Python

In [1]:
```
print("This is my first programm")
```

```
This is my first programm
```

In [2]:
```
print("HEllo World")
```

```
HEllo World
```

Variables

1. **Variables** is a container to store the value
2. **Variables** in python can store characters, integer, boolean, values etc.,

In [3]:
```
var = 10


var
```

Out[3]:
```
10
```

In [4]:
```
var = 20

var
```

Out[4]:
```
20
```

In [5]:
```
var_1 = 10

var_1
```

Out[5]:
```
10
```

In [6]:
```
1var = 10

1var
```

```
  Cell In[6], line 1
    1var = 10
    ^
SyntaxError: invalid decimal literal
```

In [7]:
```
var var = 10

var var
```

```
  Cell In[7], line 1
    var var = 10
        ^
SyntaxError: invalid syntax
```

In [8]:
```
@var = 10
```

```
  Cell In[8], line 1
    @var = 10
     ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

Keywords:

1. **Keywords** are reserved words that **cannot be considered as variables.**
2. There are a total of **35 reserved words.**

In [9]:
```
import keyword

print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'fo
r', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'no
t', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

In [10]:
```
len(keyword.kwlist)
```

Out[10]: 35

In [11]:
```
False = 10
```

```
  Cell In[11], line 1
    False = 10
    ^
SyntaxError: cannot assign to False
```

Data Types

It might be we have to store different set values like **intgers,strings,booleans,flaot,complex**
EX:

1. Integres: 1,10,100,901,783
2. Float : 5.25,89.5,1000.7
3. Complex: 3+4j,5+2j **where j denotes the imaginary unit**
4. Strings: 'hii',"Hello world' **where strings are enclosed with quotes**

In [12]:
```
x = 10

type(x)
```

Out[12]:
```
int
```

In [13]:
```
x = 79.5

type(x)
```

Out[13]:
```
float
```

In [14]:
```
#Whenever a string contains a number, it should be converted into an integer.
x = "100"

print(int(x))
```

```
100
```

In [15]:
```
x = 'rama'

type(x)
```

Out[15]:
```
str
```

In [16]:
```
x = 10+2j

print(type(x))

print("imaginary_value",x.imag)
print("real_Value",x.real)
```

```
<class 'complex'>
imaginary_value 2.0
real_Value 10.0
```

In [17]:
```
#only use j for imaginary unit
x = 10+2i

print(x)
```

```
  Cell In[17], line 2
    x = 10+2i
          ^
SyntaxError: invalid decimal literal
```

In [18]:
```
x = True

type(x)
```

Out[18]:
```
bool
```

Type casting

It means to convert into one data type into another data type is called Type castig

Numeric values such as [integers, floats, and complex] numbers can be converted into strings. However, strings cannot be converted into numeric values

In [19]:
```
#int
x = 10

print("flaot value", float(x))
print("string value", str(x))
print("boolean value", bool(x))
print("complex value", complex(x))
```

```
flaot value 10.0
string value 10
boolean value True
complex value (10+0j)
```

In [20]:
```
#flaot
x = 10.7

print("int value", int(x))
print("string value", str(x))
print("boolean value", bool(x))
print("complex value", complex(x))
```

```
int value 10
string value 10.7
boolean value True
complex value (10.7+0j)
```

In [21]:
```
#Complex
x = 10+2j

print("int value_imageinary", int(x.imag))
print("int value_real_Value",int(x.real))
print("float value_imageinary", float(x.imag))
print("float value_real_Value",float(x.real))
print("string value", str(x))
print("boolean value", bool(x))
```

```
int value_imageinary 2
int value_real_Value 10
float value_imageinary 2.0
float value_real_Value 10.0
string value (10+2j)
boolean value True
```

In [22]:
```
x = 'Ram'

print(int(x))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[22], line 3
      1 x = 'Ram'
----> 3 print(int(x))

ValueError: invalid literal for int() with base 10: 'Ram'
```

In [23]:
```
x = 'Ram'

print(float(x))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[23], line 3
      1 x = 'Ram'
----> 3 print(float(x))

ValueError: could not convert string to float: 'Ram'
```

In [24]:
```
print(float(y))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[24], line 1
----> 1 print(float(y))

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

In [25]:
```
int('12.25')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[25], line 1
----> 1 int('12.25')

ValueError: invalid literal for int() with base 10: '12.25'
```

In [26]:
```
type(float('12.25'))
```

Out[26]:
```
float
```

In [27]:
```
type(int(float("12.25")))
```

Out[27]:
```
int
```

In [28]:
```
int(float("12.25"))
```

Out[28]:
```
12
```

When using zero, it is converted into a boolean value, which evaluates to false.

In [29]:
```
x = 0.0
X = 0
y = 0+0j

print(bool(x))
print(bool(X))
print(bool(y))
```

```
False
False
False
```

In [30]:
```
x = -1.0
X = 2
y = 2+0j

print(bool(x))
print(bool(X))
print(bool(y))
```

```
True
True
True
```

User input

We can get an user input using the **input keyword**

When using the **input** keyword, it always stores the data as a string type.

In [32]:
```
input('enter your name: ')
```

```
enter your name: RAMA GOPALA KRISHNA
```

Out[32]:
```
'RAMA GOPALA KRISHNA'
```

In [33]:
```
input()
```

```
8
```

Out[33]:
```
'8'
```

In [34]:
```
input("Enter your Age: ")
```

```
Enter your Age: 22
```

Out[34]:
```
'22'
```

In [35]:
```
x = input("Enter your Age: ")
```

```
Enter your Age: RAMA
```

In [36]:
```
x
```

Out[36]:
```
'RAMA'
```

In [37]:
```
type(x)
```

Out[37]:
```
str
```

When using the input keyword, it always stores the data as a **string type,** which may then need to be converted into **numerical data.**

In [39]:
```
x = int(input("enter your age: "))
```

```
enter your age: 22
```

In [40]:
```
type(x)
```

Out[40]:
```
int
```

In [41]:
```
x = float(input("enter your age: "))
```

```
enter your age: 22.0
```

In [42]:
```
type(x)
```

Out[42]:
```
float
```

In [43]:
```
x = int(input("enter your age: "))
```

```
enter your age: 22
```

print statement

We can print anything on console using **print keyword**

In [44]:
```
x = 25

print(x)
```

```
25
```

In [45]:
```
print('my age is', x)
```

```
my age is 25
```

In [46]:
```
name = input("enter your name:")
your_age = int(input("enter your age: "))

print("your name is",name,"your age is",your_age)
```

```
enter your name:RAMA GOPALA KRISHNA
enter your age: 22
your name is RAMA GOPALA KRISHNA your age is 22
```

Formattiing

In [48]:
```
name = input("enter your name:")
your_age = int(input("enter your age: "))

print("your name {a} and your age {b}".format(a=name,b=your_age))
```

```
enter your name:RAMA GOPALA KRISHNA
enter your age: 22
your name RAMA GOPALA KRISHNA and your age 22
```

In [49]:
```
print("your name {a} and your age {b}".format(a = 'rama',b = 20))
```

```
your name rama and your age 20
```

In [50]:
```
#using the index as for formatting
x = 25
y = 19

print("your name {0} and your age {1}".format(x,y))
```

```
your name 25 and your age 19
```

In [51]:
```
#using the index as for formatting
x = 25
y = 19

print("your name {0} and your age {0}".format(x,y))
```

```
your name 25 and your age 25
```

In [52]:
```
x = 25
y = 19

print("your name {} and your age {}".format(x,y))
```

```
your name 25 and your age 19
```

Operators

Arithamatic Operators

Addition

In [53]:
```
x = 10+4

print(x)
```

```
14
```

In [54]:
```
x = 10+20+30

print(x)
```

```
60
```

In [55]:
```
x = 10.3+20.0+19.0

print(x)
```

```
49.3
```

In [56]:
```
x = 10+20.9+30

print(x)
```

```
60.9
```

In [57]:
```
x = 21+2j+334+4
print(x)
```

```
(359+2j)
```

Subtraction

In [58]:
```
x = 10-4
```

In [59]:
```
10-10-89
```

Out[59]: -89

In [60]:
```
100-2332-293299
```

Out[60]: -295531

In [61]:
```
x = 21+2j-334-4
print(x)
```

```
(-317+2j)
```

Multiplication

In [62]:
```
x = 10*4
print(x)
```

```
40
```

In [63]:
```
y = 10*7
print(y)
```

```
70
```

In [64]:
```
x = 10*20*5

print(x)
```

```
1000
```

In [65]:
```
x = 0.02*8*10

print(x)
```

```
1.6
```

In [66]:
```
x = 21+2j*4*5
print(x)
```

```
(21+40j)
```

Division

In [67]:
```
x = 10/4

print(x)
```

```
2.5
```

In [68]:
```
x = 100/25

print(x)
```

```
4.0
```

In [69]:
```
x = 10025/34

print(x)
```

```
294.8529411764706
```

In [70]:
```
x = 1000/40/90
print(x)
```

```
0.2777777777777778
```

In [71]:
```
x = 21+2j/4/5
print(x)
```

```
(21+0.1j)
```

Exponentiation

Exponentiation is the mathematical operation of raising a number to a **certain power.**

In [72]:
```
x = 2**2

print(x)
```

```
4
```

In [73]:
```
x = 2**10

print(x)
```

```
1024
```

In [74]:
```
x = 10000**10

print(x)
```

```
100000000000000000000000000000000000000000
```

In [75]:
```
#using bultin function like pow
pow(2,3)
```

Out[75]: 8

In [76]:
```
pow(10000,10)
```

Out[76]: 100000000000000000000000000000000000000000

Quotient

In [77]:
```
#without using the Quotient
x = 10/3
print(x)
```

```
3.3333333333333335
```

In [78]:
```
x = 10//3
print(x)
```

```
3
```

In [79]:
```
x = 99//10

print(x)
```

```
9
```

In [80]:
```
x = 110//10

print(x)
```

```
11
```

Modules

Modulus refers to the **remainder of a division** operation between two numbers.

In [81]:
```
x = 10%3

print(x)
```

```
1
```

In [82]:
```
x  = 68%11

print(x)
```

```
2
```

In [83]:
```
x = 100 % 10

print(x)
```

```
0
```

In [84]:
```
x = 70.75%5

print(x)
```

```
0.75
```

Assignment operartors

Assignment Operators are used to assign a **value to a variable**

In [85]:
```
# x is a variable it assign the value 5
x = 5

print(x)
```

```
5
```

In [86]:
```
x = 5

y = x+10

print(y)
```

```
15
```

In [87]:
```
x = 5

x+=10

print(x)
```

```
15
```

In [88]:
```
x = 5

x-=20

print(x)
```

```
-15
```

In [89]:
```
x = 5

x*=20

print(x)
```

```
100
```

In [90]:
```
x = 20

x/=5

print(x)
```

```
4.0
```

In [91]:
```
x = 20

x//=5

print(x)
```

```
4
```

In [92]:
```
x = 5

x**=2

print(x)
```

```
25
```

In [93]:
```
x = 5

x%=2

print(x)
```

```
1
```

Relational operators

1. Relational operators compare two values in programming to ascertain their relationship, such as greater than, less than, equal to, or not equal to.
2. Relational Operators when used returns a boolean value(True,False)

In [94]:
```
x = 10 > 7

print(x)
```

```
True
```

In [95]:
```
x = 10 < 7

print(x)
```

```
False
```

less then

In [96]:
```
x = 10
y = 12

print(x<y)
```

```
True
```

less then equal to

In [97]:
```
x = 11
y = 12

print(x<=y)
```

```
True
```

In [98]:
```
x = 13
y = 12

print(x<=y)
```

```
False
```

Greater than

In [99]:
```
x = 5
y = 3

print(x > y)
```

```
True
```

In [100]:
```
x = 12
y = 11

print(x > y)
print(y > x)
```

```
True
False
```

greater then equal to

In [101]:
```
x = 5
y = 3

print(x >= y)
```

```
True
```

In [102]:
```
x = 12
y = 16

print(x >= y)
```

```
False
```

Equal to Equal

In [103]:
```
x = 5
y = 3

print(x == y)
```

```
False
```

In [104]:
```
x = 15
y = 15

x == y
```

Out[104]:
```
True
```

Not equal to

In [105]:
```
x = 5
y = 3

print(x != y)
```

```
True
```

In [106]:
```
x = 20
y = 20

x != y
```

Out[106]:
```
False
```

Boolean value

In [107]:
```
bool(0)
```

Out[107]: False

In [108]:
```
bool(1)
```

Out[108]: True

In [109]:
```
bool(-12345)
```

Out[109]: True

In [110]:
```
bool(0.00000000000000001)
```

Out[110]: True

In [111]:
```
True*True
```

Out[111]: 1

In [112]:
```
True*False
```

Out[112]: 0

In [113]:
```
False*False
```

Out[113]: 0

In [114]:
```
True*False
```

Out[114]: 0

In [115]:
```
True**False
```

Out[115]: 1

In [116]:
```
True+True+True
```

Out[116]: 3

In [117]:
```
False+False
```

Out[117]: 0

In [118]:
```
#and
True & True
```

Out[118]: True

In [119]:
```
True & False
```

Out[119]: False

In [120]:
```
False & False
```

Out[120]: False

In [121]:
```
#or
True | True
```

Out[121]: True

In [122]:
```
True | False
```

Out[122]: True

In [123]:
```
False | False
```

Out[123]: False

In [124]:
```
x = 10

(x>5) & (x>9)
```

Out[124]: True

In [125]:
```
x = 10

print((x>5) & (x<9))
print((x>5) | (x<9))
```

```
False
True
```

Logical Operators

Logical operators manipulate boolean values to produce boolean outcomes in programming, commonly used in conditional statements and boolean expressions [True = 1 False = 0]

they are classified into 3 types **1.and,2.or,3.not**

In [126]:
```
int(True)
```

Out[126]: 1

In [127]:
```
int(False)
```

Out[127]: 0

In [128]:
```
#and
x = 5

print(x > 3 and x < 10)
```

```
True
```

In [129]:
```
#and
x = 5

print(x > 3 and x > 10)
```

```
False
```

Identity Operators

In [130]:
```
x = ["apple", "banana"]
y = ["apple", "banana"]


print(x is y)
```

```
False
```

In [131]:
```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is z)
```

```
True
```

In [132]:
```
x = ["apple", "banana"]
y = ["apple", "banana"]

print(x == y)
```

```
True
```

In [133]:
```
x = ["apple", "banana"]
y = ["apple", "banana"]

z = x

print(x is not z)
```

```
False
```

In [134]:
```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is not y)
```

```
True
```

In [135]:
```
x = ["apple", "banana"]
y = ["apple", "banana"]

print(x != y)
```

```
False
```

Membership Operators

In [136]:
```
x = ["Ram", "gopala"]

print("gopala" in x)
```

```
True
```

In [137]:
```
x = ["Ram", "gopala"]

print("gopala" not in x)
```

```
False
```

conditional statement:

In [138]:
```
x = 10
if x > 5:
    print ("X is greater than 5")
else:
    print("x is less than x")
```

```
X is greater than 5
```

In [139]:
```
x = 15
if x < 5:
    print("x is less than 5")
else:
    print("5 is greater than x")
```

```
5 is greater than x
```

In [140]:
```
x = 15
if (x >= 15):
    print("x is lesser than equl to 5")
elif((x > 5) and (x < 10)):
    print("x is between 5 nd 15")
else:
    print("x is greater than equal to 10")
```

```
x is lesser than equl to 5
```

In [141]:
```
x = 15
if (x < 15):
    print("x is lesser than equl to 5")
elif((x > 5) and (x < 10)):
    print("x is between 5 nd 15")
else:
    print("x is greater than equal to 10")
```

```
x is greater than equal to 10
```

In [143]:
```
# even and odd
n = int(input("enter your number"))
if (n%2 == 0):
    print(n,"n is even")
elif(n%2 == 1):
    print(n,"n is odd")
else:
    print("check your input")
```

```
enter your number88
88 n is even
```

For loop

In [144]:
```
for i in range(0,5):
    print(i)
```

```
0
1
2
3
4
```

In [145]:
```
for i in range(0,100,10):
    print(i)
```

```
0
10
20
30
40
50
60
70
80
90
```

In [146]:
```
for i in range(0,5):
    print("Hello")
```

```
Hello
Hello
Hello
Hello
Hello
```

In [147]:
```
x = 1
for i in range(0,5):
    print(x)
```

```
1
1
1
1
1
```

In [148]:
```
for i in range(0,5):
    print(i * 3)
```

```
0
3
6
9
12
```

In [149]:
```
x = 10
for i in range(0,5):
    y=x-i
    print(x,i,"=",y)
```

```
10 0 = 10
10 1 = 9
10 2 = 8
10 3 = 7
10 4 = 6
```

In [150]:
```
number = int(input("enter your number"))
for i in range(1,11):
    print(number,"*",i,'=',number*i)
```

```
enter your number12
12 * 1 = 12
12 * 2 = 24
12 * 3 = 36
12 * 4 = 48
12 * 5 = 60
12 * 6 = 72
12 * 7 = 84
12 * 8 = 96
12 * 9 = 108
12 * 10 = 120
```

In [152]:
```
#factorial number
num = int(input("enter your number"))
factrial = 1
if num <=1 :
    prinr("factorial is not equal to 1 and less than 0")
else:
    for  i in range(1,num+1):
        factrial = factrial*i
    print("factorial value =",factrial)
```

```
enter your number5
factorial value = 120
```

In [153]:
```
#factorial number
num = int(input("enter your number"))
factrial = 1
if num <=1 :
    prinr("factorial is not equal to 1 and less than 0")
else:
    for  i in range(1,num+1):
        factrial = factrial*i
        print("factorial value =",factrial)
```

```
enter your number8
factorial value = 1
factorial value = 2
factorial value = 6
factorial value = 24
factorial value = 120
factorial value = 720
factorial value = 5040
factorial value = 40320
```

While loop

In [154]:
```
i = 0
while i<5:
    print(i)
    i=i+1
```

```
0
1
2
3
4
```

In [155]:
```
i = 0
j = 0
while(i+j<10):
    print(i,j,"=",i+j)
    i = i+1
    j = j+1
```

