In [1]:
```
print("hello")
```
```
hello
```

In [5]:
```
user = {"name" : "Adharsh","age" : 23}
train_start_time = {100 : ["9.00","11.00"],200:["8.00","12.00"]}
true_caller = {}
while True:
    number = int(input("enter number"))
    if number == 0:
        break
    if number not in true_caller:
        name = input("enter name")
        true_caller[number] = name
    else:
        print(f"name : {true_caller[number]}")

print(true_caller)
```
```
name : manu
name : adharsh
{1234567891: 'adharsh', 1231231231: 'manu'}
```

In [6]:
```
# always false use none in if
if -1: # None , 0,
    print("number is positive")
else:
    print("this will be printed always")
```
```
number is positive
```

In [ ]:
```
#  case 2
num = 10
if num >= 10:
    print("number is positive")
```

In [7]:
```
#nested loop
num = 15
if (num > 10 or 10 > num):
    if num > 10:
        print("number is positive")
    else:
        print("2nd division")
else:
    print("Fail")
```
```
number is positive
```

In [11]:
```
# if elif else
num = 10
if num > 10:
    print("greater then 10")
elif num < 10:
    print("number less than 10")
else:
    print("number is 10")
```
```
number is 10
```

In [ ]:
```
a = 10
b = 20
c = 30

if((a >= b) and (a >= c)):
    bigger = a
```

In [12]:
```
# Range always exclude the stop value
# default start is 0 and step is 1 but stop is mandatory
product = 2
for ele in range(10,51,10):
    product *= ele
print(product)
```

```
24000000
```

In [16]:
```
for i in range(1,10,2):
    print(i)
else: # runs after completing from the for loop
    print("data not available")
```

```
1
3
5
7
9
data not available
```

In [21]:
```
# prime number
def prime(n):
    if n <2:
        return False
    for i in range(2,int(n**.5)+1):
        if n % i  == 0:
            return False
    return True

start  = int(input("enter the start"))
stop = int(input("enter the stop"))
for i in range(start,stop+1):
    if prime(i):
        print(f"{i} : prime")
    else:
        print(f"{i}: not prime")
```

```
20: not prime
21: not prime
22: not prime
23 : prime
24: not prime
25: not prime
26: not prime
27: not prime
28: not prime
29 : prime
30: not prime
31 : prime
32: not prime
33: not prime
34: not prime
35: not prime
36: not prime
37 : prime
38: not prime
39: not prime
40: not prime
41 : prime
42: not prime
43 : prime
44: not prime
45: not prime
46: not prime
47 : prime
48: not prime
49: not prime
50: not prime
51: not prime
52: not prime
53 : prime
54: not prime
55: not prime
56: not prime
57: not prime
58: not prime
59 : prime
60: not prime
```

In [7]:

```
# break --> used to break out of the loop with out executing the next statements
# continue ---> is used to skip the current iteration once we reach the countinue
numbers = [1,2,3,4,5,6,7,8,9,10]
for num in numbers:
    if num % 2 == 0:
        continue
    else:
        print(num)
else:
    print("loop over")
```

```
1
3
5
7
9
loop over
```

In [11]:
```
# username validator
usernames = set()
while True:
    username = input("enter username")
    if len(username) < 8:
        print("you entered a username with less than 8 characters")
        break
    elif username in usernames:
        print("username already used")
    else:
        print("this username is good to go")
        usernames.add(username)
```

```
this username is good to go
username already used
you entered a username with less than 8 characters
```

In [15]:
```
#List operations
fruits = ["apple","orange"]
fruits.append("banana")
print(fruits)
fruits.insert(2,"grapes")
print(fruits)
fruits.extend(["cherry","mango"])
print(fruits)
print(fruits.reverse())
```

```
['apple', 'orange', 'banana']
['apple', 'orange', 'grapes', 'banana']
['apple', 'orange', 'grapes', 'banana', 'cherry', 'mango']
None
```

In [19]:
```
a = [num for num in range(1,11)]
a.sort(reverse = True)
b = sorted(a)
print(b)
```

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

In [22]:
```
m = [1,1.0,2.0,3,2]
b = sorted(m)
m.sort()
print(b)
```

```
[1, 1.0, 2.0, 2, 3]
```

In [26]:
```
lst = "hello i am  studying python"
new_list = lst.split()
print(lst.split())
print(new_list[-1])
print(new_list[1:3])
```

```
['hello', 'i', 'am', 'studying', 'python']
python
['i', 'am']
```

In [28]:
```
lst1 = [1,2,3]
lst2 = [4,5,6]
print(lst1 + lst2)
print(lst1.count(1))
```

```
[1, 2, 3, 4, 5, 6]
1
```

In [29]:
```
power = [i**2 for i in range(10)]
print(power)
```

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

In [41]:
```
matrix = [
    [1,2,3,4],[6,7,8,9],[10,11,12,13]
]
mat = [[row[i] for row in matrix] for i in range(len(matrix)+1)]
print(mat)
```

```
[[1, 6, 10], [2, 7, 11], [3, 8, 12], [4, 9, 13]]
```

In [ ]:
```
# -------------------assignment 1 -----------------
# compute transpose of matrix using function and without using function
```

In [52]:
```
# using tanspose function
matrix = [
    [1,2,3,4],
    [5,6,7,8],
    [9,10,11,12]
]
```

In [43]:
```
# Tuple
tpl = (1,2,3,5,[1,2,3])
print(tpl)
tpl[4].append(10)
print(tpl)
```

```
(1, 2, 3, 5, [1, 2, 3])
(1, 2, 3, 5, [1, 2, 3, 10])
```

In [46]:
```
# kyc example
data = ("adharsh",[1234123412,23,100000])
print(data)
data = data + data
print(data)
```

```
('adharsh', [1234123412, 23, 100000])
('adharsh', [1234123412, 23, 100000], 'adharsh', [1234123412, 23, 100000])
```

In [47]:
```
tpl = ('hi')
print(type(tpl))
```

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

In [48]:
```
tpl = ('hi',) # put comma to make this tuple
print(type(tpl))
```

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

In [53]:
```
tpl = ('hi',[1,2,3],(4,5,6))
tpl[1] = 30
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 2
      1 tpl = ('hi',[1,2,3],(4,5,6))
----> 2 tpl[1] = 30

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

In [55]:
```
# string also immutable
s = "python"
s[2] = "p"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[55], line 3
      1 # string also immutable
      2 s = "python"
----> 3 s[2] = "p"

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

In [56]:
```
# repeting elements
tpl = (('hi',) * 5)
print(tpl)
```

```
('hi', 'hi', 'hi', 'hi', 'hi')
```

In [57]:
```
#Tuple deletion
# using del to delete entire tuple at once
tpl = (10,20,30)
print(tpl)
del tpl
print(tpl)
```

```
(10, 20, 30)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[57], line 6
      4 print(tpl)
      5 del tpl
----> 6 print(tpl)

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

In [65]:
```
# Tuple count
tpl = (1,2,3,4,1,2,3,4)
print(tpl.count(1))
# index give the index of the first occurance of the element
print(tpl.index(4))
# tuple membership using in and not in
print(1 in tpl)
print(2 not in tpl)
# tuple length using len() function
print(len(tpl))
new_tpl = sorted(tpl,reverse = True)
print(new_tpl)
# max,min,sum in tuple
print(min(tpl),max(tpl),sum(tpl))
```

```
2
3
True
False
8
[4, 4, 3, 3, 2, 2, 1, 1]
1 4 20
```

In [66]:
```
import statistics as st # to find the mean median
print(st.mean((1,2,3,4)))
```

```
2.5
```

In [67]:
```
import math # for mathamatical functions
math.factorial(5)
```

Out[67]:
```
120
```

In [68]:
```
# sets
# it is immutable we can add ,remove data
# set used for all kind of mathematical functions union,intersection etc..
# set doen't allow duplicate values
# provide unique ans sorted output
st = {1,2,3,4}
print(type(st))
```

```
<class 'set'>
```

In [73]:
```
#--------------------Assignment 2 -------------------------------------
# create a list from other list using copy
lst1 = [1,2,3,4]
lst2 = []
lst3 = lst1
lst2 = list.copy(lst1)
print(lst2)
print(id(lst2))
print(id(lst3))
print(id(lst1))
```

```
[1, 2, 3, 4]
1947984229440
1947984224128
1947984224128
```

In [78]:
```
# adding value using add and update
st = set()
st.add(1)
st.update((2,3,4,1),[5,6,7]) # update take only iterable objects
print(st)
```

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

In [80]:
```
# remove an element from set

# remove
st.remove(10)
print(st)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[80], line 4
      1 # remove an element from set
      2 
      3 # remove
----> 4 st.remove(10)
      5 print(st)

KeyError: 10
```

In [81]:
```
# using discrd
st = {1,2,3,4,5,6,7,8,9}
st.discard(10)
print(st)
```

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

In [ ]:
```
# ---------------assignment 3 --------------------
# reassign the poped value
```

In [ ]:
```
# union,intersction and difference
st1 = {1,2,3,4,5}
st2 = {1,3,6,7,10}
print(st1 | st2)
print(st1 & st2)
print(st1 - st2)

#udhdd
```

In [1]:
```
# Ductionary store as key value pair
# key must be unique
my_dict = {}
print(type(my_dict))
```

```
<class 'dict'>
```

In [3]:
```
my_dict = {1:'a',2:'b',2:'c'} # update the key 2 with value c
print(my_dict)
```

```
{1: 'a', 2: 'c'}
```

In [4]:
```
# acessing
my_dict = {1:'a',2:'b',2:'c'}
print(my_dict[2])
```

```
c
```

In [6]:
```
# get to handle exception
my_dict = {1:'a',2:'b',2:'c'}
print(my_dict.get(3,0)) # if key 3 is not present it will  return 0 here
```

```
0
```

In [7]:
```
# add or modify
my_dict= {1:'a',2:'b'}
my_dict[3] = 'c'
print(my_dict)
```

```
{1: 'a', 2: 'b', 3: 'c'}
```

In [11]:
```
# pop and popitem
my_dict = {1:'a',2:'b',3:'c'}
print(my_dict.pop(3))
print(my_dict)
print(my_dict.popitem()) # delete key and value in lifo manner return tuple of key
```

```
c
{1: 'a', 2: 'b'}
(2, 'b')
```

In [13]:
```
# del by key
my_dict = {1:'a',2:'b',3:'c'}
del my_dict[2]
print(my_dict)
my_dict.clear() # it clear the data but the structure  exist
print(my_dict)
del my_dict # it deletes the entire dictionary the object
print(my_dict)
```

```
{1: 'a', 3: 'c'}
{}
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 8
      6 print(my_dict)
      7 del my_dict
----> 8 print(my_dict)

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

In [16]:
```
my_dict = {1:'a',2:'b',3:'c'}
new_dict = my_dict.copy()
print(new_dict)
print(id(new_dict),id(my_dict))
```

```
{1: 'a', 2: 'b', 3: 'c'}
1738727691136 1738727695168
```

In [19]:
```
# forming dictionary from list using fromkeys
sub = dict.fromkeys(['maths','science','history'],'0') # all key have the same val
print(sub)
```

```
{'maths': '0', 'science': '0', 'history': '0'}
```

In [21]:
```
my_dict = {1:'a',2:'b',3:'c'}
print(my_dict.items())
print(my_dict.keys())
print(my_dict.values())
```

```
dict_items([(1, 'a'), (2, 'b'), (3, 'c')])
dict_keys([1, 2, 3])
dict_values(['a', 'b', 'c'])
```

In [25]:
```
lst = ()
print(dir(lst))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__dir
__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem_
_', '__getnewargs__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_sub
class__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new_
_', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof
__', '__str__', '__subclasshook__', 'count', 'index']
```

In [29]:

```
# dict comprehension
my_dict = {k:v**2 for k,v in enumerate(range(10)) if v % 2 == 0}
print(my_dict)
```

```
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64}
```

In [ ]: