# **Tuple unpacking:-**

tup = (1,2,3,4,5)

for i in tup:

print(i)

A screenshot of a computer

Description automatically generated

tup = ((1,2),(3,4),(5))

for a,b in tup:

print(a)

A screen shot of a computer

Description automatically generated

tup = ((1,2),(3,4),(5,6))

for i in tup:

print(i)

A screenshot of a computer

Description automatically generated

tup = ((1,2),(3,4),(5,6))

for a,b in tup:

print(a)

print(b)

A number on a blue background

Description automatically generated

tup = [(1,2),(3,4),(5,6)]

for a,b in tup:

print(a)

print(b)

A number on a blue background

Description automatically generated

tup = [(1,2),(3,4),(5,6),[7,8],{'apple':1, 'mango':[12,34]}]

for a,b in tup:

print(a)

print(b)

A blue background with white text

Description automatically generated

# **Dictionary**

d = {'name': 'Muskan', 'age': 22}

for i in d:

print(i)

A screenshot of a computer

Description automatically generated

d = {'name': 'Muskan', 'age': 22}

for i in d.items():

print(i)

A screenshot of a computer

Description automatically generated

d = {'name': 'Muskan', 'age': 22}

for a,b in d:

print(a)

print(b)

A screen shot of a computer code

Description automatically generated

d = {'name': 'Muskan', 'age': 22}

for a,b in d.items():

print(a)

print(b)

A screenshot of a phone

Description automatically generated

# **\*args**

Python has \*args which allow us to pass the variable number of non keyword arguments to function.

In the function, we should use an asterisk \* before the parameter name to pass variable length arguments.The arguments are passed as a [tuple](https://www.programiz.com/python-programming/tuple) and these passed arguments make tuple inside the function with same name as the parameter excluding asterisk \*.

def operate(\*num):

for i in num:

print(i\*\*2)

operate(1,2,3,4,5)

A screenshot of a computer

Description automatically generated

# **\*\*kwargs**

Python passes variable length non keyword argument to function using \*args but we cannot use this to pass keyword argument. For this problem Python has got a solution called \*\*kwargs, it allows us to pass the variable length of keyword arguments to the function.

In the function, we use the double asterisk \*\* before the parameter name to denote this type of argument. The arguments are passed as a dictionary, and these arguments make a dictionary inside function with name same as the parameter excluding double asterisk \*\*.

def operate(\*\*num):

for i,j in num.items():

print(j\*\*2)

operate(ok=1, lie=2)

A screenshot of a computer

Description automatically generated

def operate(\*a, \*\*num):

for i,j in num.items():

print(j\*\*2)

for k in a:

print(k)

operate(3,5,ok=1, lie=2)

A screenshot of a computer

Description automatically generated

def operate(\*\*num, \*a):

for i,j in num.items():

print(j\*\*2)

for k in a:

print(k)

operate(ok=1, lie=2,3,5)

A screenshot of a computer program

Description automatically generated

def operate(\*a,\*\*num):

for i,j in num.items():

print(j\*\*2)

for k in a:

print(k)

operate(3,ok=1, lie=2,5)

A screenshot of a computer program

Description automatically generated

# **List**

In list, we have insert(), append(), extend() to add values

del, remove(), pop(), clear() to remove values

l =[1,2,3,4]

l.insert(2,7) #insert value at the specified index

print(l)

l.append(9) #attach value at the end of the list

print(l)

l.extend([8,10,11]) #attach list at the end of l

print(l)

l.extend(8) #cannot take a single value, it take an iterator

print(l)

A screenshot of a computer program

Description automatically generated

l = [1, 2, 7, 3, 4, 9]

l.extend('Muskan')

print(l)

[1, 2, 7, 3, 4, 9, 'M', 'u', 's', 'k', 'a', 'n']

l.extend((8,10,11))

print(l)

[1, 2, 7, 3, 4, 9, 8, 10, 11]

l =[1,2,3,4,5,6,7,8,9,10]

l.remove(4) #removes the specific value

print(l)

l.pop() #removes the last item

print(l)

l.pop(2) #removes the specified index

print(l)

del l[3] #deletes the specific element

print(l)

del l #delete the list

l=[9,8,7,6,5]

l.clear() #empties the list

print(l)

A screenshot of a computer

Description automatically generated

# **Set**

In set we have add(), update() to add values

remove(), discard(), pop(), clear(), del to remove values

S = {1,2,3,4,5,6,7,8,9,10}

s.remove(2) #removes specific value

**Note:** If the item to remove does not exist, remove() will raise an error.

s.discard(4)

**Note:** If the item to remove does not exist, discard() will **NOT** raise an error.

s.pop()

**Note:** Sets are *unordered*, so when using the pop() method, you do not know which item that gets removed.

del s #deletes the set completely

s.clear() #empties the set

# **Dictionary**

In dictionary, we can directly change value/add value with the help of key or use update()

pop()

d = {"brand": "Ford", "model": "Mustang", "year": 1964}  
d.update({"year": 2020})

d[“year”] = 2025

d.pop(‘model’) #removes the item with the specified key name

d.popitem() #removes the last inserted item

del d[‘brand’] #removes the item of the specified key

del d #deletes the dictionary completely

d.clear() #empties the dictionary

# **Decorator**

1. Write a Python program to create a decorator function to measure the execution time of a function.

import time

def exec\_time(func):

def inner(x):

entry\_time = time.time()

func(x)

exit\_time = time.time()

execution\_time = exit\_time - entry\_time

return execution\_time

return inner

@exec\_time

def Calculate\_Square(x):

print(f'Square of x is {x\*\*2}')

time\_taken = Calculate\_Square(4)

print(f'Time taken to caluculate is : {time\_taken}')

# Generator

1. Write a Python program to implement a generator that generates random numbers within a given range.

A screen shot of a computer code

Description automatically generated A close up of a screen

Description automatically generated

1. Write a Python program that creates a generator function that generates all prime numbers between two given numbers.

A screenshot of a computer program

Description automatically generated A number on a blue background

Description automatically generated

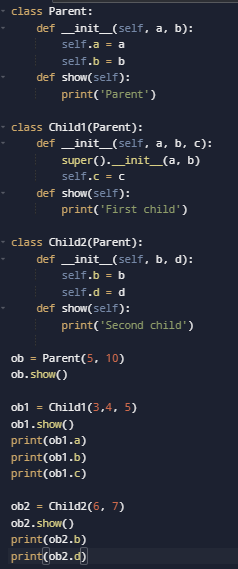
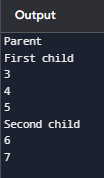
# Abstraction

A screen shot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

# Inheritance

ob1.show() is executing the method of the child class and is overriding the show method of the parent class