- keywords
  - keywords are the reserved words in python
  - It has special meaning

In [71]:
```
import keyword
keyword.kwlist
```

Out[71]:
```
['False',
 'None',
 'True',
 '__peg_parser__',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

In [2]:
```
import keyword
print(len(keyword.kwlist))
```

```
36
```

variable

```
- it is used to store the values
```

rules to create a variable

- we didn't use keywords as a variable name
- we shouldn't starts with digits and special symbols expect(-(underscore))
- we didn't use spaces also

In [3]:
```
a=9
print(a)
```

```
9
```

In [4]:
```
-s=88
print(-s)
```

```
  Input In [4]
    -s=88
    ^
SyntaxError: cannot assign to operator
```

In [5]:
```
# multiple variable assignments
A=B=C=45
print(A)
print(B)
print(C)
```

```
45
45
45
```

In [6]:
```
a1,b1,c1=3,2,1
print(a1)
b1
c1
print(a1,b1,c1)
```

```
3
3 2 1
```

Datatypes

- int
- float
- string
- boolean
- complex

In [7]:
```
s=3;g=5;h=3
print(s,g,h)
```

```
3 5 3
```

In [8]:
```
# integer
w=10
print(w)
```

```
10
```

In [9]:
```
# float
z=5.2
print(z)
```

```
5.2
```

In [10]:
```
# string Declaration
name='SRK'
print(name)
```

```
SRK
```

In [11]:
```
# Dynamic
a=int(input("enter any number:"))
print(a)
```

```
enter any number:10
10
```

In [12]:
```
st = input()
st
```

```
'hello'
```

Out[12]:
```
"'hello'"
```

In [13]:
```
b=float(input("enter b value:"))
b
```

```
enter b value:10.1
```

Out[13]:
```
10.1
```

In [14]:
```
c=str(input("enter name:"))
c
```

```
enter name:'siva sai'
```

Out[14]:
```
"'siva sai'"
```

Type conversions

1.int -int(variable name)
2.float-float(variable name)
3.string-str(variable name)

In [18]:
```
h=88
print(type(h))
j=float(g)
print(j)
print(type(j))
```

```
<class 'int'>
5.0
<class 'float'>
```

In [19]:
```
y=10.2
x=int(y)
print(type(y))
print(x)
print(type(x))
```

```
<class 'float'>
10
<class 'int'>
```

In [20]:
```
name='hello'
c=int(name)
print(c)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [20], in <cell line: 2>()
      1 name='hello'
----> 2 c=int(name)
      3 print(c)

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

In [22]:
```
z=10
print(type(z))
h=str(z)
print(h)
print(type(h))
```

```
<class 'int'>
10
<class 'str'>
```

In [26]:
```
h=10
str(h)
```

Out[26]: '10'

Operators

- Arithmetic
  --> +,-,*,/,%,//(FLOOR),power(**)
- Assignment
  --> =,+=,-=,*=,/=,//=
- Comparision
  --> ==,!=,>,<,>=,<=
- Logical
  --> and,or,not
- Bitwise
  --> &,|,^(xor),<<,>>
- Membership
  --> in,not in
- Identity
  --> is,is not

In [28]:
```
# Arithemtic
# This operators are used to perform the mathematical operations
a=10
b=15
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
print(a//b)
print(a**b)
```

```
25
-5
150
0.6666666666666666
10
0
1000000000000000
```

In [29]:
```
# floor
6//2
```

Out[29]: 3

In [30]:
```
10//4
```

Out[30]: 2

In [31]:
```
# power
5**10
```

Out[31]: 9765625

In [32]:
```
# Assignment
v=4
v+=2 # v=v+2
print(v)
```

```
6
```

In [33]:
```
v1=5
v1+=v
v1
```

Out[33]:
```
11
```

In [34]:
```
c1=4
c1-=v1
c1
```

Out[34]:
```
-7
```

In [35]:
```
d1=5
d1*=c1
d1
```

Out[35]:
```
-35
```

In [36]:
```
e1=10
e1/=d1
e1
```

Out[36]:
```
-0.2857142857142857
```

In [43]:
```
# Comparison
g1,g2=9,3
print(g1==g2)
print(g1!=g2)
print(g1>g2)
print(g1<g2)
print(g1>=g2)
print(g1<=g2)
```

```
False
True
True
False
True
False
```

In [53]:
```
i=int(input("enter first number:"))
j=int(input("enter second number:"))
print("addition of 7 and 5 is:",i+j)
print("subtraction of 7 and 5 is:",i-j)
print("multiplication of 7 and 5 is:",i*j)
print("division of 7 and 5 is:",i/j)
print("modular of 7 and 5 is:",i%j)
print("floor division of 7 and 5 is:",i//j)
print("power of 7 and 5 is:",i**j)
```

```
enter first number:7
enter second number:5
addition of 7 and 5 is: 12
subtraction of 7 and 5 is: 2
multiplication of 7 and 5 is: 35
division of 7 and 5 is: 1.4
modular of 7 and 5 is: 2
floor division of 7 and 5 is: 1
power of 7 and 5 is: 16807
```

In [55]:
```
# Logical
d,h=6,3
print(d>h and d!=h) # t and t
print(d>h and d==h)
print(d<h and d>h)
```

```
True
False
False
```

In [56]:
```
d,h=6,3
print(d>h or d!=h)
print(d>h or d==h)
print(d<h or d>h)
```

```
True
True
True
```

In [59]:
```
d,h=6,3
print(d>h  d!=h)
print(d>h & d==h)
print(d<h & d>h)
```

```
False
False
False
```

In [60]:
```
bin(10)
```

Out[60]:
```
'0b1010'
```

In [61]:
```
s=10
v=20
print(s&v)
print(s|v)
print(s^v)
```

```
0
30
30
```

Membership

These operators are used to chech whether a value or variable exists in a sequence (string,list,tuple,dictionary,set)

In [62]:
```
s="apssdc"
print("a" in s)
print("p" in s)
print("c" in s)
print("h" in s)
```

```
True
True
True
False
```

In [63]:
```
print("j" not in s)
print("aps" not in s)
print("sda" not in s)
```

```
True
False
True
```

Identity Operators

-> used to check the memory locations of the objects

In [69]:
```
v,b=8,9
print(id(v),id(b))
print(v is b)
print(v is not b)
```

```
2538190826000 2538190826032
False
True
```

In [70]:
```
v1,v2=4,4.0
print(id(v1),id(v2))
print(v1 is v2)
```

```
2538190825872 2538305836752
False
```

In [76]:
```
u=int(input("enter number1:"))
o=int(input("enter number2"))
#comparison
print(u==o)
print(u!=o)
print(u>o)
print(u<o)
print(u>=o)
print(u<=o)
#identity
print(u is v)
print(u is not v)
#boolean
print(bool(u>v))
print(bool(u==v))
```

```
enter number1:15
enter number27
False
True
True
False
True
False
False
True
True
False
```

In [ ]: