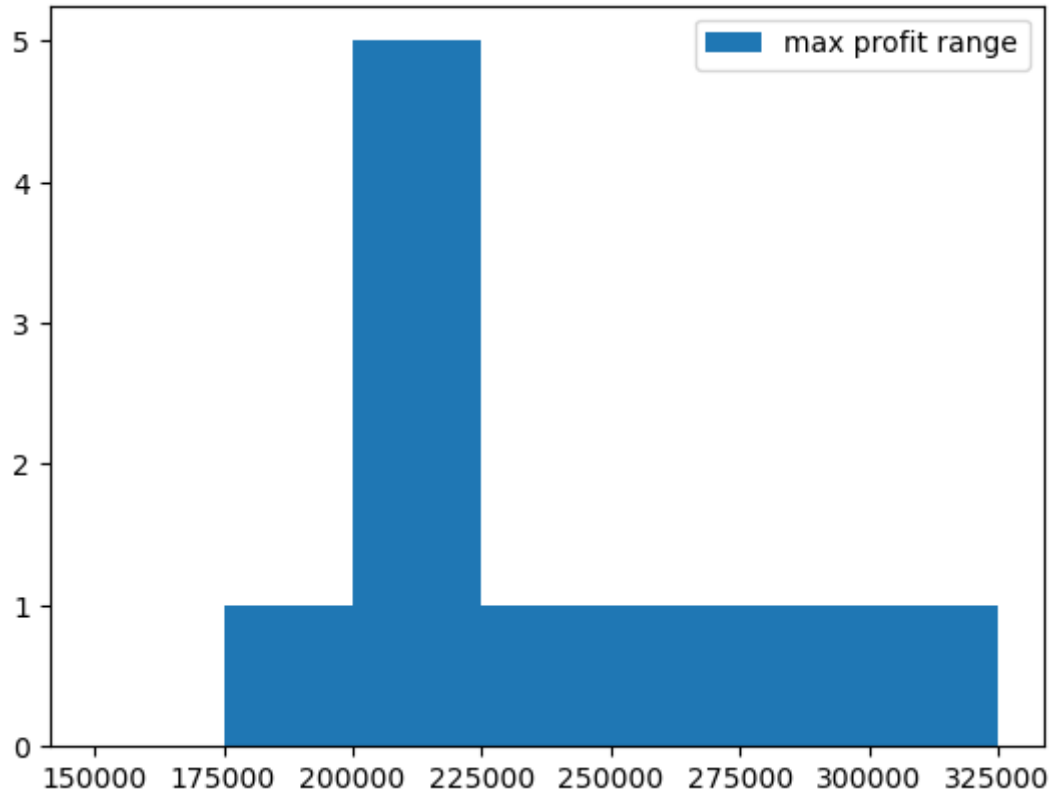
```
profitList=df["total_profit"].tolist()
```

In [151]:

```
profit_range=np.arange(150000,350000,25000)
```

In [156]:

```
plt.hist(profitList,profit_range,label="max profit range")
plt.legend()
plt.show()
```



Exercise 8: Calculate total sale data for last year for each product and show it using a Pie chart

In [174]:

```
sales_data.head(2)
```

Out[174]:

	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	tota
0	1	2500	1500	5200	9200	1200	1500	
1	2	2630	1200	5100	6100	2100	1200	

In [159]:

```
saleData=[df["facecream"].sum(),df["facewash"].sum(),df["toothpaste"].sum(),df["bathingsc
saleData
```

Out[159]:

```
[34480, 18515, 69910, 114010, 25410, 18515]
```

In [169]:

```
labels = ['FaceCream', 'FaseWash', 'ToothPaste', 'Bathing soap', 'Shampoo', 'Moisturizer'
```

In [173]:

```
plt.pie(saleData,autopct="%1.1f%%",labels=labels)
plt.legend()
plt.show()
```

Exercise 9: Read Bathing soap and facewash of all months and display it using the Subplot

In [198]:

```
plt.subplot(2,1,1)
plt.plot("month_number","bathingsoap",data=sales_data,color="black",marker="o")
plt.title("sales data of bathing soap")

plt.subplot(2,1,2)
plt.plot("month_number","facewash",data=sales_data,color="red",marker="o")
plt.xlabel("Month Number")
plt.title("sales data of facewash")
plt.ylabel("sales units in numbers")

plt.subplots_adjust(bottom=0.1,wspace=0.4,hspace=0.3)

plt.show()
```

18. Python random Data generation Exercise

Exercise 1: Generate 3 random integers between 100 and 999 which is divisible by 5

In [205]:

```
import random

print("generating three digit number from 100 to 999 which id divisible by 5")

for i in range(3):
    print(random.randrange(100,999,5))
```

```
generating three digit number from 100 to 999 which id divisible by 5
365
660
815
```

Exercise 2: Random Lottery Pick. Generate 100 random lottery tickets and pick two lucky tickets from it as a winner.

Note you must adhere to the following conditions:

The lottery number must be 10 digits long.

All 100 ticket number must be unique.

In [200]:

```
import random

lottery_tickets_list = []
print("creating 100 random lottery tickets")
# to get 100 ticket
for i in range(100):
    # ticket number must be 10 digit (1000000000, 9999999999)
    lottery_tickets_list.append(random.randrange(1000000000, 9999999999))
# pick 2 luck tickets
winners = random.sample(lottery_tickets_list, 2)
print("Lucky 2 lottery tickets are", winners)
```

```
creating 100 random lottery tickets
Lucky 2 lottery tickets are [5072541791, 3990165714]
```

Exercise 3: Generate 6 digit random secure OTP

In [202]:

```
import secrets

OTPgenerator=secrets.SystemRandom()

print("six digit random OTP")

otp=OTPgenerator.randrange(100000,999999)

print("secure OTP is ",otp)
```

```
six digit random OTP
secure OTP is  360353
```

Exercise 4: Pick a random character from a given String

In [207]:

```
name="Milind Mali"

char=random.choice(name)

print(char)
```

```
M
```

Exercise 5: Generate random String of length 5

Note: String must be the combination of the UPPER case and lower case letters only. No numbers and a special symbol.

In [209]:

```
import random
import string

def randomstring(stringLength):
    """gernerate random string of five characters"""

    letters=string.ascii_letters
    return "".join(random.choice(letters) for i in range(stringLength))

print("random string is ",randomstring(5))
```

```
random string is  ryLoK
```

Exercise 6: Generate a random Password which meets the following conditions

Password length must be 10 characters long.

It must contain at least 2 upper case letters, 1 digit, and 1 special symbol.

In [212]:

```
import random
import string

def randomPassword():
    randomSource=string.ascii_letters+string.digits+string.punctuation
    password=random.sample(randomSource,6)
    password+=random.sample(string.ascii_uppercase,2)
    password+=random.choice(string.digits)
    password+=random.choice(string.punctuation)

    passwordList=list(password)

    random.SystemRandom().shuffle(passwordList)
    password="".join(passwordList)
    return password

print("password is: ",randomPassword())
```

```
password is:  b|VW4D4N.T
```

Exercise 7: Calculate multiplication of two random float numbers

Note:

First random float number must be between 0.1 and 1 Second random float number must be between 9.5 and 99.5

In [215]:

```
num1=random.random()

num2=random.uniform(9.5,99.5)

num3=num1+num2

print(num3)
```

```
10.93030484756581
```

Exercise 8: Generate random secure token of 64 bytes and random URL

In [216]:

```
import secrets

print("Random secure Hexadecimal token is ", secrets.token_hex(64))
print("Random secure URL is ", secrets.token_urlsafe(64))
```

```
Random secure Hexadecimal token is  71782a425a21f04c97b2e0d57eda580b1fda8c
a27c39ac29a7a10fd04e49e841770028f487993af55794640e3f4c00f9e384b9acf4e895bb
1cc04e669b3a9789
Random secure URL is  u4hJfIfWVSZwxsqsLDxJ58O67dlcVWgBLFQuSKBKDi3fXYxivdpN
oPlU0zN-ikgHg80bUWhNJSUCPV8qTGNUEg
```

Exercise 9: Roll dice in such a way that every time you get the same number

Dice has 6 numbers (from 1 to 6). Roll dice in such a way that every time you must get the same output number. do this 5 times.

In [218]:

```
import random

dice = [1, 2, 3, 4, 5, 6]
print("Randomly selecting same number of a dice")
for i in range(5):
    random.seed(25)
    print(random.choice(dice))
```

```
Randomly selecting same number of a dice
4
4
4
4
4
```

Exercise 10: Generate a random date between given start and end dates

In [219]:

```
import random
import time

def getRandomDate(startDate, endDate ):
    print("Printing random date between", startDate, " and ", endDate)
    randomGenerator = random.random()
    dateFormat = '%m/%d/%Y'

    startTime = time.mktime(time.strptime(startDate, dateFormat))
    endTime = time.mktime(time.strptime(endDate, dateFormat))

    randomTime = startTime + randomGenerator * (endTime - startTime)
    randomDate = time.strftime(dateFormat, time.localtime(randomTime))
    return randomDate

print ("Random Date = ", getRandomDate("1/1/2016", "12/12/2018"))
```

```
Printing random date between 1/1/2016  and  12/12/2018
Random Date =  04/06/2018
```

Thank you very much !!!