x=int(input())

y=int(input())

max\_of\_numbers= x if x>y else y

print(max\_of\_numbers)

print(max(x,y))

print(bool([2]))

print(bool(None))

a=input()

if a:

# do operations on a

else :

print("pls give valid a")

#while loop........

name = "jithu"

counter = 0

while counter<3:

print(name)

counter=counter+1

num1=float(input())

num2=float(input())

counter=0

while counter<=4:

num1=num1+num2

counter=counter+1

print(num1)

num=int(input())

sum\_of\_numbers = 0

counter = 1

while num > 1:

sum\_of\_numbers = sum\_of\_numbers + counter

counter = counter + 1

num = num - 1

print(sum\_of\_numbers)

Fop10

list1=[1,3,4,"5"]

list2=[4,3,4,"hello"]

string="hello"

print(list1==list2)

list also specifically see order

i=0

while i < len(list1):

print(list1[i])

i=i+1

else:

print("I ended properly")

in for loop

for item in string:

print(item)

output = 'h e l l o' in five lines

for item in range(len(list1)):

print(list1[item])

def initial\_spaces(N, i):

for \_ in range(0, N-i)

def left\_part(i):

for item in range(1,i+1):

print(item, end=" ")

def right\_part(i):

for item in range(i-1,0,-1):

print(item, end=" ")

x=10

x=20

y=x

print(y)

y=900

list can also takes -ve index

y=[1,2,3,4,5]

print(y[-1])

output=5

print(y[2:4])

to delet

y[1:4] =[]

print(y)

output=[1,5]

# ABOUT LIST ##

list1=[1,2,3,4]

list2=[11,22,33]

print(list1+list2)

it is doing concatination and creating new list and giving it as a argument to print

\*\* num1+=num2 is not same function in case of list

list1+="abcd" will works but

list1=list1+"abcd" will not work

list3 = list1 \* 3

print(list3)

output=[1,2,3,4,1,2,3,4,1,2,3,4]

list3=list1.append(23)

list4=list1+[23]

print(list1)

append adds the numbwr 23 to list1 but when we print the list it gives None. it just changes the list only

so the output is [1,2,3,4,23]

whatever we give to append it will add as a last element

nested list== list lopala list lopala list and soo on .........

list3=[1,[2,[3,[4]]]]

how to print 4...?

print(list3[1][1][1][0]) -> output=4

list1.extend([1,2,3])-->extend lo em rasina adhi itrabile avvali

print(list1)

output=[1,2,3,4,1,2,3]

list1.insert(2,6)

print(list1)--> output=[1,2,6,4]

in insert first argument is index and 2nd argument is the value that we need to change

list1.remove(5)--> it will remove element 5 in the list if numbers are repeated the it will remove first element

list1.pop(5)-->it will takes index of the list.

# \*\*TUPLE\*\* ##

Collection of elements same like list

very similar to list

tuple1= (5, [1,2])

print(tuple1[-1])-->output=[1,2]

\*\*\* Dictionary \*\*\*

Dictionary (dict)

syntaax of dict

my\_dict {key,values}

key value pair

keys can be any thing like str or int as for values also

must be written in {}

my\_dict = {"a":2,

"b":1,

"c":3

}

# print(my\_dict['a']) --> 2

x = my\_dict.get("d",0) --> 0 --> bcuz d is not present and we given it defalt value as 0

print(x)

my\_dict.items() --> it gives all elements in tuple form

my\_dict.keys() --> it gives all the keys

my\_dict.values() --> it gives all the values asssigend for keys

popitem() --> it takes keys and removes last elements and gives every element except last values if we print

my\_dict.popitem()

print(my\_dict) --> {'a':1,'b':2}

my\_dict.pop('a')

print(my\_dict) --> removes a and its value --> {'b': 2, 'c': 3}

update(key,value) --> it adds the given key value pair to the given Dictionary

d = {

'a':[{'b':[3]}]

}

print(d["a"][0]["b"][0]) --> output = 3 this is called as nested dict

for i in my\_dict.items():

print(i) --> true bcuz it is itreble

<<<< SETS >>>>>>

Unorder, it will not alloed repeated values

set1 = {1,2,3,4}

s2 = { 2,3,4,5,6}

x= set("123")

print(x) ----> which give a set of elements in Unorder

x = set(map(int,input().split()))

print(x)---> {1,2,3,4}

x = set([1,2,3])

print(x) ---> {1,2,3}

print(set1 ^ s2)

set operations 👇

x.union(y) or x | y

x.intersection(y) or x & y

x - y ----> diffecrence

x.symmetric\_difference(y) or x ^ y ----> it removes all the common elements in x & y and give the remaining elements

x.issubset(y)---> that means is x is subset or not --> for checking ---> x <= y same meaning

x.issuperset(y)---> that means is x is superset or not ---> x >= y same meaning

x.isdisjoint(y)---->that means is x intersection y is null set or not

for porper superset --> x > y

for proper subset --> x < y

<<<<agumentation opetations>>>>

besides union we can directly do update --> x.update(y) or x |= y

x intersection\_update(y) ---> x &= y

x difference\_update(y) ---> x -= y

x symmetric\_difference\_update(y) ---> x ^= y