Integers - This value is represented by int class

In [4]:
```
#Python program to demonstrate numeric value

a = 5
print("Type of a: ", type(a))

b = 5.0
print("Type of b: ", type(b))

c = 2 + 4j
print("\nType of c: ", type(c))
```

```
Type of a:  <class 'int'>
Type of b:  <class 'float'>

Type of c:  <class 'complex'>
```

Strings

Strings are used in Python to record text information, such as names. Strings in Python are actually a *sequence*, which basically means Python keeps track of every element in the string as a sequence

or

A String is a data structure in Python that represents a sequence of characters. It is an immutable data type, meaning that once you have created a string, you cannot change it.

Creating a string

In [5]:
```
# Single word
'hello'
```

Out[5]:
```
'hello'
```

In [6]:
```
#Entire phrase
'This is also a string'
```

Out[6]:
```
'This is also a string'
```

In [7]:
```
# We can also use double quote
"String built with double quotes"
```

Out[7]:
```
'String built with double quotes'
```

In [10]:
```
# Using Variables
x = "Priting string using variables"
```

```
print("",  x, "\n", type(x))
```

```
Priting string using variables
<class 'str'>
```

In [11]:
```
# Escape Character
x = 'Hello I\'m Aayush'
print(x)
```

```
Hello I'm Aayush
```

We use the escape character '' to tell the program in the above case that to escape the ending.

In [12]:
```
print('Hello \t\t Jupyter')
```

```
Hello		 Jupyter
```

We use '\t' to give 5 spaces (one tap space) in between a string.

In [13]:
```
len("I am Aayush")
```

Out[13]: 11

We can use 'len()' to find out the length of a string or a string variable

String Indexing

We know that strings are a sequence, which means we can use Python can use indexes to call parts of a string

In [14]:
```
# Assign x as a string
x = "Hello World"
```

In [17]:
```
#check
x[0:5]
```

Out[17]: 'Hello'

In [22]:
```
s = "GeeksForGeeks"
s[0:13]
```

Out[22]: 'GeeksForGeeks'

In [26]:
```
s[-1]
```

Out[26]: 's'

In [29]:
```
s[5:8]
```

Out[29]: 'For'

In [32]:
```
s[4:]
```

Out[32]: 'sForGeeks'

In [33]:
```
s[:4]
```

Out[33]: 'Geek'

In [34]:
```
s[:]
```

Out[34]: 'GeeksForGeeks'

In [35]:
```
s[::1]
```

Out[35]: 'GeeksForGeeks'

In [36]:
```
s[::2]
```

Out[36]: 'GesoGes'

In [37]:
```
s[::3]
```

Out[37]: 'Gkoes'

Using double colon skips some characters from between

In [38]:
```
s[::-1]
```

Out[38]: 'skeeGroFskeeG'

In [39]:
```
s[::-2]
```

Out[39]: 'seGoseG'

In [3]:
```
s = 'GeeksForGeeks'
```

In [4]:
```
#Concatenate strings
s + ' ' + 'Here'
```

Out[4]: 'GeeksForGeeks Here'

Concatenating

In [5]:
```
s = 'hello world'
```

In [6]:
```
s.upper()
```

Out[6]: 'HELLO WORLD'

In [7]:
```
s.lower()
```

Out[7]: 'hello world'

In [ ]:

In [8]:
```
x = "Hello Planet Earth"
result = x.split()
print(result)
```

['Hell', ' Planet Earth']