In [1]:
```
print("Hello")
a='hello'
print(a)
```

```
Hello
hello
```

In [2]:
```
a,b='apple','banana'
print(a,b)
```

```
apple banana
```

In [3]:
```
x,y,z=10,20,30
print(x,y,z)
```

```
10 20 30
```

In [4]:
```
a="Hello People"
print(a)
```

```
Hello People
```

In [24]:
```
a='Hello People'
print(a[0])
print(a[1])
print(a[2])
print(a[3])
print(a[4])
print(a[6])
print(a[7])
print(a[8])
print(a[9])
print(a[10])
print(a[11])
print(len(a))
```

```
H
e
l
l
o
P
e
o
p
l
e
12
```

In [188]:
```
for x in "welcome to kpi":
  print(x)
```

```
w
e
l
c
o
m
e

t
o

k
p
i
```

In [26]:
```
z='WelcometoKPI'
print(len(z))
```

```
12
```

In [64]:
```
y = 'The best is yet to come'
if "come" in y:
    print("Yes, 'come' is present.")
y = 'The best is yet to come'
if "are" not in y:
    print("Yes, 'are' is not present.")
```

```
Yes, 'come' is present.
Yes, 'are' is not present.
```

In [65]:
```
txt = "The best things in life are free!"
if "free" in txt:
    print("Yes, 'free' is present.")
txt = "The best things in life are free!"
if "expensive" not in txt:
    print("Yes, 'expensive' is NOT present.")
```

```
Yes, 'free' is present.
Yes, 'expensive' is NOT present.
```

In [66]:
```
b = "Hello, World!"
print(b[2:5])
# Negative Indexing
b = "Hello, World!"
print(b[-5:-2])
```

```
llo
orl
```

In [69]:
```
c="welcome to kpi"
print(c[2:5])
print(c[0:6])
print(c[-10:-2])
```

```
lco
welcom
ome to k
```

In [70]:
```
a = "Hello, World!"
print(a.upper())
#Lower Case
a = "Hello, World!"
print(a.lower())
#Remove Whitespace
a = " Hello, World! "
print(a.strip())
#Replace String
a = "Hello, World!"
print(a.replace("H", "J"))
#Split String
a = "Hello, World!"
print(a.split(","))
```

```
HELLO, WORLD!
hello, world!
Hello, World!
Jello, World!
['Hello', ' World!']
```

In [75]:
```
b='hello, people!'
print(b.upper())
print(b.lower())
print(b.strip())
print(b.replace('h','H'))
print(b.split(","))
```

```
HELLO, PEOPLE!
hello, people!
hello, people!
Hello, people!
['hello', ' people!']
```

In [76]:
```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

```
['apple', 'banana', 'cherry']
```

In [79]:
```
l=['Venkata','Sai','Tanuja']
print(l)
```

```
['Venkata', 'Sai', 'Tanuja']
```

In [80]:
```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

```
3
```

In [1]:
```
l=['Venkata','Sai','Tanuja','Nekkanti']
print(len(l))
```

```
4
```

In [3]:
```
thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
print(thislist)
```

```
['apple', 'banana', 'cherry']
```

In [86]:
```
l = list(('Venkata','Sai','Tanuja','Nekkanti'))
print(l)
```

```
['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
```

In [87]:
```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

```
banana
```

In [90]:
```
l = list(('Venkata','Sai','Tanuja','Nekkanti'))
print(l[3])
print(l[2])
print(l[0])
```

```
Nekkanti
Tanuja
Venkata
```

In [91]:
```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

```
['cherry', 'orange', 'kiwi']
```

In [94]:
```
l = list(('Venkata','Sai','Tanuja','Nekkanti'))
print(l[-3:-1])
```

```
['Sai', 'Tanuja']
```

In [96]:
```
l = list(('Venkata','Sai','Tanuja','Nekkanti'))
print(l[2:3])
print(l[1:3])
```

```
['Tanuja']
['Sai', 'Tanuja']
```

In [97]:
```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
 print("Yes, 'apple' is in the fruits list")
```

```
Yes, 'apple' is in the fruits list
```

In [99]:
```
l = list(('Venkata','Sai','Tanuja','Nekkanti'))
if 'Tanuja' in l:
 print("Yes,'Tanuja' is in l")

l = list(('Venkata','Sai','Tanuja','Nekkanti'))
if 'Chaitanya' not in l:
 print("Yes,'Chaitanya' is not in l")
```

```
Yes,'Tanuja' is in l
Yes,'Chaitanya' is not in l
```

In [100]:
```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
#Using the append() method to append an item
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

```
['apple', 'banana', 'watermelon', 'cherry']
['apple', 'banana', 'cherry', 'orange']
```

In [102]:
```
l=['Venkata','Sai','Tanuja']
l.insert(3,'Nekkanti')
print(l)
l.insert(2,'Nekkanti')
print(l)
l.insert(0,'Nekkanti')
print(l)
```

```
['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
['Venkata', 'Sai', 'Nekkanti', 'Tanuja', 'Nekkanti']
['Nekkanti', 'Venkata', 'Sai', 'Nekkanti', 'Tanuja', 'Nekkanti']
```

In [103]:
```
print(l)
```

```
['Nekkanti', 'Venkata', 'Sai', 'Nekkanti', 'Tanuja', 'Nekkanti']
```

In [104]:
```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
#The pop() method removes the specified index
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
#If you do not specify the index, the pop() method removes the last item
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
#The del keyword can also delete the list completely
thislist = ["apple", "banana", "cherry"]
del thislist
#The clear() method empties the list.
#The list still remains, but it has no content
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

```
['apple', 'cherry']
['apple', 'cherry']
['apple', 'banana']
[]
```

In [114]:
```
l=['Nekkanti', 'Venkata', 'Sai', 'Nekkanti', 'Tanuja', 'Nekkanti']
l.remove('Nekkanti')
print(l)
l.pop(2)
print(l)
l.pop()
print(l)
del l
l=['Venkata', 'Sai', 'Tanuja']
l.clear()
print(l)
```

```
['Venkata', 'Sai', 'Nekkanti', 'Tanuja', 'Nekkanti']
['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
['Venkata', 'Sai', 'Tanuja']
[]
```

In [115]:
```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
 print(x)
#Use the range() and len() functions to create a suitable iterable
#Print all items by referring to their index number
print("\n")
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
 print(thislist[i])
#Print all items, using a while loop to go through all the index numbers
print("\n")
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
 print(thislist[i])
 i = i + 1


#Looping Using List Comprehension
print("\n")
thelist = ["apple", "banana", "cherry"]
[print(x) for x in thelist]
```

```
apple
banana
cherry


apple
banana
cherry


apple
banana
cherry


apple
banana
cherry
```

Out[115]:
```
[None, None, None]
```

In [123]:
```
l=['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
for x in l:
    print(x)

print('\n')
l=['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
for i in range(len(l)):
    print(l[i])

print('\n')
l=['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
i = 0
while i < len(l):
 print(l[i])
 i = i + 1

print('\n')
l=['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
[print(x) for x in l]
```

```
Venkata
Sai
Tanuja
Nekkanti


Venkata
Sai
Tanuja
Nekkanti


Venkata
Sai
Tanuja
Nekkanti


Venkata
Sai
Tanuja
Nekkanti
```

Out[123]:
```
[None, None, None, None]
```

In [125]:
```
 fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
#Loop through the letters in the word "banana"
print("\n")
for x in "banana":
    print(x)
print("\n")
for x in range(6):
    print(x)
print("\n")
#Increment the sequence with a particualr value(default is 1)
for x in range(2, 30, 3):
    print(x)
```

```
apple
banana
cherry


b
a
n
a
n
a


0
1
2
3
4
5


2
5
8
11
14
17
20
23
26
29
```

In [136]:
```
l=['Venkata', 'Sai', 'Tanuja', 'Nekkanti']
for x in "tanuja":
    print(x)

print('\n')
for x in range(10):
    print(x)

print('\n')
for x in range(2,30,3):
    print(x)
```

```
t
a
n
u
j
a


0
1
2
3
4
5
6
7
8
9


2
5
8
11
14
17
20
23
26
29
```

In [137]:
```
i = 1
while i < 6:
 print(i)
 i += 1
```

```
1
2
3
4
5
```

In [140]:
```
i=1
while i<10:
  print(i)
  i=i+1
```

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

In [144]:
```
print("Example for 'type' built-in function: ",type([]))
print("Example for 'print' built-in function")
def call(x):
 return x
print(callable(call))
```

```
Example for 'type' built-in function:  <class 'list'>
Example for 'print' built-in function
True
```

In [146]:
```
def foo():
 pass
print(type(foo))
print("foo name attribute: ",foo.__name__)
```

```
<class 'function'>
foo name attribute:  foo
```

In [147]:
```
def low():
    pass
print(type(low))
print("low name attribute: ",low.__name__)
```

```
<class 'function'>
low name attribute:  low
```

In [148]:
```
y = lambda x:x*2
print(y(5))
#Eg2:
lst = [1,5,7,14]
newlst = list(filter(lambda a:a%7==0,lst))
print(newlst)
```

```
10
[7, 14]
```

In [152]:
```
x=lambda y:y*3
print(x(10))

lst=[2,4,6,9,3,10]
newlst=list(filter(lambda a:a%2==0,lst))
print(newlst)

lst=[1,2,3,4,5,6,7,8,9]
newlst=list(filter(lambda b:b%3==0,lst))
print(newlst)
```

```
30
[2, 4, 6, 10]
[3, 6, 9]
```

In [153]:
```
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')

# Add two numbers
sum = float(num1) + float(num2)

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

```
Enter first number: 11
Enter second number: 22
The sum of 11 and 22 is 33.0
```

In [165]:
```
num1=10
num2=30
sum=float(num1)+float(num2)
print('sum is :',(sum))
```

```
sum is : 40.0
```

In [175]:
```
a=50
b=90
c=float(a+b)
print('sum of 50 and 90 is :',(c))
```

```
sum of 50 and 90 is : 140.0
```

In [176]:
```
x=99
y=10
temp=x
x=y
y=temp
print('the value of x after swapping is:',x)
print('the value of y after swapping is:',y)
```

```
the value of x after swapping is: 10
the value of y after swapping is: 99
```

In [177]:
```
x = 5
y = 10
temp=x
x=y
y=temp
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))
```

```
The value of x after swapping: 10
The value of y after swapping: 5
```

In [178]:
```
x = input('Enter value of x: ')
y = input('Enter value of y: ')
temp=x
x=y
y=temp
print('the value of x after swapping:',x)
print('the value of y after swapping:',y)
```

```
Enter value of x: 20
Enter value of y: 70
the value of x after swapping: 70
the value of y after swapping: 20
```

In [180]:
```
a="Hello team!!"
print(a.replace("t","T"))
```

```
Hello Team!!
```

In [182]:
```
l=['apple','banana','organe']
l.remove("organe")
print(l)
l.pop()
print(l)
```

```
['apple', 'banana']
['apple']
```

In [186]:
```
l=['apple','banana','organe']
l.pop(1)
print(l)
```

```
['apple', 'organe']
```

In [187]:
```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

In [189]:
```
t=('sailaja','chaitanya','satyanarayana')
print(t)
```

```
('sailaja', 'chaitanya', 'satyanarayana')
```

In [190]:
```
t=('satyanarayana','sailaja','chaitanya')
t[2]="tanuja"
print(t)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\TANUJA~1\AppData\Local\Temp/ipykernel_14184/109997667.py in <module>
      1 t=('satyanarayana','sailaja','chaitanya')
----> 2 t[2]="tanuja"
      3 print(t)

TypeError: 'tuple' object does not support item assignment
```

In [214]:
```
set={'rose','jasmine','lily'}
print(set)
set.update(['mango','organe','pineapple'])
print(set)
#set.add('cherry')

print('\n')
s={'pink','blue','yellow'}
print(s)
s.add('red')
print(s)
```

```
{'jasmine', 'rose', 'lily'}
{'rose', 'jasmine', 'lily', 'pineapple', 'organe', 'mango'}


{'blue', 'yellow', 'pink'}
{'blue', 'yellow', 'red', 'pink'}
```

In [198]:
```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
for x, y in thisdict.items():
  print(x, y)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
brand Ford
model Mustang
year 1964
```

In [201]:
```
n=10
id(n)
```

Out[201]:
```
2581650369104
```

In [221]:
```
stu={'sno':1,'sname':'tanuja','sage':22}
print(stu)
print(stu['sname'])
print(stu['sage'])

for x,y in stu.items():
    print(x,y)
```

```
{'sno': 1, 'sname': 'tanuja', 'sage': 22}
tanuja
22
sno 1
sname tanuja
sage 22
```

In [227]:
```
a,b=33,33
if b>a:
   print('b is greater than a')
elif a==b:
   print('a and b are equal')
```

```
a and b are equal
```

In [228]:
```
a,b=22,44
if b>a:
    print('b is greatr than a')
elif a<b:
    print('a is less than b')
```

```
b is greatr than a
```

In [242]:
```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)


colour=["pink","blue","green"]
for x in colour:
    if x == "green":
        continue
    print(x)
```

```
apple
cherry
pink
blue
```

In [245]:
```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")

print("\n")
def my_function(country = "India"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil


I am from Sweden
I am from India
I am from India
I am from Brazil
```

In [246]:
```
x = lambda a, b, c: a + b + c
print(x(5, 6, 2))
```

```
13
```

In [248]:
```
y=lambda a,b,c:a+b+c
print(y(10,10,10))
```

```
30
```

In [2]:
```
x=y=z='orange'
print(x)
print(y)
print(z)
```

```
orange
orange
orange
```

In [4]:
```
fruits=["apple","organe","grapes"]
x,y,z=fruits
print(x)
print(y)
print(z)
```

```
apple
organe
grapes
```

In [5]:
```
cars = ["Ford", "Volvo", "BMW"]
for x in cars:
    print(x)
```

```
Ford
Volvo
BMW
```

In [6]:
```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

```
Hello my name is John
```

In [9]:
```
class person:
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def myfunc(self):
        print('hello my name is '+ self.name)
p1 = person('tanuja',22)
p1.myfunc()
```

```
hello my name is tanuja
```

In [11]:
```
class person:
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def myfunc(self):
        print('hello my name is '+ self.name)
p1=person('sai',22)
p1.myfunc()
```

```
hello my name is sai
```

In [12]:
```
class MyNumbers:
  def __iter__(self):
    self.a = 1
    return self

  def __next__(self):
    x = self.a
    self.a += 1
    return x

myclass = MyNumbers()
myiter = iter(myclass)

print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
```

```
1
2
3
4
5
```

In [23]:
```
x='awesome'
print('python is' + x)
```

```
python isawesome
```

In [24]:
```
x='python is'
y='easy'
z=x+y
print(z)
```

```
python iseasy
```

In [28]:
```
x = 'Awesome'
def myfunc():
  print('python is' +x)
myfunc()

x = 'Awesome'
def myfunc():
    x='fantastic'
    print('python is ' +x)
myfunc()
print('python is '+x)
```

```
python isAwesome
python is fantastic
python is Awesome
```

In [29]:
```
def myfunc():
    global x
    x='fantastic'
myfunc()
print('python is '+x)
```

```
python is fantastic
```

In [37]:
```
class sample:
 def __init__(self):
    self.a = 13
    self.b = 15
s = sample() # instance creation
print(s.a)
print(s.b)
```

```
13
15
```

In [39]:
```
class sample:
 def __call__(self):
     pass
s = sample()
print(callable(s))
```

```
True
```

In [1]:
```
def local_function():
 print("This is a local function")
local_function()
```

```
This is a local function
```

In [2]:
```
def a():
    def b():
        print('inside b')
    print('inside a')
    b()
a()
```

```
inside a
inside b
```

In [4]:
```
def outer_func():
    def inner_func():
        print('nested function')
    inner_func()
outer_func()
```

```
nested function
```

In [6]:
```
def outer_func(a):
    def inner_func():
        print('nested function',a)
    inner_func()
outer_func("'hi!!'")
```

```
nested function 'hi!!'
```

In [11]:
```
def abc(x):
    return x**2
    def xyz(func):
        num=10
        return func(num)
xyz(abc)
```

Out[11]:
```
100
```

In [13]:
```
def generate_power(exponent):
 def power(base):
    return base ** exponent
 return power
g = generate_power(3)
print(g(2))
```

```
8
```

In [15]:
```
def generate_power(exponent):
    def power(base):
        return base** exponent
    return power
g= generate_power(4)
print(g(2))
```

```
16
```

In [18]:
```
def addexclamation(function):
 def add():
    func = function()
    return func +" !!!"
 return add
def sentence():
 return "hello all"
msg = addexclamation(sentence)
print(msg())
```

```
hello all !!!
```

In [22]:
```
def addexclamation(function):
    def add():
        func=function()
        return func+"!!!!!"
    return add
@addexclamation
def sentence():
    return "hello people"
print(sentence())
```

```
hello people!!!!!
```

In [25]:
```
def exclamation(function):
    def add():
        func=function()
        return func+"!!!"
    return add
@exclamation
def sentence():
    return "hiiiiii"
print(sentence())
```

```
hiiiiii!!!
```

In [28]:
```
def addstar(func):
    def star():
        return "*"+func()+"*"
    return star
@addstar
@addexclamation
def sentence():
    return "hello all"
print(sentence())
```

```
*hello all!!!!!*
```

In [34]:
```
import json

# some JSON:
x = '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["age"])
print(y['city'])
print(y['name'])
```

```
30
New York
John
```

In [36]:
```
def args_function(func):
 def getargs(arg1,arg2):
     print(arg1,arg2)
     func(arg1,arg2)
 return getargs
@args_function
def decorator_with_args(num1,num2):
 print("arguments are {} and {}".format(num1,num2))
decorator_with_args(5,6)
```

```
5 6
arguments are 5 and 6
```

In [38]:
```
def args_function(func):
    def getargs(arg1,arg2):
        print(arg1,arg2)
        func(arg1,arg2)
    return getargs
@args_function
def decorator_with_args(num1,num2):
    print("arguments are",num1,num2)
decorator_with_args(5,6)
```

```
5 6
arguments are 5 6
```

In [40]:
```
def args_function(func):
    def getargs(arg1,arg2):
        print(arg1,arg2)
        func(arg1,arg2)
    return getargs
@args_function
def decorator_with_args(str1,str2):
    print("arguments are",str1,str2)
decorator_with_args('d','e')
```

```
d e
arguments are d e
```

In [44]:
```
list1=[x**2 for x in range(10)]
print(list1)
```

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

In [48]:
```
list2=[1,2,3,4,5,66,77,22,44]
list3=[x for x in list2 if x%2==0]
print(list3)
```

```
[2, 4, 66, 22, 44]
```

In [59]:
```
x=lambda a:a**2
print(x(5))
y=lambda b:b*2
print(y(5))
z=lambda a,b:a-b
print(z(10,1))
```

```
25
10
9
```

In [66]:
```
l=[1,2,3,4,5,6,7]
newlst=list(filter(lambda x:x%3==0,l))
print(newlst)
```

```
[3, 6]
```

In [69]:
```
num = []
for i in range(10):
 num.append(i)
newnum = list(map(lambda x:x+2,num))
print(newnum)
```

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

In [76]:
```
s='1 2 3 4'
l=s.split()
print(l)

m=list(map(int,l))
print(m)

n=list(map(lambda x:x**3,m))
print(n)

o=list(filter(lambda x:(x%2==0),n))
print(o)
```

```
['1', '2', '3', '4']
[1, 2, 3, 4]
[1, 8, 27, 64]
[8, 64]
```

In [90]:
```
s='1 2 3 4 5 6 7 8 9'
l=s.split()
print(l)

m=list(map(int,l))
print(m)

n=list(map(lambda x:x**2,m))
print(n)

o=list(filter(lambda x:(x%2==0),n))
print(o)
```

```
['1', '2', '3', '4', '5', '6', '7', '8', '9']
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
[4, 16, 36, 64]
```

In [91]:
```
s="tanuja sai venkata"
l=s.split()
print(l)
```

```
['tanuja', 'sai', 'venkata']
```

In [95]:
```
lst=[2,3,4,5,6,7,8]
dic={x:x+5 for x in lst if x%2==0}
print(dic)
```

```
{2: 7, 4: 9, 6: 11, 8: 13}
```

In [96]:
```
l1=[1,2,3]
l2=[4,5,6]
dic={key:value for (key,value) in zip(l1,l2)}
print(dic)
```

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

In [102]:
```
dic={'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}
new_dic={k:v for (k,v) in dic.items() if v%2!=0 if v<40}
print(new_dic)
new_dic={k:v for (k,v) in dic.items() if v%2!=0 if v>40}
print(new_dic)
```

```
{'john': 33}
{'guido': 57}
```

In [106]:
```
dic={'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}
new_dic=dic
print(new_dic)
print(dic)
```

```
{'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}
{'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}
```

In [107]:
```
dic={'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}
new_dic={k:('old' if v>40 else 'young') for (k,v) in dic.items()}
print(new_dic)
```

```
{'jack': 'young', 'michael': 'old', 'guido': 'old', 'john': 'young'}
```

In [110]:
```
dictionary = {k1: {k2: k1 * k2 for k2 in range(1, 6)} for k1 in range(2, 5)}
print(dictionary)
```

```
{2: {1: 2, 2: 4, 3: 6, 4: 8, 5: 10}, 3: {1: 3, 2: 6, 3: 9, 4: 12, 5: 15}, 4:
{1: 4, 2: 8, 3: 12, 4: 16, 5: 20}}
```

In [112]:
```
l=[1,2,3,4,5,4,3,2,1]
s={x for x in l}
print(s)
```

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

In [116]:
```
input_list= [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]
s={x for x in input_list if x%2==0}
print(s)
```

```
{2, 4, 6}
```

In [118]:
```
l=[2,7,5,0,4,6]
gen=(x for x in l if x%2==0 )
for i in gen:
    print(i)
```

```
2
0
4
6
```

In [119]:
```
gen=(i**2 for i in range(5))
for item in gen :
    print(item)
```

```
0
1
4
9
16
```

In [120]:
```
gen=(i*2 for i in range(10))
for item in gen:
    print(item)
```

```
0
2
4
6
8
10
12
14
16
18
```

In [124]:
```
t=(33,41,58,66,17)
print(t[1])
print(t[4])

print(len(t))
```

```
41
17
5
```

In [126]:
```
t=(33,42,57,17,24)
for i in t:
    print(i)
```

```
33
42
57
17
24
```

In [127]:
```
t=(34,47,59,17,24)
index=0
while index<len(t):
    print(t[index])
    index=index+1
```

```
34
47
59
17
24
```

In [128]:
```
my_tuple = ()
print(my_tuple)
```

```
()
```

In [132]:
```
my_tuple=(1,2,3,4)
print(my_tuple)

my_tuple1=(1,"tanuja",2.3,5)
print(my_tuple1)

my_tuple2=("mouse",[1,2,3,4],{2,8,6,4},(22,69,79))
print(my_tuple2)
```

```
(1, 2, 3, 4)
(1, 'tanuja', 2.3, 5)
('mouse', [1, 2, 3, 4], {8, 2, 4, 6}, (22, 69, 79))
```

In [138]:
```
my_tuple=2,3.4,"mouse","dog"
print(my_tuple)

a,b,c,d=my_tuple
print(a)
print(b)
print(c)
print(d)
```

```
(2, 3.4, 'mouse', 'dog')
2
3.4
mouse
dog
```

In [141]:
```
my_tuple=('hello')
print(type(my_tuple))

my_tuple=('hello',)
print(type(my_tuple))

my_tuple='hello',
print(type(my_tuple))
```

```
<class 'str'>
<class 'tuple'>
<class 'tuple'>
```

In [166]:
```
my_tuple=('k','p','i','p','a','r','t','n','e','r','s')
print(my_tuple[5])
print(my_tuple[6])
```

```
r
t
```

In [169]:
```
my_tuple=('mouse',[1,2,3,4],(9,8,7,6))
print(my_tuple[0][3])
print(my_tuple[1][2])
print(my_tuple[2][2])
```

```
s
3
7
```

In [174]:
```
my_tuple=('t','a','n','u','j','a','n','e','k','k','a','n','t','i')
print(my_tuple[1:7])

print(my_tuple[:7])

print(my_tuple[1:])
print(my_tuple[:])
```

```
('a', 'n', 'u', 'j', 'a', 'n')
('t', 'a', 'n', 'u', 'j', 'a', 'n')
('a', 'n', 'u', 'j', 'a', 'n', 'e', 'k', 'k', 'a', 'n', 't', 'i')
('t', 'a', 'n', 'u', 'j', 'a', 'n', 'e', 'k', 'k', 'a', 'n', 't', 'i')
```

In [177]:
```
my_tuple=('t','a','n','u','j','a','n','e','k','k','a','n','t','i')
print(my_tuple)
del my_tuple
print(my_tuple)
```

```
('t', 'a', 'n', 'u', 'j', 'a', 'n', 'e', 'k', 'k', 'a', 'n', 't', 'i')

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\TANUJA~1\AppData\Local\Temp/ipykernel_4340/2162892029.py in <module>
      2 print(my_tuple)
      3 del my_tuple
----> 4 print(my_tuple)

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

In [181]:
```
my_tuple='t','a','n','u','j','a'
print('a' in my_tuple)
print('b' in my_tuple)
```

```
True
False
```

In [183]:
```
my_dic={}
print(my_dic)


my_dic={1:'apple',2:'boy'}
print(my_dic)
```

```
{}
{1: 'apple', 2: 'boy'}
```

In [188]:
```
my_dic={'name':'tanuja',1:[2,3,4]}
print(my_dic)
```

```
{'name': 'tanuja', 1: [2, 3, 4]}
```

In [191]:
```
my_dict = dict({1:'apple', 2:'ball'})
print(my_dict)

print('\n')
my_dic=dict([(1,'apple'),(2,'ball')])
```

```
{1: 'apple', 2: 'ball'}
```

In [200]:
```
my_dict={'name':'tanuja','age':22}
print(my_dict['name'])
print(my_dict.get('age'))
print(my_dict.get('address'))

print(my_dict['address'])#error
```

```
tanuja
22
None
```

In [202]:
```
my_dict={'name':'sai','age':21}
my_dict['age']=22
print(my_dict)
my_dict['address']='palakol'
print(my_dict)
```

```
{'name': 'sai', 'age': 22}
{'name': 'sai', 'age': 22, 'address': 'palakol'}
```

In [211]:
```
squares = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
print(squares.pop(4))
print(squares)
print(squares.popitem())
print(squares)
squares.clear()
print(squares)
#del squares  #error
#print(squares)
```

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

In [218]:
```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print(1 in squares)
print(2 not in squares)
print(49 in squares)
print( in squares)
```

```
True
True
False
False
```

In [221]:
```
squares = {0: 0, 1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print(all(squares))
print(any(squares))
print(len(squares))
print(sorted(squares))
```

```
False
True
6
[0, 1, 3, 5, 7, 9]
```

In [222]:
```
my_set={1,2,3}
print(my_set)
```

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

In [223]:
```
my_set={1.0,'hello',(1,2,3)}
print(my_set)
```

```
{1.0, (1, 2, 3), 'hello'}
```

In [224]:
```
my_set={1,2,3,4,5,6,5,4,3,2}
print(my_set)
```

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

In [225]:
```
my_set=set([1,2,3,4,5])
print(my_set)
```

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

In [230]:
```
a={}
print(type(a))
print(a)

print('\n')
a=set()
print(type(a))
print(a)
```

```
<class 'dict'>
{}


<class 'set'>
set()
```

In [233]:
```
my_set={1,3}
my_set.add(2)
print(my_set)
print('\n')
my_set.update([7,8,9])
print(my_set)
print('\n')
my_set.update([4, 5], {1, 6, 8})
print(my_set)
```

```
{1, 2, 3}


{1, 2, 3, 7, 8, 9}


{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [235]:
```
square=lambda a:a*a
print(square(6))
```

```
36
```

In [237]:
```
mul=lambda a,b:a*b
result=mul(5,6)
print(result)
```

```
30
```

In [238]:
```
six=lambda :6
result=six()
print(result)
```

```
6
```

In [239]:
```
factorial=lambda a:a*factorial(a-1) if (a>1) else 1
result=factorial(5)
print(result)
```

```
120
```

In [240]:
```
import math
def myfunc(n):
  return lambda a : math.pow(a, n)

square = myfunc(2) #square = lambda a : math.pow(a, 2)
cube = myfunc(3) #cube =  = lambda a : math.pow(a, 3)
squareroot = myfunc(0.5) #squareroot = lambda a : math.pow(a, 0.5)

print(square(3))
print(cube(3))
print(squareroot(3))
```

```
9.0
27.0
1.7320508075688772
```

In [241]:
```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

```
[2, 10, 8, 12, 16, 22, 6, 24]
```

In [1]:
```
class abc:
    pass
```

In [2]:
```
class college:
    def __init__(self):
        print("welcome")
s=college()
```

```
welcome
```

In [5]:
```
class values:
    def __init__(self):
        self.a=13
        self.b=15
s= values()
print(s.a)
print(s.b)
```

```
13
15
```

In [7]:
```
class values:
    def __init__(hi,a,b):
        hi.a=a
        hi.b=b
s=values(int(input()),int(input()))
print(s.a,s.b)
```

```
10
15
10 15
```

In [15]:
```
class student:
    def __init__(self,name,age):
        self.name=name
        self.age=age
s=student('tanuja',22)
print(s.name)
print(s.age)
s1=student('chaitanya',18)
print(s1.name)
print(s1.age)
s2=student('lokesh',23)
print(s2.name)
print(s2.age)
del s2
print(s2.name)
```

```
tanuja
22
chaitanya
18
lokesh
23

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\TANUJA~1\AppData\Local\Temp/ipykernel_17240/1145315473.py in <module>
     13 print(s2.age)
     14 del s2
---> 15 print(s2.name)

NameError: name 's2' is not defined
```

In [27]:
```
class campus:
    def __init__(self,code,name):
        self.code=code
        self.name=name
    def show(self):
        print(self.code,self.name)
c=campus(123,'SATE')
c.show()
```

```
123 SATE
```

In [28]:
```
class student(campus):
    pass
s=student(456,'ciet')
s.show()
```

```
456 ciet
```

In [35]:
```
class student2(campus):
    def __init__(self,code,name,address):

        campus.__init__(self,code,name)
        self.address=address
    def all3(self):
         print(self.code,self.name,self.address)
st=student2(111,'tanuja','palakol')
st.show()
st.all3()
```

```
111 tanuja
111 tanuja palakol
```

In [45]:
```
class A:
    def __init__(self):
        self.str='hello'
        print('A')
class B:
    def __init__(self):
        self.str1='hi'
        print('B')
class C(A,B):
    def __init__(self):
        A.__init__(self)
        B.__init__(self)
        self.str2='bye'
        print('C')
    def printstr(self):
        print(self.str,self.str1,self.str2)
cobj= C()
cobj.printstr()
```

```
A
B
C
hello hi bye
```

In [47]:
```
class A:
    def __init__(self,num1):
        self.num1=num1
    def printnum1(self):
        print(self.num1)
class B(A):
    def __init__(self,num1,num2):
        A.__init__(self,num1)
        self.num2=num2
    def printnum2(self):
        print(self.num2)
class C(B):
    def __init__(self,num1,num2,num3):
        A.__init__(self,num1)
        B.__init__(self,num1,num2)
        self.num3=num3
    def printnum3(self):
        print(self.num3)
cobj=C(1,2,3)
cobj.printnum1()
cobj.printnum2()
cobj.printnum3()
```

```
1
2
3
```

In [49]:
```
class hide:
    def __init__(self):
        self.a=12
        self.__b=13
class view(hide):
    def __init__(self):
        super().__init__()
        self.c=14
v = view()
print(v.a)
#print(v.b)
print(v.c)
```

```
12
14
```

In [53]:
```
class MyDecorator():
    def __init__(self,function):
        self.function=function
    def __call__(self):
        var=self.function()
        print(var,'all')
@MyDecorator
def function():
    return 'hi'
function()
```

```
hi all
```

In [54]:
```
class MyDecorator:
    def __init__(self, function):
        self.function = function

    def __call__(self, *args, **kwargs):
        self.function(*args, **kwargs)

@MyDecorator
def function(name, message ='Hello'):
    print("{} {}!".format(message, name))
function("arathi")
function("arathi","Hi")
```

```
Hello arathi!
Hi arathi!
```

In [56]:
```
class CubeDecorator:

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

    def __call__(self, *args, **kwargs):
        result = self.function(*args, **kwargs)
        return result
@CubeDecorator
def get_cube(n):
    print("given number is:", n)
    return n*n*n

print("Cube of number is:", get_cube(5))
```

```
given number is: 5
Cube of number is: 125
```

In [57]:
```
class Makeupper:
    def __init__(self,func):
        self.func = func

    def __call__(self):
        string = self.func()
        return string.upper()
@Makeupper
def enter_str():
     return "arathi"
enter_str()
```

Out[57]:
```
'ARATHI'
```

In [58]:
```
try:
    f = open('demo1.txt')
    if f.name == 'demo123.txt':
        raise Exception
    except IOError as e:
        print('First!')
    except Exception as e:
        print('Second')
    else:
        print(f.read())
        f.close()
finally:
    print("Executing Finally...")
print('End of program')
```

```
First!
Executing Finally...
End of program
```

In [59]:
```
amount = 10000
if(amount>2999):
    print("You are eligible to purchase Dsa Self Paced")
```

```
You are eligible to purchase Dsa Self Paced
```

In [60]:
```
def AbyB(a , b):
    try:
        c = ((a+b) / (a-b))
    except ZeroDivisionError:
        print("a/b result in 0")
    else:
        print(c)
# Driver program to test above function
AbyB(2.0, 3.0)
AbyB(3.0, 3.0)
```

```
-5.0
a/b result in 0
```

In [62]:
```
try:
    num = int(input("Enter a number: "))
    assert num % 2 == 0
except:
    print("Not an even number!")
else:
    reciprocal = 1/num
    print(reciprocal)
```

```
Enter a number: 5
Not an even number!
```

In [63]:
```
try:
    k = 5//0
    print(k)

except ZeroDivisionError:
    print("Can't divide by zero")
finally:
    print('This is always executed')
```

```
Can't divide by zero
This is always executed
```

In [64]:
```
try:
    raise NameError("Hi there") # Raise Error
except NameError:
    print("An exception")
```

```
An exception
```

In [65]:
```
a = 10
print("Type of a: ", type(a))
print(int(2.5))
```

```
Type of a:  <class 'int'>
2
```

In [66]:
```
b = 24.0
print("Type of b: ", type(b))       #use float(x) -> converts x to float type
print(float(37))
```

```
Type of b:  <class 'float'>
37.0
```

In [67]:
```
c = 3 + 5j
print("Type of c: ", type(c))              #complex(x) is used to convert x to co
myComplex = complex(21)
print(myComplex)
```

```
Type of c:  <class 'complex'>
(21+0j)
```

In [68]:
```
import decimal
print(decimal.Decimal(0.1))
```

```
0.1000000000000000055511151231257827021181583404541015625
```

In [69]:
```
from decimal import Decimal as D
print(D('1.1') + D('2.2'))
print(D('1.2') * D('2.50'))
```

```
3.3
3.000
```

In [70]:
```
from fractions import Fraction
print(Fraction(0.125))
print(Fraction(1.1))
```

```
1/8
2476979795053773/2251799813685248
```

In [71]:
```
from fractions import Fraction
print(Fraction('1.1'))
```

```
11/10
```

In [73]:
```
from fractions import Fraction as F
print(F(1, 3) + F(1, 3))
print(1 / F(5, 6))
```

```
2/3
6/5
```

In [74]:
```
import datetime as dt
dtobj = dt.datetime.now()
print(dtobj)
print(dtobj.year)
print(dtobj.strftime("%A"))
```

```
2021-12-15 22:39:31.137737
2021
Wednesday
```

In [76]:
```
import datetime
x = datetime.datetime(2021, 12, 15)
print(x)
```

```
2021-12-15 00:00:00
```

In [77]:
```
x = min(5, 10, 25)
y = max(5, 10, 25)
print(x)
print(y)
```

```
5
25
```

In [79]:
```
a = abs(-7.25)
print("abs(-7.25) is: ",a)
b = pow(4, 3)
print("pow(4, 3) is: ",b)
```

```
abs(-7.25) is:  7.25
pow(4, 3) is:  64
```

In [80]:
```
import math
print(math.pi)
print(math.exp(2))
print(math.log10(1000))
print(math.factorial(6))
print(math.sqrt(64))
print(math.ceil(1.4))
print(math.floor(1.4))
```

```
3.141592653589793
7.38905609893065
3.0
720
8.0
2
1
```

In [81]:
```
x = 100
y = 3.256
print(x + y)
print(x - y)
print(x * y)
print(x / y)
print(x // y) #floor division
print(x % 7) #modulus operator
print(12 ** 3) #exponent
```

```
103.256
96.744
325.59999999999997
30.712530712530715
30.0
2
1728
```

In [3]:
```
def bubblesort(list):
    for iter_num in range(len(list)-1,0,-1):
        for idx in range(iter_num):
            if list[idx]>list[idx+1]:
                temp = list[idx]
                list[idx] = list[idx+1]
                list[idx+1] = temp
list = [19,2,31,45,6,11,121,27]
bubblesort(list)
print(list)
```

```
[2, 6, 11, 19, 27, 31, 45, 121]
```

In [7]:
```
def bubblesort(l):
    for i in range(len(l)-1,0,-1):
        for index in range(i):
            if l[index]>l[index+1]:
                temp=l[index]
                l[index]=l[index+1]
                l[index+1]=temp
l=[12,88,4,33,76,25,93,67,36]
bubblesort(l)
print(l)
```

```
[4, 12, 25, 33, 36, 67, 76, 88, 93]
```

In [10]:
```
def merge_sort(unsorted_list):
    if len(unsorted_list) <= 1:
        return unsorted_list

    middle = len(unsorted_list) // 2
    left_list = unsorted_list[:middle]
    right_list = unsorted_list[middle:]

    left_list = merge_sort(left_list)
    right_list = merge_sort(right_list)
    return list(merge(left_list, right_list))


def merge(left_half,right_half):
    res = []
    while len(left_half) != 0 and len(right_half) != 0:
        if left_half[0] < right_half[0]:
            res.append(left_half[0])
            left_half.remove(left_half[0])
        else:
            res.append(right_half[0])
            right_half.remove(right_half[0])
    if len(left_half) == 0:
        res = res + right_half
    else:
        res = res + left_half
    return res
unsorted_list = [64, 34, 25, 12, 22, 11, 90]
print(merge_sort(unsorted_list))
```

```
[11, 12, 22, 25, 34, 64, 90]
```

