(3.9) range & None

range Data Type represents a sequence of numbers

The elements present in range Data type are not modifiable. i.e range Data type is immutable

Form-1: range(10)

generate numbers from 0 to 9

In [1]:
```
r=range(10)
for i in r : print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

Form-2: range(10,20)

generate numbers from 10 to 19

In [2]:
```
r = range(10,20)
for i in r : print(i)
```

```
10
11
12
13
14
15
16
17
18
19
```

Form-3: range(10,20,2)

2 means increment value

In [3]:
```
r = range(10,20,2)
for i in r : print(i)
```

```
10
12
14
16
18
```

We can access elements present in the range Data Type by using index.

In [4]:
```
r=range(10,20)
r[0]
```

Out[4]:
```
10
```

We cannot modify the values of range data type

In [5]:
```
r[0] = 100
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_4212/3589217873.py in <module>
----> 1 r[0] = 100

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

We can create a list of values with range data type

In [6]:
```
l = list(range(10))
print(l)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

(3.9.1) None

None means Nothing or No value associated.
If the value is not available,then to handle such type of cases None introduced.
It is something like null value in Java.

In [1]:
```
def m1():
    a=10
print(m1())
```

```
None
```

Summary of Datatypes in Python3

In [5]:
```
from IPython.display import Image
Image(filename='summary1.png')
```

Out[5]:

Datatype	Description	Is Immutable	Example

Int	We can use to represent the whole/integral numbers	Immutable	>>> a=10
			>>> type(a)
			<class 'int'>

Float	We can use to represent the decimal/floating point numbers	Immutable	>>> b=10.5
			>>> type(b)
			<class 'float'>

Complex	We can use to represent the complex numbers	Immutable	>>> c=10+5j
			>>> type(c)
			<class 'complex'>
			>>> c.real
			10.0
			>>> c.imag
			5.0

Bool	We can use to represent the logical values(Only allowed values are True and False)	Immutable	>>> flag=True
			>>> flag=False
			>>> type(flag)
			<class 'bool'>

Str	To represent sequence of Characters	Immutable	>>> s='durga'
			>>> type(s)
			<class 'str'>
			>>> s="durga"
			>>> s='''Durga Software Solutions
			... Ameerpet'''
			>>> type(s)
			<class 'str'>

In [6]:
```
Image(filename='summary2.png')
```

Out[6]:

bytes	To represent a sequence of byte values from 0-255	Immutable	>>> list=[1,2,3,4]
			>>> b=bytes(list)
			>>> type(b)
			<class 'bytes'>

bytearray	To represent a sequence of byte values from 0-255	Mutable	>>> list=[10,20,30]
			>>> ba=bytearray(list)
			>>> type(ba)
			<class 'bytearray'>

range	To represent a range of values	Immutable	>>> r=range(10)
			>>> r1=range(0,10)
			>>> r2=range(0,10,2)

list	To represent an ordered collection of objects	Mutable	>>> l=[10,11,12,13,14,15]
			>>> type(l)
			<class 'list'>

tuple	To represent an ordered collections of objects	Immutable	>>> t=(1,2,3,4,5)
			>>> type(t)
			<class 'tuple'>

set	To represent an unordered collection of unique objects	Mutable	>>> s={1,2,3,4,5,6} >>> type(s)

In [7]:
```
Image(filename='summary3.png')
```

Out[7]:

frozenset	To represent an unordered collection of unique objects	Immutable	>>> s={11,2,3,'Durga',100,'Ramu'}
			>>> fs=frozenset(s)
			>>> type(fs)
			<class 'frozenset'>

dict	To represent a group of key value pairs	Mutable	>>> d={101:'durga',102:'ramu',103:'hari'} >>> type(d) <class 'dict'>