In [1]:
```
import sys
import keyword
import operator
from datetime import datetime
import os
```

Keywords Keywords are the reserved words in Python and can't be used as an identifier

In [2]:
```
print(keyword.kwlist) #list all python keywords
```

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

In [3]:
```
len(keyword.kwlist) # python contain 35 keywords
```

Out[3]: 35

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

In [4]:
```
1var= 10 #Identifier can't start with a digit
```

```
  Cell In[4], line 1
    1var= 10 #Identifier can't start with a digit
    ^
SyntaxError: invalid decimal literal
```

In [5]:
```
val2@ = 35 # Identifier can't use special symbols
```

```
  Cell In[5], line 1
    val2@ = 35 # Identifier can't use special symbols
         ^
SyntaxError: invalid syntax
```

In [6]:
```
import = 125 # Keywords can't be used as identifiers
```

```
  Cell In[6], line 1
    import = 125 # Keywords can't be used as identifiers
           ^
SyntaxError: invalid syntax
```

In [7]:
```
"""
Correct way of defining an identifier
(Identifiers can be a combination of letters in lowercase (a to z) or uppercase
"""
val2 = 10
```

In [8]:
```
val_ = 99
```

Comments in Python

Comments can be used to explain the code for more readabilty.

In [9]:
```
# Single line comment
val1 = 10
```

In [10]:
```
# Multiple
# line
# comment
val1 = 10
```

In [11]:
```
'''
Multiple
line
comment
'''
val1 = 10
```

In [12]:
```
"""
Multiple
line
comment
"""
val1 = 10
```

Statements

Instructions that a Python interpreter can execute.

In [14]:
```
p = 20 #Creates an integer object with value 20 and assigns the variable p to p
q = 20 # Create new reference q which will point to value 20. p & q will be poi
r = q # variable r will also point to the same location where p & q are pointing
p , type(p), hex(id(p)) # Variable P is pointing to memory location '0x7fff6d71a
```

Out[14]:
```
(20, int, '0x7ff85a49b608')
```

In [15]:
```
q , type(q), hex(id(q))
```

Out[15]:
```
(20, int, '0x7ff85a49b608')
```

In [16]:
```
r , type(r), hex(id(r))
```

Out[16]:
```
(20, int, '0x7ff85a49b608')
```

In [17]:
```
p = 20
p = p + 10 # Variable Overwriting
p
```

Out[17]:
```
30
```

Variable Assigment

In [18]:
```
intvar = 10 # Integer variable
floatvar = 2.57 # Float Variable
strvar = "Python Language" # String variable
print(intvar)
print(floatvar)
print(strvar)
```

```
10
2.57
Python Language
```

Multiple Assignments

In [19]:
```
intvar , floatvar , strvar = 10,2.57,"Python Language" # Using commas to separat
print(intvar)
print(floatvar)
print(strvar)
```

```
10
2.57
Python Language
```

In [20]:
```
p1 = p2 = p3 = p4 = 44 # All variables pointing to same value
print(p1,p2,p3,p4)
```

```
44 44 44 44
```

Data Types

Numeric

In [22]:
```
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)) # val1 is an instance of int
```

```
10
<class 'int'>
28
10  is Integer? True
```

In [23]:
```
val2 = 92.78 # 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)) # Val2 is an instance of float
```

```
92.78
<class 'float'>
24
92.78  is float? True
```

In [24]:
```
val3 = 25 + 10j # Complex data type
print(val3)
print(type(val3)) # type of object
```

```
print(sys.getsizeof(val3)) # size of float object in bytes
print(val3, " is complex?", isinstance(val3, complex)) # val3 is an instance of
```

```
(25+10j)
<class 'complex'>
32
(25+10j)  is complex? True
```

In [25]:
```
sys.getsizeof(int()) # size of integer object in bytes
```

Out[25]: 28

In [26]:
```
sys.getsizeof(float()) # size of float object in bytes
```

Out[26]: 24

In [27]:
```
sys.getsizeof(complex()) # size of complex object in bytes
```

Out[27]: 32

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

In [28]:
```
bool1 = True
```

In [29]:
```
bool2 = False
```

In [30]:
```
print(type(bool1))
```

```
<class 'bool'>
```

In [31]:
```
print(type(bool2))
```

```
<class 'bool'>
```

In [32]:
```
isinstance(bool1, bool)
```

Out[32]: True

In [33]:
```
bool(0)
```

Out[33]: False

In [34]:
```
bool(1)
```

Out[34]: True

In [35]:
```
bool(None)
```

Out[35]: False

In [36]:
```
bool (False)
```

Out[36]: False

Strings

String Creation

In [37]:
```
str1 = "HELLO PYTHON"
print(str1)
```

```
HELLO PYTHON
```

In [38]:
```
mystr = 'Hello World' # Define string using single quotes
print(mystr)
```

```
Hello World
```

In [39]:
```
mystr = "Hello World" # Define string using double quotes
print(mystr)
```

```
Hello World
```

In [40]:
```
mystr = '''Hello
World '''
print(mystr)
# Define string using triple quotes
```

```
Hello
World
```

In [41]:
```
mystr = """Hello
World""" # Define string using triple quotes
print(mystr)
```

```
Hello
World
```

In [42]:
```
mystr = ('Happy '
'Monday '
'Everyone')
print(mystr)
```

```
Happy Monday Everyone
```

In [43]:
```
mystr2 = 'Woohoo '
mystr2 = mystr2*5
mystr2
```

Out[43]:
```
'Woohoo Woohoo Woohoo Woohoo Woohoo '
```

In [44]:
```
len(mystr2) # Length of string
```

Out[44]:
```
35
```

String Indexing

In [45]:
```
str1
```

Out[45]:
```
'HELLO PYTHON'
```

In [46]:
```
str1[0] # First character in string "str1"
```

Out[46]: 'H'

In [47]:
```
str1[len(str1)-1] # Last character in string using len function
```

Out[47]: 'N'

In [48]:
```
str1[-1] # Last character in string
```

Out[48]: 'N'

In [49]:
```
str1[6] #Fetch 7th element of the string
```

Out[49]: 'P'

In [50]:
```
str1[5]
```

Out[50]: ' '

String Slicing

In [51]:
```
str1[0:5] # String slicing - Fetch all characters from 0 to 5 index location exc
```

Out[51]: 'HELLO'

In [52]:
```
str1[6:12] # String slicing - Retreive all characters between 6 - 12 index loc e
```

Out[52]: 'PYTHON'

In [53]:
```
str1[-4:] # Retreive last four characters of the string
```

Out[53]: 'THON'

In [54]:
```
str1[-6:] # Retreive last six characters of the string
```

Out[54]: 'PYTHON'

In [55]:
```
str1[:4] # Retreive first four characters of the string
```

Out[55]: 'HELL'

In [56]:
```
str1[:6] # Retreive first six characters of the string
```

Out[56]: 'HELLO '

Update & Delete String

In [57]:
```
str1
```

Out[57]: 'HELLO PYTHON'

In [58]:
```
#Strings are immutable which means elements of a string cannot be changed once t
str1[0:5] = 'HOLAA'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[58], line 2
      1 #Strings are immutable which means elements of a string cannot be changed on
ce t
----> 2 str1[0:5] = 'HOLAA'

TypeError: 'str' object does not support item assignment
```

In [59]:
```
del str1 # Delete a string
print(srt1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[59], line 2
      1 del str1 # Delete a string
----> 2 print(srt1)

NameError: name 'srt1' is not defined
```

In [ ]: