# Keywords

In [20]:
```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def'
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pa
e', 'with', 'yield']
```

In [3]:
```
len(keyword.kwlist)
```

```
35
```

**35 RESERVED WORDS---

True, False, None ==> Represent Boolean data types
and, or, not, is ==> Represent the operators
if, else, elif ==> Represent the statement (# python switch,do..while statament is n
while, for, break, continue, return, in, yield ==> Represent the loop concept
try, except, finally, raise, assert ==> Represent for functionallity
import,from,as,class,def,pass,global,nonlocal,lambda,del,with==>Represent the cl
*NOTES -- 35 RESERVED WORDS ARE (ALPHABET)
*EXCEPT (True,False,None)

In [21]:

```
import pandas as pd # pandas is the module to create a dataframe
df = pd.DataFrame(keyword.kwlist)
print(df)            # Always remember python Index begins with '0'
```

```
           0
0      False
1       None
2       True
3        and
4         as
5     assert
6      async
7      await
8      break
9      class
10  continue
11       def
12       del
13      elif
14      else
15    except
16   finally
17       for
18      from
19    global
20        if
21    import
22        in
23        is
24    lambda
25  nonlocal
26       not
27        or
28      pass
29     raise
30    return
31       try
32     while
33      with
34     yield
```

# Identifier/variable/object

No length limit, lower cases, upper case, underscore, space is not allowed, keywo
not be number.

In [5]:
```
#eg
a = 4
abc = 18000
Nit_recording = 25000
hi2 = 80
#error, 2hi, nit recording, True
```

In [6]:
```
print(a)
print(abc)
print(Nit_recording)
print(hi2)
```

```
4
18000
25000
80
```

# Multiple parameters assigning

In [7]:
```
a, b, c, d, e = 10, 2.5, 'nit', True, 1+2j
```

In [8]:
```
print(a)
print(b)
print(c)
print(d)
print(e)
```

```
10
2.5
nit
True
(1+2j)
```

In [9]:
```
x1 = 20 # x,y = 20
y1 = 20
print(x1 is y1)
print(id(y1) is id(x1))
```

```
True
False
```

In [10]:
```
print(id(x1), id(y1))
```

```
140732336412040 140732336412040
```

In [11]:
```
x = True
y = True
z = False
print(x is y)
print(y is z)
print(z is x)
print(z is y)
print(id(x) is id(y))
```

```
True
False
False
False
False
```

In [12]:
```
a1, b1, c1 = 1, 2, 3            #i, f, s, c, b = 45(error)
print(a1, b1, c1)
```

```
1 2 3
```

In [13]:
```
p1 = 20 + 30 \
 + 40 + 50 +\
  70 + 80
p1
```

```
290
```

In [14]:
```
#a, b, c = 1, 2
#a, b = 1, 2, 3
```

In [15]:
```
a, b, _ = 1, 2, 3
print(a, b, _)
print(a, b)
print(b)
```

```
1 2 3
1 2
2
```

# Data types

int, float, complex, bool, str, bytes, bytearray

In [16]:
```
p = 70
q = 25.5
q1 = 100.e0
q2 = 123456789.e1
r = 1+2j
s = True
s1 = False
_t = 20
has_0_in_it = "Still Valid"
```

In [17]:
```
print(p)
print(q)
print(q1,q2)
print(r)
print(s)
print(s1)
print(_t)
print(has_0_in_it)
```

```
70
25.5
100.0 1234567890.0
(1+2j)
True
False
20
Still Valid
```

In [18]:
```
a = 10
b = 0b10
c = 0o100
print(a)
print(b)
print(c)
```

```
10
2
64
```

In [19]:
```
print(type(a))
print(type(b))
print(type(c))
```

```
<class 'int'>
<class 'int'>
<class 'int'>
```

In [20]:
```
b = 0b1111 # Now pvm convert value to binary value(0b)
print(type(b))
print(b)
```

```
<class 'int'>
15
```

In [21]:
```
print(r.real, r.imag)
```

```
1.0 2.0
```

In [22]:
```
#c = 15+0b111j # Imaginary part cannot be binary,octal
d = 0b11+15j # Real part can be binary,octal
d
```

```
(3+15j)
```

