In [7]:
```
import sys
import keyword
import operator
from datetime import datetime
import os
```

Keywords

In [8]:
```
# Keywords are the reserved words in python and can't be used as an identifier
```

In [9]:
```
print(keyword.kwlist)  #list all python keywords
```

```
['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
```

Identifiers

```
An identifier is a name given to entities like class , functions, variables,
etc. It helps to differentiate
one entity from another
```

In [11]:
```
1var = 10 # Identifiers can't start with a digit
```

```
  Cell In[11], line 1
    1var = 10 # Identifiers can't start with a digit
    ^
SyntaxError: invalid decimal literal
```

In [12]:
```
val2@ = 35 # Identifier can't use special symbols
```

```
  Cell In[12], line 1
    val2@ = 35 # Identifier can't use special symbols
          ^
SyntaxError: invalid syntax
```

In [13]:
```
import = 125 # keywords can't be used as identifiers
```
```
  Cell In[13], line 1
    import = 125 # keywords can't be used as identifiers
           ^
SyntaxError: invalid syntax
```

In [14]:
```
"""
Correct way of defining an identifier
(Identifiers can be a combination of letters in lowercase (a to z) or uppercas
"""
val2 = 10
```

In [15]:
```
val_ = 100
```

Comments in Python

```
Comments can be used to explain the code for more readability
```

In [16]:
```
# Single line comment
val1 = 23
```

In [17]:
```
# multiple
#line
#comment
val1 = 10
```

In [18]:
```
"""
Multiple
line
comments
"""
val1 = 10
```

In [19]:
```
'''
Multiple
line
comments
'''
val1 = 10
```

Statements

Instructions that a python interpreter can execute

```
In [20]: p = 20   #Creates an integer object with value 20 and assigns the variable p t
         q = 20   # Create new reference q which will point to value 20. p & q will be
         r = q    # variable r will also point to the same location where p & q are poi

         p, type(p) , hex(id(p))

Out[20]: (20, int, '0x7fff7c3f9588')

In [21]: q , type(q) , hex(id(q))

Out[21]: (20, int, '0x7fff7c3f9588')

In [22]: r , type(r), hex(id(r))

Out[22]: (20, int, '0x7fff7c3f9588')

In [23]: p =20
         p =p+10
         p

Out[23]: 30
```

Variable assignment

```
In [24]: intvar = 10 # integer variable
         floatvar = 2.57 #float variable
         strvar = "Python Language"

In [25]: print(intvar)
         print(floatvar)
         print(strvar)

         10
         2.57
         Python Language
```

Multiple assignments

```
In [26]: intvar, floatvar , strvar = 10 , 2.57,"Python Language" #Using commas to separ
         print(intvar)
         print(floatvar)
         print(strvar)

         10
         2.57
         Python Language
```

In [27]:
```
p1 = p2 = p3 = p4 = 44 # All variables pointing to same value
print(p1,p2,p3,p4)
```
```
44 44 44 44
```

Data Types

Numeric

In [29]:
```
Val1 = 10 # Integer data type
print(Val1)
print(type(Val1))  # type of object
print(sys.getsizeof(Val1)) #size of integer object in bytes
print(Val1, "is Integer?", isinstance(Val1, int))
```
```
10
<class 'int'>
28
10 is Integer? True
```

In [33]:
```
Val2 = 95.25 # Float data type
print(Val2)
print(type(Val2))  # type of object
print(sys.getsizeof(Val2)) #size of float object in bytes
print(Val2, "is float?", isinstance(Val2, float))
```
```
95.25
<class 'float'>
24
95.25 is float? True
```

In [34]:
```
Val3 = 25 + 10j # complex data type
print(Val3)
print(type(Val3))  # type of object
print(sys.getsizeof(Val3)) #size of complex object in bytes
print(Val2, "is complex?", isinstance(Val2, complex))
```
```
(25+10j)
<class 'complex'>
32
95.25 is complex? False
```

In [35]:
```
sys.getsizeof(int())   # size of integer object in bytes
```
Out[35]:
```
28
```

In [36]:
```
sys.getsizeof(float())   # size of float object in bytes
```
Out[36]:
```
24
```

```
In [37]: sys.getsizeof(complex())    # size of complex object in bytes

Out[37]: 32
```

Boolean

```
Boolean data type can have only two possible values true or false
```

```
In [38]: bool1 = True

In [39]: bool2 = False

In [40]: print(type(bool1))

         <class 'bool'>

In [41]: print(type(bool2))

         <class 'bool'>

In [42]: isinstance(bool1 , bool)

Out[42]: True

In [43]: bool(0)

Out[43]: False

In [44]: bool(1)

Out[44]: True

In [46]: bool(None)

Out[46]: False

In [47]: bool(False)

Out[47]: False
```