```
!conda install -c conda-forge pandoc

#if statment

val=input("Enter the number")
val_float=float(val)
if(val_float>100):
    print("the number is grater than 100")


#Even or odd if-Else
val=input("Enter the number")
val_float=float(val)
if(val_float%2==0):
    print("the number is even")
else:
    print("the number is odd")

age=float(input("enter the age"))
if(age<18):
    print("you are not allow to vote")
else:
    print("you are allow to vote")


#Nested if-else
age=float(input("enter the number"))
if(age<18):
    Print("you are small boy")
elif(age>18 and age<=30):
    print("you are boy")
elif(age>30 and age<=50):
    print("you are man")
else:
    print("senior citizen")

#Loop Statment
#For Loop While Loop
lst=[1,2,3,4,5,6]
for i in lst:
  print(i)

lst=[1,2,3,4,5,6]
for i in lst:
  print(i+2)

lst=[1,2,3,4,5,6]
for i in lst:
  print(i*2)
```

```
lst=[1,2,3,4,5,6]
sum1=0
for i in lst:
    sum1=sum1+i


print(sum1)

#Find sum of even odd number
lst=[1,2,3,4,5,6]
esum=0
osum=0
for i in lst:
        if(i%2==0):
         esum=esum+i
        else:
            osum=osum+i


print("sum of even no is{}".format(esum))
print("sum of odd no is{}".format(osum))

i=1
while(i<=10):
    print(i)
    i=i+1

i=10
while(i>=0):
    print(i)
    i=i-1

##break
x=1
while(x<7):

    if x==4:
        break
    print(x)
    x=x+1

## continue
x=0
while x<7:
    x=x+1
    if x==4:
        continue
    print(x)
```

```
#Python Operator
#Logical Operator
True and False
```

Python Operators Logical Equality Comparision Arithmetic 4.1.1 Logical Operators In python following keywords are used for boolean operations -

Keywords Meaning not unary negation and conditional AND or conditional OR

```
True or False

not True

not False

#AND MEANS TRUE OR TRUE=TRUE
age=int(input("enter the age"))
if age>18 and age<=40:
        print("mid age")
else:
    print("not in range")



#OR MEANS TRUE OR FALSE=TRUE
age=int(input("enter the age"))
if age>18 or age<=40:
        print("mid age")
else:
    print("not in range")

age=int(input("Enter the age"))

if age==18:
    print("You are in the teenager age")
```

Equality Operators Following operations are present in python for equlity check operation-

Operators Meaning is a is b returns true if variable/identifiers a and b points to the same object is not a is not b returns true if variable/identifiers a and b points to the different object == a == b returns true if variable/identifiers a and b has same value != a != b returns true if variable/identifiers a and b has different value

```
a="Krish"
b="Krish"
print(id(a))
print(id(b))

a is b
```

```
lst1=[1,2,3]
lst2=[1,2,3]
print(id(lst1))
print(id(lst2))

lst1 is lst2

a is not b

lst1 is not lst2

lst1 != lst2

"Siddhesh" != "Siddhesh1"
```

comparison operators Operation Meaning < less than <= less than or equal to

```
greater than
```

= greater than or equal to

```
marks= float(input("Enter the marks"))

if marks>=35:
    print("Pass")
    if marks>=50 and marks<=70:
        print("First")
elif marks<35:
    print("Fail")

#Arithmatic opertor
10+10
```

Arithmetic Operators Operation Meaning

- addition
- subtraction
- multiplication / true division // integer division % the modulo operator

```
20-10

10*20

20%2

20/2

20//2

#Pthon Number Method
abs()
```

```
abs(x) will return the absolute value of a number x which we pass in
argument. The number x can be integer, float, complex,..

#Absolute Method
abs(10)

abs(-10)
```

ceil() ceil(x) will return the ceiling value of a number x which we pass in argument. The ceiling value of a number x will be the smallest integer not less than x

Note :- This function will not be accessible directly using ceil() method. Math module will be required to access this method.

```
#ceil Method

import math

math.ceil(43.67)

math.ceil(46.20)
```

##floor

```
math.floor(43.1)

math.floor(44.2)

math.floor(48.1)
```

exp() exp(x) will return the exponential value of a number x which we pass in argument.

```
math.exp(10)
```

fabs()

```
math.fabs(10.53)

math.fabs(-10)
```

log(x)

```
math.log(10)

math.log(15)
```

Max()

```
max(10,12,5,76,100)
```

```
max(-55,-44,-33)
```

Min()

```
min(0,100,4,5,6,3)

min(-1,0)
```

sqrt()

```
math.sqrt(16)

math.sqrt(9)

math.sqrt(25)
```

pow()

```
import math
math.pow(20,5)
```

Triggnometric functions

```
import math
math.sin(0)

math.cos(90)

math.cos(0)

math.tan(90)
```

Lists A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets [ ]

```
str1="siddhesh"
print(str1)
type(str1)

str1="Siddhesh"
print(str1)

##Lists
##mutable
lst=[1,2,3,4,"Siddhesh","Badakh"]
print(lst)
```

```
lst[4]="badakh"
print(lst[4])

lst=list((1,2,3,4,5))
type(lst)
print(lst)

for i in lst:
    print(i**2)

min(lst)

type([])

lst=['Mathematics', 'chemistry', 100, 200, 300, 204]

len(lst)

type(lst)
```

Append

```
lst

lst.append("siddhesh")
lst

lst.append(["sid","Badakh"])
lst

lst[2:6]

lst[6]

lst[1:6]
```

Insert

```
lst

lst.insert(2,"Siddhesh")

lst

lst=[1,2,3]

lst.append([4,5])

lst
```

Extend Method

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

lst.append([8,9])
lst

lst
```

Various Operations that we can perform in List

```
lst=[1,2,3,4,5]

sum(lst)

lst*5

for i in lst:
    print(i/5)

lst
```

Pop() Method

```
lst.pop()

lst

lst.pop(2)

lst
```

count():Calculates total occurrence of given element of List

```
lst=[1,1,2,3,4,5]
lst.count(1)

#length:Calculates total length of List
len(lst)

# index(): Returns the index of first occurrence. Start and End index
are not necessary parameters
lst.index(1,1,4)

##Min and Max
min(lst)

max(lst)
```

SETS A Set is an unordered collection data type that is iterable, mutable, and has no duplicate elements. Python's set class represents the mathematical notion of a set.This is based on a data structure known as a hash table

```
set_var={1,2,3,4,3}

set_var

set_var={"Avengers","IronMan",'Hitman'}
print(set_var)
type(set_var)

set_var.add("Hulk")

print(set_var)

set1={"Avengers","IronMan",'Hitman'}
set2={"Avengers","IronMan",'Hitman','Hulk2'}

set2.intersection_update(set1)

set2

set2.difference_update(set1)

print(set2)
```

Dictionaries A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

```
dic={}

type(dic)

type(dict())

set_ex={1,2,3,4,5}

type(set_ex)

## Let create a dictionary

my_dict={"Car1": "Audi", "Car2":"BMW","Car3":"Mercidies Benz"}

type(my_dict)

my_dict['Car1']

# We can even loop throught the dictionaries keys

for x in my_dict:
    print(x)

# We can even loop throught the dictionaries values

for x in my_dict.values():
    print(x)
```

```
# We can also check both keys and values
for x in my_dict.items():
    print(x)

## Adding items in Dictionaries

my_dict['car4']='Audi 2.0'

my_dict

my_dict['Car1']='MAruti'

my_dict
```

Nested Dictionary

```
car1_model={'Mercedes':1960}
car2_model={'Audi':1970}
car3_model={'Ambassador':1980}

car_type={'car1':car1_model,'car2':car2_model,'car3':car3_model}

print(car_type)

## Accessing the items in the dictionary

print(car_type['car1'])

print(car_type['car1']['Mercedes'])
```

Tuples

```
## create an empty Tuples

my_tuple=tuple()

type(my_tuple)

my_tuple=("Krish","Ankur","John")

my_tuple=('Hello','World')

print(type(my_tuple))
print(my_tuple)

type(my_tuple)
```