# Operations - boolean, complex

In [23]:
```
r1 = 3+4j
```

In [24]:
```
x = r + r1
x
```

```
(4+6j)
```

In [25]:
```
s + s1
```

```
1
```

In [26]:
```
s - s1
```

```
1
```

In [27]:
```
s1 - s
```

```
-1
```

In [28]:
```
s1/s
```

```
0.0
```

In [29]:
```
# s/s1 error
```

# String

In [9]:
```
g = 'Welcome to Python'
```

In [10]:
```
h = "Programming"
```

In [12]:
```
g +" " + h
```

```
'Welcome to Python Programming'
```

In [32]:
```
i = '''nit
    technology
        hyderabad'''
```

In [33]:
```
print(g)
print(h)
print(i)
```

```
Welcome to Python
Programming
nit
    technology
        hyderabad
```

# String Operations

In [34]:
```
mystr2 = 'Woohoo '
mystr2 = mystr2*5
mystr2
```

```
'Woohoo Woohoo Woohoo Woohoo Woohoo '
```

In [3]:
```
str1 = 'Hello Python'
```

In [4]:
```
print(str1[0])        #String indexing
print(str1[-1])
```

```
H
n
```

In [5]:
```
print(str1[0:5],    #string slicing
    str1[6:12],
     str1[-4:],
     str1[-6:])
```

```
Hello Python thon Python
```

In [13]:
```
j = 'hello'
for p in j:
    print(p)
```

```
h
e
l
l
o
```

In [14]:
```
for q in enumerate(j):
    print(q)
```

```
(0, 'h')
(1, 'e')
(2, 'l')
(3, 'l')
(4, 'o')
```

In [17]:
```
#list[enumerate(j)]
list(enumerate(j))
```

```
[(0, 'h'), (1, 'e'), (2, 'l'), (3, 'l'), (4, 'o')]
```

## String function

strip(), rstrip(), lstrip(), strip('*'), lower(), upper(), replace(2 arg), count(1 arg), starts
split(), format - Combining string & numbers, center(), rjust(), find(), index(), rindex
isinstance(), isalpha() - all are letters, isalnum() - all are either letters or numbers, i
islower(), isupper()

In [19]:
```
#str1[0:5] = 'HOLAA'
str1.replace("He" , "Ho")
```

```
'Hollo Python'
```

In [ ]:
```

```

## standard functions

In [38]:
```
type(r)
```

```
complex
```

In [39]:
```
id(i)
```

```
1435548905136
```

In [40]:
```
print(type(r))
```

```
<class 'complex'>
```

In [41]:
```
print(id(i))
```

```
1435548905136
```

In [42]:
```
u = reversed('hello')
str(u)
#u
```

```
'<reversed object at 0x0000014E3D65A1A0>'
```

In [44]:
```
import sys
print(sys.getsizeof(r))
```

```
32
```

In [45]:
```
print(isinstance(r, complex))
```

```
True
```

# User defined function

In [18]:
```
def greet(): # user define greet()
    print('welcome nit') # statement
    print('please do hard work')
greet()
```

```
welcome nit
please do hard work
```

# Typecasting

```
convert one data type to other
```

## Others data type to int -- except complex and string in text

In [22]:
```
print(int(2.56))       #float to int
print(int('10'))       #string to int
print(int(True))       #boolean to int
#print(int(1+2j))
#print(int('ten'))
```

```
2
10
1
```

## Other data type to float -- except complex and string in text

In [23]:
```
print(float(2))          #int to float
print(float('10'))       #string to float
print(float(False))      #boolean to float
#print(float(1+2j))
#print(float('ten'))
```

```
2.0
10.0
0.0
```

## Other data type to complex -- except string in text

In [24]:

```
print(complex(10))          #int to complex
print(complex(10, 20))      #int to complex
print(complex(90.8))        #float to complex
print(complex(True))        #boolean to complex
print(complex('10'))        #string to complex only when one argument is pas
#print(complex('90',80))
#print(complex('80','70'))
#print(complex('ten'))
```

```
(10+0j)
(10+20j)
(90.8+0j)
(1+0j)
(10+0j)
```

## Other data type to boolean

In [25]:

```
print(bool(26))
print(bool(5.6))
print(bool('0'))
print(bool('zero'))
print(bool(0+0j))
print(bool(False))
print(bool( ))
print(bool())
```

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

# Other data type to string

In [26]:

```
print(str(2))
print(str(9.8))
print(str(True))
print(str(0+0j))
```

```
2
9.8
True
0j
```