**Data type**

**String:**

name = "irfan"  
print(name)

output: Irfan

name = "irfan"  
print(type(name))

output: str

name = "irfan"  
print("My name is "+name)

output: My name is Irfan

first\_name = "Irfan"  
middle\_name="Shah"  
last\_name ="Mayeen"  
full\_name =first\_name +" " +middle\_name +" "+last\_name  
print(full\_name)

output: Irfan Shah Mayeen

**Integer:**

year = 21  
printf(year)

output:21

year = 21  
print(type(year))

output: int

year = 21  
year=year+4  
print(year)

output:25

Now the problem is

year = 21  
printf("My age is "+year)

output: Errror.

Solution: typecasting variable

year = 21  
print("My age is "+str(year))

Output: My age is 21

**Float:**

area = 15.5  
print(area)

Output:15.5

area = 15.5  
print(type(area))

Output:float

**Boolean:**

ans = True  
print(ans)

Output:true

ans = True  
print(type(ans))

Output: bool

ans = True  
print("Are you human? "+str(ans))

Output: Are you human?t true

Multiple Variable in single line

name , age , area,ans ="Irfan",21,15.5,True

Multiple Variable Contain same value

irfan=sibbir=miraz=22

**String**

Length

Length =len(String)

Find index

name ="Irfan Shah"  
print(name.find("I"))

Output: 0

name ="Irfan Shah"  
print(name.find("r"))

Output: 1

name ="Irfan Shah"  
print(name.find("p"))

Output: -1

Capitalize( first letter will be upper case)

name ="irfan shah"  
print(name.capitalize())

Output: Irfan shah

Upper and Lowe case letter( it will not change the string just print)

name ="Irfan Shah"  
print(name.lower())  
print(name.upper())

Output: irfan shah

IRFAN SHAH

Digit or not(full string)

name ="Irfan Shah"  
print(name.isdigit())

output:False

name ="1236"  
print(name.isdigit())

output: True

name ="123 6"  
print(name.isdigit())

output:False ( because here a space after 3. It is not a digit)

Alphabet or not(full string)

name="Irfan shah"  
print(name.isalpha())

output: false ( because space after n)

name="Irfanshah"  
print(name.isalpha())

output:True

name="123"  
print(name.isalpha())

Output:False

Count frequency

name ="aaabbbbccc"

print(name.count("a"))

output:3 .(here a arrives 3 time)

name ="aaabbbbccc"

print(name.count("b"))

output:4

Replace

name ="aaaccc"

print(name.replace("a","b"))

output: bbbccc ( all a replaced by b)

String Print n times

name ="Bangladesh"

print(name\*3)

output: BangladeshBangladeshBangladesh ( bangladesh print 3 times)

name ="Irfan Shah"

print(name)

print(len(name))

print(name.find("f"))

print(name.capitalize())

print(name.lower())

print(name.upper())

print(name.isdigit())

print(name.isalpha())

print(name.count("a"))

print(name\*3)

**Typecasting**

Type casting = convert the data type of a value to another data type

x=1 #int  
y=2.0 #float  
z="3" #str  
  
print(int(x))  
print(int(y))  
print(int(z))

Output: 1

2

3

y=2.0 #float  
z="3" #str

y=int(y)  
z=int(z)  
print(y)  
print(z)

Output: 1

2

3

**User input**

Input();

name = input("what is your name:")  
age=input("How old are you: ")  
print(name)  
print(age)

output: what is your name:irfan

How old are you: 18

irfan

18

age=input("How old are you: ")  
age=age+1

print(age)

Output: Error

Solution: typecasting variable

age=int(input("How old are you: "))

**Math Functions**

import math  
pi=3.14  
print(round(pi))  
print(math.ceil(pi))  
print(math.floor(pi))  
print(abs(pi))  
print(pow(3,2))  
print(math.sqrt(64))

3

4

3

3.14

9

8.0

maximum && minimum

x=2  
y=3  
z=6  
print(max(x,y,z))

print(min(x,y,z)

Output:6 (maximum number)

2 (minimum number)

**String Slicing**

name ="Irfan Shah"  
first\_name= name[0:5]  
print(first\_name)

output: Irfan

name ="Irfan Shah"  
last\_name=name[6:10]  
print(last\_name)

output: shah

name ="Irfan Shah"  
funky\_name=name[0:8:2]  
print(funky\_name)

Output : IfnSa (print 1 skip character)

name ="Irfan Shah"

funky\_name=name[0:10:3]  
print(funky\_name)

Output: IaSh

name ="Irfan Shah"

funky\_name=name[::2]  
print(funky\_name)

output: IfnSa ( here name[ : :🡪 python assume first empty ,last empty

Reverse string

name ="Irfan Shah"

reverse\_name= name[::-1]  
print(reverse\_name)

Output: hahS nafrI

Slice() function

website = "http://google.com"  
slice = slice(7,-4)  
print(website[slice])

Output: google

website = "http://google.com"  
website2= "http://facebook.com"  
slice = slice(7,-4)  
print(website[slice])  
print(website2[slice])

Output: google

facebook

**Else if**

age =int(input("Enter your age:"))  
if age>=18 :  
 print("Your are an adult")  
else:  
 print("you are a child")

age =int(input("Enter your age:"))  
if age>=18 :  
 print("Your are an adult")  
elif age<0:  
 print("You have not born")  
else:  
 print("you are a child")

age =int(input("Enter your age:"))  
if age>=18 :  
 print("Your are an adult")  
elif age==12:  
 print("you are 12 years old")  
elif age<0:  
 print("You have not born")  
else:  
 print("you are a child"

Conditional operator

temp=float(input("Enter temperature"))  
  
if temp >= 0 and temp <= 30:  
 print("Weather is good")  
 print("Go outside!")  
elif temp<0 or temp>30:  
 print("the weather bad today")  
 print("Don't Go!")

**Loops**

While loop

name =""  
while len(name)==0:  
 name = input("Enter your name: ")  
print("Hello "+name)

or

name = None  
while len(name)==0:  
 name = input("Enter your name: ")  
print("Hello "+name)

For loop

for i in range(5):  
 print(i)

Output:

0

1

2

3

4

for i in range (2,5):  
 print(i)

output:

2

3

4

for i in range (2,10,2):  
 print(i)

output:

2

4

6

8

for i in range (1,10,2):  
 print(i)

output:

1

3

5

7

9

for i in "Irfan":  
 print(i)

outputs:

I

r

f

a

n

## Timer

for second in range(5,0,-1):  
 print(second)  
 time.sleep(1)  
print("Happy Birthday!")

Nested loop

row = int(input("Enter rows:"))  
col = int(input("Enter colums:"))  
symbol = input("Enter a symbol:")  
for i in range(row):  
 for j in range(col):  
 print(symbol,end="")  
 print()

outputs:

Enter rows:3

Enter colums:3

Enter a symbol:\*

\*\*\*

\*\*\*

\*\*\*

For control’

#break

while True:  
 name = input("Enter your name:")  
 if name!="":  
 break

#continue

phone\_number="0199-334-3853"  
for i in phone\_number:  
 if i=="-":  
 continue  
 print(i,end="")

#pass

for i in range(1,10):  
 if i== 5:  
 pass  
 else:  
 print(i,end=" ")

outputs: 1 2 3 4 6 7 8 9

**List**

# list = used to store multiple items in a asingle variable

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
print(food)

outputs: ['pizza', 'burger', 'chicken', 'hotdog']

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
print(food[0])  
print(food[1])  
print(food[2])

outputs:

pizza

burger

chicken

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
food[0]="rool"  
print(food[0])

outputs: rool

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
for x in food:  
 print(x)

outputs:

pizza

burger

chicken

hot dog

#add element in the last

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
food.append("ice cream")  
for x in food:  
 print(x)

outputs:

pizza

burger

chicken

ice cream

#remove element from last

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
food.pop()  
for x in food:  
 print(x)

outputs:

pizza

burger

chicken

#insert given index

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
food.insert(0,"cake")  
for x in food:  
 print(x)

outputs:

cake

pizza

burger

chicken

ice cream

#sort

food = ["pizza" , "burger" , "chicken" ,"hotdog"]  
food.sort()  
for x in food:  
 print(x)

outputs:

burger

chicken

hotdog

roll

#clear list

food.clear()

**2D List**

drinks = ["coffe","tea","soda"]  
dinner= ["pizza","hotdog","kacchi"]  
desert = ["cake","ice cream"]  
food =[drinks,dinner,desert]  
print(food)

outputs: [['coffe', 'tea', 'soda'], ['pizza', 'hotdog', 'kacchi'], ['cake', 'ice cream']]

drinks = ["coffe","tea","soda"]  
dinner= ["pizza","hotdog","kacchi"]  
desert = ["cake","ice cream"]  
food =[drinks,dinner,desert]  
print(food[0])

outputs : ['coffe', 'tea', 'soda']

drinks = ["coffe","tea","soda"]  
dinner= ["pizza","hotdog","kacchi"]  
desert = ["cake","ice cream"]  
food =[drinks,dinner,desert]  
print(food[0])  
print(food[1])  
print(food[2])

outputs :

['coffe', 'tea', 'soda']

['pizza', 'hotdog', 'kacchi']

['cake', 'ice cream']

drinks = ["coffe","tea","soda"]  
dinner= ["pizza","hotdog","kacchi"]  
desert = ["cake","ice cream"]  
food =[drinks,dinner,desert]  
print(food[0][0])

outputs : coffe

drinks = ["coffe","tea","soda"]  
dinner= ["pizza","hotdog","kacchi"]  
desert = ["cake","ice cream"]  
food =[drinks,dinner,desert]  
print(food[1][2])

outputs : kacchi

**Tuple**

#student (name,age,gender  
student = ("Irfan",21,"male")  
print(student.count("Irfan"))

Outputs: 1 ( because here only one Irfan)

student = ("Irfan",21,"male")  
print(student.index("male"))

Outputs: 2 ( because male index number is 2)

student = ("Irfan",21,"male")  
for x in student:  
 print(x,end=" ")

outputs : Irfan 21 male

student = ("Irfan",21,"male")  
if "Irfan" in student:  
 print("Irfan is here")

outputs : Irfan is here

#problems of tuple

Tuple are unchanble

Student[0]= “Rahim” -> it is not allowed

student = ("Irfan",21,"male")  
print(student[0])  
print(student[1])

outputs : Irfan 21

#can be create multiple tuple

teachers = (("Ishan",26,"male"),("ashik",35,"male"))  
print(teachers[0])

outputs : ('Ishan', 26, 'male')

teachers = (("Ishan",26,"male"),("ashik",35,"male"))  
print(teachers[0])  
print(teachers[1][1])

outputs : 35

**Set**

food = {"pizza","burger","kacchi"}  
for x in food:  
 print(x,end=" ")

outputs : pizza burger kacchi

#if we add duplicate , it will print only one

subject ={"English","Bangla","computer","Bangla","Bangla"}  
for x in subject:  
 print(x,end=" ")

outputs : computer Bangla English 🡪(herew Bangla is more than one but print only one time)

food = {"pizza","burger","kacchi"}  
food.add("rool")  
food.remove("burger")  
food.clear()

#add two ser

game = {"cricket","football","hockey"}  
team = {"Bangladesh","India","Argentina"}  
game.update(team)  
for x in game:  
 print(x,end=" ")

outputs: cricket Argentina India hockey Bangladesh football 🡪 (random)

game = {"cricket","football","hockey"}  
team = {"Bangladesh","India","Argentina"}  
field = game.union(team)  
for x in field:  
 print(x,end=" ")

outputs: India football hockey Bangladesh Argentina cricket🡪 (random)

game = {"cricket","football","hockey"}  
team = {"Bangladesh","India","Argentina"}  
field = game.union(team)  
print(field)

outputs: {'Argentina', 'cricket', 'football', 'hockey', 'Bangladesh', 'India'}

#difference

animal = {"Tiger","Lion","Hen"}  
bird = {"Doel","Hen","Duck"}  
print(animal.difference(bird))

output: {'Lion', 'Tiger'} 🡪 from animal hen removed

animal = {"Tiger","Lion","Hen"}  
bird = {"Doel","Hen","Duck"}  
print(bird.difference(animal))

output: {'Duck', 'Doel'} 🡪 from bird Hen removed

animal = {"Tiger","Lion","Hen"}  
bird = {"Doel","Hen","Duck"}  
print(bird.difference(bird))

output: set() 🡪 every element remove form bird …because difference with bird to bird

#intersection (commom)

animal = {"Tiger","Lion","Hen"}  
bird = {"Doel","Hen","Duck"}  
print(animal.intersection(bird))

output: {‘Hen’} 🡪 only Hen is both in set

animal = {"Tiger","Lion","Hen"}  
bird = {"Doel","Hen","Duck"}  
print(animal.intersection(animal))

output: {'Tiger', 'Lion', 'Hen'} 🡪 because every element is available in both set ,.. cause both set are same animal

**Dictionary**

#dictionary = A changeable , unordered collection of unique key: value pairs  
# Fast beacuse they are use hashing , allow us to access a value quickly

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}  
print(capital['Russia'])  
print(capital['Bangladesh'])

output: Moscow

Dhaka

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
print(capital['USA'])

output: Error 🡪 because USA is not in our dictionary

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}  
print(capital.get('Bangladesh'))  
print(capital.get('USA'))

output: Dhaka

None

#print keys

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}  
print(capital.keys())

output: dict\_keys(['Bangladesh', 'India', 'China', 'Russia'])

#print values

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}

print(capital.values())

outputs: dict\_values(['Dhaka', 'Dehli', 'Beijing', 'Moscow'])

#entire dictionary print

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}  
print(capital.items())

output: dict\_items([('Bangladesh', 'Dhaka'), ('India', 'Dehli'), ('China', 'Beijing'), ('Russia', 'Moscow')])

#or

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}  
for key,value in capital.items():  
 print(key,value)

output:

Bangladesh Dhaka

India Dehli

China Beijing

Russia Moscow

#update (add)

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}

capital.update({'Germany':'Berlin'})  
for key,value in capital.items():  
 print(key,value)

output:

Bangladesh Dhaka

India Dehli

China Beijing

Russia Moscow

Germany Berlin

#update(change)

apital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}

capital.update({'Bangladesh': 'Munshiganj'})  
for key,value in capital.items():  
 print(key,value)

output:

Bangladesh Munshiganj

India Dehli

China Beijing

Russia Moscow

#remove

capital={'Bangladesh':'Dhaka',  
 'India':'Dehli',  
 'China':'Beijing',  
 'Russia':'Moscow'}

capital.pop('India')  
for key,value in capital.items():

output:

Bangladesh Munshiganj

China Beijing

Russia Moscow

**Indexing**

name = "irfan"  
if(name[0].islower()):  
 name=name.capitalize()  
print(name)

output: Irfan

name = "irfan"  
first\_name= name[0:5].upper()  
print(first\_name)

output: IRFAN

name = "irfan Shah"

last\_name=name[6:10].lower()  
print(last\_name)

output: shah

name = "irfan Shah!"  
last\_character = name[-1]  
second\_last =name[-2]  
print(last\_character)  
print(second\_last)

output: !

h

**Function**

def hello():  
 print("hello!")  
  
hello()  
hello()

output: hello!

Hello!

#with parameter

def hello1(name):  
 print("hello "+name)  
 print("Nice to meet you")  
  
hello1("Irfan")

output: hello Irfan

Nice to meet you

def printName(first\_name,last\_name):  
 print("Hello "+first\_name+" "+last\_name)  
  
printName("Irfan","Shah")

output: Hello Irfan Shah

#return

def multiply(num1,num2):  
 result = num1\*num2  
 return result  
  
x=multiply(5,2)  
print(x)

print(multiply(5,3))

output: 10

15

**Keyword Argument**

#keyword arguments = argument preceded by an indentifier when we pass them to a function  
# The order of the arguments doesn't matter,unlike positional arguments  
# python knows the names of the arguments that our function receieves

def hello(first,middle ,last):  
 print("Hello "+first+" "+middle+" "+last)  
hello("Irfan","shah","Mayeen")

output: Hello Irfan Shah Mayeen

#but

def hello(first,middle ,last):  
 print("Hello "+first+" "+middle+" "+last)  
hello("Shah","Irfan","Mayeen")

output: Hello shah Irfan Mayeen 🡪 caller order matter.

##now solve it using keyword argument

def hello(first,middle ,last):  
 print("Hello "+first+" "+middle+" "+last)  
hello(middle="shah",first="Irfan",last="Mayeen")

output: Hello Irfan Shah Mayeen 🡪 pass argument with the same key as parameter

**Nested Function**

num = input("Enter a number :")  
num = float(num)  
num=abs(num)  
num=round(num)  
print(num)

#or

print(round(abs(float(input("Enter a number: ")))))

**Scope**

name= "Irfan"  
def display\_name():  
 name ="code" # local scope (available only inside this function  
 print(name)  
  
print(name)  
display\_name()

output: Irfan 🡪 from Global

code 🡪 from local 🡪ffrom function

**Args\***

#agrs = parameter that will pack all argument into a tuple  
# useful so that a function can accept a varying amount of arguments

def add(\*args):  
 sum=0  
 for i in args:  
 sum+=i  
 return sum  
print(add(1,2,3,4,5,6))

output: 21

def add(\*stuff):  
 sum=0;  
 for i in stuff:  
 sum+=i  
 return sum  
  
print(add(1,2,3,4,5,6))

output: 21

def add(\*stuff):  
 sum=0  
 stuff = list(stuff)  
 stuff[0]=0  
 for i in stuff:  
 sum+=i  
 return sum  
  
print(add(1,2,3,4,5,6))

output: 20 🡪 stuff[0] =1 changed stuff[0]=0

stuff = list(stuff)  
 stuff[0]=0  
 stuff[1]=0  
 for i in stuff:  
 sum+=i  
 return sum  
  
print(add(1,2,3,4,5,6))

output: 18 🡪 stuf[0] =1 changed to 0 , stuff[1] =2 changed to 0 . so total 3 will decrease

**Kwargs\***

def hello(first,last):  
 print("Hello! "+first+" "+last)  
  
hello(first="Irfan",middle="Shah",last="Mayeen" )

output: error 🡪 argument 3 , but function parameter 2

#solve using \*\*kwargs

def hello(\*\*kwargs):  
 print("Hello! "+kwargs['first']+" "+kwargs['last'])  
  
hello(first="Irfan",middle="Shah",last="Mayeen" )

output: Hello! Irfan Mayeen

#full list print

def hello(\*\*kwargs):  
 print("Hello",end=" ")  
 for key,value in kwargs.items():  
 print(value,end=" ")  
  
hello(title="Mr.",first="Irfan",middle="Shah",last="Mayeen")

output: Hello mr. Irfan Shah Mayeen

#if we do For loop like this

def hello(\*\*kwargs):  
 print("Hello",end=" ")  
 for value in kwargs.items():  
 print(value,end=" ")  
  
hello(title="Mr.",first="Irfan",middle="Shah",last="Mayeen")

output: Hello ('title', 'Mr.') ('first', 'Irfan') ('middle', 'Shah') ('last', 'Mayeen')

**format function**

animal = "cow"  
item = "moon"  
print("The "+animal+" jumped over the "+item)

output: The cow jumped over the moon

#uisng format method

print("The {} jumped over the {}".format("cow","moon"))

output: The cow jumped over the moon

#or

animal = "cow"  
item = "moon"  
print("The {} jumped over the {}".format(animal,item))

output: The cow jumped over the moon

#using index positional argument

animal = "cow"  
item = "moon"  
print("The {0} jumped over the {1}".format(animal,item))

output: The cow jumped over the moon

animal = "cow"  
item = "moon"  
print("The {1} jumped over the {0}".format(animal,item))

output: The moon jumped over the cow

print("The {3} jumped over the {2}".format("tiger","Lion","Moon","Cat"))

output: The Cat jumped over the Moon

#keyword argument

print("The {animal} jumped over the {item}".format(animal="lion",item="moon"))

output: The lion jumped over the moon

#using string

text = "The {} jupmed over the {}"  
print(text.format("Tiger","moon"))

output: The Tiger jumped over the moon

#space

name="Irfan"  
print("hello,my name is {}.Nice to meet you".format(name))  
print("hello,my name is {:20}. Nice to meet you".format(name))

output: hello,my name is Irfan.Nice to meet you

hello,my name is Irfan . Nice to meet you

name="Irfan"  
print("hello,my name is {}.Nice to meet you".format(name))  
print("hello,my name is {:20}. Nice to meet you".format(name))  
print("hello,my name is {:<20}. Nice to meet you".format(name)) #right  
print("hello,my name is {:>20}. Nice to meet you".format(name)) #left  
print("hello,my name is {:^20}. Nice to meet you".format(name)) #central

output:

hello,my name is Irfan.Nice to meet you

hello,my name is Irfan . Nice to meet you

hello,my name is Irfan . Nice to meet you

hello,my name is Irfan. Nice to meet you

hello,my name is Irfan . Nice to meet you

#number

number = 3.14159  
print("The number pi is {}".format(number))  
print("The number pi is {:.2f}".format(number))  
print("The number pi is {:.3f}".format(number))

output:

The number pi is 3.14159

The number pi is 3.14

The number pi is 3.142

number = 1000  
print("The number is {}".format(number))  
print("The number is {:,}".format(number))  
print("The number is {:b}".format(number)) #binary  
print("The number is {:o}".format(number))#ocatal  
print("The number is {:x}".format(number)) #hexa-decimal lowercase  
print("The number is {:X}".format(number)) #hexa-decimal uppercase  
print("The number is {:E}".format(number)) #sciencific notation

output:

The number is 1000

The number is 1,000

The number is 1111101000

The number is 1750

The number is 3e8

The number is 3E8

The number is 1.000000E+03

**Random Number**

import random  
x = random.randint(1,6) #integer number  
print(x)  
y = random.random() #floating number  
print(y)

#ramdom choice from a list

myList =['rock','paper','scissors']  
z= random.choice(myList)  
print(z)

#random suffle

cards=[1,2,3,4,5,6,7,8,9,"J","Q","K","A"]  
random.shuffle(cards)  
print(cards)

**Exception Handle**

print(5/0)

it is math maticaly impossible … so it is an exception

numerator = int(input("Enter a number to divide: "))  
denominator = int (input("Enter a numbr to divide by:"))  
result = numerator / denominator  
print(result)

if user input denominator =0 .. it will exception and stops the programso

solution:

try:  
 numerator = int(input("Enter a number to divide: "))  
 denominator = int (input("Enter a numbr to divide by:"))  
 result = numerator / denominator  
 print(result)  
except Exception:  
 print("You enter denominator as 0")

try:  
 numerator = int(input("Enter a number to divide: "))  
 denominator = int (input("Enter a numbr to divide by:"))  
 result = numerator / denominator  
 print(result)  
except ZeroDivisionError as e:  
 print(e)  
 print("you can't divided by zero! idiot!")  
except ValueError as e:  
 print(e)  
 print("Enter only number plz")  
except Exception as e:  
 print(e)  
 print("Something went wronng:")  
else:  
 print(result)  
finally:  
 print("this is alyaws execute")

**File**

**Import os**

import os  
path ="C:\\Users\\Admin\\PycharmProjects\\File\\test.txt"  
if os.path.exists(path):  
 print("that location exists")  
 if os.path.isfile(path):  
 print("That is a file")  
else:  
 print("That location doesn't exists")  
  
location ="C:\\Users\\Admin\\PycharmProjects\\File\\Folder"  
if os.path.exists(location):  
 print("that location exists")  
 if os.path.isdir(location):  
 print("That is a Directory/folder")  
else:  
 print("That location doesn't exists")

output:

that location exists

That is a file

that location exists

That is a Directory/folder

#read a file

#if the file inside the project folder

with open('test.txt') as file:  
 print(file.read())

#if the file is not in the project folder

with open('C:\\Users\\Admin\\PycharmProjects\\File\\test.txt') as file:  
 print(file.read())

#finally

try:  
 with open('test.txt') as file:  
 print(file.read())  
except FileNotFoundError  
 print("File was not found")

#write a file

text = "Irfan"  
with open('test.txt','w') as file:  
 file.write(text)

#but the problem is , the text it overwritten.. / previous text delete

Solve : use append

with open('test.txt','a') as file:

**copy file**

#copyfile()= copies contents of a file  
#copy() = copyfile() + permission mode+destination ca be a directory  
#copy2()= copy() + copies metadata (fie's creation and modification times)  
  
import shutil  
shutil.copfile('test.txt','copy.txt') #(source , destination)

n('test.txt','a') as file:

**move file**

import os  
source ="text.txt"  
destination="C:\\Users\\Admin\\Desktop\\text.txt"  
try:  
 if os.path.exists(destination):  
 print("There is already a file there")  
 else:  
 os.replace(source,destination)  
 print(source+" was moved")  
  
except FileNotFoundError:  
 print(source+" "+"was not found")

**Delete file**

import os  
os.remove('test.txt')

#or

import os  
os.remove('test.txt')  
  
#or  
  
path = "test.txt"  
try:  
 os.remove(path)  
 print("File removed")  
except FileNotFoundError:  
 print("File not found")

**Modules**

#1st python file (message)

def hello():  
 print("Hello!Have a nice day")  
def bye():  
 print("Bye!Have a wonderful day")

#2nd python file (main)

import message  
  
message.hello()  
message.bye()

output: Hello!Have a nice day

Bye!Have a wonderful day

#or

#2nd python file (main)

import message as msg  
msg.hello()

#or

from message import hello  
hello()

#or

from message import hello,bye  
hello()  
bye()

#import all

from message import \*

**Object Oriented**

**car file:**

class Car:  
  
 def \_\_init\_\_(self,make,model,year,color): #constructor  
 self.make = make  
 self.model = model  
 self.year = year  
 self.color = color  
 def drive(self):  
 print("this "+self.model+" car is driving ")  
 def stop(self):  
 print("this car is stop")

**main file:**

from car import Car  
  
car1= Car("Chevy","Corvette",2024,"red")  
car2= Car("ford","mustang",2022,"blue")  
  
print(car1.make)  
print(car1.model)  
print(car1.year)  
print(car1.color)  
  
car1.drive()  
car1.stop()  
  
print(car2.make)  
print(car2.model)  
print(car2.year)  
print(car2.color)  
car2.drive()

**class variable**

**car file:**

class Car:  
 wheels =4 #class variable  
  
 def \_\_init\_\_(self,make,model,year,color):  
 self.make = make #instance variable  
 self.model = model #instance variable  
 self.year = year #instance variable  
 self.color = color #instance variable  
 def drive(self):  
 print("this "+self.model+" car is driving ")  
 def stop(self):  
 print("this car is stop")

**main file**

from car import Car

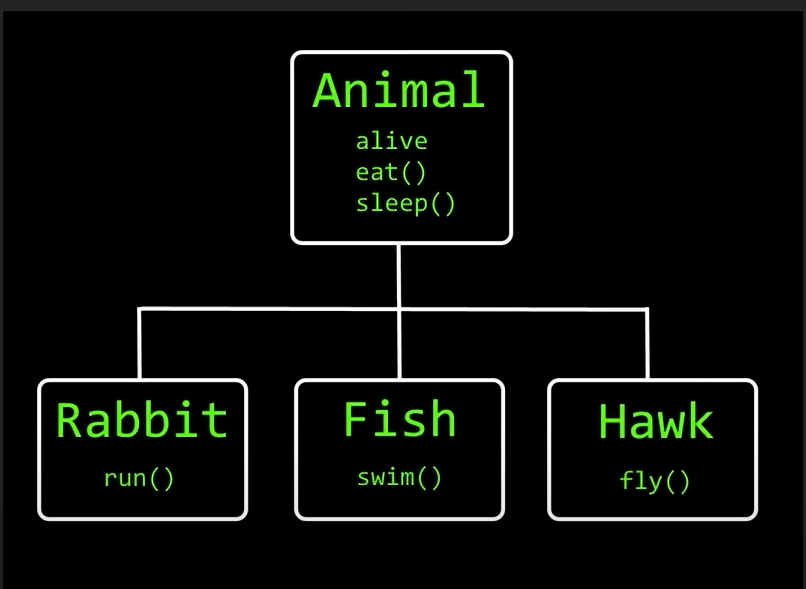
car1= Car("Chevy","Corvette",2024,"red")  
car2= Car("ford","mustang",2022,"blue")  
car3 =Car("motorcycle","yamaha",2022,"black")

print(car1.make)  
print(car1.model)  
print(car1.year)  
print(car1.color)

car1.drive()  
car1.stop()

print(car2.make)  
print(car2.model)  
print(car2.year)  
print(car2.color)  
car2.drive()  
print(car1.wheels)  
print(car2.wheels)  
car3.wheels=2  
print(car3.wheels) #it will change only car3 wheels  
car.wheels=2 # it will change for all wheel

**Inheritance**

****

class Animal:  
 alive = True  
 def eat(self):  
 print("This animal is eating")  
 def sleep(self):  
 print("This animal is sleeping")  
class Rabbit(Animal): #Rabbit child class and,Animal parent class  
 def run(self):  
 print("The rabbit is running")  
class Fish(Animal):  
 def swim(self):  
 print("This fish is swimming")  
class Hawk(Animal):  
 def fly(self):  
 print("This hawk is flying")  
rabbit = Rabbit()  
fish = Fish()  
hawk=Hawk()  
  
print(rabbit.alive)  
fish.eat()  
hawk.sleep()  
  
rabbit.run()  
fish.swim()  
hawk.fly()

**Multi level inheritance**

#multi level inheritance = when a derived (child) class inherits another derived (child)class  
  
class Organism:  
 alive = True  
class Animal(Organism):  
 def eat(self):  
 print("This animal is eating")  
class Dog(Animal):  
 def bark(self):  
 print("This dog is barking")  
dog = Dog()  
print(dog.alive)  
dog.eat()  
dog.bark()  
  
  
#Dog ----> Aimal --->Organism

**Multiple Inheritance**

#multiple inheritance = when a childclass is derived from more than one parent class  
class Prey:  
 def flee(self):  
 print("The animal fless")  
  
class Predator:  
 def hunt(self):  
 print("The animal is hunting")  
class Rabbit(Prey):  
 pass  
class Hawk(Predator):  
 pass  
class Fish(Prey,Predator): #multiple inheritance  
 pass  
  
rabbit = Rabbit()  
hawk = Hawk()  
fish = Fish()  
  
rabbit.flee()  
hawk.hunt()  
  
fish.flee() #fish can use both Prey and Predator class function's  
fish.hunt()

**Method overriding**

class Animal:  
 def eat(self):  
 print("This animal is eating")  
class Rabbit(Animal):  
 def eat(self):  
 print("The Rabbit is eating")  
  
rabbit = Rabbit()  
rabbit.eat()  
  
animal = Animal()  
animal.eat()

**method chaining**

#method chainig = celling multiple methods sequentially  
#each cell performs an action on the same object and return self  
  
class Car:  
 def turn\_on(self):  
 print("You start the engine")  
 return self  
 def drive(self):  
 print("you drive the car")  
 return self  
 def brake(self):  
 print("You step on the brake")  
 return self  
 def tunr\_off(self):  
 print("You turn off the engine")  
 return self  
  
car = Car()  
car.turn\_on().drive().brake().tunr\_off()

Output:

You start the engine

you drive the car

You step on the brake

You turn off the engine

**Super function**

#super() = Function used to give access to the methods a parent class.  
# return a temporary object of a parent class when used  
class Rectangle:  
 def \_\_init\_\_(self, length,width):  
 self.length=length  
 self.width=width  
  
class Square(Rectangle):  
 def \_\_inti\_\_(self,length,width):  
 # self.length=length  
 #self.width=width  
 super().\_\_init\_\_(length,width)  
  
 def area(self):  
 return self.length \* self.width  
class Cube(Rectangle):  
 def \_\_init\_\_(self,length,width,height):  
 # self.length=length  
 #self.width=width  
 super().\_\_init\_\_(length,width)  
 self.height=height  
 def volume(self):  
 return self.length\*self.width\*self.height  
  
  
square = Square(3,3)  
cube = Cube(3,3,3,)  
  
print(square.area())  
print(cube.volume())

**Abstract**

Cant create an object of abstract class ..

#prevents a user from creating an object of that class  
# + compels a user to override abstract methods in a child class  
  
#abstract cass = a class which contains one or more abstract method  
#abstract method = a method that has a decleration but does not an implementations  
  
from abc import ABC,abstractmethod  
class Vehicle(ABC):  
 @abstractmethod  
 def go(self):  
 pass  
 @abstractmethod  
 def stop(self):  
 pass  
  
class Car(Vehicle):  
 def go(self):  
 print("you drive the car")  
 def stop(self):  
 print("This car is stop")  
  
class Motorcycle(Vehicle):  
 def go(self):  
 print("you ride the motorcycle")  
 def stop(self):  
 print("This motorcyle is stop")  
  
#vehicle = Vehicle()  
car = Car()  
motorcycle = Motorcycle()  
  
#vehicle.go()  
car.go()  
motorcycle.go()  
  
car.stop()  
motorcycle.stop()

**Object pass as argument**

class Car:  
 color =None  
class Motorcycle:  
 color=None  
def change\_color(vehicle,color):  
 vehicle.color=color  
  
car1=Car()  
car2=Car()  
car3=Car()  
  
bike1=Motorcycle()  
change\_color(bike1,"black")  
  
change\_color(car1,"red")  
change\_color(car2,"blue")  
  
print(car1.color)  
print(car2.color)  
print(car3.color)  
print(bike1.color)

output:

red

blue

None

black

**Duck Typing**

#Duck Typing = concept where the class of an object is less important than the method /attributes  
# class type is not checked if minimum methods/attributes are present  
#"If it walks like a duck , and it quacks like a duck , then it must be a duck  
class Duck:  
 def walk(self):  
 print("This duck is walking")  
 def talk(self):  
 print("This duck is talking")  
class Chicken:  
 def walk(self):  
 print("This chicken is walking")  
 def talk(self):  
 print("This chicken is talking")  
class Person:  
 def catch(self,duck):  
 duck.walk()  
 duck.talk()  
 print("You caught the critter!")  
  
duck = Duck()  
chicken=Chicken()  
person=Person()  
person.catch(duck)

Output:

This duck is walking

This duck is talking

You caught the critter!

**Walrus operator ( : = )**

class Duck:  
 def walk(self):  
 print("This duck is walking")  
 def talk(self):  
 print("This duck is talking")  
class Chicken:  
 def walk(self):  
 print("This chicken is walking")  
 def talk(self):  
 print("This chicken is talking")  
class Person:  
 def catch(self,duck):  
 duck.walk()  
 duck.talk()  
 print("You caught the critter!")  
  
duck = Duck()  
chicken=Chicken()  
person=Person()  
person.catch(duck)  
person.catch(chicken)

output:

This duck is

This duck is talking

You caught the critter!

This chicken is walking

This chicken is talking

You caught the critter!

#walrus operator :=  
#new to python 3.0  
#assignment expression aka walrus operator  
#assigns values to variables as part of a larger expression  
  
#happy =True  
#print(happy)  
 #or  
#print(happy=True) # it will error  
 #walrus operator  
  
print(happy := True)  
  
#example  
  
#foods = list()  
  
#while True:  
# food = input("What food do you like : ")  
# if food =="quit":  
# break  
# foods.append(food)  
  
#print(foods)  
  
 #or  
foods = list()  
while food := input("what food do you like: ")!="quit":  
 foods.append(food)  
  
print(foods[0])

**Function to variable**

def hello():  
 print("This is hello function")  
  
hello()

output: This is hello function

print(hello)

output: memory address of hello function ( hexadecimal)

def hello():  
 print("This is hello function")  
  
hello()  
print(hello)  
memory\_address = hello  
memory\_address()

we can store memory address of a function into a variable .. then if we call the variable as like the function , it will work same as the function

#so now change print to say

say = print  
say("it will work")

**Higher order function(function as argument)**

#1. accepts a function as an argument

#higher order function = a function that either  
  
def loud(text):  
 return text.upper()  
def quiet(text):  
 return text.lower()  
def hello(func):  
 text=func("hello")  
 print(text)  
  
hello(loud)  
hello(quiet)

Output: HELLO

Hello

#2.return a function (In python ,function are also treated as objects)

def divisor(x):  
 def dividend(y):  
 return y/x  
 return dividend  
divide = divisor(2)  
print(divide(10))

output: 5.0

**Lamba Function**

def double(x):  
 return x \* 2  
  
print(double(5))

output : 10

#we can make it using lambda function

double = lambda x:x\*2  
print(double(5))

output: 10

#2 parameter

multiply = lambda x,y : x\*y  
print(multiply(5,6))

output : 30

#3 parametr

add = lambda x,y,z : x+y+z  
print(add(5,6,7))

output: 18

#string

full\_name = lambda first\_name,last\_name : first\_name+" "+last\_name  
print(full\_name("Irfan","Shah"))

output: Irfan Shah

#check

age\_check = lambda age : True if age >=18 else False  
print(age\_check(18))

output: True

**Sort**

# sort() method = used with lists  
#sort() function = used with iterables  
  
  
#for list  
students = ["Irfan","nirjon","alamin","durjoy","touhed"]  
print("------Before Sort-------")  
for i in students:  
 print(i)  
  
  
students.sort()  
print("------After Sort----------")  
for i in students:  
 print(i)  
  
  
students.sort(reverse=True)  
print("------reverse Sort----------")  
for i in students:  
 print(i)  
  
#for touple  
students = ("Irfan","nirjon","alamin","durjoy","touhed")  
sorted\_students = sorted(students) #create a sorted list from touple  
print("---------from touple to list-----")  
for i in sorted\_students:  
 print(i)  
  
sorted\_students = sorted(students,reverse=True) # create a sorted list from touple  
print("---------Reverse from touple to list-----")  
for i in sorted\_students:  
 print(i)

#leve; 2

students =[("Irfan","A",25),  
 ("Nirjon","D",23),  
 ("Alamin","C",31),  
 ("Durjoy","B",19),  
 ("Touhed","F",22)]  
#sort alphabatically  
students.sort()  
print("--------sort alphabatically-------")  
for i in students:  
 print(i)  
  
#sort by grade  
  
grade = lambda grades : grades[1]  
students.sort(key=grade)  
print("-------sort by grade-------")  
for i in students:  
 print(i)  
  
#rerverse  
grade = lambda grades : grades[1]  
students.sort(key=grade,reverse=True)  
print("-------Reverse sort by grade-------")  
for i in students:  
 print(i)  
  
#sort by age  
age = lambda ages : ages[2]  
students.sort(key=age)  
print("------sort bye age------")  
for i in students:  
 print(i)

#touple of touple

#touple of touple  
students =(("Irfan","A",25),  
 ("Nirjon","D",23),  
 ("Alamin","C",31),  
 ("Durjoy","B",19),  
 ("Touhed","F",22))  
  
age = lambda ages : ages[2]  
sorted\_student = sorted(students,key=age)  
print("-----touple of touple sort-----")  
for i in sorted\_student:  
 print(i)

**Map**

#map() applies a function to each item in an iterable(list ,tuple,etc)  
#  
#map(function,iterable)  
  
store =[("shirt",20.00),  
 ("pants",25.00),  
 ("jacket",50.00),  
 ("socks",10.00)]  
to\_Tk = lambda data: (data[0],data[1]\*100)  
store\_Tk = list(map(to\_Tk,store))  
  
for i in store\_Tk:  
 print(i)

**Filter**

#filter() = create a collection of elements from an iterable for which a function return true  
#  
#filter(function,iterable)  
  
friends = [("Irfan",22),  
 ("Nirjon",19),  
 ("Ross",17),  
 ("joy",15),  
 ("Rachel",14)]  
age = lambda data : data[1]>=18  
drinking\_buddies=list(filter(age,friends))  
  
for i in drinking\_buddies:  
 print(i)

Output: ('Irfan', 22)

('Nirjon', 19)

**Filter**

#reduce()= apply a function to an iterable and reduce it to a single cumulative value.  
# performs function on first two element and repeats process until 1 value remain  
#  
#reduce(function,iterable)

import functools  
letters = ["H" ,"E" ,"L" ,"L" ,"O"]  
word = functools.reduce(lambda x,y, :x+y,letters)  
print(word)

output: HELLO

factorial =[5,4,3,2,1]  
result = functools.reduce(lambda x,y:x\*y,factorial)  
print(result)

output: 120

**List comprehensions**

**#** a way to create a new list with less syntax

#list comprehension = a way to create a new list with less syntax  
# can mimic certain lamba functions,easier to read  
# list = [expression for item in iterable]  
  
squares =[] #create an empty lis  
for i in range(1,11): #create a for loop  
 squares.append(i\*i) #define what each loop iteration should do  
print(squares)  
  
 #or using list comprehension  
  
squares=[i\*i for i in range(1,11)]  
print(squares)

Output:

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

# can mimic certain lamba functions,easier to read

students = [100,90,80,70,60,50,40,30,0]  
passed\_students = list(filter(lambda x:x>=60,students))  
print(passed\_students)  
  
 #or using list comprehension  
students = [100,90,80,70,60,50,40,30,0]  
passed\_students = [i if i>=60 else "FAILED" for i in students]  
print(passed\_students)

Output:

[100, 90, 80, 70, 60]

[100, 90, 80, 70, 60, 'FAILED', 'FAILED', 'FAILED', 'FAILED']

**Zip**

#zip(\*iterables) = aggregate elements from two or more iterables(list,tuple,sets,etc)  
# creates a zip object with paried elements stored in tuple for each element

usernames = ["Irfan","Shah","Mayeen"]  
passwords =("p@assword","abc123","guest")  
  
users = zip(usernames,passwords)  
for i in users:  
 print(i)

Output:

('Irfan', 'p@assword')

('Shah', 'abc123')

('Mayeen', 'guest')

users = list(zip(usernames,passwords))  
for i in users:  
 print(i)  
print(type(users))  
  
users = dict(zip(usernames,passwords))  
for key,value in users.items():  
 print(key+" : "+value)  
print(type(users))

#3

usernames = ["Irfan","Shah","Mayeen"]  
passwords =("p@assword","abc123","guest")  
login\_date =["1/1/2023" , "1/3/2023"]  
users=zip(usernames,passwords,login\_date)  
for i in users:  
 print(i)

usernames = ["Irfan","Shah","Mayeen"]  
passwords =("p@assword","abc123","guest")  
login\_date =["1/1/2023" , "1/3/2023"]  
users=zip(usernames,passwords,login\_date)  
for i in users:  
 print(i)

output:

('Irfan', 'p@assword', '1/1/2023')

('Shah', 'abc123', '1/3/2023')

('Mayeen', 'guest', '2/4/2023')

**Time**

import time  
  
print(time.ctime(0))#1 #convert a time expressed in seconds since epoch to a readable string  
 #epoch=when your computer thinks time began(reference point)  
  
print(time.time())#2 #return current seconds since epoch  
  
print(time.ctime(time.time())) #3 # current date and time  
#or  
time\_object = time.localtime()  
#time\_object = time.gmtime() #utc time  
print(time\_object) #4  
#time object convert as a readable format  
local\_time=time.strftime("%B %d %Y %H : %M :%S",time\_object)  
print(local\_time) #5 # %B = month , %d =date %y =year ,%H =hour, %M=minute %S = second  
  
##  
time\_string="28 April, 2023"  
time\_object=time.strptime(time\_string,"%d %B, %Y")  
print(time\_object) #6  
  
##  
time\_tuple=(2023,4,20,4,20,10,0,0,0) #(year,month,day,hour.minute,second,week day (start from monday =0,day of year,daylight saving time(0/-1)  
time\_string=time.asctime(time\_tuple)  
print(time\_string)#7

**outputs:**

1.Thu Jan 1 06:00:00 1970

2.1687957372.4725606

3.Wed Jun 28 19:02:52 2023

4.time.struct\_time(tm\_year=2023, tm\_mon=6, tm\_mday=28, tm\_hour=19, tm\_min=2, tm\_sec=52, tm\_wday=2, tm\_yday=179, tm\_isdst=0)

5.June 28 2023 19 : 02 :52

6.time.struct\_time(tm\_year=2023, tm\_mon=4, tm\_mday=28, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=4, tm\_yday=118, tm\_isdst=-1)

7.Mon Apr 20 04:20:10 2023

**Thread**

#thread = a flow of execution.Like a separate order of instructions.  
# however each thread takes a turn running to achieve concurrency  
# GIL - global interpreter lock  
# allows only one thread to hold the control of the python interpreter  
  
#cpu bound = program/task spends most of it's time waiting for the internal events (cpu intensive) use multiprocessor)  
  
#IO bound = program/task spends most of it's time waiting for external events (user input,web scraping) use multithreading)

import threading  
import time  
  
print(threading.active\_count())  
print(threading.enumerate())

**##**

import threading  
import time

print(threading.active\_count())  
def eat\_breakfast():  
 time.sleep(3)  
 print("You eat breakfast")  
def drink\_coffe():  
 time.sleep(4)  
 print("You drink coffe")  
def study():  
 time.sleep(5)  
 print("you finished study")  
eat\_breakfast()

drink\_coffe()  
study()

this program will take 12 second

**##**

import threading  
import time  
  
print(threading.active\_count())  
print(threading.enumerate())  
  
def eat\_breakfast():  
 time.sleep(3)  
 print("You eat breakfast")  
def drink\_coffe():  
 time.sleep(4)  
 print("You drink coffe")  
def study():  
 time.sleep(5)  
 print("you finished study")  
  
x = threading.Thread(target=eat\_breakfast,args=())  
x.start()  
  
y = threading.Thread(target=drink\_coffe,args=())  
y.start()  
  
z = threading.Thread(target=study,args=())  
z.start()  
  
print(threading.active\_count())  
print(threading.enumerate())

this program will take 5 second instead of 12 is because before our main thread was in charge of running these tasks sequentially in order but now since , se have a thread dedicated to each task we can run them all concurrently instead of sequentially.So that this program now took about five seconds to complete

