```
# introduction to numpy
# numpy is an open source library that is used for handle scintific calculations.
# different b/w numpy and array---->
# (1) list hetrogeneous data type and numpy is homogeneous data type.
# (2) list takes more memory as compare to numpy.
# (3) list takes more execution time as compare to numpy.
```

```
import numpy as np
li=[x for x in range(1,10)]
```

```
import numpy as np
a=np.ones((5,5))
a.shape
a.size
a.ndim
a
```

```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

```
a=np.array([1,25])
a
b=np.ones((5,5))
b
```

```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

```
np.zeros((5,5),dtype=int)
```

```
array([[0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0]])
```

```
#import numpy as np
#a=np.array([1,2,3,4,5])
#np.arange(1,91,1).reshape(8,8)
```

```
import numpy as np
```

```
a=[1,2,34,4,5,67]
type(a)
```

```
b=np.array(a)
b
```

```
type(b)
```

```
# user defined array
```

```
a=[]
size=int(input("enter size "))
for i in range(size):
  val=int(input("enter number"))
  a.append(val)
b=np.array(a)
b
```

```
import numpy as np
a= [[[1,2,3],[4,5,6],[4,5,6]]]
b = np.array(a)
b
```

```
type(a)
```

```
b=np.arange(2,20,2).reshape(3,3)
b
```

```
# matrix----> rows coloms
# row(1)=[1,2,3]
# row(2)=[4,5,6]
# row(3)=[4,7,8]
# col(1)=[1,4,4]
#col(2)=[2,5,7]
#col(3)=[3,6,8]
# total element=col*row=3*3=9
```

```
print("total rows and  columns:",b.shape)
print("total element:",b.size)
```

```
a=[[1,2,4,7],[3,4,5,3],[3,6,7,5]]
b=np.array(a)
b
```

```
print("total rows and  columns:",b.shape)
print("total element:",b.size)
```

```
# how to check dimension of an matrix.
```

```
a=np.array([1,2,3])
a
```

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
a
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
a.ndim
```

```
2
```

```
a=np.array([[[1,2,3],[4,5,6],[7,8,9]]])
a
```

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

```
a.ndim
```

```
3
```

```
# images----> pixels(0-255)----> normalize(0,1)----> matrix----> operations
# dl----> artifical neurous---> weight----> randonly creation----> numpy
# datasets----> dummy entry----> random module----> attach pandas
```

```
# predefined functions in numpy.
```

```
#(1) zero()----> it will create an array in which all the value will be one in either
```

```
import numpy as np

a=np.zeros(3)
a
```

```
array([0., 0., 0.])
```

```
a=np.zeros((3,4))
a
```

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

```
#(2) ones()----> it will create an array in which all the values will be 1 in either
```

```
import numpy as np
a=np.ones((3))
a
```

```
array([1., 1., 1.])
```

```
import numpy as np
a=np.ones((3,5))
a
```

```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
```

```
       [1., 1., 1., 1., 1.]])
```

```
# (3).eye---> it will create an array in which digonal positional element will be 1 a
# symmetric matrix
a=np.eye(3)
a
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
# assymetric matrix
a=np.eye(3,4)
a
```

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.]])
```

```
# (4) diag---->it will create an array in which we can set any custom values at digon
a=np.diag([1,2,3,4])
a
```

```
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

```
# (5) random module
# (a) randint()---> it will create an array create randomly in a given range.
# syntax: np.random.randint(min_num,max_num,total_num)
```

```
import numpy as np
```

```
import numpy as np
a=np.random.randint(1,5,2)
a
```

```
array([4, 4])
```

```
a=np.random.rand(4)
a
```

```
array([0.88416894, 0.92705441, 0.61903215, 0.46683803])
```

```
np.random.seed(23)
a=np.random.randint(1,5,2)
a
```

```
array([4, 3])
```

```
# view vs copy
```

```
a=np.array([1,2,3,4,5,6])
a
```

```
array([1, 2, 3, 4, 5, 6])
```

```
b=a[1:4]
b[:]=0
b
```

```
array([0, 0, 0])
```

```
b=a[1:5]
b[:]=0
b
```

```
array([0, 0, 0, 0])
```

```
a
```

```
array([1, 0, 0, 0, 0, 6])
```

```
# copy
```

```
a=np.array([1,2,3,4,5,6])
b=a[1:4].copy()
b[:]=0
a
```

```
array([1, 2, 3, 4, 5, 6])
```

```
# reshaping the array
# total_row*coloum=total_element
```

```
a=np.random.randint(1,50,12)
a
```

```
array([41, 10, 41, 32, 46, 13, 28, 40, 27, 26,  7, 46])
```

```
# 1*4  4*1
# 2*4 4*2
```

```
a.reshape(2,6)
```

```
array([[41, 10, 41, 32, 46, 13],
       [28, 40, 27, 26,  7, 46]])
```

```
import numpy as np

a.reshape(3,3)
```

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

```
a.reshape(3,3)
```

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

```
a.shape
```

```
(12,)
```

```
import numpy as np
a.reshape(-1, 3)
```

```
array([[41, 10, 41],
       [32, 46, 13],
       [28, 40, 27],
       [26,  7, 46]])
```

```
a.reshape(1,-2)
```

```
array([[41, 10, 41, 32, 46, 13, 28, 40, 27, 26,  7, 46]])
```

```
# operations on array
```

```
a=np.arange(1,9,3)
a
```

```
array([1, 4, 7])
```

```
a=np.arange(1,10)
a
```

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

```
a*2
```

```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18])
```

```
a=np.arange(5,9)
a
```

```
array([5, 6, 7, 8])
```

```
a=np.arange(1,10).reshape(3,3)
a
```

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

```
a=np.arange(4,8).reshape(2,2)
a
```

```
array([[4, 5],
       [6, 7]])
```

```
b=np.arange(5,9).reshape(2,2)
b
```

```
array([[5, 6],
       [7, 8]])
```

```
a+b
```

```
array([[ 9, 11],
       [13, 15]])
```

```
a-b
```

```
array([[-1, -1],
       [-1, -1]])
```

```
a.dot(b)
```

```
array([[55, 64],
       [79, 92]])
```

```
a=np.arange(1,8)
a
```

```
array([1, 2, 3, 4, 5, 6, 7])
```

```
a>5
```

```
array([False, False, False, False, False,  True,  True])
```

```
b=a>5
a[b]
```

```
array([6, 7])
```

```
a=np.arange(1,8)
a**2
```

```
array([ 1,  4,  9, 16, 25, 36, 49])
```

```
a**3
```

```
array([  1,   8,  27,  64, 125, 216, 343])
```

```
a=np.arange(1,10).reshape(3,3)
a
```

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

```
np.sum(a)
```

```
np.int64(45)
```

```
np.sum(a,axis=1)                    # row operation
a
```

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

```
a=np.sum(a,axis=0)                # column operation
a
```

```
array([12, 15, 18])
```

```
# unique()----> it will create  an array in which it will remove duplicate values and i
# arr ----> unique elements of array.
# return_index = true ----> indexing of unique element.
# return_counts = true ----> frequency of each unique element.
```

```
a = np.array([1,2,3,4,1,3,5,2,7,2,5,6,7])
a
```

```
array([1, 2, 3, 4, 1, 3, 5, 2, 7, 2, 5, 6, 7])
```

```
b = np.unique(a,return_index = True, return_counts = True)
b
```

```
(array([1, 2, 3, 4, 5, 6, 7]),
 array([ 0,  1,  2,  3,  6, 11,  8]),
 array([2, 3, 2, 1, 2, 1, 2]))
```

```
# linspace()----> it will return an array in which element will be same gap in a give
# syntax : np.linspace(min_num , max_num ,total_numbers)
```

```
a = np.linspace(1,2,5)
a
```

```
array([1.  , 1.25, 1.5 , 1.75, 2.  ])
```

```
# horizontal and vertical stacking---->
# horizontal stacking ----> it will add 2 and more arrays in horizontally.
```

```
a = np.arange(1,5)
a
```

```
array([1, 2, 3, 4])
```

```
b = np.arange(5,9)
b
```

```
array([5, 6, 7, 8])
```

```
c = np.arange(9,13)
c
```

```
array([ 9, 10, 11, 12])
```

```
np.hstack((a,b,c))
```

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

```
np.vstack((a,b,c))
```

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

```
# introduction in pandas---->
# pandas is an open source library of python which is used for handle data manipulati
```

```
import pandas as pd
```

```
# data structure of pandas---->
#(1)series
#(2)dataframe
```

```
# series----> it is one dimensional array and it return only value not column name.
```

```
a=pd.Series([1,2,3,4,5])
a
```

	0
0	1
1	2
2	3
3	4
4	5

dtype: int64

```
type(a)
```

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy:
bool | None=None, fastpath: bool | lib.NoDefault=lib.no_default) -> None
```

```
One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
methods for performing operations involving the index. Statistical
methods from ndarray have been overridden to automatically exclude
```

```
# dataframe---->  it is multi dimensional array and it return column with value name.
```

```
# task:: we have to create a dataframe of employees in which column are
# emp_id , emp_name,department,salary,working_hours
```

```
a={"emp_id":['101','102','103','104','105','106'],
    "emp_name":['divya','chiya','sourabh','anil','vartika','keshav'],
    "department":['abc','dfg','tyu','yio','sdf','hjkl'],
    "salary":['1000','6789','34546','34567','34567','34567'],
    "working hours":['1','2','4','6','8','8']}
```

```
import pandas as pd
df=pd.DataFrame(a)
df
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
# how can we export our dataframe into csv data ?
```

```
df.to_csv("emp_info.csv")
df                                          # TO_CSV MEAN EXPORT YOUR DATA INTO CSV
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
df.to_csv('new_emp_info.csv',index=False)            # INDEX=FALSE MEAN IT WILL REMO
```

```
# HOW TO IMPORT CSV DATA USING PANDAS
```

```
import pandas as pd
df=pd.read_csv("/content/covid_toy - covid_toy.csv")
df.head()
```

	age	gender	fever	cough	city	has_covid
0	60	Male	103.0	Mild	Kolkata	No
1	27	Male	100.0	Mild	Delhi	Yes
2	42	Male	101.0	Mild	Delhi	No
3	31	Female	98.0	Mild	Kolkata	No
4	65	Female	101.0	Mild	Mumbai	No

```
# df
```

```
import pandas as pd
df=pd.read_csv("/content/covid_toy - covid_toy.csv",encoding='latin-1')
df
```

AI prompt cell

Show code

Double-click (or enter) to edit

```
# how can we check total column in this dataframe
```

```
df.columns
```

```
Index(['emp_id', 'emp_name', 'department', 'salary', 'working hours'], dtype='object')
```

```
# how can we check top 5 data from the dataframe.
```

```
df.head(2)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2

```
# how can we check bottom data from the dataframe.
```

```
df.tail(2)
```

	emp_id	emp_name	department	salary	working hours
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
df.sample(2)
```

	age	gender	fever	cough	city	has_covid
86	25	Male	104.0	Mild	Bangalore	Yes
78	11	Male	100.0	Mild	Bangalore	Yes

```
# how can we check datatype of all column
```

```
df.dtypes
```

	0
age	int64
gender	object
fever	float64
cough	object
city	object
has_covid	object

dtype: object

```
# how can we check statically view of dataframe.
```

```
df.describe()
```

	age	fever
count	100.000000	90.000000
mean	44.220000	100.844444
std	24.878931	2.054926
min	5.000000	98.000000
25%	20.000000	99.000000
50%	45.000000	101.000000
75%	66.500000	102.750000
max	84.000000	104.000000

```
# how can we check ovarall information in the dataframe
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   age        100 non-null    int64  
 1   gender     100 non-null    object 
 2   fever      90 non-null     float64
```

```
 3   cough      100 non-null    object
 4   city       100 non-null    object
 5   has_covid  100 non-null    object
dtypes: float64(1), int64(1), object(4)
memory usage: 4.8+ KB
```

```
df.head(2)
```

	age	gender	fever	cough	city	has_covid
0	60	Male	103.0	Mild	Kolkata	No
1	27	Male	100.0	Mild	Delhi	Yes

```
# how can we check missing value in a dataframe
```

```
df.isnull().sum()
```

	0
age	0
gender	0
fever	10
cough	0
city	0
has_covid	0

dtype: int64

```
df['fever'].mean()
```

```
np.float64(100.84444444444445)
```

```
df['fever']=df['fever'].fillna(df['fever'].mean())
```

```
df.isnull().sum()
```

	0
age	0
gender	0
fever	10
cough	0
city	0
has_covid	0

dtype: int64

```
df.head(3)
```

	age	gender	fever	cough	city	has_covid
0	60	Male	103.0	Mild	Kolkata	No
1	27	Male	100.0	Mild	Delhi	Yes
2	42	Male	101.0	Mild	Delhi	No

```
# how can we check total sub-categories in a column.
```

```
df['gender'].value_counts()
```

gender	count
Female	59
Male	41

dtype: int64

```
df['has_covid'].value_counts()
```

has_covid	count
No	55
Yes	45

dtype: int64

```
df['gender']=df['gender'].map({'male':0,'female':1})
```

```
df['cough']=df['cough'].map({'mild':0,'strong':1})
```

```
df['city']=df['city'].map({'mumbai':0,'delhi':1})
```

```
df['has_covid']=df['has_covid'].map({'yes':1,'no':0})
```

```
# data
# categorical(descride)----> we cannot divide futher
# numerical(continuous)-----> we can divide futher
```

```
# loc() and iloc()
# df.loc[row_range , column_name]  ----> start value include and last value include.
# df.iloc[row_range , column_range]   ----> start value include but last value exclud
```

```
import pandas as pd
df.loc[5:10,['age','fever','cough']]
```

	age	fever	cough
5	84	NaN	Mild
6	14	101.0	Strong
7	20	NaN	Strong
8	19	100.0	Strong
9	64	101.0	Mild
10	75	NaN	Mild

```
df.iloc[5:10 ,[0,2]]=55
```

```
df.head(11)
```

	age	gender	fever	cough	city	has_covid
0	60	Male	103.0	Mild	Kolkata	No
1	27	Male	100.0	Mild	Delhi	Yes
2	42	Male	101.0	Mild	Delhi	No
3	31	Female	98.0	Mild	Kolkata	No
4	65	Female	101.0	Mild	Mumbai	No
5	55	Female	55.0	Mild	Bangalore	Yes
6	55	Male	55.0	Strong	Bangalore	No
7	55	Female	55.0	Strong	Mumbai	Yes
8	55	Female	55.0	Strong	Bangalore	No
9	55	Female	55.0	Mild	Delhi	No
10	75	Female	NaN	Mild	Delhi	No

```
df['gender'].value_counts()
```

gender	count
Female	59
Male	41

dtype: int64

```
df.head(2)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2

```
df['gender']=df['gender'].map({'male':0,'female':1})
df['cough']=df['cough'].map({"mild":0,"strong":1})
```

```
df['has_covid']=df['has_covid'].map({"yes":1,"NO":0})
```

```
df.sample(3)
```

	age	gender	fever	cough	city	has_covid
86	25	NaN	104.0	NaN	Bangalore	Yes
19	42	NaN	NaN	NaN	Bangalore	Yes
53	83	NaN	98.0	NaN	Delhi	Yes

```
df.drop(columns=['city'])
```

	age	gender	fever	cough	has_covid
0	60	NaN	103.0	NaN	No
1	27	NaN	100.0	NaN	Yes
2	42	NaN	101.0	NaN	No
3	31	NaN	98.0	NaN	No
4	65	NaN	101.0	NaN	No
...	...	...	...	...	...
95	12	NaN	104.0	NaN	No
96	51	NaN	101.0	NaN	Yes
97	20	NaN	101.0	NaN	No
98	5	NaN	98.0	NaN	No
99	10	NaN	98.0	NaN	Yes

100 rows × 5 columns

```
df=df.drop(columns=['city'])   # how to drop or delete one or more columns
```

```
# df.head(2)
```

```
df.sort_index()      # sort the data
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
import pandas as pd
df = df.rename(columns={"cough":"updated_cough"})
```

```
df
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
import pandas as pd
df.head(3)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4

```
df.rename(columns={"updated_city":"city"})
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
df.head(2)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2

```
df.iloc[2:5 ,0:4:2]
```

	emp_id	department
2	103	tyu
3	104	yio
4	105	sdf

```
df.head(2)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2

```
df['Age']=df['age'].add(10)
```

```
df.head()
```

```
df['fever']=df['fever'].sub(100)
```

```
df.head(2)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2

```
df=df.drop(columns=['Age'])
```

```
df.head()
```

```
df[['age','fever']]      # using this method ,we can filter data
```

```
import pandas as pd
```

```
df.head(3)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4

```
df.tail(3)
```

	emp_id	emp_name	department	salary	working hours
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
df.sample(3)
```

	emp_id	emp_name	department	salary	working hours
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
5	106	keshav	hjkl	34567	8

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   emp_id         6 non-null      object
 1   emp_name       6 non-null      object
 2   department     6 non-null      object
 3   salary         6 non-null      object
 4   working hours  6 non-null      object
dtypes: object(5)
memory usage: 372.0+ bytes
```

```
df.describe()
```

	emp_id	emp_name	department	salary	working hours
count	6	6	6	6	6
unique	6	6	6	4	5
top	101	divya	abc	34567	8
freq	1	1	1	3	2

```
df.loc[5:10,['gender','age']]
```

	gender	age
5	NaN	55
6	NaN	55
7	NaN	55
8	NaN	55
9	NaN	55
10	NaN	75

```
import pandas as pd

s=df.axes
s
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-1070355205.py in <cell line: 0>()
      1 import pandas as pd
      2
----> 3 s=df.axes
      4 s

NameError: name 'df' is not defined
```

```
p=df.dtypes
p
```

