```
#Even or Odd
a = int(input('Enter a number'))
if a%2 == 0:
    print('Number is even')
else:
    print('Number is odd')



#Fibonnaci
a = 0
b = 1
c = a+b
print(a)
print(b)
print(c)
for i in range(3,11):
    a = b
    b = c
    c = a+b
    print(c)



#Reverse_Number
def rev(a):
    s = 0
    while(a!=0):
        e = a%10
        s = s*10+e
        a = a//10
        print(s)
a = int(input('Enter a number'))
```

```
rev(a)



#Armstrong
def armstrong(a):
    s = 0
    num = a
    while(a!=0):
        c = a%10
        s = s+(c*c*c)
        a = a//10
    if(s == num):
        print(num,'is a armstrong')
    else:
        print(num,'not armstrong')

a = int(input('Enter the number'))
armstrong(a)



#Palindrome
def palindrome(a):
    s = 0
    num = a
    while(a!=0):
        c = a%10
        s = s*10+c
        a = a//10
    if(s == num):
        print(s,'is palindrome')
    else:
```

```
        print(s,'not palindrome')

a = int(input('Enter the number'))
palindrome(a)



#Recurssive Factorial
def fact(a):
    if(a == 0):
        return 1
    else:
        return (a*fact(a-1))
a = int(input('Enter the number'))
print(fact(a))



#Vowel[normal code]
def check(a):
    s = "aeiou"
    found = False
    for char in a:
        if char in s:
            print('It is a vowel')
            found = True
        if not found:
            print('It is not vowel')
a = input('Enter the characters: ')
check(a.lower())



#Vowel[string code]
```

```
def check_vowels(a):
    vowels = "aeiou"
    found = False
    for char in a:
        if char in vowels:
            print(f'{char} is a vowel')
            found = True
    if not found:
        print("No vowels found")

a = input('Enter a string: ').lower()
check_vowels(a)



#Length of String
def findlen(a):
    print(len(a))
a = input('Enter the string: ')
findlen(a)



#Histogram
def histogram(a):
    for i in a:
        print('*'*i)
histogram([4,9,7])



#User input
def histogram(a):
    for i in a:
```

```
        print('*' * i)

user_input = input("Enter numbers separated by spaces: ")
numbers = list(map(int, user_input.split()))
histogram(numbers)



#panagram
def panagram(sen):
    a1 = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
    for i in sen:
        if i.lower() in a1:
            a1.remove(i.lower())
    if i not in a1:
        print('The sentence is a panagram')
    else:
        print('The sentence is not a panagram')
sen = "The quick brown fox jumps over the lazy dog"
panagram(sen)



#User-input
def panagram(sen):
    a1 = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
    for i in sen:
        if i.lower() in a1:
            a1.remove(i.lower())
    if not a1:
        print('The sentence is a pangram')
    else:
        print('The sentence is not a pangram')
```

```
user_sentence = input("Enter a sentence: ")
panagram(user_sentence)


#Common
def common(a1,a2):
    r = False
    for i in a1:
        if i in a2:
            r = True
    return r
a1 = [1,2,3,4]
a2 = [6,8,9,1]
print(common(a1,a2))


#Print list after removal of specific elements
a1 = [1,2,3,4,5,6]
a2 = []
for i,e in enumerate(a1):
    if i not in[0,2,4,5]:
        a2.append(e)
print(a2)


#Ascending Descending dictionary by value
import operator
d = {1: 1, 2: 2, 3: 3, 4: 13, 5: 4, 6: 8, 7: 2, 8: 11, 9: 0, 10: 23}
print("Original dictionary:", d)
```

```
t = sorted(d.items(), key=operator.itemgetter(1))
print("Ascending order:", t)

a = sorted(d.items(), key=operator.itemgetter(1), reverse=True)
print("Descending order:", a)



class Person:
    def __init__(self):
        self.__rollno = None
        self.__name = None

    def getdata(self, r, n):
        self.__rollno = r
        self.__name = n

    def putdata(self):
        print(self.__rollno)
        print(self.__name)

p = Person()