**##**

import threading  
import time  
  
print(threading.active\_count())  
print(threading.enumerate())  
  
def eat\_breakfast():  
 time.sleep(3)  
 print("You eat breakfast")  
def drink\_coffe():  
 time.sleep(4)  
 print("You drink coffe")  
def study():  
 time.sleep(5)  
 print("you finished study")  
  
x = threading.Thread(target=eat\_breakfast,args=())  
x.start()  
  
y = threading.Thread(target=drink\_coffe,args=())  
y.start()  
  
z = threading.Thread(target=study,args=())  
z.start()  
  
x.join()  
y.join()  
z.join()  
  
print(threading.active\_count())  
print(threading.enumerate())  
print(time.perf\_counter())

**Deamon Thread**

#deamon thread = a thread that in the backround , not imporatnt for program to run  
# your program will not wait for daemon threads to complete before exiting  
# non-daemon threads cannot normally be killed ,stay alive until task is complete  
  
# ex. backround tasks,garbage collection,waiting for input,long running process

import threading  
import time  
  
def timer():  
 print()  
 count=0  
 while True:  
 time.sleep(1)  
 count+=1  
 print("logged in for: ",count,"seconds")  
x = threading.Thread(target=timer)  
x.start()  
answer = input("Do you wish to exit?")

this program will never end

to stop this

x = threading.Thread(target=timer, daemon=True)

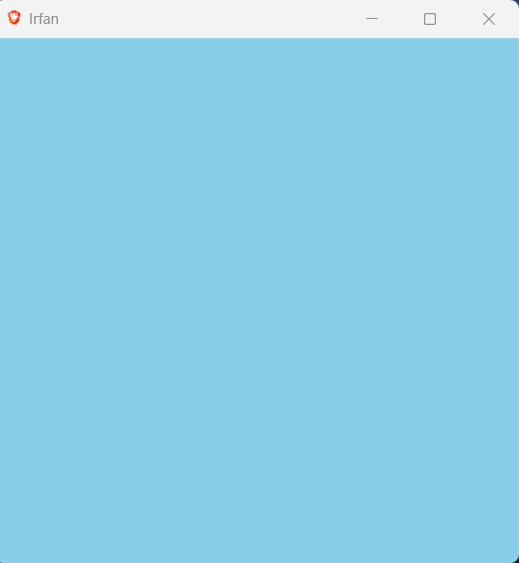
**##**

import threading  
import time  
  
def timer():  
 print()  
 count=0  
 while True:  
 time.sleep(1)  
 count+=1  
 print("logged in for: ",count,"seconds")  
x = threading.Thread(target=timer, daemon=True)  
x.start()  
answer = input("Do you wish to exit?")

**Gui.**

**#window**

from tkinter import \*  
#WIDGETS = gui ELEMENT: BUTTON,TEXTBOXES,LABELS,images  
#windows = serve as a container to hold or contain these widgets  
  
window = Tk() #instantiate an instance of a window  
window.geometry("420x420") # ("width x height "  
window.title("Irfan") #set title  
  
#set icon  
icon =PhotoImage(file='logo.png' )  
window.iconphoto(True,icon)  
  
#change backround color  
window.config(background="skyblue")  
  
#display  
window.mainloop() #place window on computer screen,listen for event

****

**Label**

**#**

from tkinter import \*  
#label = an area widget that holds text and/or an image within a window

window = Tk()  
  
label =Label(window,text="Hello wordld")  
  
label.pack()  
  
  
window.mainloop()

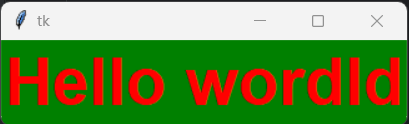
**#**

from tkinter import \*  
#label = an area widget that holds text and/or an image within a window  
  
window = Tk()  
  
label =Label(window,text="Hello wordld")  
  
label.place(x=0,y=0)  
label.pack()  
  
window.mainloop()

**#add font**

wordld",font=('Arial',40,'bold'),fg="red",bg="green")

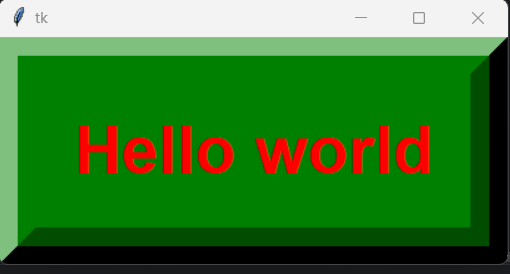
from tkinter import \*  
#label = an area widget that holds text and/or an image within a window  
  
window = Tk()  
  
label =Label(window,text="Hello wordld",font=('Arial',40,'bold'),fg="red",bg="green")  
label.pack()  
  
window.mainloop()

****

**#border,**

relief=RAISED,bd=30,padx=30,pady=30

from tkinter import \*  
#label = an area widget that holds text and/or an image within a window  
  
window = Tk()  
  
label =Label(window,text="Hello world",font=('Arial',40,'bold'),fg="red",bg="green",relief=RAISED,bd=30,padx=30,pady=30)  
label.pack()  
#or  
#label.place(x=0,y=0)  
  
  
  
window.mainloop()

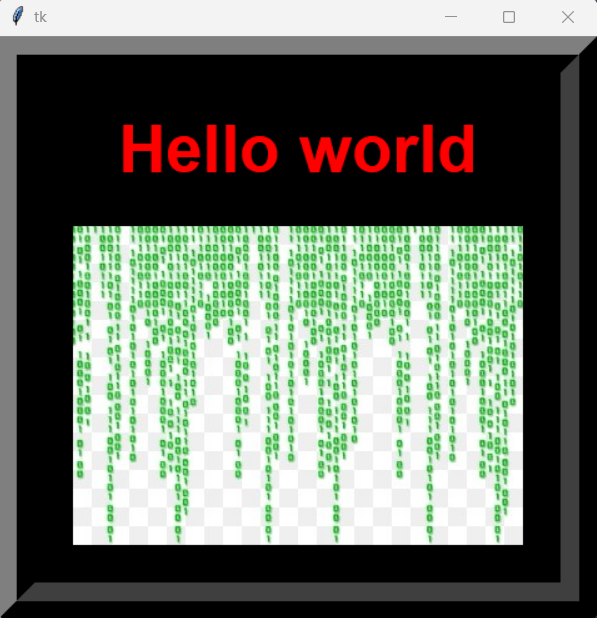
****

**#img label**

photo = PhotoImage(file='img.png')

image=photo,compound='bottom')

from tkinter import \*  
#label = an area widget that holds text and/or an image within a window  
  
window = Tk()  
photo = PhotoImage(file='img.png')  
  
label =Label(window,text="Hello world",  
 font=('Arial',40,'bold'),fg="red",bg="black",  
 relief=RAISED,bd=30,padx=30,pady=30,  
 image=photo,compound='bottom')  
label.pack()  
#or  
#label.place(x=0,y=0)  
  
  
  
window.mainloop()



**Button**

from tkinter import \*  
  
#button = you click it , the it does stuff  
def click():  
 print("You clicked the button")  
window = Tk()  
photo = PhotoImage(file='cplus.png')  
  
button = Button(window,  
 text="Click me", #button name  
 command=click, #function call(action)  
 font=("Comic Sans",30), #font  
 fg="red",   
 bg="green",  
 activeforeground="blue", #after click  
 activebackground="black", #afte click  
 state =ACTIVE, #if state =DISABLEd ,button will not work  
 image=photo, #umage add  
 compound='bottom')  
  
button.pack()  
window.mainloop()

**Entry Box**

**#**

from tkinter import \*  
#entry widget = textbox that accepts a single line of user input  
  
window =Tk()  
  
entry=Entry(window,  
 font=("Arial",50))  
  
entry.pack()  
window.mainloop()

****

**#input and output in console**

from tkinter import \*  
#entry widget = textbox that accepts a single line of user input  
  
window =Tk()  
def submit():  
 username = entry.get()  
 print(username)  
  
entry=Entry(window,  
 font=("Arial",50))  
  
entry.pack(side=LEFT)  
  
submit\_button = Button(window,text="Submit",command=submit)  
submit\_button.pack(side=RIGHT)  
window.mainloop()

**##delete , backspace**

from tkinter import \*  
#entry widget = textbox that accepts a single line of user input  
  
window =Tk()  
def submit():  
 username = entry.get()  
 print(username)  
  
def delete():  
 entry.delete(0,END)  
def backspace():  
 entry.delete(len(entry.get())-1,END)  
  
entry=Entry(window,  
 font=("Arial",50),  
 fg="green",  
 bg="black")  
  
entry.pack(side=LEFT)  
  
submit\_button = Button(window,text="Submit",command=submit)  
submit\_button.pack(side=RIGHT)  
  
delete\_button = Button(window,text="Delete",command=delete)  
delete\_button.pack(side=RIGHT)  
  
backspace\_button = Button(window,text="Back\_Space",command=backspace)  
backspace\_button.pack(side=RIGHT)  
  
  
window.mainloop()

**#default text**

entry.insert(0,'This is default text')

**#after submit text , disable**

def submit():  
 username = entry.get()  
 print(username)  
 entry.config(state=DISABLED)

**##show character password type**

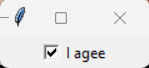
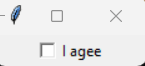
entry=Entry(window,  
 font=("Arial",50),  
 fg="green",  
 bg="black",  
 show="\*")

****

**CheckButton**

**#**

from tkinter import \*  
  
window=Tk()  
check\_button=Checkbutton(window,text="I agee")  
check\_button.pack()  
window.mainloop()

