Python Tutorial

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

Keywords

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

In [8]:
```
print(keyword.kwlist)
```

```
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'awai
t', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finall
y', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'n
ot', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

In [9]:
```
len(keyword.kwlist)
```

Out[9]: 36

Identifiers

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

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

```
  File "C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14668\525829194.py", line 1
    1var =10
     ^
SyntaxError: invalid syntax
```

In [11]:
```
val2@ =35
```

```
  File "C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14668\3625693022.py", line 1
    val2@ =35
         ^
SyntaxError: invalid syntax
```

In [12]:
```
1 import = 125
```

```
  File "C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14668\4166699667.py", line 1
    import = 125
           ^
SyntaxError: invalid syntax
```

In [13]:
```
1 val2 =10
```

In [14]:
```
1 val2
```

Out[14]:
```
10
```

Comment

Comments can be used to explain the code for readability

In [15]:
```
1 #single line comment
2 val1=10
```

In [16]:
```
1 '''
2 Multi-line
3 comment
4 '''
5 val2=10
```

In [17]:
```
1 """
2 Multi-line
3 comment
4 """
5
6 val3= 30
```

Statement

Instructions that python interpreter can execute

In [18]:
```
1 p = 20 #Creates an integer object with value 20 and assigns the variable p to
2 q = 20 # Create new reference q which will point to value 20. p & q will be po
3 r = q # variable r will also point to the same location where p & q are pointi
4 p , type(p), hex(id(p)) # Variable P is pointing to memory location
```

Out[18]:
```
(20, int, '0x1d871906b90')
```

In [19]:
```
q , type(q), hex(id(q))
```

Out[19]:
```
(20, int, '0x1d871906b90')
```

In [20]:
```
r , type(r), hex(id(r))
```

Out[20]:
```
(20, int, '0x1d871906b90')
```

In [21]:
```
p = 20
p = p + 10 # Variable Overwriting
p
```

Out[21]:
```
30
```

Variable Assigment

In [22]:
```
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 Assigments

In [23]:
```
intvar , floatvar , strvar = 10,2.57,"Python Language" # Using commas to separ
print(intvar)
print(floatvar)
print(strvar)

```

```
10
2.57
Python Language
```

In [24]:
```
p1 = p2 = p3 = p4 = 44 # All variables pointing to same value
print(p1,p2,p3,p4)

```

```
44 44 44 44
```

Data types

Numeric

In [31]:
```
1 val1 = 10 # Integer data type
2 print(val1)
3 print(type(val1)) # type of object
4 print(sys.getsizeof(val1)) # size of integer object in bytes
5 print(val1, " is Integer?", isinstance(val1, int)) # val1 is an instance of in
```

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

In [33]:
```
1 val2 = 92.78 # Float data type
2 print(val2)
3 print(type(val2)) # type of object
4 print(sys.getsizeof(val2)) # size of float object in bytes
5 print(val2, " is float?", isinstance(val2, float)) # Val2 is an instance of fl
```

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

In [34]:
```
1 val3 = 25 + 10j # Complex data type
2 print(val3)
3 print(type(val3)) # type of object
4 print(sys.getsizeof(val3)) # size of float object in bytes
5 print(val3, " is complex?", isinstance(val3, complex)) # val3 is an instance o
```

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

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

Out[37]: 24

In [38]:
```
1 sys.getsizeof(float()) # size of float object in bytes
2
```

Out[38]: 24

In [39]:
```
1 sys.getsizeof(complex()) # size of complex object in bytes
```

Out[39]: 32

In [40]:
```
sys.getsizeof(str()) # size of string object in bytes
```

Out[40]: 51

In [41]:
```
sys.getsizeof(list())
```

Out[41]: 56

In [42]:
```
sys.getsizeof(set())
```

Out[42]: 216

In [43]:
```
sys.getsizeof(tuple())
```

Out[43]: 40

In [44]:
```
sys.getsizeof(dict())
```

Out[44]: 232

Boolean

Boolean data types can have only two possible values true or false

In [46]:
```
bool1 = True
bool2 = False
print(type(bool1))
print(type(bool2))
print(isinstance(bool1, bool))
```

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

In [47]:
```
bool(0)
```

Out[47]: False

In [48]:
```
bool(1)
```

Out[48]: True

In [49]:
```
bool(None)
```

Out[49]: False

In [50]:
```
bool (False)
```

Out[50]: False

In [51]:
```
1 bool(4)
```

Out[51]:
```
True
```

In [52]:
```
1 bool('T')
```

Out[52]:
```
True
```

Strings

String Creation

In [53]:
```
1 str1 = "HELLO PYTHON"
2 print(str1)
```

```
HELLO PYTHON
```

In [54]:
```
1 mystr = 'Hello World' # Define string using single quotes
2 print(mystr)
3
```

```
Hello World
```

In [55]:
```
1 mystr = "Hello World" # Define string using double quotes
2 print(mystr)
3
```

```
Hello World
```

In [56]:
```
1 mystr = '''Hello
2 World ''' # Define string using triple quotes
3 print(mystr)
```

```
Hello
World
```

In [57]:
```
1 mystr = """Hello
2 World""" # Define string using triple quotes
3 print(mystr)
4
```

```
Hello
World
```

In [58]:
```
mystr = ('Happy '
'Monday '
'Everyone')
print(mystr)

```

```
Happy Monday Everyone
```

In [59]:
```
mystr2 = 'Woohoo '
mystr2 = mystr2*5
mystr2
```

Out[59]: 'Woohoo Woohoo Woohoo Woohoo Woohoo '

In [60]:
```
len(mystr2) # Length of string
```

Out[60]: 35

String Indexing

In [61]:
```
str1
```

Out[61]: 'HELLO PYTHON'

In [62]:
```
str1[0] # First character in string "str1"
```

Out[62]: 'H'

In [63]:
```
str1[len(str1)-1] # Last character in string using len function
```

Out[63]: 'N'

In [64]:
```
str1[-1] # Last character in string
```

Out[64]: 'N'

In [65]:
```
str1[6] #Fetch 7th element of the string
```

Out[65]: 'P'

In [66]:
```
str1[5]
```

Out[66]: ' '

String Slicing

In [67]:
```
1 str1[0:5] # String slicing - Fetch all characters from 0 to 5 index location
```

Out[67]: 'HELLO'

In [68]:
```
1 str1[6:12] # String slicing - Retreive all characters between 6 - 12 index
```

Out[68]: 'PYTHON'

In [69]:
```
1 str1[-4:] # Retreive last four characters of the string
```

Out[69]: 'THON'

In [70]:
```
1 str1[-6:] # Retreive last six characters of the string
```

Out[70]: 'PYTHON'

In [71]:
```
1 str1[:4] # Retreive first four characters of the string
```

Out[71]: 'HELL'

In [72]:
```
1 str1[:6] # Retreive first six characters of the strin
```

Out[72]: 'HELLO '

Update & Delete String

In [73]:
```
1 str1
```

Out[73]: 'HELLO PYTHON'

In [75]:
```
1 #Strings are immutable which means elements of a string cannot be changed
2 str1[0:5] = 'HOLAA'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14668\2984603276.py in <module>
      1 #Strings are immutable which means elements of a string cannot be changed
----> 2 str1[0:5] = 'HOLAA'

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

In [76]:
```
del str1 # Delete a string
print(srt1)

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14668\1861387540.py in <module>
      1 del str1 # Delete a string
----> 2 print(srt1)

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

String Concatenation

In [78]:
```
# String concatenation
s1 = "Hello"
s2 = "Asif"
s3 = s1 +' ' + s2
print(s3)
```

```
Hello Asif
```

In [ ]:
```

```

In [ ]:
```

```