p.getdata(81, 'Shubham')
p.putdata()



#Private variables
class Check:
    x = 2
    __y = 10
```

```
    def disp(self):
        print('Welcome')

    def get_private_y(self):
        return self.__y

c = Check()

print(c.x)

print(c.get_private_y())

c.disp()



'''Create class student 2 members rollno & name
as public create methods to get data and put data'''
class Student:
    def __init__(self):
        self.rollno = None
        self.name = None

    def getdata(self, r, n):
        self.rollno = r
        self.name = n

    def putdata(self):
        print("Roll No:", self.rollno)
        print("Name:", self.name)

s = Student()
```

```
a = int(input('Enter your roll no: '))
b = input('Enter your name: ')
s.getdata(a, b)
s.putdata()



#fset,fget,fdel
class St:
    def __init__(self, t):
        self._t = t
    def get_t(self):
        return self._t
    def set_t(self, value):
        self._t = value
    def del_t(self):
        del self._t
    t = property(get_t, set_t, del_t)

s1 = St(4)
print(s1.t)
s1.t = 10
print(s1.t)
del s1.t



#Built-in class attributes
class St:
    s = 2
    __t = 3
```

```
print('st.s =', St.s)
print('st.__doc__ =', St.__doc__)
print('st.__name__ =', St.__name__)
print('st.__module__ =', St.__module__)
print('st.__dict__ =', St.__dict__)



#Constructor
class Student:
    def __init__(self):
        print('Inside the default constructor')

    def show(self):
        print('Inside the show method')

s = Student()
s.show()



#Parameterized Constructor
class Student:
    def __init__(self, r, n):
        self.__rollno = r
        self.__name = n
    def show(self):
        print(self.__rollno)
        print(self.__name)

s1 = Student(2, 'Shubham')
s1.show()
```

```
#Deconstructor
class ABC:
    def __init__(self, a):
        print('Hello:', a)
    def __del__(self):
        print('Object deleted')

a1 = ABC('Shubham')
a2 = ABC('Aditya')
del a1
del a2



#Arithmetic Operations in Class(add,sub,mul,div)
class Operation:
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def add(self):
        return self.a + self.b
    def sub(self):
        return self.a - self.b
    def mul(self):
        return self.a * self.b
    def div(self):
        return self.a / self.b

n1 = int(input('Enter the first number: '))
n2 = int(input('Enter the second number: '))
```

```
op1 = Operation(n1, n2)

print("Addition:", op1.add())
print("Subtraction:", op1.sub())
print("Multiplication:", op1.mul())
print("Division:", op1.div())



#Super_KEyword using Overriding
class b1:
    def show(self):
        print('Inside base class')
class derived(b1):
    def show(self):
        print('Inside derived class')
        super().show()
d = derived()
d.show()



#Super_Keyword using constructor
class Base:
    def __init__(self, a):
        self.num = a

class Derived(Base):
    def __init__(self, a, b):
        super().__init__(a)
        self.num2 = b
    def add(self):
        print(self.num + self.num2)
```

```
d = Derived(10, 20)
d.add()



#Static_methods
class example:
    def show(self):
        print('Inside show method')
    @staticmethod
    def display():
        print('Inside display method')
e = example()
e.show()
example.display()



#Polymorphism
def show(a):
    print('Value', a)
show(1)
show(1.0)
show('a')
show(True)



#Single_Inhertiance
class abc:
    def show(self):
        print('Inside abc')
class defg(abc):
    def disp(self):
```

```
        print('Inside defg')
d = defg()
d.disp()
d.show()



#Multiple_Inheritance
class b1:
    def setnum(self):
        self.a = 5
    def dispnum(self):
        print(self.a)
class b2:
    def setnum(self):
        self.b = 10
    def dispnum1(self):
        print(self.b)
class der(b1, b2):
    def add(self):
        return self.a + self.b
d = der()
d.setnum()
d.dispnum()
d2 = b2()
d2.setnum()
d2.dispnum1()



#New Screen
from tkinter import *
import tkinter
```

```
top=tkinter.Tk()
top.title("new screen")
top.mainloop()


#Widget name as anchors
from tkinter import *
import tkinter
top=tkinter.Tk()
b1=tkinter.Button(top,text="Flat",relief="flat")
b2=tkinter.Button(top,text="RAISED",relief="raised")
b1.pack()
b2.pack()
top.mainloop()


#Implementing widget as bitmap
from tkinter import *
import tkinter
top=tkinter.Tk()
b1=tkinter.Button(top,text="raised",bitmap="error")
b2=tkinter.Button(top,text="raised",bitmap="info")
b3=tkinter.Button(top,text="raised",bitmap="question")
b1.pack()
b2.pack()
b3.pack()
top.mainloop()


#Implementing Cursor
from tkinter import *
```

```
import tkinter
top=tkinter.Tk()
b1=tkinter.Button(top,text="Circle",relief="flat",cursor="circle")
b2=tkinter.Button(top,text="Plus",relief="raised",cursor="plus")
b1.pack()
b2.pack()
top.mainloop()


#Implementing lable widget
from tkinter import *
import tkinter

root = tkinter.Tk()
var = StringVar()
label = Label(root, textvariable=var, bg="red", fg="blue")
var.set("Python Variable Label")
label.pack()

root.mainloop()


#Message Box
from tkinter import *
from tkinter import messagebox

root = Tk()
var=StringVar()
label=Message(root,textvariable=var,relief="raised")
var.set("hello world")
label.pack()
```

```
root.mainloop()



#Implement checkbutton widget
from tkinter import *
import tkinter
top=tkinter.Tk()
c1=Checkbutton(top,text="music",onvalue=1,offvalue=0,height=5,width=20)
c2=Checkbutton(top,text="video",onvalue=1,offvalue=0,height=5,width=20)
c1.pack()
c2.pack()
top.mainloop()



#Radio Button
from tkinter import *

def sel():
   selection = "You selected the option: " + str(var.get())
   label.config(text=selection)

root = Tk()
var = IntVar()



r1 = Radiobutton(root, text="Coffee", variable=var, value=1, command=sel)
r1.pack(anchor=W)
r2 = Radiobutton(root, text="Juice", variable=var, value=2, command=sel)
r2.pack(anchor=W)
```

```
label = Label(root)
label.pack()

root.mainloop()



#Implement frame widget
from tkinter import *

root = Tk()
frame = Frame(root)
frame.pack()

bottomframe = Frame(root)
bottomframe.pack(side=BOTTOM)



greenbutton = Button(frame, text="green", fg="green")
greenbutton.pack(side=LEFT)

redbutton = Button(frame, text="red", fg="red")
redbutton.pack(side=LEFT)

root.mainloop()



import tkinter

# Create the main window
top = tkinter.Tk()
```

```
# Create a canvas with a blue background
c = tkinter.Canvas(top, bg="blue", height=250, width=300)
c.pack()

# Define the coordinates for the arc
coord = 10, 50, 240, 210

# Create an arc with specified start and extent
arc = c.create_arc(coord, start=0, extent=150, fill="red")

# Start the Tkinter main loop
top.mainloop()



#menubutton
from tkinter import *



top = Tk()



mb = Menubutton(top, text="File", relief="raised")
mb.grid()



mb.menu = Menu(mb, tearoff=0)
mb["menu"] = mb.menu



mb.menu.add_checkbutton(label="Open")
mb.menu.add_checkbutton(label="Close")
```

```
mb.pack()

top.mainloop()



#import menu
import tkinter as tk
from tkinter import Menu, Toplevel, Button

def donating():
    filewin = Toplevel(root)
    button = Button(filewin, text="Donating Button")
    button.pack()

root = tk.Tk()
menubar = Menu(root)

filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=donating)
filemenu.add_command(label="Open", command=donating)
filemenu.add_command(label="Save", command=donating)
menubar.add_cascade(label="File", menu=filemenu)

root.config(menu=menubar)
root.mainloop()



#listbox
from tkinter import *
```

```
top = Tk()
listbox = Listbox(top)
listbox.insert(1, "Python")
listbox.insert(2, "Data Structures")
listbox.insert(3, "Algorithms")
listbox.insert(4, "Operating Systems")
listbox.pack()

top.mainloop()
```