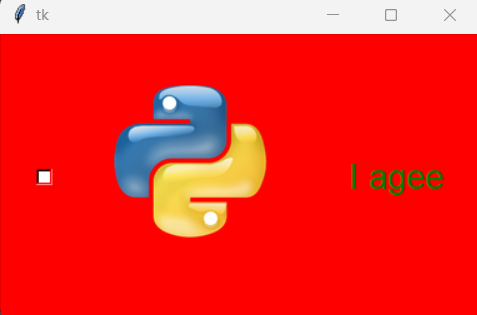
****

**##console show**

from tkinter import \*  
  
def display():  
 if(x.get()==1):  
 print("You agree")  
 else:  
 print("You don't agree")  
  
window=Tk()  
x=IntVar()  
check\_button=Checkbutton(window,text="I agee",  
 font=('Arial',20),  
 fg="Green",  
 bg="RED",  
 activeforeground="orange",  
 activebackground="blue",  
 variable=x,  
 onvalue=1,  
 offvalue=0,  
 command=display,  
 padx=25, pady=10)  
  
check\_button.pack()  
window.mainloop()

**##image add**

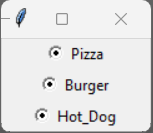
from tkinter import \*  
  
def display():  
 if(x.get()==1):  
 print("You agree")  
 else:  
 print("You don't agree")  
  
window=Tk()  
x=IntVar()  
  
python\_photo = PhotoImage(file='img.png')  
check\_button=Checkbutton(window,text="I agee",  
 font=('Arial',20),  
 fg="Green",  
 bg="RED",  
 activeforeground="orange",  
 activebackground="blue",  
 variable=x,  
 onvalue=1,  
 offvalue=0,  
 command=display,  
 padx=25, pady=10,  
 image=python\_photo,  
 compound='left')  
  
check\_button.pack()  
window.mainloop()

****

**Radio Button**

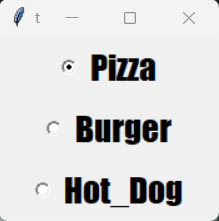
**##default all select**

from tkinter import \*  
  
food =["Pizza","Burger","Hot\_Dog"]  
  
window=Tk()  
  
for i in range(len(food)):  
 radiobutton=Radiobutton(window,text=food[i])  
 radiobutton.pack()  
  
window.mainloop()

****

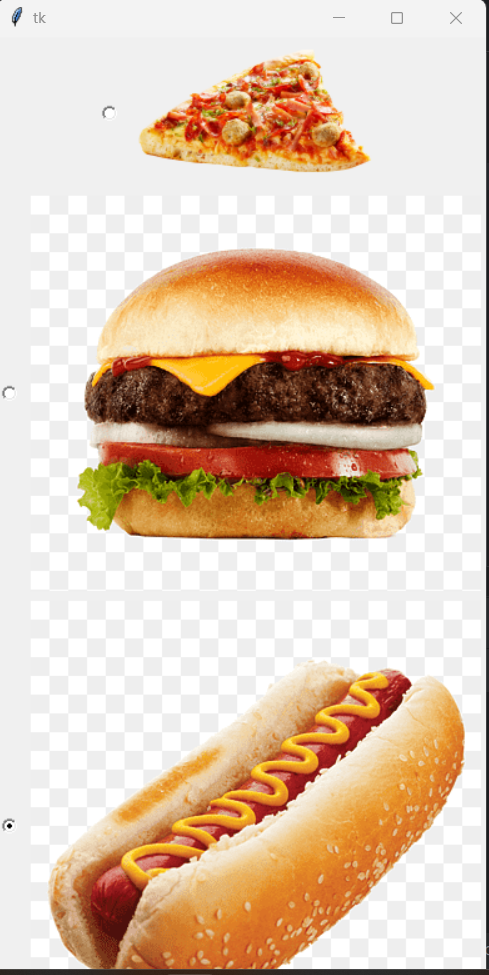
**##select one, then select all**

#radio button = similar to check box, but you can select one from a group  
  
from tkinter import \*  
  
food =["Pizza","Burger","Hot\_Dog"]  
  
window=Tk()  
  
x=IntVar()  
  
for i in range(len(food)):  
 radiobutton=Radiobutton(window,text=food[i],variable=x)  
 radiobutton.pack()  
  
window.mainloop()

****

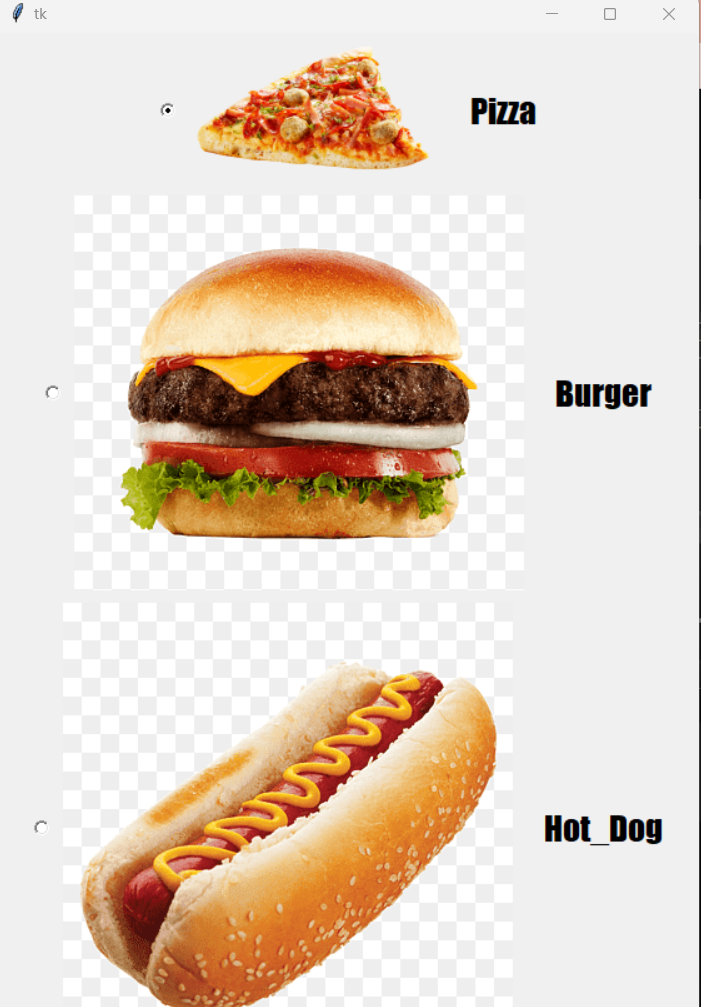
**##imageadd**

#radio button = similar to check box, but you can select one from a group  
  
from tkinter import \*  
  
food=["Pizza","Burger","Hot\_Dog"]  
  
window=Tk()  
  
pizzaImage=PhotoImage(file='pizza.png')  
burgerImage =PhotoImage(file='burger.png')  
hotdogImage=PhotoImage(file='hotdog.png')  
foodImages = [pizzaImage,burgerImage,hotdogImage]  
  
x=IntVar()  
  
for i in range(len(food)):  
 radiobutton=Radiobutton(window,  
 text=food[i], #adds text to radio button  
 variable=x, #groups radiobutton together if they share the samne variable  
 value=i , #assigns each radiobutton a differenet value  
 padx=25 , # adds padding on x-axis  
 font=("Impact",20),  
 image=foodImages[i] )#adds images radiobutton  
  
 radiobutton.pack()  
  
window.mainloop()

****

**#add compound**

#radio button = similar to check box, but you can select one from a group  
  
from tkinter import \*  
  
food=["Pizza","Burger","Hot\_Dog"]  
  
window=Tk()  
  
pizzaImage=PhotoImage(file='pizza.png')  
burgerImage =PhotoImage(file='burger.png')  
hotdogImage=PhotoImage(file='hotdog.png')  
foodImages = [pizzaImage,burgerImage,hotdogImage]  
  
x=IntVar()  
  
for i in range(len(food)):  
 radiobutton=Radiobutton(window,  
 text=food[i], #adds text to radio button  
 variable=x, #groups radiobutton together if they share the samne variable  
 value=i , #assigns each radiobutton a differenet value  
 padx=25 , # adds padding on x-axis  
 font=("Impact",20),  
 image=foodImages[i], #adds images radiobutton  
 compound='left') #adds image & text(left-side)  
 radiobutton.pack()  
  
window.mainloop()

****

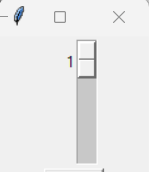
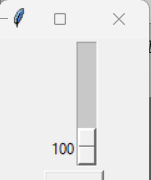
**#console output**

#radio button = similar to check box, but you can select one from a group  
  
from tkinter import \*  
  
food=["Pizza","Burger","Hot\_Dog"]  
  
def order():  
 if(x.get()==0):  
 print("You select Pizza")  
 elif(x.get()==1):  
 print("You order Burger")  
 elif(x.get()==2):  
 print("You order Hotdog")  
 elif (x.get() == 1):  
 print("You order Burger")  
 else:  
 print("Nothing")  
  
  
window=Tk()  
  
pizzaImage=PhotoImage(file='pizza.png')  
burgerImage =PhotoImage(file='burger.png')  
hotdogImage=PhotoImage(file='hotdog.png')  
foodImages = [pizzaImage,burgerImage,hotdogImage]  
  
x=IntVar()  
  
for i in range(len(food)):  
 radiobutton=Radiobutton(window,  
 text=food[i], #adds text to radio button  
 variable=x, #groups radiobutton together if they share the samne variable  
 value=i , #assigns each radiobutton a differenet value  
 padx=25 , # adds padding on x-axis  
 font=("Impact",20),  
 image=foodImages[i], #adds images radiobutton  
 compound='left', #adds image & text(left-side)  
 indicatoron=0, #eliminate circle indicator  
 command=order) #set cammand of radiobutton to function  
  
 radiobutton.pack()  
  
window.mainloop()

**Scale**

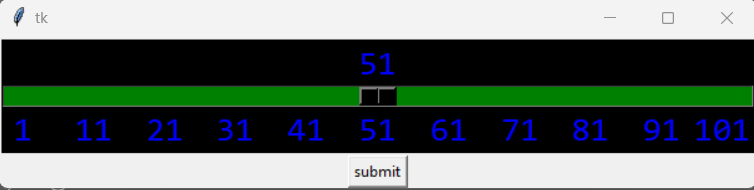
from tkinter import \*

window=Tk()  
  
scale = Scale(window,from\_=1,to=100)  
scale.pack()  
  
window.mainloop()

** **

**##**

from tkinter import \*  
  
def submit():  
 print("The temperaure is : "+ str(scale.get())+" degree C")  
  
window=Tk()  
  
scale = Scale(window,from\_=1,to=100,  
 length=600,  
 orient=HORIZONTAL,  
 font=('Consolas',20),  
 tickinterval=10, #adds numeric indicator for value  
 #showvalue=0) #hide current value  
 resolution=5, #increment of slider  
 troughcolor="Green",  
 fg="Blue",  
 bg="black")  
  
  
  
#scale.set(50)  
#or  
scale.set(((scale['from']-scale['to'])/2)+scale['to'])  
  
scale.pack()  
  
button =Button(window,text="submit",command=submit)  
button.pack()  
window.mainloop()

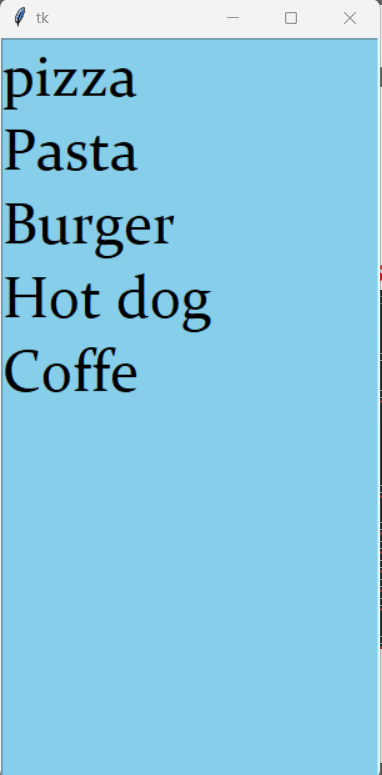
****

**##image add**

from tkinter import \*  
  
def submit():  
 print("The temperaure is : "+ str(scale.get())+" degree C")  
  
window=Tk()  
  
hotImage = PhotoImage(file='flame.png',width=200,height=200)  
hotLabel=Label(image=hotImage)  
hotLabel.pack(side="top")  
  
coldImage=PhotoImage(file='snow.png',width=200,height=200)  
coldLabel=Label(image=coldImage)  
coldLabel.pack(side="bottom")  
  
scale = Scale(window,from\_=1,to=100,  
 length=600,  
 orient=VERTICAL,  
 font=('Consolas',20),  
 tickinterval=10, #adds numeric indicator for value  
 #showvalue=0) #hide current value  
 resolution=5, #increment of slider  
 troughcolor="Green",  
 fg="Blue",  
 bg="black")  
  
  
  
#scale.set(50)  
#or  
scale.set(((scale['from']-scale['to'])/2)+scale['to'])  
  
scale.pack()  
  
button =Button(window,text="submit",command=submit)  
button.pack()  
window.mainloop()

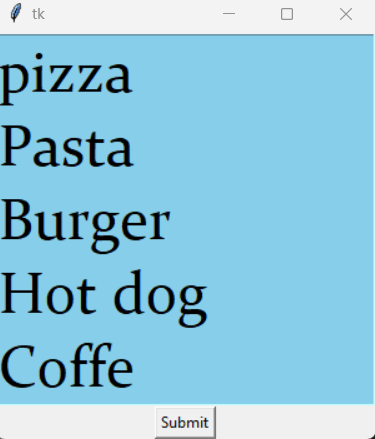
**List Box**

#list box = A listing of selectable text items within it's own container  
  
from tkinter import \*  
window = Tk()  
  
listbox = Listbox(window,  
 bg="SKYblue",  
 font=("Constantia",35),  
 width=12)  
  
listbox.pack()  
  
listbox.insert(1,"pizza")  
listbox.insert(2,"Pasta")  
listbox.insert(3,"Burger")  
listbox.insert(4,"Hot dog")  
listbox.insert(5,"Coffe")  
  
window.mainloop()

****

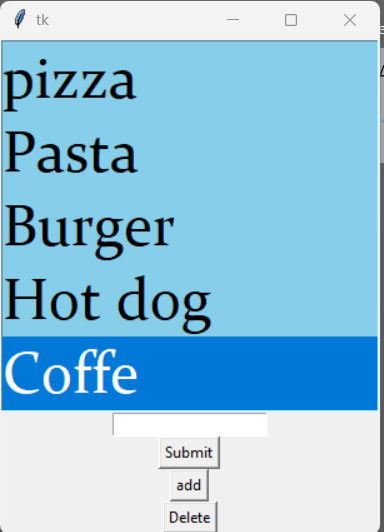
**##ad submit button**

#list box = A listing of selectable text items within it's own container  
  
from tkinter import \*  
def submit():  
 print("You have ordered:")  
 print(listbox.get(listbox.curselection()))  
window = Tk()  
  
listbox = Listbox(window,  
 bg="SKYblue",  
 font=("Constantia",35),  
 width=12)  
  
listbox.pack()  
  
listbox.insert(1,"pizza")  
listbox.insert(2,"Pasta")  
listbox.insert(3,"Burger")  
listbox.insert(4,"Hot dog")  
listbox.insert(5,"Coffe")  
  
listbox.config(height=listbox.size())  
  
submit\_buttom=Button(window,text="Submit",command=submit)  
submit\_buttom.pack()  
window.mainloop()

****

**##add,delete**

#list box = A listing of selectable text items within it's own container  
  
from tkinter import \*  
def submit():  
 print("You have ordered:")  
 print(listbox.get(listbox.curselection()))  
def add():  
 listbox.insert(listbox.size(),entryBox.get())  
 listbox.config(height=listbox.size())  
def delete():  
 listbox.delete(listbox.curselection())  
 listbox.config(height=listbox.size())  
  
window = Tk()  
  
listbox = Listbox(window,  
 bg="SKYblue",  
 font=("Constantia",35),  
 width=12,)  
   
  
listbox.pack()  
  
listbox.insert(1,"pizza")  
listbox.insert(2,"Pasta")  
listbox.insert(3,"Burger")  
listbox.insert(4,"Hot dog")  
listbox.insert(5,"Coffe")  
  
listbox.config(height=listbox.size())  
  
entryBox=Entry(window)  
entryBox.pack()  
  
submit\_buttom=Button(window,text="Submit",command=submit)  
submit\_buttom.pack()  
  
addButton = Button(window,text="add",command=add)  
addButton.pack()  
  
deleteButton =Button(window,text="Delete",command=delete)  
deleteButton.pack()  
window.mainloop()

****

**##multiple selection**

#list box = A listing of selectable text items within it's own container  
  
from tkinter import \*  
def submit():  
 food=[]  
 for i in listbox.curselection():  
 food.insert(i,listbox.get(i))  
 print("You have ordered:")  
 for i in food:  
 print(i)  
 # print(listbox.get(listbox.curselection()))  
def add():  
 listbox.insert(listbox.size(),entryBox.get())  
 listbox.config(height=listbox.size())  
def delete():  
 for i in listbox.curselection():  
 listbox.delete(i)  
 #listbox.delete(listbox.curselection())  
 listbox.config(height=listbox.size())  
  
window = Tk()  
  
listbox = Listbox(window,  
 bg="SKYblue",  
 font=("Constantia",35),  
 width=12,  
 selectmode=MULTIPLE)  
  
  
listbox.pack()  
  
listbox.insert(1,"pizza")  
listbox.insert(2,"Pasta")  
listbox.insert(3,"Burger")  
listbox.insert(4,"Hot dog")  
listbox.insert(5,"Coffe")  
  
listbox.config(height=listbox.size())  
  
entryBox=Entry(window)  
entryBox.pack()  
  
submit\_buttom=Button(window,text="Submit",command=submit)  
submit\_buttom.pack()  
  
addButton = Button(window,text="add",command=add)  
addButton.pack()  
  
deleteButton =Button(window,text="Delete",command=delete)  
deleteButton.pack()  
window.mainloop()

**MessageBox**

from tkinter import \*  
from tkinter import messagebox #import messagebox library  
  
def click():  
 #messagebox.showinfo(title='Irfan',message="You are a person")  
 # messagebox.showwarning(title="Waring",message="It;s virus")  
 #messagebox.showerror(title="Error", message="Something is wring")  
 if messagebox.askokcancel(title="ASK", message="Do You agree?"):  
 print("You agreed")  
 else:  
 print("You don't agree")  
window =Tk()  
  
button = Button(window,text="click me",command=click)  
button.pack()  
  
window.mainloop()

**Color Chooser**