Data Types...

- 1) STR
- 2) INT
- 3) FLOAT
- 4) COMPLEX
- 5) LIST
- 6) TUPLE
- 7) RANGE
- 8) DICT
- 9) SET
- 10) FROZENSET
- 11) BOOL
- 12) BYTES
- 13) BYTEARRAY
- 14) MEMORYVIEW

String Data Type

- Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes.
- Python treats single quotes the same as double quotes.
- Creating strings is as simple as assigning a value to a variable

In [5]:
```
y_str = "I'm practicing python"
x = "Hello World"       #str
print('String Data Type: ',type(x))
print(x)
print(y_str)
print(type(y_str))
```

```
String Data Type:  <class 'str'>
Hello World
I'm practicing python
<class 'str'>
```

Int Data Type

- they are positive or negative whole numbers with no decimal point. Integers in Python are of unlimited size.

In [6]:
```
a=55
a
type(a)
```

Out[6]:
```
int
```

In [7]:
```
x = 34  #int
y = 534 #int
print('Int Data Type : ',type(x))
print(x)
print(type(y))
```

```
Int Data Type :  <class 'int'>
34
<class 'int'>
```

Float Data Type

- they represent real numbers and are written with a decimal point dividing the integer and the fractional parts.
- Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 102 = 250).

In [8]:
```
x = 20.2342    #float
print('Float Data Type : ',type(x))
print(x)
```

```
Float Data Type :  <class 'float'>
20.2342
```

List

- List is a collection which is ordered and changeable. Allows duplicate members.
- In Python lists are written with square brackets.

In [9]:
```
bool_v =[1,2,3,"sfsdf",True,'sdfsd']
type(bool_v)
```

Out[9]:
```
list
```

In [10]:
```
a = [1,0,"this is index 2",1,2,2,2,3,4,5,6,7,8,'String','NAme',44.33,True]
print(type(a))
print(a[2])
```

```
<class 'list'>
this is index 2
```

In [11]:
```
x = ["apple",23,2323,34.34, "banana", "cherry",11,22,11,22,1,1,2,2,2,3,3]       #list
print(' List Data Type : ',type(x))
print(x[2])
```

```
 List Data Type :  <class 'list'>
2323
```

In [12]:
```
num_list=[1,2,3,4,5,6,7,7,7,6,6,5,5,8,8,9,9,]
print(type(num_list))
print(num_list)
```

```
<class 'list'>
[1, 2, 3, 4, 5, 6, 7, 7, 7, 6, 6, 5, 5, 8, 8, 9, 9]
```

Tuple

- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- In Python tuples are written with round brackets.

In [13]:
```
a_tuple=(1,2,3,4)
type(a_tuple)
```

Out[13]:
```
tuple
```

In [14]:
```
x = ("apple", 12,1212,121.12,"banana", "cherry")        #tuple
print('Tuple Data Type : ', type(x))
print(x[0])
```

```
Tuple Data Type :  <class 'tuple'>
apple
```

In [15]:
```
num_tuple = (1,2,3,4,5,6,6,7,7,8,8,9,9,10)
print(type(num_tuple))
print(num_tuple)
```

```
<class 'tuple'>
(1, 2, 3, 4, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10)
```

Dictionary

- Dictionary is a collection which is unordered, changeable and indexed. No duplicate members.
- In Python dictionaries are written with curly brackets, and they have keys and values.

In [16]:
```
x = {"name" : "Govardhan", "age" : 23,"loc":"Hyd"}      #dict
print('Dictionary Data Type : ',type(x))
print(x['loc'])
```

```
Dictionary Data Type :  <class 'dict'>
Hyd
```

In [17]:
```
a_set = set()
```

Sets

- A set is a collection which is unordered and unindexed. No duplicate members.
- In Python, sets are written with curly brackets.

In [18]:
```
x = {"apple", "banana", "cherry","cherry"}      #set
print('Set Data Type : ',type(x))
print(x)
```

```
Set Data Type :  <class 'set'>
{'cherry', 'apple', 'banana'}
```

In [19]:
```
a_set = {1,1,1,2,2,2,3,4,4,5,5,5,6,7,8,9}
a_set
```

Out[19]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

frozenset()

- The frozenset() is an inbuilt function is Python which takes an iterable object as input and makes them immutable. Simply it freezes the iterable objects and makes them unchangeable.

In [20]:
```
x = frozenset({"apple", "banana", "cherry"})    #frozenset
print('Frozenset Data Type : ',type(x))
print(x)
```

```
Frozenset Data Type :  <class 'frozenset'>
frozenset({'cherry', 'apple', 'banana'})
```

Bool

- The boolean data type is either True or False. In Python, boolean variables are defined by the True and False keywords.

In [21]:
```
a=False
b=True
print('Variable a type is : ',type(a))
print('Variable b type is : ',type(b))
```

```
variable a type is :  <class 'bool'>
variable b type is :  <class 'bool'>
```

In [22]:
```
num =0
bool(num)
# y or n  or True or False or 1 or 0 or active or inactive
```

Out[22]:
```
False
```

In [23]:
```
x = False       #bool
print('Bool Data Type : ',type(x))
print(x)
```

```
Bool Data Type :  <class 'bool'>
False
```

In [24]:
```
bool_f=False
print(type(bool_f))
print(bool_f)
```

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

Bytes

- Return a new "bytes" object, which is an immutable sequence of small integers in the range 0 <= x < 256, print as ASCII characters when displayed. bytes is an immutable version of bytearray – it has the same non-mutating methods and the same indexing and slicing behavior.

In [25]:
```
x = b"Hello"   #bytes
print('Bytes Data Type : ',type(x))
print(x)
```

```
Bytes Data Type :  <class 'bytes'>
b'Hello'
```

In [26]:
```
type(x)
```

Out[26]:
```
bytes
```

bytearray()

- bytearray() method returns a bytearray object which is an array of given bytes. It gives a mutable sequence of integers in the range 0 <= x < 256.
- The difference between bytes() and bytearray() is that bytes() returns an object that cannot be modified, and bytearray() returns an object that can be modified.

In [27]:
```
# Converting inter into bytearray
x = bytearray(14))     #bytearray
print('Byte Array Data Type : ',type(x))
print(x)
```

```
Byte Array Data Type :  <class 'bytearray'>
bytearray(b'\x0e')
```

Memoryview()

- The memoryview() function returns a memory view object from a specified object.
- A memory view is a safe way to expose the buffer protocol in Python.
- It allows you to access the internal buffers of an object by creating a memory view object.

In [28]:
```
x = memoryview(bytes(5))        #memoryview
print('Memory View Data Type : ',type(x))
x = memoryview(b"Hello")
print(x)
#return the Unicode of the first character
print(x[0])
#return the Unicode of the second character
print(x[1])
```

```
Memory View Data Type :  <class 'memoryview'>
<memory at 0x000001E900AA2B80>
72
101
```

In [29]:
```
type(x)
```

Out[29]:
```
memoryview
```

Date & Time Types

In [30]:
```
import datetime
x = datetime.datetime.now()
print(x)
type(x)
```

```
2022-12-13 20:00:16.386989
```

Out[30]:
```
datetime.datetime
```