WORKING WITH CORE PYTHON PROGRAMMING
Variables in PYTHON
Define Variable?
A variable is a name that refers to a value. Variables point to the memory location where data is read and modified.

What is an identifier?
An identifier is just the name of the variable.

Python Variable Name Rules
1 Python Variables Must begin with a letter (a - z, A - Z) or underscore (_)
2 Python Variables should be characters like letters, numbers or _
3 Python Variables are Case Sensitive
4 Python Variables can be any reasonable length
5 Python Variables must not be reserved word or Keyword.
6 The variable names should be written in camelCase or PascalCase

Illegal Variable Names:
```
76tones = 'Big Data'
#illegal because it begin with number

more@=1000000
#illegal because it contains special  character

class = 'Advanced Theoretical'
#class is one of Python's keywords.
```

camel & Pascal-Case:
In Real Time Projects, camelCase &PasCal are a naming convention in which a name is formed of multiple words that are joined together as a single word for better readability.

Examples:
"iPhone ", "eBay", "FedEx", "PayPal", etc...!!

EXAMPLES:
```
>>> Age=10
>>> Age10=10
>>> 10Age=10
SyntaxError: invalid syntax
>>> _Age=10
>>> #Age=10
>>> Age.22=10#Special Character must not use
SyntaxError: invalid syntax
>>> for=10#It is a keyword
SyntaxError: invalid syntax
```

Constants:
Fixed values such as numbers, letters & strings,are called "constants"

What is Literal?
a literal is a notation for representing a fixed value.

Python supports the following literals:
I. String literals:

II.Numeric literals: 1 int 2 long 3 float 4 complex
III. Boolean literals: True or False
IV. Special literals:None ==> In Python is same as "null" , means non existent, not known, or empty.
V.Literal Collections: list, tuple, set & dict

NOTE:
Literals concept is not applicable in PYTHON, but it is convention to use only uppercase characters. It is just convention but we can change the value.

Syntax:
MIN_VALUE, MAX_VALUE

Example:
```
MAX_VAL=10; MIN_VAL=1
print(MAX_VAL+MIN_VAL)
MAX_VAL=100
print(MAX_VAL+MIN_VAL)
```

Example:
```
PYSTR="HELLO"#String literals:
PYINT=10#Integer Literal
PYFLOAT=10.99#Float Literal
PYCOM=1+2j#Complex Literal
PYBOOL=True#Boolean Literal
PYBOOL=False#Boolean Literal
PYSPE=None#Special Literal
PYLIST=[1,2,3,4,5]#LIST Literal
PYTUPLE=(1,2,3,4,5)#Tuple Literal
PYSET={1,2,3,4,5}#Set Literal
PYDICT={1:"RajuSir"}#Dictionary Literal
```

Data types in Python:
Data Type represent the type of data present inside a variable. Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instances (objects) of these classes.

Some built-in Python data types are:
1 Numeric data types: int, float, complex
2 String data types: str
3 Sequence types: list, tuple, range
4 Binary types: bytes, bytearray, memoryview
5 Mapping data type: dict
6 Boolean type: bool
7 Set data types: set, frozenset

type() function: It returns type of the given object.

Syntax:
type(object)

Example:
```
#Numbers, Numerical Data type
#Integer
#Declartion is an Object or Instance or Ref.
```

```
PyInt=10#int Object
print(type(PyInt))#<class 'int'>
print(PyInt)#10
#Float
PyFloat=10.001#float Object
print(type(PyFloat))#<class 'float'>
print(PyFloat)#10.001
#String
PyStr="PYTHON"#string Object
print(type(PyStr))#<class 'str'>
print(PyStr)#PYTHON
#List
PyList=[1,2,3,4]#list Object
print(type(PyList))#<class 'list'>
print(PyList)#[1, 2, 3, 4]
#Tuple
PyTuple=(1,2,3,4)#tuple Object
print(type(PyTuple))#<class 'tuple'>
print(PyTuple)#(1, 2, 3, 4)
#Set
PySet={1,2,3}#set Object
print(type(PySet))#<class 'set'>
print(PySet)#{1, 2, 3}
#Dictionary
PyDict={1:"PY",2:"DS"}#Dictionary Object
print(type(PyDict))#<class 'dict'>
print(PyDict)#{1: 'PY', 2: 'DS'}
```

format() function:
It returns a formatted representation of the given value controlled by the format specifier.

Syntax:
format(value[, format_spec])

```
'<'     The field will be left-aligned within the available space
'>'     The field will be right-aligned within the available space
'^'     Forces the field to be centered within the available space.
```

Example:

```
print(format(123, "d"))
print(format(123.4567898, "f"))
print(format(8/9,".3f"))
```

Example:

```
print(format(123,'d'))#d-digit, integer
print(format(123,'f'))#f-float
print(format(2/3,'f'))
print(format(2/3,'.3f'))#.3f 3-float values
print(format(2/3,'.1f'))#.1f 1-float value
print(format("Hello"))
print(format("Hello",'s'))
print(format("Hello",'.1s'))
print(format("Hello",'.4s'))
print(format("Hello",'.6s'))
```

Example:

```
print(format(123, "<40"))
print(format(123, ">40"))
print(format("Hello", "^40"))
```

Swap variables
In Python swap values in a single line and this applies to all objects in python.

Syntax
```
var1, var2 = var2, var1
```

Example :
```
x = 10;y = 20
print(x,y)  #10 20
x, y = y, x
print(x,y) #20 10
```

Example:
```
PyStr1="Hello";PyStr2="Bye"
print(PyStr1,PyStr2)#Hello Bye
PyStr1,PyStr2=PyStr2,PyStr1
print(PyStr1,PyStr2)#Bye Hello
```

Example:
```
#Classic way to swapping
a=23;b=42
tmp=a
a=b
b=tmp
print(a,b)
#PYTHONIC Way
a=23;b=42
a,b=b,a
print(a,b)
```