In [22]:
```
def insertion_sort(InputList):
    for i in range(1, len(InputList)):
        j = i-1
        nxt_element = InputList[i]
# Compare the current element with next one
    while (InputList[j] > nxt_element) and (j >= 0):
        InputList[j+1] = InputList[j]
        j=j-1
        InputList[j+1] = nxt_element
    return InputList
list = [19,2,31,45,30,11,121,27]
print('sorted list:',insertion_sort(list))
print(list)
```

```
sorted list: [19, 2, 31, 45, 30, 11, 27, 121]
[19, 2, 31, 45, 30, 11, 27, 121]
```

In [5]:
```
def insertion_sort(list1):
    for i in range(1,len(list1)):
        value=list1[1]
        j=i-1
        while j>=0 and value<list1[j]:
            list1[j+1]=list1[j]
            j-=1
            list1[j+1]=value
        return list1
list1 = [10, 5, 13, 8, 2]
print("The unsorted list is:", list1)
print("The sorted list1 is:", insertion_sort(list1))
```

```
The unsorted list is: [10, 5, 13, 8, 2]
The sorted list1 is: [5, 10, 13, 8, 2]
```

In [37]:
```
def linear_search(values, search_for):
    search_at = 0
    search_res = False
# Match the value with each data element
    while search_at < len(values) and search_res is False:
        if values[search_at] == search_for:
            search_res = True
        else:
            search_at = search_at + 1
    return search_res
l = [64, 34, 25, 12, 22, 11, 90]
print(linear_search(l, 12))
print(linear_search(l, 91))
```

```
True
False
```

In [40]:
```
for i in range(10):
    print(i, end =" ")
print('\n')
for i in range(3, 10, 2):
    print(i, end =" ")
```

```
0 1 2 3 4 5 6 7 8 9

3 5 7 9
```

In [13]:
```
l=[1,0,7,2]
l.sort()
print(l)
```

```
[0, 1, 2, 7]
```

In [14]:
```
import os
entries = os.listdir('my_directory/')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
C:\Users\TANUJA~1\AppData\Local\Temp/ipykernel_23280/1649859026.py in <module>
      1 import os
----> 2 entries = os.listdir('my_directory/')

FileNotFoundError: [WinError 3] The system cannot find the path specified: 'my_
directory/'
```

In [17]:
```
entries = os.listdir('my_directory/')
for entry in entries:
    print(entry)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
C:\Users\TANUJA~1\AppData\Local\Temp/ipykernel_23280/14316079.py in <module>
----> 1 entries = os.listdir('my_directory/')
      2 for entry in entries:
      3     print(entry)

FileNotFoundError: [WinError 3] The system cannot find the path specified: 'my_
directory/'
```

In [20]:
```
os.makedir('./tanuja',exist_ok=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
C:\Users\TANUJA~1\AppData\Local\Temp/ipykernel_23280/2562547992.py in <module>
----> 1 os.makedir('./tanuja',exist_ok=True)

AttributeError: module 'os' has no attribute 'makedir'
```

os.listdir('./data')

In [21]:
```
import os
```

In [22]:
```
os.makedirs('./tanuja',exist_ok=True)
```

In [23]:
```
'tanuja' in os.listdir('.')
```

Out[23]:
```
True
```

In [24]:
```
os.listdir('./tanuja')
```

Out[24]:
```
[]
```

In [27]:
```
my_list=[4, 7, 0, 3]
my_iter = iter(my_list)
print(next(my_iter))
print(next(my_iter))
print(my_iter.__next__())
print(my_iter.__next__())
```

```
4
7
0
3
```

In [28]:
```
print(dir(my_iter))
```

```
['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__g
e__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass_
_', '__iter__', '__le__', '__length_hint__', '__lt__', '__ne__', '__new__', '__
next__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setstate_
_', '__sizeof__', '__str__', '__subclasshook__']
```

In [30]:
```
for city in ["Berlin", "Vienna", "Zurich"]:
    print(city)

print("\n")
for city in ("Python", "Perl", "Ruby"):
    print(city)
for char in "Iteration is easy":
    print(char, end = " ")
```

```
Berlin
Vienna
Zurich


Python
Perl
Ruby
I t e r a t i o n   i s   e a s y
```

In [32]:
```
cities = ["Berlin", "Vienna", "Zurich"]
iterator_object= iter(cities)
print(next(iterator_object))
print(next(iterator_object))
print(next(iterator_object))
```

```
Berlin
Vienna
Zurich
```

In [33]:
```
def iterable(obj):
    try:
        iter(obj)
        return True
    except TypeError:
        return False
for element in [34, [4, 5], (4, 5),
       {"a":4}, "dfsdf", 4.5]:
    print(element, " is iterable : ", iterable(element))
```

```
34  is iterable :  False
[4, 5]  is iterable :  True
(4, 5)  is iterable :  True
{'a': 4}  is iterable :  True
dfsdf  is iterable :  True
4.5  is iterable :  False
```

In [36]:
```
from collections import Counter
lst=[1,3,2,5,7,9,3,2,6,9]
Counter(lst)
```

Out[36]:
```
Counter({1: 1, 3: 2, 2: 2, 5: 1, 7: 1, 9: 2, 6: 1})
```

In [46]:
```
print(lst)
cnt = Counter(lst)
print(cnt[1])
print(cnt.most_common())
```

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

In [48]:
```
cnt=Counter({1:4,2:2,3:3,4:5})
print(list(cnt.elements()))
```

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

In [49]:
```
deduct={1:1,4:1,2:1}
cnt.subtract(deduct)
print(cnt)
```

```
Counter({4: 4, 1: 3, 3: 3, 2: 1})
```

In [50]:
```
deduct={1:2,2:1,3:2,4:2}
cnt.subtract(deduct)
print(cnt)
```

```
Counter({4: 2, 1: 1, 3: 1, 2: 0})
```

In [53]:
```
from collections import namedtuple
a=namedtuple('details','name,age')
s=a('tanuja',22)
print(s)
```

```
details(name='tanuja', age=22)
```

In [60]:
```
from collections import namedtuple
a=namedtuple('details','idno,name')
s=a(12,'tanuja')
print(s)
```

```
details(idno=12, name='tanuja')
```

In [59]:
```
s=a._make([122,'lokesh'])
print(s)
```

```
details(idno=122, name='lokesh')
```

In [61]:
```
from collections import namedtuple
a=namedtuple('courses','name,technology')
s=a('data science','python')
print(s)
```

```
courses(name='data science', technology='python')
```

In [62]:
```
for i in s:
    print(s)
```

```
courses(name='data science', technology='python')
courses(name='data science', technology='python')
```

In [65]:
```
itr=iter(s)
while True:
    try:
        print(next(itr))
    except StopIteration:
        break
```

```
data science
python
```