	0
emp_id	object
emp_name	object
department	object
salary	object
working hours	object

dtype: object

```
b=df.empty
b                        ### it return true if there any missing data in dataframe ot
```

```
False
```

```
df.isnull().sum      ### for check missing values.
```

```
pandas.core.frame.DataFrame.sum
def sum(axis: Axis | None=0, skipna: bool=True, numeric_only: bool=False,
min_count: int=0, **kwargs)
```

```
Return the sum of the values over the requested axis.

This is equivalent to the method ``numpy.sum``.

Parameters
----------
```

```
df=df.dropna()
df                          # for remove missing row
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
i = df.ndim
i                          # number of axes(it is 2)
```

```
2
```

```
t=df.shape
t
df.shape[0]
df.shape [1]                      #shape=n(rows),n(column)
                                  # total row
                                  # total column
```

```
5
```

```
d=df.size                       # row count* column count
d
```

```
30
```

```
a=df.values                   #get a numpy array for df
a
```

```
array([['101', 'divya', 'abc', '1000', '1'],
       ['102', 'chiya', 'dfg', '6789', '2'],
       ['103', 'sourabh', 'tyu', '34546', '4'],
       ['104', 'anil', 'yio', '34567', '6'],
       ['105', 'vartika', 'sdf', '34567', '8'],
       ['106', 'keshav', 'hjkl', '34567', '8']], dtype=object)
```

```
df=df.copy()
df
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
p=df.sort_values(by='resume_score')
p
```

```
r=df.sort_index()
r
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
df['salary']=df['salary'].astype(int)
df
```

```
df.head(3)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4

Double-click (or enter) to edit

```
df['salary']=df['salary'].astype(str)
```

```
df.dtypes
```

```
t=df.add(4)
t
```

```
df['cgpa']=df['cgpa'].add(4)
df
```

```
s=df.count()
s
```

	0
emp_id	6
emp_name	6
department	6
salary	6
working hours	6

dtype: int64

```
p=df.max()
p
```

	0
emp_id	106
emp_name	vartika
department	yio
salary	6789
working hours	8

dtype: object

```
q=df.min()
q
```

	0
emp_id	101
emp_name	anil
department	abc
salary	1000
working hours	1

dtype: object

```
g=df.filter(items=['cgpa','placed'])
g
df[['cgpa','placed']]                  # second option of filtering
```

```
s=df.filter(items=[5,6],axis=0)     #axis=0 means row wise operation
s
```

	emp_id	emp_name	department	salary	working hours
5	106	keshav	hjkl	34567	8

```
t=df.filter(like='5',axis=0)
t
```

	emp_id	emp_name	department	salary	working hours
5	106	keshav	hjkl	34567	8

```
p=df.rename(columns={'cgpa':'updated_cgpa','resume_score':'semester_marks'})  #how to
p
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
1	102	chiya	dfg	6789	2
2	103	sourabh	tyu	34546	4
3	104	anil	yio	34567	6
4	105	vartika	sdf	34567	8
5	106	keshav	hjkl	34567	8

```
import pandas as pd
df['half']=df['cgpa'].where(df['cgpa']>7,other=0)
df.head(10)
```

```
df.groupby('has_covid').size()
```

```
#EDA==> exploratory data analysis
```

```
#parts of eda
#(1).univariate analysis==> analysis on a single independent column
#(2).bivariate analysis==> analysis on two columns
# (3).multivariate analysis==> analysis on more than two columns
```

```
# DATA types
# (1).numerical data ==> continuous data ==> age(year,month,days),height,
#(2). categorical data==> descrete data==> total number of employees,marital
```

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt                    # VISUALIZATION LIBARY
import seaborn as sns                             # MATPLOTLIB'S UPDATED VERSION IS SEA
```

Double-click (or enter) to edit

```
df=pd.read_csv('/content/Attrition - Attrition1.csv')
df.head(2)
```

```
df.head()
```

	age	gender	fever	updated_cough	city	has_covid
0	60	NaN	103.0	NaN	Kolkata	No
1	27	NaN	100.0	NaN	Delhi	Yes
2	42	NaN	101.0	NaN	Delhi	No
3	31	NaN	98.0	NaN	Kolkata	No
4	65	NaN	101.0	NaN	Mumbai	No

```
# (1).univariate analysis
```

```
df.columns
```

```
Index(['emp_id', 'emp_name', 'department', 'salary', 'working hours'], dtype='object')
```

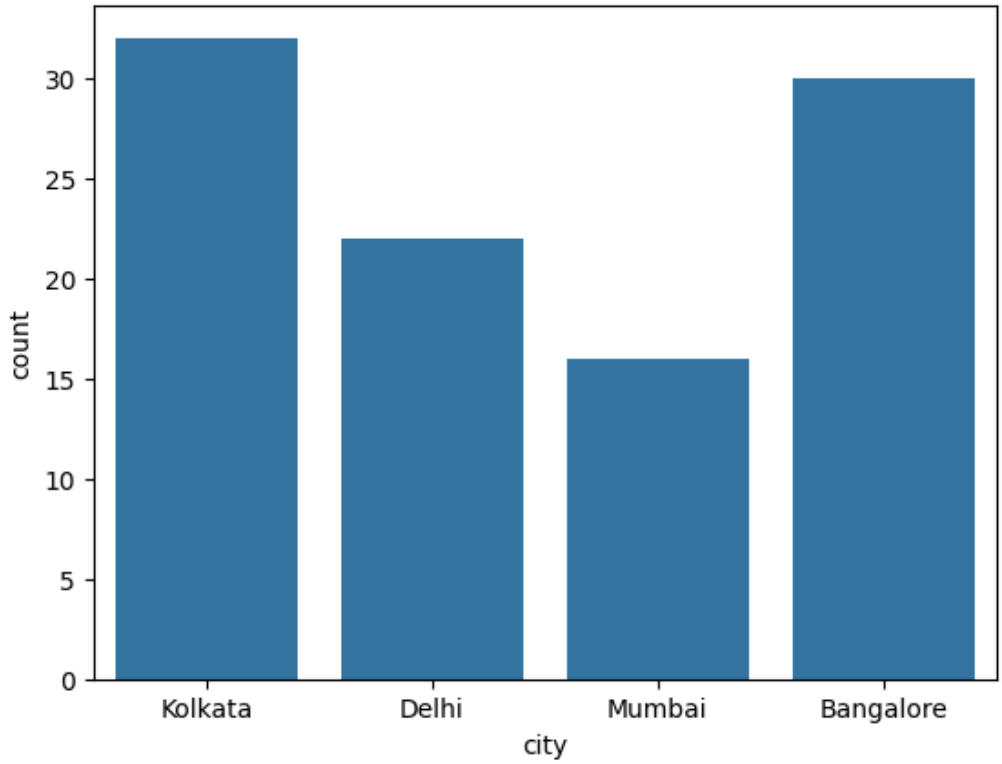
```
df['city'].value_counts()
```

city	count
Kolkata	32
Bangalore	30
Delhi	22
Mumbai	16

dtype: int64

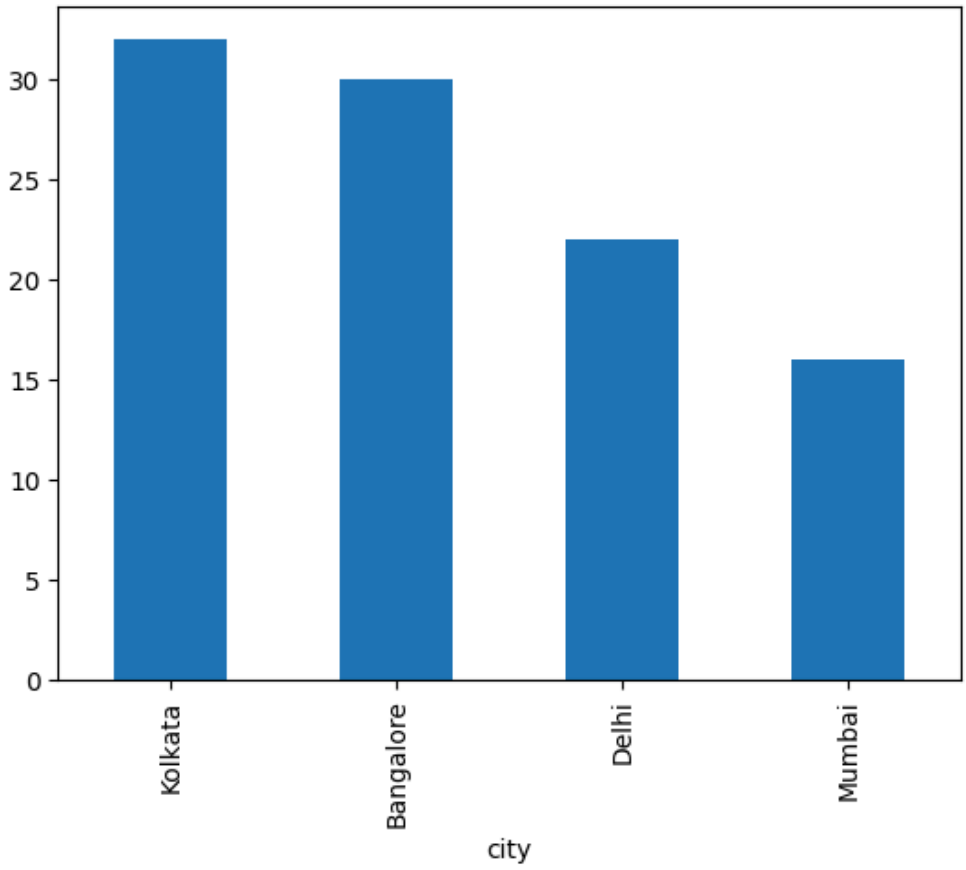
```
sns.countplot(x=df['city'])
```

```
<Axes: xlabel='city', ylabel='count'>
```



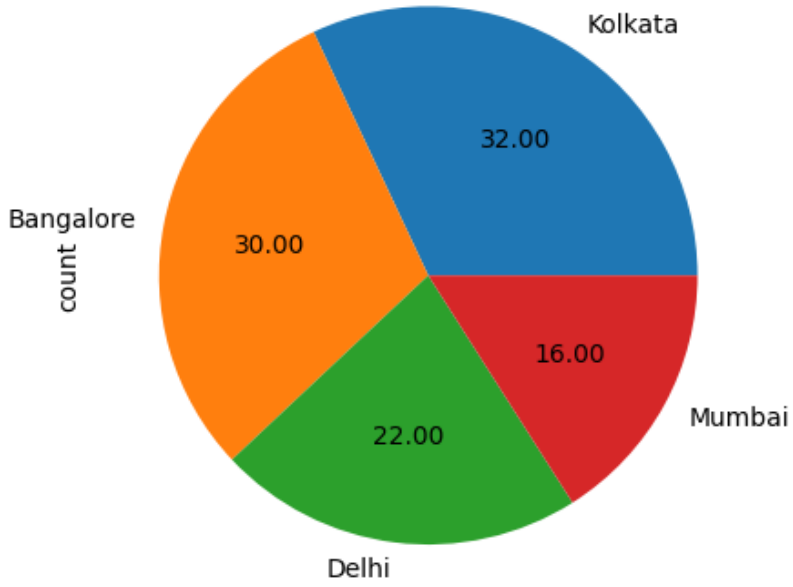

```
df['city'].value_counts().plot(kind='bar')
```

```
<Axes: xlabel='city'>
```



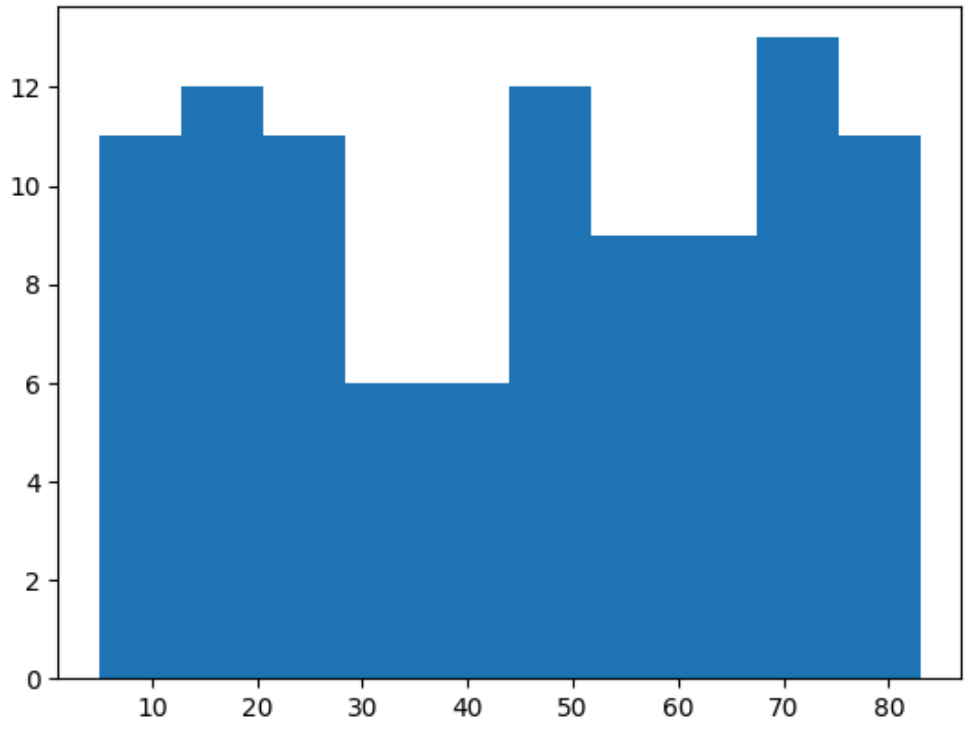

```
df['city'].value_counts().plot(kind='pie',autopct='%.2f')
```

```
<Axes: ylabel='count'>
```



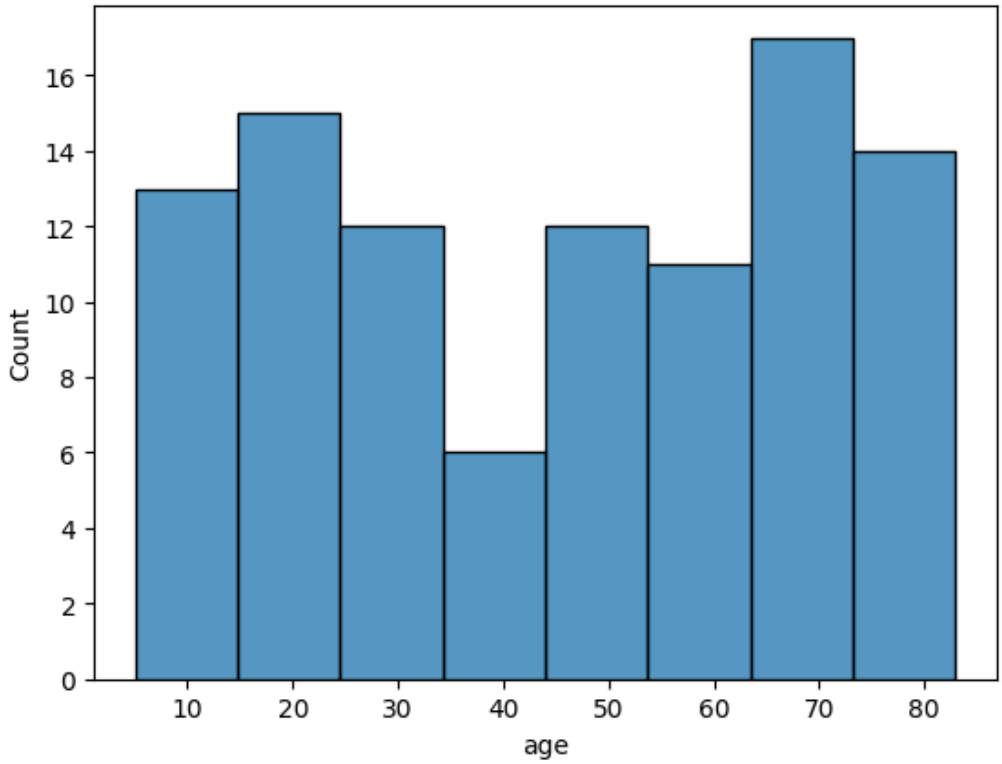

```
# if we have numerical data then we use histogram because it finds the distribution
```

```
plt.hist(x=df['age'])
plt.show()
```




```
sns.histplot(x=df['age'])
```

```
<Axes: xlabel='age', ylabel='Count'>
```

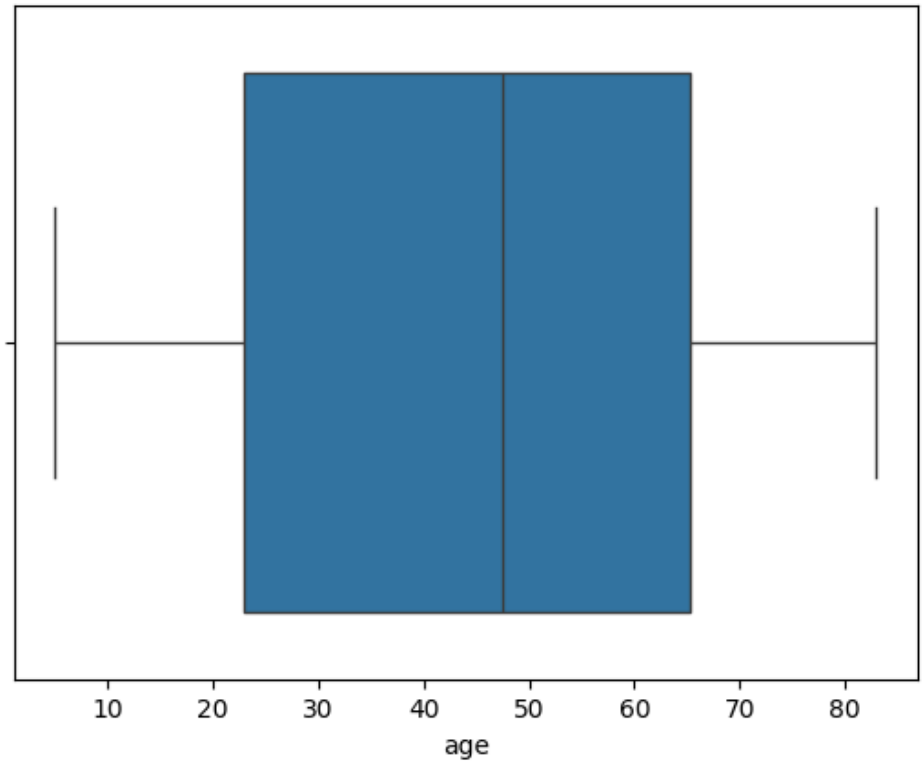