```
0 0 = 0
1 1 = 2
2 2 = 4
3 3 = 6
4 4 = 8
```

break

when ever we want to exit tp the loop and does not need to iterate further, we use break keyword

In [156]:
```
for i in range(0,10):
    if i==3:
        print("i is 3")
    print(i)
```

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

In [157]:
```
for i in range(0,10):
    if i==3:
        break
        print("i is 3")
    print(i)
```

```
0
1
2
```

In [158]:
```
li =[10,20,30,50,60]
for i in li:
    if i == 30:
        break
    print(i)
```

```
10
20
```

continue

whenever we want to skip th iterator in the loop, we use continue keyword

In [159]:
```
for i in range(1,10):
    if i == 5:
        print("this is 5")
    print(i)
```

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

In [160]:
```
for i in range(1,10):
    if i == 5:
        print("skip the number")
        continue
    print(i)
```

```
1
2
3
4
skip the number
6
7
8
9
```

In [161]:
```
for i in range(1,10):
    if i == 5:
        continue
    print(i)
```

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

List

list is object is an orderd collection of one or more data items, which can be different data types list is a mutable,i.e, changable

In [162]:
```
li = []
type(li)
```

Out[162]:
```
list
```

In [163]:
```
li = [1,2.3,3+2j,"hello",True]

print(li)
```

```
[1, 2.3, (3+2j), 'hello', True]
```

list indexing

In [164]:
```
list1= ["ram","gopala","krishna","masani"]
```

In [165]:
```
#positive index
list1[0]
```

Out[165]: 'ram'

In [166]:
```
list1[1]
```

Out[166]: 'gopala'

In [167]:
```
list1[2]
```

Out[167]: 'krishna'

In [168]:
```
#negative index
list1[-1]
```

Out[168]: 'masani'

In [169]:
```
list1[-2]
```

Out[169]: 'krishna'

In [170]:
```
list1[-0]
```

Out[170]: 'ram'

In [171]:
```
#slicing[start:stop:skip]
list2 = [10,20,30,40,50,60,70,80]
```

In [172]:
```
list2[::]
```

Out[172]: [10, 20, 30, 40, 50, 60, 70, 80]

In [173]:
```
list2[0:4]
```

Out[173]: [10, 20, 30, 40]

In [174]:
```
list2[0:5:2]
```

Out[174]: [10, 30, 50]

In [175]:
```
list2[::-1]
```

Out[175]: [80, 70, 60, 50, 40, 30, 20, 10]

In [176]:
```
#using for loop
for i in list1:
    print(i)
```

```
ram
gopala
krishna
masani
```

In [177]:
```
print("ram" in list1)
```

```
True
```

List methods

In [178]:
```
list3 = [10,20,30,40,50,60,70,80,90]
list3
```

Out[178]:
```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

append

In [179]:
```
list3.append(100)
list3
```

Out[179]:
```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

In [180]:
```
list3.append(-130)

list3
```

Out[180]:
```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, -130]
```

In [181]:
```
#nested list
list3.append([10,20,30])
list3
```

Out[181]:
```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, -130, [10, 20, 30]]
```

extende

In [182]:
```
list4 = [10,20,30,40,50,60,70,80,90]
list4
```

Out[182]:
```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

In [183]:
```
list4.extend([100,110,120])
list4
```

Out[183]:
```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
```

In [184]:
```
list4.extend([130,140,150,160,170])
list4
```

Out[184]:
```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170]
```

Insert

In [185]:
```
list5 = [10,20,30,40,50,60,70,80,90]
list5
```

Out[185]: [10, 20, 30, 40, 50, 60, 70, 80, 90]

In [186]:
```
list5.insert(2,100)
```

In [187]:
```
list5
```

Out[187]: [10, 20, 100, 30, 40, 50, 60, 70, 80, 90]

In [188]:
```
list5.insert(-2,100)
```

In [189]:
```
list5
```

Out[189]: [10, 20, 100, 30, 40, 50, 60, 70, 100, 80, 90]

pop

In [190]:
```
list5
```

Out[190]: [10, 20, 100, 30, 40, 50, 60, 70, 100, 80, 90]

In [191]:
```
list5.pop()
```

Out[191]: 90

In [192]:
```
list5.pop(-1)
```

Out[192]: 80

In [193]:
```
list5
```

Out[193]: [10, 20, 100, 30, 40, 50, 60, 70, 100]

remove

In [194]:
```
list6 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160,
```

In [195]:
```
list6.remove(10)
```

In [196]:
```
list6
```

Out[196]: [20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170]

Count

In [197]:
```
list6 = [20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 17
```

In [198]:
```
list6.count(100)
```

Out[198]: 3

In [199]:
```
list6.count(40)
```

Out[199]: 1

len

In [200]:
```
len(list6)
```

Out[200]: 18

In [201]:
```
len(list5)
```

Out[201]: 9

In [202]:
```
len(list2)
```

Out[202]: 8

index

In [203]:
```
list5
```

Out[203]: [10, 20, 100, 30, 40, 50, 60, 70, 100]

In [204]:
```
list5.index(30)
```

Out[204]: 3

In [205]:
```
list5.index(100)
```

Out[205]: 2

In [206]:
```
list5.index(50)
```

Out[206]: 5

reverse

In [207]:
```
list7 = [10, 20, 100, 30, 40, 50, 60, 70, 100]
```

In [208]:
```
list7.reverse()
list7
```

Out[208]:
```
[100, 70, 60, 50, 40, 30, 100, 20, 10]
```

In [209]:
```
list7
```

Out[209]:
```
[100, 70, 60, 50, 40, 30, 100, 20, 10]
```

In [210]:
```
list7[::-1]
```

Out[210]:
```
[10, 20, 100, 30, 40, 50, 60, 70, 100]
```

sort

In [211]:
```
list8 = [1,3,8,7,6,0,7,5,2,-1]
```

In [212]:
```
list8.sort()
list8
```

Out[212]:
```
[-1, 0, 1, 2, 3, 5, 6, 7, 7, 8]
```

In [213]:
```
list8.sort(reverse = True)

list8
```

Out[213]:
```
[8, 7, 7, 6, 5, 3, 2, 1, 0, -1]
```

strings

string is defined in python either by using single quotation marks or double quotation marks and three double quotes

In [214]:
```
a = 'Hello'

a
```

Out[214]:
```
'Hello'
```

In [215]:
```
b = "Hello"

b
```

Out[215]:
```
'Hello'
```

In [216]:
```
c = """Hello"""

c
```

Out[216]:
```
'Hello'
```

In [217]:
```
print(type(a))
print(type(b))
print(type(c))
```

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

In [218]:
```
e = "hello, my self rama goapala krishna"
e
```

Out[218]:
```
'hello, my self rama goapala krishna'
```

Indexing

String indexing works exatlu the same way list indexing works. first character starts with 0th index and last index -1

In [219]:
```
A = "Hello World"
```

In [220]:
```
A[0]
```

Out[220]:
```
'H'
```

In [221]:
```
A[5]
```

Out[221]:
```
' '
```

In [222]:
```
A[-1]
```

Out[222]:
```
'd'
```

slicing

[start:stop:skip]

In [223]:
```
A[:]
```

Out[223]:
```
'Hello World'
```

In [224]:
```
A[::-1]
```

Out[224]:
```
'dlroW olleH'
```

In [225]:
```
A[0:5]
```

Out[225]: 'Hello'

slicing elements while slicing

In [226]:
```
A[0:9:2]
```

Out[226]: 'HloWr'

In [227]:
```
A[0:6:3]
```

Out[227]: 'Hl'

In [228]:
```
A[-1::-1]
```

Out[228]: 'dlroW olleH'

Methods

In [229]:
```
A =" HEllo World "
B ="RAma GopalA KRisHnA"
```

Upper

In [230]:
```
A.upper()
```

Out[230]: ' HELLO WORLD '

In [231]:
```
B.upper()
```

Out[231]: 'RAMA GOPALA KRISHNA'

lower

In [232]:
```
A.lower()
```

Out[232]: ' hello world '

In [233]:
```
B.lower()
```

Out[233]: 'rama gopala krishna'

Strip

In [234]:
```
A.strip()
```

Out[234]: 'HEllo World'

In [235]:
```
D = "Hello World!!!"
```

In [236]:
```
D.strip("!")
```

Out[236]: 'Hello World'

Replace

In [237]:
```
E = "Hello world Welcome to my jupter"
```

In [238]:
```
E.replace("jupter","jupyter note book")
```

Out[238]: 'Hello world Welcome to my jupyter note book'

In [239]:
```
E.replace("Hello","Hi")
```

Out[239]: 'Hi world Welcome to my jupter'

Split

In [240]:
```
F = "Hello world"
```

In [241]:
```
F.split()
```

Out[241]: ['Hello', 'world']

In [242]:
```
F.split("   ")
```

Out[242]: ['Hello world']

In [243]:
```
"Hello" in F
```

Out[243]: True

In [244]:
```
"Hi" in F
```

Out[244]: False

In [245]:
```
x = "He is a\"legned\"in cricket"
x
```

Out[245]:
```
'He is a"legned"in cricket'
```

\t

In [246]:
```
G = "Hello\tWorld"
G
```

Out[246]:
```
'Hello\tWorld'
```

In [247]:
```
print(G)
```

```
Hello	World
```

\n

In [248]:
```
H = "Hello\nWorld"
H
```

Out[248]:
```
'Hello\nWorld'
```

In [249]:
```
print(H)
```

```
Hello
World
```

Tuples

Tuple is a collection of orderd data items which can be different data types

Tuple is a immutable,i.e, not changable

In [250]:
```
tple = ()
type(tple)
```

Out[250]:
```
tuple
```

In [251]:
```
A = (1,2.3,"Ram",True)

A
```

Out[251]:
```
(1, 2.3, 'Ram', True)
```

In [252]:
```
type(A)
```

Out[252]:
```
tuple
```

In [253]:
```
B = [10,20,40,50]
type(B)
```

Out[253]:
```
list
```

In [254]:
```
c=tuple(B)
```

In [255]:
```
type(c)
```

Out[255]:
```
tuple
```

In [256]:
```
A = (1,2.3,"Ram",True)
for i in A:
    print(i)
```

```
1
2.3
Ram
True
```

Indexing

In [257]:
```
Tuple = ("Rama","Gopala","Krishna","Masani")
Tuple
```

Out[257]:
```
('Rama', 'Gopala', 'Krishna', 'Masani')
```

In [258]:
```
Tuple[0]
```

Out[258]:
```
'Rama'
```

In [259]:
```
Tuple[1]
```

Out[259]:
```
'Gopala'
```

In [260]:
```
Tuple[3]
```

Out[260]:
```
'Masani'
```

In [261]:
```
Tuple[-1]
```

Out[261]:
```
'Masani'
```

In [262]:
```
Tuple[-2]
```

Out[262]:
```
'Krishna'
```

In [263]:
```
Tuple[len(Tuple)-1]
```

Out[263]:
```
'Masani'
```

slicing

In [264]:
```
Tuple = ("Rama","Gopala","Krishna","Masani")
Tuple
```

Out[264]: ('Rama', 'Gopala', 'Krishna', 'Masani')

In [265]:
```
Tuple[:]
```

Out[265]: ('Rama', 'Gopala', 'Krishna', 'Masani')

In [266]:
```
Tuple[:-1]
```

Out[266]: ('Rama', 'Gopala', 'Krishna')

In [267]:
```
Tuple[::-1]
```

Out[267]: ('Masani', 'Krishna', 'Gopala', 'Rama')

In [268]:
```
Tuple[::]
```

Out[268]: ('Rama', 'Gopala', 'Krishna', 'Masani')

slicing elements while slicing

In [269]:
```
Tuple[0:3:2]
```

Out[269]: ('Rama', 'Krishna')

In [270]:
```
Tuple[0:2:2]
```

Out[270]: ('Rama',)

methods

In [271]:
```
A_Tuple = (10,20,30,40,50,10,10,10)
```

In [272]:
```
len(A_Tuple)
```

Out[272]: 8

In [273]:
```
A_Tuple.count(10)
```

Out[273]: 4

In [274]:
```
A_Tuple.index(30)
```

Out[274]: 2

In [275]:
```
del(A_Tuple)
```

Set

set is a collection of unordered data items which can be od different data type

Don't allow duplicates

In [276]:
```
SET = {}
```

In [277]:
```
type(SET)
```

Out[277]: dict

In [278]:
```
Set = {1,2,"Ram",True}
```

In [279]:
```
type(Set)
```

Out[279]: set

In [280]:
```
A  = { 2,3,"Hello",True,"Sam"}
```

In [281]:
```
A
```

Out[281]: {2, 3, 'Hello', 'Sam', True}

In [282]:
```
A  = {1,2,3,"Hello",True,"Sam"}
```

In [283]:
```
A
```

Out[283]: {1, 2, 3, 'Hello', 'Sam'}

In [284]:
```
B  = {0,2,3,"Hello",False,"Sam"}
B
```

Out[284]: {0, 2, 3, 'Hello', 'Sam'}

In [285]:
```
c = {True,False,1,0,10,20,30,30,50,25.5,"50.5"}
c
```

Out[285]: {10, 20, 25.5, 30, 50, '50.5', False, True}

In [286]:
```
List = [1,2,3,4,1,5,7,1,9,11,1,2,2]
set(List)
```

Out[286]: {1, 2, 3, 4, 5, 7, 9, 11}

In [287]:
```
for i in c:
    print(i)
```

```
False
True
10
50.5
50
20
25.5
30
```

Methods

In [288]:
```
Set_1 = {1,2,3,4,54,65,76,68}

Set_1.add(100)
Set_1
```

Out[288]:
```
{1, 2, 3, 4, 54, 65, 68, 76, 100}
```

In [289]:
```
len(Set_1)
```

Out[289]:
```
9
```

In [290]:
```
Set_1.update([200,150,300])
Set_1
```

Out[290]:
```
{1, 2, 3, 4, 54, 65, 68, 76, 100, 150, 200, 300}
```

In [291]:
```
len(Set_1)
```

Out[291]:
```
12
```

In [292]:
```
Set_1.remove(1)
```

In [293]:
```
Set_1
```

Out[293]:
```
{2, 3, 4, 54, 65, 68, 76, 100, 150, 200, 300}
```

In [294]:
```
Set_1.clear()
```

In [295]:
```
Set_1
```

Out[295]:
```
set()
```

Oprations

In [296]:
```
SET_1 = {1,2,3,4,5}
SET_2 = {1,7,4,5,6}
```

In [297]:
```
SET_1.union(SET_2)
```

Out[297]: {1, 2, 3, 4, 5, 6, 7}

In [298]:
```
SET_1.intersection(SET_2)
```

Out[298]: {1, 4, 5}

In [299]:
```
SET_1-SET_2
```

Out[299]: {2, 3}

In [300]:
```
SET_2-Set_1
```

Out[300]: {1, 4, 5, 6, 7}

In [301]:
```
SET_1^SET_2
```

Out[301]: {2, 3, 6, 7}

Dictionary

Dictionary is an unorderd collection of key and value pairs, Where the keys and values can be of any datatype

In [302]:
```
DICT = {}
```

In [303]:
```
type(DICT)
```

Out[303]: dict

In [304]:
```
Students = {"RAMA":78,"GOPALA":77,"KRISHNA":78,"MASANI":"RAJU"}
```

In [305]:
```
Students
```

Out[305]: {'RAMA': 78, 'GOPALA': 77, 'KRISHNA': 78, 'MASANI': 'RAJU'}

In [306]:
```
type(Students)
```

Out[306]: dict

Accesing items in Dictionary

In [307]:
```
Students["RAMA"]
```

Out[307]: 78

In [308]:
```
Students["MASANI"]
```

Out[308]: 'RAJU'

In [309]:
```
Students["KRISHNA"]
```

Out[309]: 78

In [310]:
```
for i in Students:
    print(i,Students[i])
```

```
RAMA 78
GOPALA 77
KRISHNA 78
MASANI RAJU
```

In [311]:
```
for i in Students.keys():
    print(i)
```

```
RAMA
GOPALA
KRISHNA
MASANI
```

In [312]:
```
for i in Students.values():
    print(i)
```

```
78
77
78
RAJU
```

In [313]:
```
student = {"marks":[60,70,65,67,93],"MARKS":[82,88,87,89,98]}
```

In [314]:
```
student
```

Out[314]: {'marks': [60, 70, 65, 67, 93], 'MARKS': [82, 88, 87, 89, 98]}

In [315]:
```
for i in student["MARKS"]:
    print(i)
```

```
82
88
87
89
98
```

In [316]:
```
for i in student:
    print(i,student[i])
```

```
marks [60, 70, 65, 67, 93]
MARKS [82, 88, 87, 89, 98]
```

In [317]:
```
for i in student:
    for j in student[i]:
        print(i,j)
```

```
marks 60
marks 70
marks 65
marks 67
marks 93
MARKS 82
MARKS 88
MARKS 87
MARKS 89
MARKS 98
```

In [318]:
```
student
```

Out[318]:
```
{'marks': [60, 70, 65, 67, 93], 'MARKS': [82, 88, 87, 89, 98]}
```

In [319]:
```
student["rama"] = [78,77,79,80,81]
```

In [320]:
```
student
```

Out[320]:
```
{'marks': [60, 70, 65, 67, 93],
 'MARKS': [82, 88, 87, 89, 98],
 'rama': [78, 77, 79, 80, 81]}
```

In [321]:
```
student.get("MARKS")
```

Out[321]:
```
[82, 88, 87, 89, 98]
```

In [322]:
```
len(student)
```

Out[322]:
```
3
```

In [323]:
```
len(student.get("MARKS"))
```

Out[323]:
```
5
```

In [324]:
```
for i in student:
    print(i,len(student[i]))
```

```
marks 5
MARKS 5
rama 5
```

In [325]:
```
student.pop("rama")
```

Out[325]:
```
[78, 77, 79, 80, 81]
```

In [326]:
```
student
```

Out[326]:
```
{'marks': [60, 70, 65, 67, 93], 'MARKS': [82, 88, 87, 89, 98]}
```

In [327]:
```
del student["marks"]
```

In [328]:
```
student
```

Out[328]:
```
{'MARKS': [82, 88, 87, 89, 98]}
```

MY SELF RAMA GOPALA KRISHNA