In [66]:
```
from collections import deque
lst = ['a','e','i','o','u']
dequelst = deque(lst)
print(dequelst)
```

```
deque(['a', 'e', 'i', 'o', 'u'])
```

In [67]:
```
dequelst.append('b')
print(dequelst)
```

```
deque(['a', 'e', 'i', 'o', 'u', 'b'])
```

In [69]:
```
dequelst.append('t')
print(dequelst)
```

```
deque(['a', 'e', 'i', 'o', 'u', 'b', 't'])
```

In [70]:
```
dequelst.appendleft('z')
print(dequelst)
```

```
deque(['z', 'a', 'e', 'i', 'o', 'u', 'b', 't'])
```

In [72]:
```
dequelst.appendleft('s')
print(dequelst)
```

```
deque(['s', 'z', 'a', 'e', 'i', 'o', 'u', 'b', 't'])
```

In [73]:
```
dequelst.pop()
print(dequelst)
```

```
deque(['s', 'z', 'a', 'e', 'i', 'o', 'u', 'b'])
```

In [75]:
```
dequelst.pop()
print(dequelst)
```

```
deque(['s', 'z', 'a', 'e', 'i', 'o', 'u'])
```

In [76]:
```
dequelst.appendleft('z')
print(dequelst)
```

```
deque(['z', 's', 'z', 'a', 'e', 'i', 'o', 'u'])
```

In [77]:
```
dequelst.popleft()
print(dequelst)
```

```
deque(['s', 'z', 'a', 'e', 'i', 'o', 'u'])
```

In [78]:
```
dequelst.pop()
print(dequelst)
```

```
deque(['s', 'z', 'a', 'e', 'i', 'o'])
```

In [83]:
```
from collections import deque
a=['t','a','n','u','j','a']
d=deque(a)
print(d)
```

```
deque(['t', 'a', 'n', 'u', 'j', 'a'])
```

In [84]:
```
itr=iter(a)
while True:
    try:
        print(next(itr),end=' ')
    except StopIteration:
        break
```

```
t a n u j a
```

In [87]:
```
from collections import ChainMap
a={1:'m',2:'n'}
b={3:'x',4:'y'}
d=ChainMap(a,b,)
print(d)
```

```
ChainMap({1: 'm', 2: 'n'}, {3: 'x', 4: 'y'})
```

In [88]:
```
from collections import ChainMap
a={1:'m',2:'n'}
b={3:'x',4:'y'}
c={5:'t',6:'l'}
d=ChainMap(a,b,c)
print(d)
```

```
ChainMap({1: 'm', 2: 'n'}, {3: 'x', 4: 'y'}, {5: 't', 6: 'l'})
```

In [89]:
```
for i in d.items():
    print(i)
```

```
(5, 't')
(6, 'l')
(3, 'x')
(4, 'y')
(1, 'm')
(2, 'n')
```

In [90]:
```
itr=iter(d.items())
while True:
    try:
        print(next(itr))
    except StopIteration:
        break
```

```
(5, 't')
(6, 'l')
(3, 'x')
(4, 'y')
(1, 'm')
(2, 'n')
```

In [91]:
```
from collections import OrderedDict
d = OrderedDict()
d[1] = 't'
d[2] = 'a'
d[3] = 'n'
d[4] = 'u'
d[5] = 'j'
d[6] = 'a'
print(d)
```

```
OrderedDict([(1, 't'), (2, 'a'), (3, 'n'), (4, 'u'), (5, 'j'), (6, 'a')])
```

In [92]:
```
d[2]='t'
print(d)
```

```
OrderedDict([(1, 't'), (2, 't'), (3, 'n'), (4, 'u'), (5, 'j'), (6, 'a')])
```

In [93]:
```
print(d.keys())
for key,val in d.items(): #similar to dictionary
    print(key,val,end=' ')
```

```
odict_keys([1, 2, 3, 4, 5, 6])
1 t 2 t 3 n 4 u 5 j 6 a
```

In [94]:
```
print(d.keys())
```

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

In [95]:
```
itr=iter(d.items())
while True:
    try:
        print(next(itr))
    except StopIteration:
        break
```

```
(1, 't')
(2, 't')
(3, 'n')
(4, 'u')
(5, 'j')
(6, 'a')
```

In [96]:
```
from collections import defaultdict
d = defaultdict(int)
d[1]='abc'
d[2]='efg'
print(d)
print(d[3])
```

```
defaultdict(<class 'int'>, {1: 'abc', 2: 'efg'})
0
```

In [97]:
```
from collections import defaultdict
d=defaultdict(int)
d[1]='AI'
d[2]='ML'
print(d)
```

```
defaultdict(<class 'int'>, {1: 'AI', 2: 'ML'})
```

In [98]:
```
for i in d.items():
    print(i)
```

```
(1, 'AI')
(2, 'ML')
```

In [99]:
```
itr=iter(d.items())
while True:
    try:
        print(next(itr))
    except StopIteration:
        break
```

```
(1, 'AI')
(2, 'ML')
```

In [100]:
```
print(d[3])
```

```
0
```

exceptions

In [3]:
```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print(x/y)
except ZeroDivisionError :
    print("Can't Divide with Zero")
except ValueError:
    print("please provide int value only")
```

```
Enter First Number: 10
Enter Second Number: 2
5.0
```

In [4]:
```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print(x/y)
except ZeroDivisionError :
    print("Can't Divide with Zero")
except ValueError:
    print("please provide int value only")
```

```
Enter First Number: 10
Enter Second Number: tanuja
please provide int value only
```

In [5]:
```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print(x/y)
except ZeroDivisionError :
    print("Can't Divide with Zero")
except ValueError:
    print("please provide int value only")
```

```
Enter First Number: 9
Enter Second Number: 0
Can't Divide with Zero
```

In [7]:
```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print(x/y)
except ArithmeticError :
    print("ArithmeticError")
except ZeroDivisionError:
    print("ZeroDivisionError")
```

```
Enter First Number: 10
Enter Second Number: 5
2.0
```

In [10]:
```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print(x/y)
except ArithmeticError :
    print("ArithmeticError")
except ZeroDivisionError:
    print("ZeroDivisionError")
```

```
Enter First Number: 10
Enter Second Number: 0
ArithmeticError
```

In [11]:
```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print(x/y)
except (ZeroDivisionError,ValueError) as msg:
    print("Plz Provide valid numbers only and problem is: ",msg)
```

```
Enter First Number: 10
Enter Second Number: sailaja
Plz Provide valid numbers only and problem is:  invalid literal for int() with
base 10: 'sailaja'
```

In [ ]:
```

```