```
# 3. boxplot
# for find our outliers
# 1.lower fence
# 2. 25% data
# 3. iqr(inter quartile range)(75%-25%)
# 4.75% data
# upper fence
```

```
# x=1,2,3,4,5
# mean = (1+2+3+4+5)/5=15/5=3     # hero 100 is outlier.
# x=1,2,3,4,5,100
# mean = 15+100/6=115/6=19.6
```

```
sns.boxplot(x=df['age'])
```

```
<Axes: xlabel='age'>
```




```
# app reviews sentiment analysis
```

```
import numpy as np
import pandas as pd
```

```
df=pd.read_csv(' /content/Untitled form (Responses) - Form responses 1 (2).d.csv')
df.head(2)
```

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            100 non-null    int64
 1   gender         0 non-null      float64
 2   fever          92 non-null     float64
 3   updated_cough  0 non-null      float64
 4   city           100 non-null    object
 5   has_covid      100 non-null    object
dtypes: float64(3), int64(1), object(2)
memory usage: 4.8+ KB
```

```
# exploratory data analysis
```

```
# plotting the distribution of ratingS
```

```
sns.set(style='whitegrid')
plt.figure(figsize=(9,5))
sns.countplot(data=df,x='Rating')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```

```
# adding sentiment labels in the data
```

```
from textblob import TextBlob
```

```
def textblob_sentiment_analysis(reviews):
  sentiment=TextBlob(reviews).sentiment.polarity
  if sentiment.polarity> 0.1:
    return 'positive'
  elif sentiment.polarity < -0.1:
    return 'negative'
  else:
    return 'neutral'
```

```
df['sentiment']=df['reviews'].apply(textblob_sentiment_analysis)
```

```
df.sample(5)
```

	emp_id	emp_name	department	salary	working hours
0	101	divya	abc	1000	1
5	106	keshav	hjkl	34567	8
4	105	vartika	sdf	34567	8
3	104	anil	yio	34567	6
1	102	chiya	dfg	6789	2

```
# analyzing app reviews sentiments
```

```
sentiment_distribution=df['sentiment'].value_counts()
sentiment_distribution
```

```
plt.figure(figsize=(9,5))
sns.barplot(x=sentiment_distribution.index,
            y=sentiment_distribution.values)
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

```
plt.figure(figsize=(10,5))
sns.countplot(data=df,
```

```
                x='rating',
                hue='sentiment')
plt.xlabel('rating')
plt.ylabel('Count')
plt.legend('title=sentiment')
plt.show()
```

```
# summary
# app reviews sentiment analysis is a valuable tool for app developers and business t
# user feedback ,prioritize feature updates and maintain a positive user community. i
# analysis techniques to determine whether the sentiments in these reviews are positi
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
tips=sns.load_dataset('tips')
tips
```

```
# bivariate analysis
```

```
# 1.scatterplot(numerical column-numerical column)
```

```
sns.scatterplot(x='total_bill',
                y='tip',data=tips)
```

```
sns.scatterplot(x='size',
                y='tip',data=tips,hue=tips['sex'])
```

```
sns.scatterplot(x='total_bill',
                y='tip',data=tips,hue=tips['sex'],style=tips['smoker'])
```

```
# heatmap(categorical-categorical)
p=pd.crosstab(tips['day'],tips['time'])
p
```

```
sns.heatmap(p)
```

```
# polymorphism= bahuT SARE FROM
# TWO TYPES =OVERRIDING,OVERLOADING


class a:
    def show(self):
        print("hello a")
class b(a):
    def show(self):
        print("hello b")
class c(b):
    def show(self):
        print("hello c")
class d(c):
```

```
class d(c):
    def show (self):
        print("hello d")
class d(b):
    pass
obj=d()
for obj in [a(),b(),c()]:
    obj.show()
```

```
hello a
hello b
hello c
```

```
class father:
  def manners(self):
    return"has good manners"

class mother:
  def bargaining(self):
    return"good bargaining"

class child:
  def gaming(self):
    return"good gaming"

c = child()
c.gaming()
```

```
'good gaming'
```

```
class teacher:
  def __init__(self,subject,expression):
    self.subject=subject
    self.expression=expression

class singer:
  def __init__(self,genre,song):
    self.genre=genre
    self.song=song

class person(teacher,singer):
  def __init__(self,name,age,genre,song,subject,expression):
    teacher.__init__(self,subject,expression)
    singer.__init__(self,genre,song)
    self.name=name
    self.age=age
p=person("rahul","23","classical","soaf","maths","12")
```

```
class grandfather:
  def message(self):
    print("hello divya")

class father(grandfather):
  def message_from_father(self):
    print("hello father")
```

```
class child(father):
  def message_from_child(self):
    print("hello child")

c = child()

c.message_from_child()
c.message_from_father()
c.message()
```

```
hello child
hello father
hello divya
```

```
class grandmother:
  def message(self):
    print("hello nani")

class mother(grandmother):
  def message_from_mother(self):
    print("hello mummy")

class child(mother):
  def message_from_child(self):
    print("hello divya")

c = child()

c.message_from_child()
```

```
hello divya
```

```
class animal:
  def sound(self):
    return "animal makes sound"
class cat(animal):
  pass
cat().sound()
```

```
'animal makes sound'
```

```
class person:
  def __init__(self,name,id,salary):
    self.name=name
    self.id=id
    self.salary=salary

  def display(self):
    print(f"Name: {self.name}, ID: {self.id}, Salary: {self.salary}")

class student(person):
  def __init__(self,name,id,salary,marks):
    super().__init__(name,id,salary)
    self.marks=marks
```

```
  def display(self):
    super().display()
    print(f"Marks: {self.marks}")


class teacher(person):
  def __init__(self,name,id,salary,department):
    super().__init__(name,id,salary)
    self.department=department

  def display(self):
    super().display()
    print(f"Department: {self.department}")


student_instance = student("divya","3456","22346","98")
teacher_instance = teacher("tushar","2345","234567","cse")

student_instance.display()
teacher_instance.display()
```

```
Name: divya, ID: 3456, Salary: 22346
Marks: 98
Name: tushar, ID: 2345, Salary: 234567
Department: cse
```

```
class animal:
  def __init__(self,name,color,age):
    self.name=name
    self.color=color
    self.age=age

  def display(self):
    print(f"name:{self.name},color:{self.color},age:{self.age}")

class dog(animal):
  def __init__(self,name,color,age,breed):
    super().__init__(name,color,age)
    self.breed=breed

  def display(self):
    super().display()
    print(f"breed:{self.breed}")

class cat(animal):
  def __init__(self,name,color,age,breed,eyes):
    super().__init__(name,color,age)
    self.breed=breed
    self.eyes=eyes

  def display(self):
    super().display()
    print(f"breed:{self.breed},eyes:{self.eyes}")

class cow(animal):
  def __init__(self,name,color,age,breed,eyes,voice):
    super().__init__(name,color,age)

    self.eyes=eyes
    self.voice=voice
```

```
animal1 = animal("sadsf","black","23")
dog2=dog("sasfdgh","color","w23","frthgjh")
animal1.display()
dog2.display()
```

```
name:sadsf,color:black,age:23
name:sasfdgh,color:color,age:w23
breed:frthgjh
```

```
class student:
  college = "g.e.c.b"

  def __init__(self,name):
    self.name=name


s1 = student("divya")
s1.college = 'ssdfdg'
```

```
class person:
  def __init__(self,name,location,salary):
    self.name=name
    self.location=location
    self.salary=salary

class student(person):
  def __init__(self,name,location,salary,id, marks):
    super().__init__(name,location,salary)
    self.id=id
    self.marks=marks

student1 = student("rahul","jaipur","34000","dsfgn","34")

student1.location
```

```
'jaipur'
```

```
class person:
  def __init__(self,name,gender,id):
    self.name=name
    self.gender=gender
    self.id=id

class student(person):
  def __init__(self,name,gender,id,marks,location):
    super().__init__(name,gender,id)
    self.location=location
    self.marks=marks

class teacher(person):
  def __init__(self,name,gender,id,department):
    super().__init__(name,gender,id)
    self.department=department

student_instance = student("divya","female","345657","78","jaipur")
teacher_instance = teacher("mohit","male","89","cse")
```

```
teacher_instance.name
student_instance.location
```

```
'jaipur'
```

```
class divya:
  def __init__(self,name,location,branch):
    self.name=name
    self.location=location
    self.branch=branch
divya1=divya("sdsdfv","sdfvfggh","sfdbghmh")
rahul=divya("divya","jaipur","cse")
print(rahul.name)
```

```
divya
```

```
class student:
  def __init__(self, name, location, branch):
    self.name = name
    self.location = location
    self.branch = branch

  def display_info(self):
    print(f"name:{self.name}")
    print(f"location:{self.location}")
    print(f"branch:{self.branch}")

student1=student("divya","jaipur","cse")
student2=student("himanshu","dholpur","ba")
student2.display_info()
```

```
name:himanshu
location:dholpur
branch:ba
```

```
class home:
  def __init__(self,name,color,cloths,sleeper,bike):
     self.name=name
     self.color=color
     self.cloths=cloths
     self.sleeper=sleeper
     self.bike=bike

  def display_info(self):
   print(f"name:{self.name},color:{self.color},sleeper:{self.sleeper},bike:{self.bike

home1 = home("rakesh","pink","pant","black","yamaha")
home1.display_info()
```

```
name:rakesh,color:pink,sleeper:black,bike:yamaha
```

```

```

```
# inheritance
class person:
  def __init__(self,name,branch,location):
    self.name=name
```

```
    self.branch=branch
    self.location=location
class student(person):
  def __init__(self,name,marks,id,gender):
    super().__init__(name,id,gender)
    self.marks=marks
student=student("rahul",10,10,"male")
student.marks
```

```
10
```

```
class person:
  def __init__(self,name,branch,gender,id):
    self.name=name
    self.id=id
    self.gender=gender
    self.branch=branch
  def display(self):
    print("name:{self.name}","id:{self.id}","gender:{self.gender}","branch:{self.gender
# write a function to show allthe details of each person (either that person or student
# inheriting all the attributes from the parent class to the child class
class student(person):
  def __init__(self,name,gender,branch,id):
    super().__init__(name,salary,department)
    self.salary=salary
    self.department=department
student=student("divya","male","cse",102)
teacher=teacher("rahul","male",123,"cse",200000)
```

```
# oops
# school management system.
first="divya"
branch="cse"
location="jaipur"

def display(first,branch,location):
  print(first,branch,location)
display(first,branch,location)
```

```
divya cse jaipur
```

```
class student:
  def __init__(self,name,branch,location):
     self.name=name
     self.branch=branch
     self.location=location
student1 = student("rahul","jaipur","cse")
student2=student("divya","jaipur","cse")
print(student2.name)
```

```
divya
```

```
class car:
  def __init__(self,name,company,engine,sunroof,color):
     self.name=name
     self.company=company
     self.engine=engine
```

```
      self.sunroof=sunroof
      self.color=color
      def display_info(self):
       print(f"name:{self.name}")
       print(f"company:{self.company}")
       print(f"engine:{self.engine}")
       print(f"sunroof:{self.sunroof}")
       print(f"color:{self.color}")
car1=car("verna","hyundai","petrol","yes","white")
print(car1.name)
```

```
divya
```

```
class student:
  college="jcerc"
  def __init__(self,name):
    self.name=name
s1=student("divya")
s1.college="sasdfgghhc"
print(s1.c)
```

```
divya
```

```
# object-----real world entity.
# class-----class is basically the blue print of the object(template of object).

class student:
  def __init__(self,name,location,branch):
    self.name=name
    self.location=location
    self.branch=branch
student1=student("rahul","jaipur","ece")
print(student1.name)
```

```
rahul
```

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

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

```
class student:
  def __init__(self,name,location,branch):
    self.name=name
    self.location=location
    self.branch=branch
  def display_info(self):
   print(f"name:{self}")
   print(f"name:{self.name}")
   print(f"location:{self.location}")
   print(f"branch:{self.branch}")
student1=student("rahul","jaipur","cse",)
print(student1.name)
```

```
rahul
```

```
class Car:
    def __init__(self, wheels, color, brand, engine, sunroof, model, bulletproof):
        self.wheels = wheels
        self.color = color
        self.brand = brand
        self.engine = engine
        self.sunroof = sunroof
        self.model = model
        self.bulletproof=bulletproof

    def display_info(self):
        print(f"Wheels: {self.wheels}")
        print(f"Color: {self.color}")
        print(f"Brand: {self.brand}")
        print(f"Engine: {self.engine}")
        print(f"Sunroof: {self.sunroof}")
        print(f"Model: {self.model}")
        print(f"Bulletproof: {self.bulletproof}")
car = Car("4","red","petrol","yes","tata","yes",True)
car.display_info()
```

```
Wheels: 4
Color: red
Brand: petrol
Engine: yes
Sunroof: tata
Model: yes
Bulletproof: True
```

```
class house:
  def __init__(self,name,color,cloths,sleeper):
    self.name=name
    self.color=color
    self.cloths=cloths
    self.sleeper=sleeper
    def display_info(self):
     print(f"name:{self.name}")
     print(f"color:{self.color}")
     print(f"cloths:{self.cloths}")
     print(f"sleeper:{self.sleeper}")
  house = house("charchil","white","pent","chapal")
  house.display_info()
```

```
# college management system
class student:
    # class level attribute
    college="xyz university"

    # intance attribute
    def __init__(self,name,course):
        course=course
        self.name=name
s1=student("divya")
s1.college="xyz universitry"
```

```
for i in range(5,51,5):
  if ( i%5==0) :
```

```
    print([i],end=" ")
```

```
[5] [10] [15] [20] [25] [30] [35] [40] [45] [50]
```

```
sum=0
for i in range(1,11):
  sum+=i
  print(i,end=" ")
print(sum)
```

```
1 2 3 4 5 6 7 8 9 10 55
```

```
for i in range(10,1,-1):
  print(i)
```

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

```
# find the largest from the given list[4,3,21,10]
a=[4,3,21,10]
max=0
for i in a:
  if i>max:
    max=i
print(max)
```

```
21
```

```
# find the second largest number from the given list
a=[4,3,21,10]
max=0
for i in a:
  if i>max:
    max=i
a.remove(max)
max1=0
for i in a:
  if i>max1:
    max1=i
print(max1)
```

```
21
```

```
# intersections of 2 list.
a=[1,2,3,4,5]
b=[3,4,5,6,7]
c=[]
for i in a:
  if i in b:
    c.append(i)
```

```
print(c)
```

```
[3, 4, 5]
```

```
# rotate list
a=[1,2,3,4,5]
n = 4
for i in range(n):
    last = a[-1]
    for i in range(len(a)-1, 0, -1):
        a[i] = a[i -1]
    a[0] = last
print(a)
```

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

```
a=int(input("enter the number"))
if a>=1000:
  print("withdrawal")
else:
  print("not")
```

```
a=int(input("enter the num"))
if a>=50000:
  print("high salary")
else:
  print("normal salary ")
```

```
enter the num51000
high salary
```

```
a=(input("enter the password"))
if a=="admin123":
  print("login successful")
else:
  print("not")
```

```
enter the passwordadmin123
login successful
```

```
li=[1,2,3,4]
left=0
right=len(li)-1
while left < right:
  li[left],li[right]=li[right],li[left]
  left+=1
  right-=1


print(li)
```

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

```
a=int(input("enter first num"))
b=int(input("enter second num"))
c=int(input("enter third num"))
if a>b and a>c:
  print("a")
elif b>c and b>a:
  print("b")
elif c>a and c>b:
  print(c)
else:
  print(d)
```

```
enter first num45
enter second num67
enter third num89
89
```

```
a=int(input("enter the temperture"))
if a>=40:
  print("v")
elif a>=25:
  print("hot")
elif a>=10:
  print("normal")
else:
  print("cold")
```

```
enter the temperture11
normal
```

```
balance = 10000
withdraw = int(input("Enter amount to withdraw: "))

if withdraw <= balance:
    balance -= withdraw
    print("Withdrawal successful. Remaining balance:", balance)
else:
    print("Insufficient Balance")
```

```
Enter amount to withdraw: 10001
Insufficient Balance
```

```
a=int(input("enter the num"))
if a<=78:
  print("positive")
else:
  print("nagitive")
```

```
enter the num56
positive
```

```
a=int(input("enter the marks"))
if a>=60:
  print("a")
elif a>=33:
  print("pass")
else:
  print("fail")
```

```
enter the marks55
pass
```

```
if age==18:
  print("divya")
else:
  print("10")
```

```
divya
```

```
i=20
if i>20:
  print("positive")
elif i<20:
       print("nagitive")
else:
  print("zero")
```

```
zero
```

```
i=int(input("enter the age"))
if i<=18:
  print("not")
else:
  print("eligible")
```

```
enter the age41
eligible
```

```
a=int(input("enter the marks"))
if a>=90:
  print("a")
elif a>=75:
  print("b")
elif a>=50:
  print("c")
else:
  print("fail")
```

```
enter the marks80
b
```

```
ch = input("Enter a character: ")

if ch.lower() in ['a', 'e', 'i', 'o', 'u']:
    print("Vowel")
else:
    print("Consonant")
```

```
Enter a character: ai
Consonant
```

```
i=int(input("enter the name"))
if i%2==0:
  print("even")
```

```
else:
  print("odd")
```

```
enter the name5
odd
```

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

if num > 0:
    print("Positive number")
elif num < 0:
    print("Negative number")
else:
    print("Zero")
```

```
Enter a number: 56
Positive number
```

```
a=10
b=2
print(a/b)
```

```
5.0
```

```
def age_validate(age):
  if age>=18:
    print("elible")
  else:
    print("not elible")
age_validate(89)
```

```
elible
```

```
def age_validate(age):
  if age<0:
    raise exception("please enter the age which is valid")
  else:
    print("age is validated")
try:
  age_validate(18)
except exception as e:
   print("you've got this exception",e)
```

```
age is validated
```

```
# bank management.
balance=100000
def withdraw(money):
   global balance
 if money > balance:
   raise exception("paisa itna hai hi nahin tumhare account main")
 else:
   balance =balance-money
   print("paisa nikala gya")
   print("bacha hua balance",balance)
try:
  withdraw(20000000)
```

```
except exception as e:
   print("kuch exception aya hai",e)
```

```
  File "<tokenize>", line 5
    if money > balance:
    ^
IndentationError: unindent does not match any outer indentation level
```

```
# index error
li=[1,2,34,56]
try:
  element=(li[5])
except IndexError:
  print("please enter the valid index")
```

```
please enter the valid index
```

```
# key error
d={"a":1,"b":2}
try:
  print(d[a])
except KeyError:
  print("please enter the valid key")
```

```
please enter the valid key
```

```
# seek and tell function into file handling.
with open('sample.txt','r') as f:
  # reading the file line of the file
  # print (f.readline())

  # finding
```

```
  File "/tmp/ipython-input-3983086469.py", line 6
    # finding
             ^
SyntaxError: incomplete input
```

```
# writelines
# take big files

big_data=['hello_world' for i in range(2000)] # Fixed syntax: removed \n and extra qu
with open('new.txt','w')as f:
          f.writelines(big_data)
```

```
# readling the big files into the small chunks.
with open('new.txt','r')as f:
   chunk_size=100
   while True: # Fixed 'true' to 'True'
    chunk=f.read(chunk_size)
    if not chunk:
```

```
    break
  print("chunk->")
  print(chunk)
```

```
# r+ mode in file handling
# this is used for the reading and writing the file but it works for the existing file
# w
with open('sample.txt''r+')as f:
  f .write('anything')
  print(f.tell())
  f.seek(0)
  print(f.read())
```

```
# w+ mode in file handling
# this mode is used for the handling and writing the file and also create the new
# file if it doesn't exits
with open('sample.txt','w+')as f:
  f.write('anything')
  f.seek(0)
  print(f.read())
```

```
anything
```

```
# write a program to create a file named hello.txt and write "hello,python!"into it

with open('hello.txt','w') as f:
f.write('hello python')
print(f.read())
```

```
# write a program to read the content of hello.txt and print it.
with open ('hello.txt','r') as f:
  print(f.read())
```

```
# write a program to append a new line "welcome to python" to hello.txt.
with open('hello.txt','a')as f:
  f.write("welcome to python")
  print(f.read())
```

```
# write a program to read the first 10 character of a file.
with open('hello.txt','r') as f:
  print(f.read(10))
```

```
welcome to
```

```
# write a program to read A FILE LINE BY LINE AND PRINT EACH LINE.
with open('hello.txt','r') as f:
  for line in f:
    print(line)
```

```
welcome to python
```

```
# write a program to copy the content of hello.txt to another file called copy.txt.
with open('hello.txt','r') as f:
  content=f.read()
```

```
with open('copy.txt','w') as f:
  f.write(content)
```

```
#
```

```
# reading csv file using python
import csv
with open('student_marks.csv','r') as f:
  content=csv.reader(f)
  for linr in content:
    print(line)
```

```
from _typeshed import SupportsRichComparison
file handling.
(read)----python Script(os)---(read)---

#python program(open,read,write)---->buffer ram----->operating system(syster call,file
```

```
# recursion=fuction khud ko call krta hai.
```

```
def is_prime(n):
    for i in range(2,n):
        if n % i == 0:
            return False
    return True
for num in range(10, 101):
    if is_prime(num):
        print(num, end=" ")
is_prime(i)
```

```
li=[x for x in range(10)]
s=lambda x:x**2
list(map(s,li))
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
filetered=list(lambda x:x%2==0,li)
print(filetered)
```

```
def dost (paisa):
  return "saman"
dost(19)
```

```
'saman'
```

```
def dost(paisa):
  if paisa==20:
    return "chips"
  elif paisa==50:
    return "colddrink"
  else:
    return "bhai paisa kam hai"
(dost(10))
```

```
'bhai paisa kam hai'
```

```
def display (name,age ,location):
  print(f"my name is{name}.my age is{age}.i live in {location}")
display("divya",20,"dholpur")
```

```
my name isdivya.my age is20.i live in dholpur
```

```
# default arguments
def greeting(name="divya"):
  print(f"hello{name}")
greeting()
```

```
hellodivya
```

```
# keyword argument : isme keyword or value sath m hoti hai
# position important nhi hoti esme.
def studentsdetails(name,location,course):
  print(f"student name:{name},course enrolled:{course},location:{location}")
studentsdetails(name="divya",location="dholpur",course="python")
```

```
student name:divya,course enrolled:python,location:dholpur
```

```
# VARIABLE LENGTH POSITIONAL ARGUMENTS
def add(*args):
  return sum(args)
print(add(1,2,3,4,5,6,7,8,9))
```

```
45
```

```
# variable length keyword argument.
def student_details(**kwargs):
  print(kwargs)
student_details(name="mohan",sub="python",location="jaipur")
```

```
{'name': 'mohan', 'sub': 'python', 'location': 'jaipur'}
```

```
# function
```

```
n=4
for i in range(1,n+1):
  print("*" * (2*i-1))
  for j in range(n,0,-1):
    print("*" * (2*i -1))
```

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

```
n = int(input("Enter the number of prime numbers: "))
count = 0

for num in range(2, 1000):
    is_prime = True
    for i in range(2, num):
        if num % i == 0:
            is_prime = False
            break
    if is_prime:
        print(num, end=" ")
        count += 1
    if count == n:
      break
```

```
Enter the number of prime numbers: 500
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109
```

```
# "write a python program using for loops to print all prafect numbers between 1 and
# (a perfect number=sum of its proper divisor =number itself ,e.g.28=1+2+4+7+14)"
for num in range(1, 1001):
    sum_divisors = 0
    for i in range(1, num):
        if num % i == 0:
            sum_divisors += i
    if sum_divisors == num:
      print(num)
```

```
6
28
496
```

```
Start coding or generate with AI.
```

```
# WRITE A FOR LOOP TO GENERATE THE SEQUENCE: 3,9,27,81,243
a=3
for i in range(5):
```

```
 print(a,end=" ")
 a*=3
```

```
3 9 27 81 243
```

```
# write a for loop that prints all prime numbers b/w 60 and 90.
for num in range(60, 91):
    if num > 1:
        for i in range(2, num):
            if num % i == 0:
                break
        else:
           print(num)
```

```
61
67
71
73
79
83
89
```

```
# write will be the output ? for i in range(1,5):
for j in range(1,5):
  print(j,end=" ")
print()
```

```
1 2 3 4
```

```
# write a program using a for loop to count how many digits in tha number
#9876543210 are odd.
num=9876543210
count=0
for d in str(num):
  if int(d)%2!=0:
    count+=1
print(count)
```

```
5
```

```
# rewrite this into a compact for loop:print(5).
for i in range(5):
  print(5**i)
```

```
1
5
25
125
625
```

```
# write a for loop to print all three-digit numbers divisible by 13
for i in range(100,1000):
```

```
  if i%13==0:
    print(i,end=" ")
```

```
104 117 130 143 156 169 182 195 208 221 234 247 260 273 286 299 312 325 338 351 364 37
```

```
# trace the output:
for i in range(1,4):
  for j in range(1,i+2):
    print(i*j,end=" ")
  print()
```

```
1 2
2 4 6
3 6 9 12
```

```
# section B.
```

```
# write a python program using a for loop to check whether a number is an armstrong
a=int(input("enter the number"))
sum=0
temp=num
while temp>0:
  digit=temp%10
  sum+=digit**3
  temp//=10
if a==sum:
  print(a,"is a armstrong number")
else:
  print(a,"is not a armstrong number")
```

```
enter the number153
```

```
# write a for loop program to generate this pattern.
a=1
for i in range(1,6):
  for j in range(i):
    print(a,end=" ")
    a=a+1
  print()
```

```
1
23
456
78910
1112131415
```

```
s=0
for i in range(1,6):
  s+=i if i%2==0 else-i
  print(s)
```

```
-1
1
-2
2
-3
```

```
# write the output of : for i in range(4):
for i in range(4):
  print(i,end=" ")
print()
```

```
0 1 2 3
```

```
# reverse the string "PYTHONIC"using a for loop (without slicing)
a="PYTHONIC"
b=""
for i in a:
  b=i+b
  print(b)
```

```
P
YP
TYP
HTYP
OHTYP
NOHTYP
INOHTYP
CINOHTYP
```

```
Start coding or generate with AI.
```

```
a=56
b=45
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
print(a**b)
print(a//b)
```

```
101
11
2520
1.2444444444444445
11
46608080274105061329009150889974173059443551178471443074145110421699840190709761
```

```
print(" hey i am a /"good girls/"/n and this viewer is also a good girls")
```

```
  File "/tmp/ipython-input-1141371553.py", line 1
    print(" hey i am a /"good girls/"/n and this viewer is also a good girls")
          ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

```
a="i love phython"
b=a[::-1]
b
```

```
'nohtyhp evol i'
```

```
a={divya:"1"}
b= dict(for k,v in k,v a.item())
b
```

```
  File "/tmp/ipython-input-606112692.py", line 2
    b= dict(for k,v in k,v a.item())
            ^
SyntaxError: invalid syntax
```

Start coding or generate with AI.

```
a={"a":1,"b":2,"c":3}
b={v:k for k,v in a.items()}
b
```

```
{1: 'a', 2: 'b', 3: 'c'}
```

```
a={"1":4565778,"2":45,"3":56766}
b=list(a.items())
b
```

```
[('1', 4565778), ('2', 45), ('3', 56766)]
```

```
a=[]
b="aeiou"
for i in b:
  a.append(i)
  print(a)
```

```
['a']
['a', 'e']
['a', 'e', 'i']
['a', 'e', 'i', 'o']
['a', 'e', 'i', 'o', 'u']
```

```
a=int(input("enter the name"))
print(a)
```

```
enter the name45
45
```

Start coding or generate with AI.

```
# create a srting in upper all srting upper vowel is upper

a="my name is divya"
b=a.upper()
b
```

```
'MY NAME IS DIVYA'
```

```
Start coding or generate with AI.
```

```
my_dict = {
    "a": 1,
    "b": 2,
    "c": 3
}

# Values को keys और keys को values बनाना
reversed_dict = {v: k for k, v in my_dict.items()}

print(reversed_dict)
```

```
{1: 'a', 2: 'b', 3: 'c'}
```

```
a="divya"
print(a)
print(type(a))
```

```
divya
<class 'str'>
```

```
for i in range(1,6):
  for j in range(1,5-i,1):
    print(" ",end="")
  for k in range(1,i+1):
    print("*",end=" ")
  print()
```

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

```
for i in range(1,6):
  for j in range(1,i):
      print("/",end=" ")
  for k in range(1,5-i+1):
      print("*",end=" ")
  print()
```

```
* * * *
/ * * *
/ / * *
/ / / *
/ / / /
```

```
for i in range(1,6):
  for k in range(1,5-i,1):
     print("-",end=" ")
  for j in range(1,i+1):
     print("*",end=" ")
  print()
```

```
- - - *
- - * *
- * * *
* * * *
* * * * *
```

```
for i in range(6,0,-1):
  for j in range(1,i+1):
    print("*",end=" ")
  else:
    print(" ",end=" ")
  print( )
```

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

```
for i in range(1,6):
  for k in range(1,i+1):
    if (k==1 or i==5 or i==k):
     print("*",end=" ")
    else:
     print("-",end=" ")
  print()
```

```
*
* *
* - *
* - - *
* * * * *
```

```
for i in range(1,6):
  for j in range(1,i+1):
    print("-",end=" ")
  for k in range(1,5-i+1):
    print("*",end=" ")
  print( )
```

```
- * * * *
- - * * *
- - - * *
- - - - *
- - - - -
```

```
for i in range(1,6):
  for k in range(1,i+1):
    print(k,end=" ")
    x=65
  for k in range(1,5-i+1):
    print(chr(x),end=" ")
    x=x+1
  print( )
```

```
1 A B C D
1 2 A B C
1 2 3 A B
1 2 3 4 A
1 2 3 4 5
```

```
i=20
while(i<=40):
  if(i%3==0):
    i=i+1
    print(i)
```

```
i=98
total=0
while(i>58):
  i=i-1
print(total)
```

```
0
```

```
# FILE OPEN
```

```
# F=OPEN (FILENAME,Mode)
```

```
# 1.*args(non-keyword arguments)
# *args allows you to pass multiple positional arguments to a functions . it collects
```

```
def add_numbers(*args):
  total=sum(args)
  print(f"sum:{total}")
add_numbers(5,6,8)
```

```
sum:19
```

```
# 3.using args and *kwargs together
# you can use both in tha same function.but args must come before *kwargs
def display_info(*args,**kwargs):
  print("posotion arguments:",args)
  print("keyword arguments:",kwargs)
display_info(1,2,3,name="divya",age="20")
```

```
posotion arguments: (1, 2, 3)
keyword arguments: {'name': 'divya', 'age': '20'}
```

```
# 4.required arguments(mandatory inputs)
# there are arguments that must be passed to a function;otherwise:phython throws an e
def green (name):
  print(f"hello ,{name}!")
green("divya")
```

```
hello ,divya!
```

```
# 5. default arguments (optinal inputs)
# these arguments have a dafault value assigned.if no value is provided,phython use t
def green(name="divya"):
  print(f"hello,{name}!")
green()
```

```
hello,divya!
```

```
# 3.return keyword(output of a function)
# the return statement sends a value back from a function.if return is not used , the
def add(a,b):
  return a+b
result=add(5,4)
print(result)
```

```
9
```

```
# q.1 find the maximum number using*args
def find_max(*args):
  return max(args)
print(find_max(16,5,6,156))
print(find_max(-3,-6,-7,-9))
```

```
156
-3
```

```
# 2.calculate THE PRODUCT OF NUMBERS USING*ARGS.
def d(*args):
  result=1
  for num in args:
    result*=num
  return result
print(d(2,3,4))
print(d(4,5))
```

```
24
20
```

```
Start coding or generate with AI.
```

```
#1.lambda()---->
# syntax:lambda condition:expression
```

```
# 2.filter()---->
# filter(logic,data)---->by default output---->object---->type cast list
```

```
# 3. map()---->                #its important very very
# map(logic,data)---->
```

```
# lambda function====< a lambda function can take any number of arguments but can onl
# syntax: lambda arguments: expressions
# ex.1
```

```
x=lambda a:a+10
print(x(5))
```

```
15
```

```
x=lambda a,b:a*b
print(x(4,5))
```

```
20
```

```
x=lambda a,b:a/b
print(x(4,2))
```

```
2.0
```

```
x=lambda a,b,c:a+b*c
print(x(3,4,5))
```

```
23
```

```
# filter functions==> the filter function return as iterator where the items are filt
ages = [34, 4, 5, 67, 2, 34, 5, 77, 4, 55]
def myfun(x):
    if x >= 18:
        return True
    else:
        return False
B= list(filter(myfun, ages))
print(B)
```

```
[34, 67, 34, 77, 55]
```

```
# map function==>
```

```
a=[1,2,4,56,2,68,3]
def d(a):
  return a*a
b=list(map(d,a))
print(b)
```

```
[1, 4, 16, 3136, 4, 4624, 9]
```

```
a=['1','2','4','56']
s=list(map ( int ,a))
print(a)
```

```
['1', '2', '4', '56']
```

```
a=[1,2,3,5,6,7]
b=list(map(lambda x:x%3==0 or x%5==0,a))
b
```

```
[False, False, True, True, True, False]
```

```
a=[1,2,3,5,6,7]
```

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

```
# functions in phython.
# pre-defined functions.
# print(),return(),sum(),input(),sum(),max(),min()
# manually created functions.
# step 1.def
# step 2.name of functions
# step 3. purpose of creation tha function
# step 4. call the functin
```

```
# q. create a function who add 2 numbers?
def my_add():
  a=1
  b=2
  c=(a+b)
  print(c)
my_add()
```

```
3
```

```
 #q. create a function who add 2 numbers?
def my_add(a,b):
  p=int(input("enter the number"))
  q=int(input("enter the second number"))
  c=a+b
  print(c)
my_add(4,6)
```

```
# arguments and parameters
```

```
# q. create a function who add 2 numbers?
def my_add(a,b):
  c=a+b
  print(c)
p=int(input("enter fisrt number"))
q=int(input("enter second number"))
my_add(p,q)
```

```
enter fisrt number45
enter second number45
90
```

```
# q.create a function who add sum of square of a given number.
def my_add(a,b):
  sum=a**2+b**2
  print(sum)
my_add(1,2)
```

```
# q. create a function who given number armstrong or not.
def my_add(a):
  a=int(input("enter the number"))
  sum=0
  temp=a
  while temp>0:
    digit=temp%10
    sum+=digit**3
    temp//=10
  if a==sum:
    print(a,"is a armstrong number")
  else:
    print(a,"is not a armstrong number")
my_add(a)
```

```
enter the number153
153 is a armstrong number
```

```
def divya():
  if a%2==0:
    print("even number")
  else:
    print("odd number")
a=int(input("enter the number"))
divya()
```

```
enter the number2
even number
```

```
def divya():
  if a%3==0 and a%5==0:
    print("divisible by 3 and 5")
  else:
    print("not divisible by 3 and 5")
a=int(input("enter the number"))
divya()
```

```
enter the number45
divisible by 3 and 5
```

```
# q create a function it is prime or not prime.
def divya():
  if a%2==0:
    print("prime number")
  else:
    print("not prime number")
a=int(input("enter the number"))
divya()
```

```
enter the number5
not prime number
```

```
# q. create a function factorial or not factorial.
def factorial(all):
  factorial = 1
  for i in range(1,a+1):
    factorial *= i
```

```
  print(factorial)
a=int(input("enter the number"))
factorial(a)
```

```
def divya(a):
  fact=1
  for i in range(1,a+1)
   fact=fact*i
   print(fact)
divya(a)
```

```
def divya(a):
  b="aeiou"
  for i in a:
    if i not in b:
      a=a.replace
    print(a)
divya("divya bansal")
```

```
def is_palindrome(num):
    s = str(num)
    if s == s[::-1]:
        print(num, "is a Palindrome number")
    else:
        print(num, "is not a Palindrome number")
is_palindrome(121)
```

```
121 is a Palindrome number
```

```
# q.Write a Python function to find the maximum of three numbers.
def max_num(a,b,c):
  if a>b and a>c:
    print(a)
  elif b>a and b>c:
    print(b)
  elif c>a and c>b:
    print(c)
max_num(1,2,3)
```

```
3
```

```
# q. write a function who reverse a string without using pre-defined functions and slic
def my_add(a):
  reverse=""
  for i in a:
    reverse=i+reverse
  print(reverse)
my_add("divya")
```

```
ayvid
```

```
# q. w.a.p who calculate second maximum number in a list without using pre-defined fu
def my_add(a):
  max_num=0
  second_max=0
```

```
  for i in a:
    if i>max_num:
      second_max=max_num
      max_num=i
    elif i>second_max and i!=max_num:
      second_max=i
  print(second_max)
my_add([1,2,3,4,5])
```

```
4
```

```
# write a function who remove vowels in a string.
#a="phython programming"
def my_add(a):
  b="aeiou"
  for i in a:
    if i  not in b:
      a=a.replace(i,"")
  print(a)
my_add("phython programming")
```

```
ooai
```

```
a={1,2,3,4,5}
print(a)
type(a)
```

```
{1, 2, 3, 4, 5}
set
```

```
a={1,23,45,5,6,78,45,78}
print(10 in a)
print(45 in a)
```

```
False
True
```

```
a={1,23,4,5,6}
a.add(34)
print(a)
```

```
{1, 34, 4, 5, 6, 23}
```

```
a={1,2,3,4,5}
a.remove(4)
print(a)
```

```
{1, 2, 3, 5}
```

```
a={1,2,3,4,5}                        # it is not show error
a.discard(8)
print(a)
```

```
{1, 2, 3, 4, 5}
```

```
a={1,2,34,5,6,77}
s=[]
for i in a:
  s.append(i)
  print(s)
```

```
[1]
[1, 2]
[1, 2, 34]
[1, 2, 34, 5]
[1, 2, 34, 5, 6]
[1, 2, 34, 5, 6, 77]
```

```
a={1,2,3,4}
b={2,3,4,5,6}
a.union(b)
print(a)
```

```
{1, 2, 3, 4}
```

```
a={1,2,3,4}
b={2,3,4,5,6}
b.union(a)
print(b)
```

```
{2, 3, 4, 5, 6}
```

```
a={1,2,3,4}
b={2,3,4,5,6}
print(a.intersection(b))
```

```
{2, 3, 4}
```

```
# q. write a function who create user defined list and we can
#add and remove and update element.
def my_add(a):
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-595189700.py in <cell line: 0>()
     11     a.update(5)
     12     print(update)
---> 13 my_add(1,2,3,4)
     14
     15

TypeError: my_add() takes 1 positional argument but 4 were given
```

```
a = {1, 2, 3}
b = {2, 3, 4}
print(a.difference(b))
```

```
{1}
```

```
a = {1, 2, 3}
b = {2, 3, 4}
print(b.difference(a))
```

```
{4}
```

```
# q. create a function who calculate average number of a list.
# a=[10,20,30]
def average(1):
  sum=0
  for i in 1:
    d=len(1)
    sum=sum+i
    print(int(sum/d))
1=[10,20,30]
average(1)
```

```
  File "/tmp/ipython-input-1097097397.py", line 3
    def average(1):
                ^
SyntaxError: invalid syntax
```

```
# q. write a function who add digits of a number.
# a=123
def my_add(a):
  sum=0
  for i in a:
     sum=sum+i
     print(sum)
my_add([1,2,3])
```

```
1
3
6
```

```
# q.write a function to find longest word in a sentence.
# a="phython makes coding interesting"
def my_add(a):
  b=a.split()
  longest_word=b[0]
  for i in b:
    if len(i)>len(longest_word):
      longest_word=i
  print(longest_word)
my_add("phython makes coding interesting")
```

```
interesting
```

```
# q. write a function to count how many digits are in a number.
# a=12345
def my_add(a):
  count=0
  while(a>0):
    a=a//10
    count=count+1
  print(count)
my_add(12345)
```

```
5
```

```
# q.write a function to remove duplicates from a list
a=[1,2,2,3,3,4,4]
def my():
 b=[]
 for i in a:
  if i not in b:
    b.append(i)
  print(b)
my()
```

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

```
#q. write a function to count the frequency of each lelement in a list.
#a=[1,2,2,3,3,3,4]
def my():
  a=[1,2,2,3,3,3,4]
  b={}
  for i in a:
    if i in b:
      b[i]=b[i]+1
    else:
      b[i]=1
  print(b)
my()
```

```
{1: 1, 2: 2, 3: 3, 4: 1}
```

```
# q.Write a Python function to sum all the numbers in a list.
def d(a,b,c):
  sum=0
  for i in a,b,c:
    sum=sum+i
    print(sum)
```

```
d(2,3,4)
```

```
2
5
9
```

```
# q.Write a Python function to multiply all the numbers in a list.
def a(a,b):
  mul=1
  for i in a,b:
    mul=mul*i
    print(mul)
a(4,6)
```

```
4
24
```

```
# introduction to set...
# identification ={obj1,obj1,,....obj n}
# it is unordered, mutable data type
# it does not allow duplicate elements
```

```
# q.write a Python program to reverse a string.
def my_add(a):
  reverse=""
  for i in a:
    reverse=i+reverse
  print(reverse)
my_add("divya")
```

```
ayvid
```

```
def my_add(a):
  reverse=""
  for i in a:
    reverse=i+reverse
  print(reverse)
my_add("divya")
```

```
# q.Write a Python function to calculate the factorial of a number (a non-negative in
def my_add(a):
  f=1
  for i in range(1,a+1):
    f=f*i
    print(f)
my_add(4)
```

```
1
2
6
24
```

```
# q.Write a Python function that accepts a string and counts the number of upper and
def my_add(a,b):
  upper=0
  lower=0
```

```
  for i in a,b:
    if i.isupper():
      upper=upper+1
    elif i.islower():
      lower=lower+1
  print(upper)
  print(lower)
my_add("divya","sasfg")
```

```
0
2
```

```
a={"sam","raj","mohit","sam","raj"}
a
```

```
{'mohit', 'raj', 'sam'}
```

```
type(a)
```

```
set
```

```
# loops in set
for i in a:
  print(i,end=" ")
```

```
raj mohit sam
```

```
len(a)
```

```
3
```

```
a={1,2,3}
b={4,5,6}                    #it is not add
#c=a+b
#c
```

```
# a={1,2,3}                  #duplicate values not allow in set so it is not working.
print(a*3)
```

```
#functions in set
```

```
# 1.len()---->it will return length of a set.
a={1,2,3}
len(a)
```

```
3
```

```
# add()----> it will add element.
a={1,2,3}
a.add(560)
a
```

```
{1, 2, 3, 560}
```

```
a={1,2,3,4}
a.remove(4)
a
```

```
{1, 2, 3}
```

```
a={1,2,3,4}
a.discard(3)
a
```

```
{1, 2, 4}
```

```
# operations in set
```

```
a={1,2,3,4}
b={3,4,5,6}
c=a.union(b)
c
```

```
{1, 2, 3, 4, 5, 6}
```

```
a={1,2,3,4}
b={3,4,5,6}
c=a|b
c
```

```
{1, 2, 3, 4, 5, 6}
```

```
a={1,2,3,4,1,2,5,6}
b={3,4,5,6,1,2}
c=a.intersection(b)
c
```

```
{1, 2, 3, 4, 5, 6}
```

```
a={1,2,3,4,1,2,5,6}
a.remove
a
```

```
{1, 2, 3, 4, 5, 6}
```

```
a={1,2,3,4}
b={3,4,5,6}
c=a.intersection(b)
c
```

```
{3, 4}
```

```
a={1,2,3,4}        # it will return common element in 2 set
b={3,4,5,6}
c=a&b
c
```

```
{3, 4}
```

```
a={1,2,3,4}                    # it will return only sets a values without including c
b={3,4,5,6}
a-b
```

```
{1, 2}
```

```
a={1,2,3,4}
b={3,4,5,6}
b=b-a
b
```

```
{5, 6}
```

```
a={1,2,3,4}                          # it using method , we can extract all elements w
b={3,4,5,6}
a^b
```

```
{1, 2, 5, 6}
```

```
a={1,2,3,4}
b=a.copy()
b
```

```
{1, 2, 3, 4}
```

```
# q. find the unique vowels in a srting using A set.
a="programming inn phython"
b=set(a)
c={'a','e','i','o','u'}
d=b&c
d
```

```
{'a', 'i', 'o'}
```

```
# q. given 2 lists , find common element using a set.
list1=[1,2,3,4,5,6]
list2=[3,4,5,6,7,8]
a=set(list1)
b=set(list2)
c=a&b
c
```

```
{3, 4, 5, 6}
```

```
# q. find element that are in list1 but not in list2.
l1=[1,2,3,4,5,6]
l2=[3,4,5,6,7,8]
a=set(l1)
b=set(l2)
c=a-b
c
```

```
{1, 2}
```

```
# q . find all duplicate element in a using a set.
num=[1,2,3,4,5,1,6,3,]
```

```
seen = set()
duplicates = set()

for n in num:
    if n in seen:
        duplicates.add(n)
    else:
        seen.add(n)

print( duplicates)
```

```
{1, 3}
```

```
# q. check if all element of one list exist in another list using sets.
a=[1,2,3]
b=[1,2,3,4,5]
set(a).issubset(b)
```

```
True
```

```
a = "programming"

b = set()
for ch in a:
    if ch in b:
        first_repeat = ch
        break
    b.add(ch)

print( first_repeat)
```

```
r
```

```
#dictionary comprehension.
# dictionary comprehension in phython offer a concise way to create
# dictionaries from iterablas
# similar to list comorehensinos , they allow you to
# construct dictionaries in a single line, with an optional condition for
# filtering
# syntax:
# { key_expression:_value_expression for item in iterable if condition}
```

```
# create a dictionary of squares
a={i:i**2 for i in range(1,6)}
a
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

```
# filter even number
a={i:i**2 for i in range(10) if i%2==0}
a
```

```
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64}
```

```
# reverse a dictionary
a={"a":1,"b":2,"c":3}
b={v:k for v,k in a.items()}
b
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
# convert list in dictionary with lengths
words=["apple","banana","cherry"]
b={word:len(word)for word in words}
b
```

```
{'apple': 5, 'banana': 6, 'cherry': 6}
```

```
# q. create a dictionary where keys are words and values are uppercase varsions.
words=["phython","code","dict"]
a={word.upper():len(word) for word in words}
a
```

```
{'PHYTHON': 7, 'CODE': 4, 'DICT': 4}
```

```
#create a dictionary from a sentence with each word and its frequency.
sentence="phython makes coding easy and phython is fun"
a={word:sentence.count(word) for word in sentence.split()}
a
```

```
{'phython': 2, 'makes': 1, 'coding': 1, 'easy': 1, 'and': 1, 'is': 1, 'fun': 1}
```

```
# create a nested dictionary of numbers from 1-3 with their square and cube.
a={x:(x**2,x**3)for x in range(1,4)}
print(a)
```

```
{1: (1, 1), 2: (4, 8), 3: (9, 27)}
```

```
# create a dictionary from a string where keys are vowels and values are their count.
a="dictionary comprehension practice"
b={i:a.count(i) for i in a if i in "aeiou"}
b
```

```
{'i': 4, 'o': 3, 'a': 2, 'e': 3}
```

```
a={i:"divisible by 3" for i in range(1,11) if i%3==0}
a
```

```
{3: 'divisible by 3', 6: 'divisible by 3', 9: 'divisible by 3'}
```

```
# create a dictionary from 2 list.
keys=["name","age","city"]
values=["same",30,"dehli"]
a=dict(zip(keys,values))
a
```

```
{'name': 'same', 'age': 30, 'city': 'dehli'}
```

```
a={k:v for k,v in zip(keys,values)}
a
```

```
{'name': 'same', 'age': 30, 'city': 'dehli'}
```

```
# q.convert a list of tuples into a dictionary.
p=[("a",1),("b",2),("c",3)]
a=dict(p)
a
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
Start coding or generate with AI.
```

```
# introduction to dicitionary
# identification ={key:value,key:value...}
# element will separate by commo.
# indexing start from 0 and length start from 1.
```

```
a={"model":"hyrender","brend":"toyota","price":"200000"}
a
```

```
{'model': 'hyrender', 'brend': 'toyota', 'price': '200000'}
```

```
type(a)
```

```
# loop in dictionary
```

```
### extract all keys
```

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

```
model
brend
price
```

```
for i in a.keys():
 print(i)
```

```
model
brend
price
```

```
a={"model":"hyrender","brend":"toyota","price":"200000"}
for i in a:
  print(a[i])
```

```
hyrender
toyota
200000
```

```
### extract all keys and values
for x,y in a.items():
```

```
  print(x,y)
```

```
a 1
b 2
c 3
```

```
a
```

```
{'model': 'hyrender', 'brend': 'toyota', 'price': '200000'}
```

```
a["model"]
```

```
'hyrender'
```

```
a["color"]="black"
```

```
a
```

```
{'model': 'hyrender', 'brend': 'toyota', 'price': '200000', 'color': 'black'}
```

```
# functions in dictionary
```

```
# 1. len()----> it will return lengthof a dictionary
len(a)
```

```
4
```

```
# 2. pop()---->it will remove key value pair.
a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2906379790.py in <cell line: 0>()
      1 # 2. pop()---->it will remove key value pair.
----> 2 a

NameError: name 'a' is not defined
```

```
a.pop
```

```
<function dict.pop>
```

```
a
```

```
{'model': 'hyrender', 'brend': 'toyota', 'price': '200000', 'color': 'black'}
```

```
# 3. popitem()----> it will remove element from the last
```

```
('price', '200000')
```

```
divya={10:"divya",39:"ddddd"}
print(divya)
```

```
{10: 'divya', 39: 'ddddd'}
```

```
divya[10]="rrrrr"
print(divya)
```

```
{10: 'rrrrr', 39: 'ddddd'}
```

```
divya[33]="gggfgfg"
print(divya)
```

```
{10: 'rrrrr', 39: 'ddddd', 33: 'gggfgfg'}
```

```
divya={"d":90,"r":45}
divya["d"]=divya["d"]+45
print(divya)
```

```
{'d': 135, 'r': 45}
```

```
#q. find the student with the second highest marks.
a={"amit":85,"sita":90,"ravi":78,"mohit":88}
b=list(a.values())
b.sort()
print(b[-2])
```

```
88
```

```
a={"amit":85,"sita":90,"ravi":78,"mohit":88}
highest=-1
for i in students:
  if students[i]>highest:
    highest=students[i]
print(highest)

second_highest=-1
for i in students:
  if students[i]>second_highest and students[i]<highest:
    second_highest=students[i]
    second_name=i
print("second topper:",second_name,"marks:",second_highest)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-4033201857.py in <cell line: 0>()
      1 a={"amit":85,"sita":90,"ravi":78,"mohit":88}
      2 highest=-1
----> 3 for i in students:
      4   if students[i]>highest:
      5     highest=students[i]

NameError: name 'students' is not defined
```

```
# q. count frequency of each character in a string using in a dictionary.
a="programming"
b={}
for i in a:
  if i in b:
    b[i]=b[i]+1
  else:
```

```
    b[i]=1`
print(b)
```

```
  File "/tmp/ipython-input-1792692906.py", line 8
    b[i]=1`
          ^
SyntaxError: invalid syntax
```

```
# q. find keys with maximum value.
a={"a":50,"b":80,"c":80,"d":60}
max_value=max(a.values())
for i in a:
  if a[i]==max_value:
    print(i)
```

```
b
c
```

```
a=tuple(i**2 for i in range(1,5))
a
```

```
(1, 4, 9, 16)
```

```
# q. remove all the keys with values less than 50.
a={"a":50,"b":80,"c":80,"d":60}
b=list(a.values())
b.sort()
for i in b:
  if i > 50:
    print(i)
```

```
60
80
80
```

```
# q. remove all the keys with values less than 50.
a={"a":50,"b":80,"c":80,"d":60}
b={}
for i,j in a.items():
  if j>=50:
    b[i]=j
b
```

```
{'a': 50, 'b': 80, 'c': 80, 'd': 60}
```

```
#q. generate a tuple of (number ,square ) pairs for numbers from 1 to 6.
a = tuple((i**2) for i in range(1, 7))
print(a)
```

```
(1, 4, 9, 16, 25, 36)
```

```
#q. create a tuple of the first letter of each word in a sentence.
a= "mathrmatics is good"

result = tuple(word[0] for word in a.split())
```

```
print(result)
```

```
('m', 'i', 'g')
```

```
#q.create a tuple of square of numbers from 1-10 but only include numbers greater the
a=tuple(i**2 for i in range(1,11) if i>5)
a
```

```
(36, 49, 64, 81, 100)
```

```
#q. for numbers 1-8 create a tuple where multiple if 3 become "three" multiple of 5 b
t = tuple( "three" if x % 3 == 0 else "five" if x % 5 == 0 else x for x in range(1, 5
print(t)
```

```
(1, 2, 'three', 4)
```

```
#q. from the list of names, create a tuple of names in uppercase if length>4 , otherw
name =["AI","PHYTHON","ML","SCIENCE"]

A = tuple(n.upper() if len(n) > 4 else n.lower() for n in name)
print(A)
```

```
('ai', 'PHYTHON', 'ml', 'SCIENCE')
```

```
#Q. from the word "developer",create a tuple of only vowels.
word = "developer"
vowels = tuple(ch for ch in word if ch.lower() in "aeiou")
print(vowels)
```

```
('e', 'e', 'o', 'e')
```

```
# q. create a tuple of (world ,length) pairs for each word in sentence.
a="phython tuple comprehension practice"

t = tuple((word, len(word)) for word in a.split())
print(t)
```

```
(('phython', 7), ('tuple', 5), ('comprehension', 13), ('practice', 8))
```

```
a=(10,20,30,40,50)
for i in a():
  print(a)
```

```
Start coding or generate with AI.
```

```
a=tuple(i**2 for i in range(1,10) if i%2==0)
a
```

```
(4, 16, 36, 64)
```

```
a=tuple("even" if x%2==0 else x for x in range(1,5))
a
```

```
(1, 'even', 3, 'even')
```

```
#q. from the string "phython" create a tuple that converts vowels to uppercase and co
s = "phython"
vowels = "aeiou"
result = tuple(ch.upper() if ch.lower() in vowels else ch.lower() for ch in s)
print(result)
```

```
('p', 'h', 'y', 't', 'h', 'O', 'n')
```

```
#q. create a tuple where numbers divisible by 3 are "div3" , otherwise keep the numbe
nums = tuple( "div3" if i%3==0 else i for i in range(1, 11))
print(nums)
```

```
(1, 2, 'div3', 4, 5, 'div3', 7, 8, 'div3', 10)
```

```
#q. in a given range(1,11) replace odd numbers to "odd" and even number to "even"
result = ["even" if i % 2 == 0 else "odd" for i in range(1, 11)]
print(result)
```

```
['odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even']
```

```
a = [10, 20, 30, 40, 50]
print(a[0], a[-1])
```

```
10 50
```

```
a = ["apple","banana","cherry"]
print(len(a))
```

```
3
```

```
d = [5,10,15,20,25]
double = [x*2 for x in d]
print(double)
```

```
[10, 20, 30, 40, 50]
```

```
a = [45,23,4,56,78,]
second_largest = sorted(a)[-2]
print(second_largest)
```

```
56
```

```
m = [10,20,30,40,50,60]
middle = m[-3:-1]
print(middle)
```

```
[40, 50]
```

```
a = [10, 25, 40, 55, 60, 75]
greater_50 = [x for x in a if x > 50]
print(greater_50)
```

```
[55, 60, 75]
```

```
a = [2,4,6,8,10]
b = [x**3 for x in a]
print(b)
```

```
[8, 64, 216, 512, 1000]
```

```
a = [1,2,3,2,4,2,5]
print(a.count(2))
```

```
3
```

```
a = [1,2,3,4,5,6]
evens = [x for x in a if x % 2 == 0]
print(evens)
```

```
[2, 4, 6]
```

```
a = [1, 2, 3, 4, 5]
b = [4, 5, 6, 7, 8]
without_common = [x for x in a + b if (x not in a) or (x not in b)]
print(without_common)
```

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

```
text = "hello divya"
reversed = [word[::-1] for word in text.split()]
print(reversed)
```

```
['olleh', 'ayvid']
```

```
x = [1, 2, 3]
y = ['a', 'b']
pairs = [(i, j) for i in x for j in y]
print(pairs)
```

```
[(1, 'a'), (1, 'b'), (2, 'a'), (2, 'b'), (3, 'a'), (3, 'b')]
```

```
matrix = [[1, 2], [3, 4], [5, 6]]
flat = [num for row in matrix for num in row]
print(flat)
```

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

```
lst = [10, 20, 30, 40, 50]
double = [x * 2 for x in lst]
```

```
print(double)
```

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

```
lst = [12, 25, 30, 45, 50, 60]
num = [x for x in lst if x % 5 == 0]
print(num)
```

```
[25, 30, 45, 50, 60]
```

Start coding or generate with AI.

```
s = "programming"
unique_chars = [ch for ch in set(s)]
print(unique_chars)
```

```
['n', 'r', 'm', 'o', 'i', 'g', 'p', 'a']
```

```
primes = [x for x in range(2, 31) if all(x % i != 0 for i in range(2, x))]
print(primes)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

```
a=(1,3,2,4,5,6,7)
for i in a:
  if i%2!=0:
    print(i)
```

```
1
3
5
7
```

```
a = (10, 20, 5, 50, 30)
b = list(set(a))
b.sort()
print(b[-2])
```

```
30
```

```
a=(1,2,3,4,5)
sum=0
for i in a:
  sum=sum+i
  print(sum)
```

```
1
3
6
10
15
```

```
#positive and negative count
a=(1,2,-4,-2,-5)
pos=neg=0
```

```
for i in a:
  if i>0:
    pos=pos+1
  else:
    neg=neg+1
print(pos,neg)
```

```
2 3
```

```
# odd even count
a=(1,2,3,5,7,4,56,7,45)
odd=even=0
for i in a:
  if i%2==0:
    even+=1
  else:
    odd+=1
print(odd,even)
```

```
6 3
```

```
a = (10, 25, 5, 60, 15)
mx = a[0]
for i in a:
    if i > mx:
        mx = i
print(mx)
```

```
60
```

```
a = (10, 25, 5, 60, 15)
mx = a[0]
for i in a:
    if i < mx:
        mx = i
print(mx)
```

```
5
```

```
a = (1, (2, 3), (4, 5))
for i in a:
    if type(i) == tuple:
        for j in i:
            print(j)
    else:
        print(i)
```

```
1
2
3
4
5
```

```
a=(10,2,30)
b=list(a)
print(b)
```

```
[10, 2, 30]
```

```
a = (5, 1, 8, 2, 3)
b = tuple(sorted(a))
print(b)
```

```
(1, 2, 3, 5, 8)
```

```
a = (10, 20, 30, 40)
print(a.index(30))
```

```
2
```

```
# Q.1 swap 2 element in a tuple.
a = (10, 20, 30, 40)
lst = list(a)
lst[1], lst[3] = lst[3], lst[1]
a = tuple(lst)
print(a)
```

```
(10, 40, 30, 20)
```

```
#Q.2 find element that appear only once in a tuple
a = (1, 2, 2,3,4,4,5)
unique = tuple(x for x in a if a.count(x) == 1)
print(unique)
```

```
(1, 3, 5)
```

```
#Q.3 check if all the element in a tuple are the same
a = (2,2,2,2)
if len(set(a)) == 1:
    print("all element are same ")
else:
    print("all element are not same")
```

```
all element are same
```

```
a = [(2,4), (2,8), (1,3), (2,5)]
result = [t for t in a if all(x % 2 == 0 for x in t)]
print(result)
```

```
[(2, 4), (2, 8)]
```

```
# Q.replace the last element of each tuple with its square.
a = [(1,2), (3,4), (5,6)]
result = []
for t in a:
    last_sq = t[1] ** 2
    new_tuple =t[:-1] + (last_sq,)
    result.append(new_tuple)
print(result)
```

```
[(1, 4), (3, 16), (5, 36)]
```

```
# q.create a tuple of unique characters from a srting.
a="programming"
```

```
unique = []
for ch in a:
    if ch not in unique:
      unique.append(ch)
print(result)
```

```
('p', 'r', 'o', 'g', 'a', 'm', 'i', 'n')
```

```
# q.find the most freqent element in a tuple.
a=(1,2,3,3,4,3,3,4,4,5)
max=0
for i in a:
  if a.count(i)>max:
    max=a.count(i)
for i in a:
  if a.count(i)==max:
    print(i)
    break
```

```
3
```

```
# q.generate all possible pairs from two tuples.
a=(1,2)
b=("a","b")
pairs=[]
for i in a:
  for j in b:
    pairs.append((i,j))
print(pairs)
```

```
[(1, 'a'), (1, 'b'), (2, 'a'), (2, 'b')]
```

```
# q. group element of a tuple into pairs
a=(1,2,3,4,5,6)
b=[]
for i in range(0,len(a),2):
  b.append((a[i],a[i+1]))
print(b)
```

```
[(1, 2), (3, 4), (5, 6)]
```

```
items=[]
size=int(input("enter size"))
for i in range(size):
  val=input("enter element")
  if val not in items:
    items.append(val)
unique_tuple=tuple(items)
print("unique tuple",unique_tuple)
```

```
enter size4
enter element3
enter element2
enter element1
enter element3
unique tuple ('3', '2', '1')
```

```
for i in range(1,6):
  for j in range(i):
     print(i,end=" ")
  else:
     print(j,end=" ")
print()
```

```
1 0 2 2 1 3 3 3 2 4 4 4 4 3 5 5 5 5 5 4
```

```
t = (1, 2, 3, 4, 5)

rev_list = []
for i in range(len(t)-1, -1, -1):
    rev_list.append(t[i])

rev_t = tuple(rev_list)

print("Original Tuple =", t)
print("Reversed Tuple =", rev_t)
```

```
Original Tuple = (1, 2, 3, 4, 5)
Reversed Tuple = (5, 4, 3, 2, 1)
```

```
# tuple in phython
# immutable datatype -----> we cannot or update values at a run time
# indexing start from 0 and length start from 1
# this is like a container in which we can store diff diff data type element
```

```
a=("sam","rahul",45,32.56)
print(a)
```

```
('sam', 'rahul', 45, 32.56)
```

```
# indexing in tuple
print(a[0])
print(a[1])
```

```
sam
rahul
```

```
# slicing in tuple
a=("sam","raj",45,34.67)
print(a[0:3])
print(a[1:4])
print(a[:2])
print(a[:])
```

```
('sam', 'raj', 45)
('raj', 45, 34.67)
('sam', 'raj')
('sam', 'raj', 45, 34.67)
```

```
## immutable datatype
a=("sam","rahul",56,45.67)
#a[1]="rahul"
```

```
# properties of tuple
```

```
# concatenation
```

```
a=(1,2,3)
b=(4,5,6)
print(a+b)
```

```
(1, 2, 3, 4, 5, 6)
```

```
# replication
a=(1,2,3)
b=a*3
print(b)
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

```
a=(1,2,3,4)
print(len(a))
```

```
4
```

```
# functions in tuple
len()--> it will return length of a tuple
```

```
a=("sam","rahul",45,6.7)
a.index(45)
```

```
2
```

```
a=("sam","raj",45,87.5,45,45)
print(count(d (45)))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1625179652.py in <cell line: 0>()
      1 a=("sam","raj",45,87.5,45,45)
----> 2 print(count(a(45)))

TypeError: 'tuple' object is not callable
```

```
# min()--> it will return minimum number in a tuple
a=(1,2,3,4)
min(a)
```

```
# min()--> it will return max number in a tuple
a=(1,2,3,4)
max(a)
```

```
# how can we create a user defined tuple?
a=[]
size=int(input("enter size"))
for i in range(size):
   val=input("enter item")
   a.append(val)
b= tuple(a)
print(b)
c=list(b)
d=input("enter item which you want to add")
c.append(d)
e= tuple(c)
print(e)
```

```
enter size4
enter item4
enter item4
enter item5
enter item6
('4', '4', '5', '6')
enter item which you want to add3
('4', '4', '5', '6', '3')
```

```
a = []
size = int(input("Enter size: "))

for i in range(size):
    val = input("Enter item: ")
    a.append(val)

b = tuple(a)
print("Tuple:", b)

c = list(b)
delete = input("Enter item to delete: ")
if delete in c:
    c.remove(delete)
else:
    print("Item not found!")
d = tuple(c)
print("Tuple after deletion:", d)

f = list(d)
index = int(input("Enter index to insert: "))
insert = input("Enter value to insert: ")
f.insert(index, insert)
print("List after insertion:", f)

g = list(f)
count_val = input("Enter value to count: ")
print("Count of", count_val, "=", g.count(count_val))
```

```
Enter size: 5
Enter item: 4
Enter item: 3
Enter item: 5
Enter item: 4
Enter item: 6
```

```
Tuple: ('4', '3', '5', '4', '6')
Enter item to delete: 5
Tuple after deletion: ('4', '3', '4', '6')
Enter index to insert: 7
Enter value to insert: 4
List after insertion: ['4', '3', '4', '6', '4']
Enter value to count: 4
Count of 4 = 3
5
```

```
for i in range(1,6):
  for j in range(1,6):
    if i==1 or i==5 or j==i :
      print(i,end=" ")
    else:
      print(" ",end=" ")
  print()
```

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

```
for i in range(1,6):
  for j in range(1,6):
    if i==1 or i==5 or i==j :
      print("*",end=" ")
    else:
      print(" ",end=" ")
  print()
```

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

```
# "list cpmprehension"
# list comprehension in phython a concise way to create lists
# syntax
# [expression for item in iterable if condition]
```

```
# white a program who print list of square?
```

```
a=[1,2,3,4]
b=[]
for i in range(1,len(a)+1):
  b.append(i**2)
print(b)
print(a)
```

```
[1, 4, 9, 16]
[1, 2, 3, 4]
```

```
squar=[x**2 for x in range(1,11)]
print(squar)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
#"nested list comprehension"
```

```
x_val=[1,2,3]
y_val=[3,4,5]
pair=[(x,y) for x in x_val for y in y_val]
pair
```

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

```
#multiplication table
```

```
a=[f"{i}x{j}={i*j}" for i in range(1,3) for j in range(1,11)]
a
```

```
['1x1=1',
 '1x2=2',
 '1x3=3',
 '1x4=4',
 '1x5=5',
 '1x6=6',
 '1x7=7',
 '1x8=8',
 '1x9=9',
 '1x10=10',
 '2x1=2',
 '2x2=4',
 '2x3=6',
 '2x4=8',
 '2x5=10',
 '2x6=12',
 '2x7=14',
 '2x8=16',
 '2x9=18',
 '2x10=20']
```

```
x_val=[1,2,3]
y_val=[3,4,5]
prod=[(x,y) for x in x_val for y in y_val if x+y>5]
prod
```

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

```
#4. create a list comprehension who filter only even numbers.
```

```
a=[i for i in range (1,10) if i**2>20]
a
```

```
[5, 6, 7, 8, 9]
```

```
#convert all string into upper case
```

```

```

```
names=["divya","charchil","kanishtha"]
upper_names=[i.upper() for i in names]
upper_names
```

```
['DIVYA', 'CHARCHIL', 'KANISHTHA']
```

```
filtered_shop = [item for item in shop if item not in (bakery + clothes + stationery)
print(filtered_shop)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-3022919384.py in <cell line: 0>()
----> 1 filtered_shop = [item for item in shop if item not in (bakery + clothes +
stationery)]
      2 print(filtered_shop)

NameError: name 'shop' is not defined
```

```
numbers = [(x, x**2) for x in range(1, 6)]
print(numbers)
```

```
[(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

```
words = ["python", "ai", "science", "data"]

long_words = [word for word in words if len(word) > 3]
print(long_words)
```

```
['python', 'science', 'data']
```

```
A = ("12333545678", "123567", "123456788456","9166790334")

numbers = [num for num in A if num.isdigit() and len(num) == 10]
print(numbers)
```

```
['9166790334']
```

```
# using the list cpmprehension list aphababats in string print
```

```
ch=["divya","charchil",45,34,57,78]
alpha_name=[ch.alpha()for i in ch]
alpha_name
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-2653559818.py in <cell line: 0>()
      1 ch=["divya","charchil",45,34,57,78]
----> 2 alpha_name=[ch.alpha()for i in ch]
      3 alpha_name

AttributeError: 'list' object has no attribute 'alpha'
```

```
a=" i love you"
c="aeiou"
b=[i for i
```

```
   in a if i in c]
print(b)
```

```
['i', 'o', 'e', 'o', 'u']
```

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

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

```
a=["sam","divya ","raj"]
m=[a[i][0] for i in range(len(a))]
m
```

```
['s', 'd', 'r']
```

```
a=[i for i in range(1,51) if i%3==0 and i%5!=0]
a
```

```
[3, 6, 9, 12, 18, 21, 24, 27, 33, 36, 39, 42, 48]
```

```
Start coding or generate with AI.
```

```
a=[i for i in range(1,11) if i%2==0]
a
```

```
[2, 4, 6, 8, 10]
```

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

```
*
```

```
n=10
for i in range(1,7):
  for j in range(i):
   print("*",end=" ")
else:
  print("",end="")
```

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

```
a=[]
size=int(input("enter size"))
for i in range(size):
```

```
  val=int(input("enter number"))
  a.append(val)
print(a)
total_sum
```

```
a=[]
size=int(input("enter size"))
for i in range(size):
  val=int(input("enter the number"))
  a.append(val)
print(a)
sum=0
for i in range(len(a)):
  sum=sum+a[i]
print("total=",sum)
```

```
enter size4
enter the number3
enter the number4
enter the number5
enter the number6
[3, 4, 5, 6]
total= 18
```

```
a=[]
size=int(input("enter size"))
for i in range(size):
  val=int(input("enter the number"))
  a.append(val)
print(a)
max=0
for i in range(len(a)):
 if a[i]>max:4
  max=a[i]
print("max total is =",max)
```

```
enter size5
enter the number6
enter the number5
enter the number6
enter the number7
enter the number8
[6, 5, 6, 7, 8]
max total is = 8
```

```
nums=[1,2,3,4,5,6,8,9,10]
total_sum=0
for num in range(1,11):
    total_sum+=i
print("total sum is",total_sum)
```

```
total sum is 40
```

```
i=121
temp=i
reverse=0
num=0=
while(i>0):
```

```
  digit=num%10
  reverse=reverse*10+digit
  i=i//10
if temp==reverse:
    print("palindrome")
else:
    print("no palindrome")
```

```
no palindrome
```

```
i=3
rev=0
fact=1
while(i>0):
  d=num%10
  rev=rev*10+d
  i=i//10
print(fact)
```

```
0
0
```

```
i=34
rev=0
sum=0
n=0
while(i>0):
  d=n%10
  rev=rev*10+d
  i=i//10
  sum=sum+d
print(sum)
```

```
0
```

```
i=int(input("enter number"))
n=int(input("enter number"))
while(i<=n):
  if i%2==0:
    print(i,end=" ")
    i=i+1
```

```
2
```

```
i=1
n=21
sum=0
while(i<=n):
  if i%2==0:
    sum=sum+i
    i=i+1
print(sum)
```

```
i=334
rev=0
while(i>0):
```

```
  d=i%10
  rev=rev*10+d
  i=i//10
print(rev)
```

```
0
```

```
i=1
n=30
while(i<=n):
  if(i%2==0):
    print(i)
    i=i+1
```

```
2
```

```
i=121
rev=0
temp=i
while(i>0):
  d=n%10
  rev=rev*10+d
  i//=10
if  temp==rev:
    print("palindrome")
else:
    print("no palindrome")
```

```
no palindrome
```

```
i=1
while(i<=10):
  print(10*i)
  i=i+1
```

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

```
i=1
while(i<=10):
  if (i%2==0):
    print(3*i)
    i+=1
```

```
n=5
fact=1
i=1
while i<=n:
  fact*=i
  i+=1
  print("sum of fact",i)
```

```
n=5
fact=1
for i in range(1,1+n):
  fact*+1
  print("sum of fact",i)
```

```
list=["d","e","t","s","w"]
list.sort()
print(list)
```

```
['d', 'e', 's', 't', 'w']
```

```
list=["janooniyat","khamosiya","saiyara"]
print(list)
```

```
['janooniyat', 'khamosiya', 'saiyara']
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-3193190764.py in <cell line: 0>()
      1 list=["janooniyat","khamosiya","saiyara"]
      2 print(list)
----> 3 list.appeand('divya')

AttributeError: 'list' object has no attribute 'appeand'
```

```
a=1
b=5
while(a<=b):
  print(a)
  a=a+1
```

```
1
2
3
4
5
```

```
i=1
n=10
while(i<=n):
  print(i,end=" ")
  i=i+1
```

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

```
i=1
n=20
while(i<=n):
  if(i%2==0):
    print(i)
    i=i+1
```

```
i=4
n=40
while(i<=n):
  if(i*4==0):
    print(i)
    i=i+1
```

```
a=["sam",45,98.7,"divya"]
for i in range(len(a)):
  print(a[i],i,end=" ")
```

```
sam 0 45 1 98.7 2 divya 3
```

```
# slicing in list
a=['sam',45,56.90,'rahul']
a[3]='python'
print(a)
```

```
['sam', 45, 56.9, 'python']
```

```
a=['sam',45,56.90,'rahul']
for i in a:
  print(i,end=" ")
```

```
sam 45 56.9 rahul
```

```
# "properties of list "
# "concatenatio"
a=[1,2,3,4]
b=[5,6,7,8]
c=a+b
print(c)
```

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

```
# "replication"
a=[1,2,3]
a*3
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
Start coding or generate with AI.
```

```
fuctions in
```

```
i=int(input("enter the number"))
rev=0
```

```
while(i>0):
  digit=i%10
  rev=rev*10+digit
  i=i//10
print(rev)
```

```
enter the number34
43
```

```
a=["sam"45,59.6,"rahul"]
print(a[-1::-1])
```

```
  File "/tmp/ipython-input-1405448812.py", line 1
    a=["sam"45,59.6,"rahul"]
       ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

```
i=2
sum=0
while(i<=10):
  sum=sum+i
  i=i+1
print(sum)
```

```
54
```

```
a=['sam',45,56.90,'rahul']
b=[]
for i in range(len(a)):
  print(a[i])
  if (a[i].isalpha()==0):
    b.append(a[i])
  else:
    c.append(a[i])
```

```
sam
45
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-716616200.py in <cell line: 0>()
      3 for i in range(len(a)):
      4   print(a[i])
----> 5   if (a[i].isalpha()==0):
      6     b.append(a[i])
      7   else:

AttributeError: 'int' object has no attribute 'isalpha'
```

```
i=int(input("enter the number"))       # palindrome quentions
temp=i
rev=0
while(i>0):
  d=i%10
  rev=rev*10+d
  i//=10
```

```
if temp==rev:
  print("palindrome")
else:
  print("no")
```

```
enter the number45
no
```

```
Start coding or generate with AI.
```

```
i="divya is developer"
rev=0
while(i>0):
  d=n%10
  rev=rev*10+d
  i=i//10
print(rev.split()[::-1])
```

```
i=1
sum=0
while(i<=10):
  sum=sum+i
  i=i+1
print(sum)
```

```
n=int(input("enter the number"))
rev=0
temp=0
while(i>0):
  d=i%10
  rev=rev*10+d
  i//=10
  if temp==rev:
    print("palindrome")
  else:
    print("not palindrome")
```

```
data="fun is phython"
rev=0
while(data>0):
  rev=rev*10+data
  data//=10
print(rev.split[]::-1)
```

```
i=4
n=40
while(i<=n):
  if(i*4==0):
    print(i)
    i+=1
```

```
i=84951
l_d=0
while(i>0):
  d=i%10
  if d>l_d:
```

```
    l_d=0
    i//=10
print(l_d)
```

```
i=84951
small=0
while(i<0):
  d=i%10
  if d<small:
    small=d
    i//=10
print(small)
```

```
i=1
n=30
sum=0
while(i<=n):
  if i%2!=0
  sum=sum+i
  i=i+1
print(sum)
```

```

```

```
i = 2
count = 0
total = 0
while count < 15:
    total += i * i
    i += 2
    count += 1
print("Sum of squares of first 15 even numbers is:", total)
```

```
text = "phython program"
vowels = "aeiou"
count = 0
i = 0
while i < len(text):
    ch = text[i]
    if ch.isalpha() and ch not in vowels:
     count += 1
     i += 1
print("Number of consonants:", count)
```

```
i=36
n=1
while(i<=n)
```

```
  if i*36==0:
    print(i,end=" ")
    i+=1
```

```
num = 36
i = 1
while i <= num:
    if num % i == 0:
        print(i, end=" ")
    i += 1
```

```
a=int(input("enter the maximum number"))
b=int(input("enter the minimum number"))
p=int(input("enter step number"))
while(a>=b):
  print(a,end=" ")
  a=a-p
```

```
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
```

```
i=1
n=4
sum=0
while(i<=n):
  sum=sum+i*i
  i=i+1
print(sum)
```

```
30
```

```
i=153
orig=i
sum=0
while(i>0):
  sum=sum+(i%10)**3
  i=i//10
if(orig==sum):
  print("yes armstong")
```

```
else:
  print("not armstong")
```

```
yes armstong
```

```
i=1234
odd=0
even=0
while(i>0):
  d=i%10
  if(d%2==0):
    even=even+d
  else:
    odd=odd+d
  i=i//10
print(odd,even)
```

```
i=123
rev=0
while(i>0):
  rev=rev*10+i%10
  i=i//10
print(rev)
```

```
i=1
n=1
sum=0
while(i<=n):
  sum=sum+i
  i=i+1
print(sum)
```

```
1
```

```
import random

# Players and their positions
players = ["Red", "Blue", "Green", "Yellow"]
positions = {player: 0 for player in players}

def roll_dice():
    return random.randint(1, 6)

def play_turn(player):
    input(f"\n{player}'s turn... Press Enter to roll dice 🎲 ")
    dice = roll_dice()
    print(f"{player} rolled {dice}")
    positions[player] += dice
    if positions[player] >= 50:   # Win condition (board size 50 steps)
        print(f"🏆 {player} wins the game!")
        return True
    else:
        print(f"{player}'s position: {positions[player]}")
        return False

# Main Game Loop
game_over = False
```

```
while not game_over:
    for player in players:
        game_over = play_turn(player)
        if game_over:
            break
```

```
for i in range(1,6):
  for j in range(1,6):
    if i==0 or i==5 or i==j or j==1:
      print("*",end=" ")
    else:
      print(" ",end=" ")
  print()
```

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

Start coding or generate with AI.

```
for i in range(1,6):
  for j in range(1,6):
    if j==0 or j==5 or i==j or i==1:
      print("*",end=" ")
    else:
      print(" ",end=" ")
  print()
```

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

```
for i in range(1,6):
  for j in range(1,6):
    if i==0 or i==5 or i==j or j==1:
      print("*",end=" ")
    else:
      print(" ",end=" ")
  print()
```

```
for i in range(1,6):
  for j in range(1,6):
    if i==j or i==5 or j==1 or j==5:
      print("*",end=" ")
    else:
      print(" ",end=" ")
print()
```

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

```
for i in range(1,6):
  for j in range(i):
    print("*",end=" ")
```

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

```
a="phython"
for i in range(len(a)):
  print(i)
```

```
0
1
2
3
4
5
6
```

```
for i in range(3):
  for j in range(5):
    print(i,j,end=" -")
```

```
0 0 -0 1 -0 2 -0 3 -0 4 -1 0 -1 1 -1 2 -1 3 -1 4 -2 0 -2 1 -2 2 -2 3 -2 4 -
```

```
for i in range(1,6,1):
   print(i *"*")
for j in range(6,0,-1):
   print(j*"*")
```

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

Default title text

```
# @title Default title text
for i in range(6,0,1):
   print(i *"*")
for j in range(6,0,-1):
   print(j *"*")
```

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

```
n=5
for i in range(n):
  for j in range(n):
```

```
    if i=0 or i=n-1 or j=0 or j=n-1:
      print(j+1,end=" ")
    else:
      print(" ",end=" ")
    print()
```

```
n=5
for i in range(n):
  for j in range(n):
    if i==0 or i==n-1 or j==0 or j==n-1:
      print(j+1,end=" ")
    else:
      print(" ",end=" ")
  print()
```

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

```
for i in range(1,6):
  for j in range(i):
    print("*",end=" ")
  print()
```

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

```
for i in range(6,0,-1):
  for j in range(i):
    print("*",end=" ")
  print()
```

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

```
for i in range(1,6):
  for j in range(i):
    print(i,end=" ")
  print()
```

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

```
c
```

```
count=0
for i in range(1,9):
```

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

```
for i in range(1,6):
  for j in range(1,i+1):
    print(j,end=" ")
  print()
```

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

```
count=0
for i in range(5,0,-1):
  for j in range(1,i+1):
    print(count,end=" ")
    count=count+1
  print(" ")
```

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

```
for i in range(5,0,-1):
  for j in range(1,i+1):
    print(j,end=" ")
  print(" ")
```

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

```
n=5
for i in range(1,n+1):
  print(" "*(n-i),end="")
  for j in range(1,i+1):
    print(j,end=" ")
  print()
```

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

```
n=5
for i in range(1,n+1):
  print(" "*(n-i),end="")
  for j in range(1,i+1):
```

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

```
    1
   12
  123
 1234
12345
```

Start coding or generate with AI.

```
fr i in range(p0-)
```

```
for i in range(9):
  for j in range(8):
    print(j,i,end=" ")
```

```
0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 0 1 1 1 2 1 3 1 4 1 5 1 6 1 7 1 0 2 1 2 2 2 3 2 4 2 5
```

```
a="apple"
print(a[-5:-1])
```

```
appl
```

```
a="divya bamsal"
print(a.replace("bamsal","bansal"))
```

```
divya bansal
```

```
data="divya dksjdkls"
print(len(data))
```

```
14
```

```
a=int(input("enter the first number"))
b=int(input("enter the second number"))
c7=int(input("enter the third number"))
if a>=b and a>=c:
  print("a is greater")
elif b>=a and b>=c:
  print("b is greater")
else:
  print("c is greater")
```

```
enter the first number56
enter the second number76
enter the third number90
c is greater
```

```
a="divya"
print(len(a))
```

```
5
```

```
a="allo"
print(a.count("l"))
```

```
2
```

```
a="123"
print(a.isdigit())
```

```
True
```

```
a="div123"
print(a.isalnum())
```

```
True
```

```
a="puhHYTCH"
print(a.lower())
```

```
puhhytch
```

```
a="oefbujfrYFEJBG"
print(a.upper())
```

```
OEFBUJFRYFEJBG
```

```
text="apple,banana,cherry"
print(text.split(","))
```

```
['apple', 'banana', 'cherry']
```

Double-click (or enter) to edit

```
fruits=["apple","banana","mango"]
print(" ".join(fruits))
```

```
apple banana mango
```

```
a="hello world"
print(a.count("o"))
```

```
2
```

```
a="  hello "
print(a.split())
```

```
['hello']
```

```
print("welcome to the bakery shop")
print("pizza:price:-240/-")
print("pasta:price:-150/-")
print("burger:price:-100/-")
print("sandwich:price:-120-")
a=int(input("enter the pizza"))
b=int(input("enter the pasta"))
c=int(input("enter the burger"))
d=int(input("enter the sandwich"))
```

```
welcome to the bakery shop
pizza:price:-240/-
pasta:price:-150/-
burger:price:-100/-
sandwich:price:-120-
enter the pizza23
enter the pasta56
enter the burger298
enter the sandwich3567
```

```
name="simar"
age=23
print(f"my name is {name} and my age is {age} year old")
```

```
my name is simar and my age is 23 year old
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
data="hello"
print(data[::-1])
```

```
olleh
```

```
s="progamming"
```

```
s="banana"
count=0
for i in s:
  if i=="a":
   count+=1
print(count)
```

```
3
```

```
s="hello world "
print(s.upper())
```

```
HELLO WORLD
```

```
s=["apple,banana,cherry"]
for fruits in s:
  print(fruits,end=" ")
```

```
apple,banana,cherry
```

```
num="123"--
ch="abc"
for ch in i:
  if not in ch:
    print("ch")
  elif i==1 and i==2 and i==3:
    print("num")
```

```
  File "/tmp/ipython-input-767681291.py", line 4
    if not in ch:
            ^
SyntaxError: invalid syntax
```

```
s=["phython","is","fun"]
join=' '.join(s)
print(join)
```

```
phython is fun
```

```
def sum_all(*range):
  s=0
  for i in range:
    s=s+i
  return s
print(sum_all(12,34,5,6,8,34,4))
```

```
103
```

```
def sum_all(*range):
  s=0
  for i in range:
    if i%5==0:
     s=s+i
  return s
print(sum_all(12,34,45,5,8,34,4))
```

```
50
```

```
def data(**args):
  return args ['a']
data(a=90,b=34)
```

```
90
```

```
def data(**divya):
  divya["a"]=divya["b"]
  print(divya)
data(a=12,b=23)
```

```
{'a': 23, 'b': 23}
```

```
def data(**args):
  print(args)
  print(type(args))
data(a=12,b=[12,45])
```

```
{'a': 12, 'b': [12, 45]}
<class 'dict'>
```

Start coding or generate with AI.

```
def f1(a=90):
    a=13
    print(a)
f1(3)
```

```
13
```

```
def f2(a=34,b=45):
   a=23
   print(a)
f2(2)
```

```
23
```

```
a=56
def d1():
  print(a)
d1()
print(a)
```

```
56
56
```

```
def diff(a,b):
  print(a-b)
  return 19
print(diff(12,34))
```

```
-22
19
```

```
def add(a,b):
  print(a+b)
  print("hello")
add(12,45)
```

```
57
hello
```

```
def f1():
  print("divya bansal")
  return 12
f1()
```

```
divya bansal
12
```

```
def f1():
  print("diva")
f1()
```

```
diva
```

```
def f1(x=20):
  print(x*9)
f1(34)
```

```
306
```

```
def f1(x=2,y=0,z=9):
  print(x*5)
f1(7)
```

```
35
```

```
def ff(x,y,z):
  print(x,y,z)
ff(x=1,y=3,z=4)
```

```
1 3 4
```

```
def divya():
  print("divya bansal")
  return 19
  print("no divya")
print(divya())
```

```
divya bansal
19
```

```
def f1(a,b):
  print(a-b)
  return 20
print(f1(12,34))
```

```
-22
20
```

```
name="divya"
print("my name is",name)
```

```
my name is divya
```

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

```
hello world
hello world
hello world
```

```
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

```
data="divya basanl"
print("from dholpu",data)
```

```
bansal divya divya basanl
```

```
def sum_call(*data):
  sum=0
  for i in data:
    s=s+i
  print(sum)
sum_call(1,3,4,56,34)
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
/tmp/ipython-input-3783630176.py in <cell line: 0>()
      4     s=s+i
      5   print(sum)
----> 6 sum_call(1,3,4,56,34)

/tmp/ipython-input-3783630176.py in sum_call(*data)
      2   sum=0
      3   for i in data:
----> 4     s=s+i
      5   print(sum)
      6 sum_call(1,3,4,56,34)

UnboundLocalError: cannot access local variable 's' where it is not associated with a
value
```

```
def sum_call(*data):
  sum=1
  for i in data:
    sum=sum*i
  print(sum)
sum_call(23,456,6,23,456,5445)
```

```
3593639364480
```

```
Start coding or generate with AI.
```

```
for i in range(1,6):
  for j in range()
```

```
for i in range(1,6):
  for j in range(1,6):
    if i==j or i==5 or j==1 or j==5:
      print("*",end=" ")
    else:
      print(" ",end=" ")
  print()
```

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

```
for i in range(1,6):
  for j in range(1,5):
    print(i,j)
  print("sdfgfh",end=" ")
```

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

```
for row in range(1,6):
  for col in range(row):
   print("*",end=" ")
  print()
```

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

```
for i in range(6,0,-1):
  for j in range(i):
   print("*",end="  ")
  print("  ")
```

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

```
for i in range(1,6):
  for j in range(6-i):
   print(" " ,end=" ")
  for k in range(i):
```

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

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

```
for i in range(1,6):
  for j in range(6,0,-1):
    if j>i:
      print(" ",end=" ")
    else:
      print("*",end=" ")
  print("  ")
```

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

```
for i in range(1,6):
  for j in range(6,0,-1):
    if j>i:
      print(" ",end=" ")
    if j<i:
      print(' dcdf',end=" ")
    else:
      print("*",end=" ")
  print("  ")
```

```
  *   *   *   *   * *
  *   *   *   * *  dcdf
  *   *   * *  dcdf  dcdf
  *   * *  dcdf  dcdf  dcdf
  * *  dcdf  dcdf  dcdf  dcdf
```

```
Start coding or generate with AI.
```

```
data="regex"
upper_count=0
lower_count=0
for i in data:
  if i.isupper():
    upper_count+=1
  elif i.islower():
    lower_count+=1
    print("upper_count")
  print("lower_count")
```

```
data="divya bansal"
for i in range(len(data)-1,-1,-1):
   print(data[i],end=" ")
```

```
l a s n a b   a y v i d
```

```
data="dfgklnfjksdbbfl"
remove_duplicates=" "
for i in data:
  if i not in remove_duplicates:
    remove_duplicates+=i
print(remove_duplicates)
```

```
 dfgklnjsb
```

```
data="welcome to regex"
print(data.count("d"))
```

```
0
```

```
Start coding or generate with AI.
```

```
data="divya"
reverse=' '
for i in range(len(data)-1,-1,-1):
  reverse+=(data[i])
if reverse==data:
  print("palindrome")
else:
  print("not palindrome")
```

```
not palindrome
```

```
data="REGEx"
a="a"
print(a.upper())
```

```
A
```

```
data="rErFE"
print(data.lower())
print(data.upper)
print(data.count())
```

```
rerfe
<built-in method upper of str object at 0x7faa764bd4b0>
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-821953608.py in <cell line: 0>()
      2 print(data.lower())
      3 print(data.upper)
----> 4 print(data.count())

TypeError: count() takes at least 1 argument (0 given)
```

```
a=19
new=str(a)
print(new[::-1])
```

```
91
```

```
a=19
num=0
while a>0:
  num=num*10+a%10
  a=a//10
print(num)
```

```
91
```

```
s="divya bansal"
print(list(s))
```

```
['d', 'i', 'v', 'y', 'a', ' ', 'b', 'a', 'n', 's', 'a', 'l']
```

```
def function(a,b):
 print(a+b)
function(10,56)
```

```
66
```

```
def divya():
  print("my name is divya bansal")
  print("i am from dholpur")
  print("i am studying in btech in govt.college")
divya()
```

```
my name is divya bansal
i am from dholpur
i am studying in btech in govt.college
```

```
def function(a,b):
  print(a+b)
function(12,45)
function(45,8)
```

```
57
53
```

```
def f():
   print("hello")
   return 10
   print("bye")
f()
```

```
hello
10
```

```
def divya():
  print("i am the best student in my college for bharatpur ")
  return 56
divya()
```

```
i am the best student in my college for bharatpur
56
```

```
def f1():
  print("hello")
  return 18
```

```
  print("bye")
f1()
```

```
hello
18
```

```
def f1():
  print("hello")
  return 56
  print("bye")
  a=f1
  print(f1(a))
```

```
def add(a,b):
  return(a+b)
print(add(90,34))
```

```
124
```

```
def f1(x=6):
   print(x*5)
f1(6)
```

```
30
```

```
def f(x,z,y):
  print(23,45,67)
f(x=1,y=2,z=3)
```

```
23 45 67
```

```
def f1():
  print("hello")
  return 56
  print("bye")
  a=f1
f1()
```

```
hello
56
```

```
a=10
count=0
for i in range(1,11):
  if a%i==0:
    print(a%i,"value of i")
    count=count+i
print(count)
```

```
0 value of i
0 value of i
0 value of i
0 value of i
18
```

```
a=int(input("enter"))
for i in range(2,a):
```

```
  if a%i==0:
    print("not a prime number")
  else:
    print("prime a number")
```

```
sum=0
for i in range(1,109):
  sum=sum+i
print(sum)
```

```
108
```

```
for i in range(10,1,-2):
  print(i,end=" ")
```

```
10 8 6 4 2
```

```
count=0
for i in range(6,11):
  count=count+1
print("count value",i)
```

```
count value 10
```

```
sum=0
for i in range(1,11):
  sum=sum+i
print("sum value",sum)
```

```
sum value 55
```

```
sum=0
for i in range(1,99):
  sum=sum+i
print("sum value",sum)
```

```
sum value 4851
```

```
count=0
for i in range(97,35,-1):
  if i%2==0:
    count=count+1
print("count value",count)
```

```
count value 31
```

Double-click (or enter) to edit

```
for i in range(97,56,-1):
  print(i,end=' ')
```

```
97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69
```

```
for i in range(1,11):
  if i==3 or i==5:
    print(i)
```

```
3
5
```

```
for i in range(1,11):
  if i%3==0 or i%4==0:
    print(i,end=" ")
```

```
3 4 6 8 9
```

```
for i in range(1,11):
  if i%3==0:
    print(i,end=" ")
```

```
3 6 9
```

```
for i in range(1,101):
  if i%2==0 or i%5==0:
    print(i,end=" ")
```

```
2 4 5 6 8 10 12 14 15 16 18 20 22 24 25 26 28 30 32 34 35 36 38 40 42 44 45 46 48 50 5
```

```
for i in range(91,20,-1):
  if i%2==0 or i%5==0:
    print(i,end=" ")
```

```
90 88 86 85 84 82 80 78 76 75 74 72 70 68 66 65 64 62 60 58 56 55 54 52 50 48 46 45 44
```

```
for i in range(1,11):
  if i%2==0:
    print(i,end=" ")
```

```
2 4 6 8 10
```

```
for i in range(1,6):
  i=i+2
  print(i,end=" ")
```

```
3 4 5 6 7
```

```
password=("set the password")
check=input("enter the password")
if password==check:
  print("success")
else:
  check=input("enter the password for second")
if password==check:
    print("success")
else:
    check=input("enter the password for third")
if password==check:
      print("success")
```

```
else:
      print("account locked")
```

```
enter the password123
enter the password for second123
```

```
data='hello divya'
for char in data:
  print(char,end='')
  if char==' ':
    print()
```

```
hello
divya
```

```
data='divya'
for char in data:
  print(char,end=' ')
  if char==' ':
   print()
```

```
d i v y a
```

```
data={'a':45,'b':34,'c':56}
for i in data:
656  print(i,data[])
```

```
a 45
b 34
c 56
```

```
def divya():

  print("dfd")
  print("divya")
divya()
```

```
dfd
divya
```

```
x=('a':34),('b':56),('c':45)
for a,b in x:
  print(a,b)
```

```
  File "/tmp/ipython-input-4284568224.py", line 1
    x=('a':34),('b':56),('c':45)
          ^
SyntaxError: invalid syntax
```

```
data=[1,23,455,]
print(data[::-1])
```

```
[455, 23, 1]
```

```
data=tuple(range(20,0,-1))
print(data)
```

```
(20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)
```

```
data=set(range(20,0,-1))
print(data)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
```

```
data=list(range(4,41,4))
print(data)
```

```
[4, 8, 12, 16, 20, 24, 28, 32, 36, 40]
```

```
data=list(range(40))
print(data[::-1])
```

```
[39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 1
```

```
data='my name is divya'
print(data[0:14:4])
```

```
maii
```

```
data=' 3wfghj,jhgsafgh'
print(data[17:0:-1])
```

```
hgfasghj,jhgfw3
```

```
data='divya bansal'
start=int(input("enter the start"))
stop=int(input("enter the stop"))
print(start,stop)
```

```
enter the start2
enter the stop7
2 7
```

```
data='welcome to regex'
print(data.split("e")[::1])
```

```
['w', 'lcom', ' to r', 'g', 'x']
```

```
data="welcome to regex"
print(data.capitalize())
print(data.lower())
print(data.title())
```

```
Welcome to regex
welcome to regex
Welcome To Regex
```

```
Start coding or generate with AI.
```

```
n1=int(input("enter the value"))
if n1>=5:
  if n1<=10:
    print("greater then 5 and lessthen or equal to 10")
  elif n1<15:
    print("greater then 10 and lessthen or equal to 15")
  else:
    print("greater then 16")
```

```
enter the value30
greater then 16
```

```
name='regex'
print(f'my institude name is{name}')
```

```
my institude name isregex
```

```
data='divya'
print(f'my name is {data}')
```

```
my name is divya
```

```
a=17
b=27
print(bin(a))
print(bin(b))
print(a^b)
```

```
0b10001
0b11011
10
```

```
a=23
b=21
print(bin(a))
print(bin(b))
print(a^b)
```

```
0b10111
0b10101
2
```

```
a=int(input("enter 1"))
b=int(input("enter 2"))
print(a+b)
```

```
practical=int(input("enter 1"))
theory=int(input("enter 2"))
total=practical+theory
if total>=33:
  print("pass")
else:
  print("fail")
```

```
enter 112
enter 223
```

```
pass
```

```
a=int(input("enter the value"))
if(a%5==0 or a%3==0):
  print("yes")
else:
  print("no")
```

```
a=17
b=27
print(bin(a))
print(bin(b))
print(13<<2)
```

```
0b10001
0b11011
52
```

```
a=17
b=27
print(bin(a))
print(bin(b))
print(a&b)
```

```
0b10001
0b11011
17
```

```
print(~5)
```

```
-6
```

```
Start coding or generate with AI.
```

```
5
```

```
print(2**2**3)
```

```
256
```

```
data=[1,3,9,2]
print(9 in data)
```

```
True
```

```
a=20
b=67
print(int (a+b))
```

```
87
```

```
a=int(input("enter the value"))
b=int(input("enter the value"))
c=int(input("enter the value"))
print(a+b+c)
```

```
a=10
b=10
add=a+b
print(a+b)
```

```
20
```

```
a=10
b=10
print(a-b)
```

```
0
```

```
a=10
b=10
print(a*b)
```

```
100
```

```
a=10
b=10
print(a/b)
```

```
1.0
```

```
a=10
b=10
print(a//b)
```

```
1
```

```
a=10
b=10
print(a**b)
```

```
10000000000
```

```
a=10
b=10
print(a%b)
```

```
0
```

```
data=20
divya="she is phython student"
print(data,divya)
```

```
20 she is phython student
```

```
data='''divya
bansal'''
print(data)
```

```
divya
bansal
```

```
data=[12,4,6,7,35,6]
even_count=0
odd_count2=0
for i in data:
  if(i%2==0 ):
    even_count+=1
  else:
    odd_count+=1
  print("even")
  print("odd")
```

```
even
odd
even
odd
even
odd
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2115653100.py in <cell line: 0>()
      6     even_count+=1
      7   else:
----> 8     odd_count+=1
      9   print("even")
     10   print("odd")

NameError: name 'odd_count' is not defined
```

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

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

```
for i in range(11,0,-1):
  print(i,end=" ")
```

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

```
for i in range(1,21):
  if i%2==0:
    print("even",i,end=" ")
  else:
    print("odd",i)
```

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

```
even 12 odd 13
even 14 odd 15
even 16 odd 17
even 18 odd 19
even 20
```

```
for i in range(1,55):
   if(i*5):
     print("divisible ",i)
```

```
divisible  1
divisible  2
divisible  3
divisible  4
divisible  5
divisible  6
divisible  7
divisible  8
divisible  9
divisible  10
divisible  11
divisible  12
divisible  13
divisible  14
divisible  15
divisible  16
divisible  17
divisible  18
divisible  19
divisible  20
divisible  21
divisible  22
divisible  23
divisible  24
divisible  25
divisible  26
divisible  27
divisible  28
divisible  29
divisible  30
divisible  31
divisible  32
divisible  33
divisible  34
divisible  35
divisible  36
divisible  37
divisible  38
divisible  39
divisible  40
divisible  41
divisible  42
divisible  43
divisible  44
divisible  45
divisible  46
divisible  47
divisible  48
divisible  49
divisible  50
divisible  51
divisible  52
divisible  53
```

```
divisible  54
```

```
Start coding or generate with AI.
```

```
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
<built-in function sum> total
```

```
data="112,3445,5,6667644,234,677653,233455676,2"
duplicates=""
```

```
for i in data:
   if i not in duplicates:
    duplicates+=i
print(duplicates)
```

```
12,34567
```

```
data=[12,34,34,56,6,33,43,6]
length=len(data)
for i in range(0,length):
  for j in range(0,length-1):
    if data[i]>data[j]:
       data[j],data[j+1]=data[j+1],data[j]
print(data)
e,sssss
```

```
[56, 34, 43, 34, 6, 33, 12, 6]
```

```
a=12
b=34
b,a=a,b
```

```
data=[12,44,,677,4,45,5,34,55,3,3,4]
maximum_no=[0]
for i in data:
  if i>max:max=i:
  print(maximum_no)
```

```
  File "/tmp/ipython-input-4213058671.py", line 1
    data=[12,44,,677,4,45,5,34,55,3,3,4]
                ^
SyntaxError: invalid syntax
```

```
data=[12,34,5,6,67,23,45]
second=[0]
for i in range(len(0)):
  if
```

```
for i in range(1,6):
 if(i==1 or i==n):
     print("*",end=' ')
else:
     print("*"+' '*(1-2)+'*')
```

```
*
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-5-216134079.py in <cell line: 0>()
      1 for i in range(1,6):
----> 2  if(i==1 or i==n):
      3      print("*",end=' ')
      4 else:
      5      print("*"+' '*(1-2)+'*')

NameError: name 'n' is not defined
```

```
data=(a ,34,56),(b ,56,34),(c ,45,34),(d ,32,12)
for i,j,k in data:
  print(i,j,k)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-4-3447917934.py in <cell line: 0>()
----> 1 data=(a ,34,56),(b ,56,34),(c ,45,34),(d ,32,12)
      2 for i,j,k in data:
      3   print(data)

NameError: name 'a' is not defined
```

```
data={'a':34,12:'twelve',True:344,4.6:[123,45]}
print(data[True])
```

```
344
```

```
data={'a':45,'b':45,'a':34}
print(data)
```

```
{'a': 34, 'b': 45}
```

```
keys=['a','b','c','d']
values=[12,34,56,23]
data=zip(keys,values)
print(list(data))
```

```
[('a', 12), ('b', 34), ('c', 56), ('d', 23)]
```

```
keys=['a','b','c','d']
values=[12,34,56,23]
data=zip(keys,values)
print(tuple(data))
```

```
(('a', 12), ('b', 34), ('c', 56), ('d', 23))
```

```
data="abacgaaha"
count=0
for i in data:
   if i=='a':
    count+=1
print(count)
```

```
5
```

```
data='dfhghjktygaswrtjjktfdafg'
keys=set(data)
values=[]
for i in keys:
  count=0
  for j in data:
    if i==j:
      count+=1
```

```
  values.append(count)
print(dict(zip(keys,values)))
```

```
{'w': 1, 'k': 2, 'r': 1, 'f': 3, 'd': 2, 'j': 3, 'y': 1, 'h': 2, 'a': 2, 's': 1, 'g':
```

```
students={'name':'divya','age':19,'course':'python'}
print(students['name'])
print(students['course'])
```

```
divya
python
```

```
data="divya"
print(data[::-1])
```

```
ayvid
```

```
data="divyabansal"
print(data[5])
```

```
b
```

```
a="i read maths"
print(a[2:10])
```

```
read mat
```

```
a="python"
  a[1]='s'
```

```
  File "/tmp/ipython-input-21-2331576152.py", line 2
    a[1]='s'
    ^
IndentationError: unexpected indent
```

```
a='12'
print(a*3)
```

```
121212
```

```
a=67
b=56
print(a%b)
```

```
11
```

```
a=45
b=456
print(a*b)
```

```
20520
```

```
a=45
b=456
print(a**b)
```

```
732666410557248774470864364285991585059711362790242016464253231842047266724411609305230
```

```
a=45
b=456
print(a/b)
```

```
0.09868421052631579
```

```
a=45
b=456
print(a//b)
```

```
0
```

```
a=45
b=456
print(a and b)
```

```
456
```

```
a=45
b=456
print(a or b)
```

```
45
```

```
a=45
b=456
print(a==b)
```

```
False
```

```
a=45
b=456
print(a>=b)
```

```
False
```

```
a=45
b=456
print(a<=b)
```

```
True
```

```
a=45
b=456
print(a<b)
```

```
True
```

```
a=45
b=456
```

```
print(a>b)
```

```
False
```

```
a=45
b=456
print(a!=b)
```

```
True
```

```
a=456
b=45
print(a==b)
```

```
False
```

```
a=45
b=456
print(a + b)
```

```
501
```

Start coding or generate with AI.

```
a=45
print(id(a))
b=67
print(id(b))
```

```
10759144
```