Data Types

1. Fundamental Data Types

integer	float	complex	boolean	string
True	42	42.0	"hello"	a+bj

2. Advanced Data Types or Data Structure or Containers

list --> [1,2,3]

tuple --> (1,2,3)

set --> {1,2,3}

dictionary --> {1:"audi",2:"Maruti"}

Int - Numeric value without any decimal

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

```
10
```

In [2]:
```
# To check data type : use type() function
type(a)
```

Out[2]:
```
int
```

Float - Numeric value with decimal

In [3]:
```
b=3.0
print(b)
```

```
3.0
```

In [4]:
```
type(b)
```

Out[4]:
```
float
```

In [5]:
```
a=3
type(a)
```

Out[5]:
```
int
```

Type Casting: Convert from one data type to another data type

In [6]:
```
b = 5.9
int(b)
```

Out[6]: 5

In [7]:
```
int(10.9999)
```

Out[7]: 10

In [8]:
```
float(5)
```

Out[8]: 5.0

Boolean - True/False

In [9]:
```
b1 = True
type(b1)
```

Out[9]: bool

In [10]:
```
type(False)
```

Out[10]: bool

Complex

Complex literals can be created by using the notation x + yj where x is the real component and y is the imaginary component.

In [11]:
```
#complex numbers: note the use of `j` to specify the imaginary part
x=10.0+20.5j
```

In [12]:
```
type(x)
```

Out[12]: complex

In [13]:
```
# to get real value
x.real
```

Out[13]: 10.0

In [14]:
```
# to get imaginary value
x.imag
```

Out[14]: 20.5

In [15]:
```
int(x.imag)
```

Out[15]:
```
20
```

In [16]:
```
complex(10.4)
```

Out[16]:
```
(10.4+0j)
```

Strings

- A string in Python consists of a series or sequence of characters - letters, numbers, and special characters.
- We can use single quotes or double quotes to represent strings.
- Multi-line strings can be denoted using triple quotes, ''' or """.
- Strings can be indexed - often synonymously called subscripted as well.
- The first character of a string has the index 0.

In [17]:
```
a='siva'
print(a)
```

```
siva
```

In [18]:
```
type(a)
```

Out[18]:
```
str
```

In [19]:
```
myString1 = 'Hello'
print(myString1)

myString2 = "Hello"
print(myString2)

myString3 = '''Hello'''
print(myString3)
```

```
Hello
Hello
Hello
```

In [20]:
```
name='''siva
rama
krishna'''

print(name)
```

```
siva
rama
krishna
```

In [21]:
```
var1  = 1234567
type(var1)
```

Out[21]:
```
int
```

In [22]:
```
var = "1234567"
type(var)
```

Out[22]:
```
str
```

In [23]:
```
int(var)
```

Out[23]:
```
1234567
```

In [24]:
```
a='si@1234'
```

In [25]:
```
int(a)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-8b2ab3991dae> in <module>
----> 1 int(a)

ValueError: invalid literal for int() with base 10: 'si@1234'
```

Length of string

In [26]:
```
d = 'This is interesting'
len(d)
```

Out[26]:
```
19
```

How to access characters in a string using indexing

In [27]:
```
myString = "Hello"
```

In [28]:
```
#print first Character
print(myString[0])
```

```
H
```

In [29]:
```
#last char d[len(d)-1] or d[-1]
print(myString[-1])
```

```
o
```

In [30]:
```
#print character using negative indexing
print(myString[-3])
```

```
l
```

In [31]:
```
# If we try to access index with out of the range, we will get index error.
print(myString[5])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-31-1c520f1abd32> in <module>
      1 # If we try to access index with out of the range, we will get index er
ror.
----> 2 print(myString[5])

IndexError: string index out of range
```

In [32]:
```
# If we try to access by using use decimal number, we will get errors.
print(myString[1.5])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-2809aa1e7e14> in <module>
      1 # If we try to access by using use decimal number, we will get errors.
----> 2 print(myString[1.5])

TypeError: string indices must be integers
```

Slicing the strings

[start index (inclusive) : end index (exclusive) : increment/decrement]

[default increment = +1]

In [33]:
```
#slicing 2nd to 5th character (start index to end-1 index)
myString[2:5]
```

Out[33]: 'llo'

In [34]:
```
myString[2:5:1]
```

Out[34]: 'llo'

In [35]:
```
myString[0:3]
```

Out[35]: 'Hel'

In [36]:
```
myString[:4]            # slicing (0 to end-1)
```

Out[36]: 'Hell'

In [37]:
```
myString[2:]            # slicing (Start index to all)
```

Out[37]:
```
'llo'
```

In [38]:
```
myString[:]             # slicing (Start to all)
```

Out[38]:
```
'Hello'
```

In [39]:
```
myString[10:45]         # slicing with increment
```

Out[39]:
```
''
```

In [40]:
```
myString[-1::-1]        # slicing with decrement
```

Out[40]:
```
'olleH'
```

In [41]:
```
#print in reverse order
myString[::-1]
```

Out[41]:
```
'olleH'
```

Concatenation

Joining of two or more strings into a single one is called concatenation.

The + operator does this in Python. Simply writing two string literals together also concatenates them.

The * operator can be used to repeat the string for a given number of times.

In [42]:
```
#concatenation of 2 strings
'Hello' +"World"
```

Out[42]:
```
'HelloWorld'
```

In [43]:
```
"1"+2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-43-5c53161196cd> in <module>
----> 1 "1"+2

TypeError: can only concatenate str (not "int") to str
```

In [44]:
```
a = "Data Science"
b = "SRK"
c = a + b
print(c)
```

```
Data ScienceSRK
```

In [45]:
```
a = "Data Science by"
b = "SRK"
c = a + " " +b
print(c)
```

```
Data Science by SRK
```

In [46]:
```
#repeat string n times
"AI" * 5
```

Out[46]:
```
'AIAIAIAIAI'
```

In [47]:
```
user = "satish"
lines = 100

print("Congratulations, " + user + " ! You just wrote " + str(lines) + " lines of
```

```
Congratulations, satish ! You just wrote 100 lines of code
```

String Immutable

In [48]:
```
myString = "Hello"
myString[4]
```

Out[48]:
```
'o'
```

In [49]:
```
# Strings are immutable. This means that elements of a string cannot be changed o
# We can simply reassign different strings to the same name.

myString = "Hello"
myString[4]='s' # strings are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-49-833279607c78> in <module>
      3
      4 myString = "Hello"
----> 5 myString[4]='s' # strings are immutable

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

String Methods

Some of the commonly used methods are lower(), upper(), join(), split(), find(), replace() etc

Capitalize

In [50]:
```
# Capatilise only first letter and remaning all to small  --> Sentence case in MS

a="gooD moRNing"
a.capitalize()
```

Out[50]:
```
'Good morning'
```

Title

In [51]:
```
# Capatilise first letter of each word and remaning all to small
a = "gOOd moRning"
a.title()
```

Out[51]:
```
'Good Morning'
```

Upper

In [52]:
```
# Convert complete string to upper case
a.upper()
```

Out[52]:
```
'GOOD MORNING'
```

Lower

In [53]:
```
# Convert complete string to lower case
a.lower()
```

Out[53]:
```
'good morning'
```

Split

In [54]:
```
# Split with space
st = 'hello, my name is Sam'
st.split(' ')
```

Out[54]:
```
['hello,', 'my', 'name', 'is', 'Sam']
```

Join

In [55]:
```
' '.join(['This', 'will', 'split', 'all', 'words', 'in', 'a', 'list'])
```

Out[55]:
```
'This will split all words in a list'
```

Find

In [56]:
```
# To find a string in the input string
"Good Morning Mahesh".find("M")

# It returns the index position of string if available
# It returns -1 if not available
```

Out[56]: 5

Count

In [57]:
```
#To count how many times the string available in the main string
"Good Morning".count("oo")
```

Out[57]: 1

Replace

In [58]:
```
# To replace the old string  with new string
s1 = "Good morning"
s2 = s1.replace("Good", "Bad")
```

In [59]:
```
s2
```

Out[59]: 'Bad morning'

In [60]:
```
s1
```

Out[60]: 'Good morning'

In [61]:
```
my_str='Siva123'

print(my_str.isdigit()) #test if string contains only numbers

print(my_str.isalnum()) #check if it is alphanumeric

print(my_str.isalpha()) #check if all char in the string are alphabetic

print(my_str.istitle()) #test if string contains title words

print(my_str.isupper()) #test if string contains upper case

print(my_str.islower()) #test if string contains lower case

print(my_str.isspace()) #test if string contains spaces

print(my_str.endswith('s')) #test if string endswith a s

print(my_str.startswith('S')) #test if string startswith S
```

```
False
True
False
True
False
False
False
False
True
```

Input

- want to take the input from the user. In Python, we have the input() function to allow this.
- by default it takes as "str" datatype

In [65]:
```
input()
```

```
siva
```

Out[65]:
```
'siva'
```

In [66]:
```
n=input()
```

```
srk
```

In [67]:
```
n
```

Out[67]:
```
'srk'
```

In [68]:
```
type(n)
```

Out[68]:
```
str
```

In [69]:
```
n1=input("enter your age:")
n1
```

```
enter your age:24
```

Out[69]:
```
'24'
```

In [70]:
```
type(n1)
```

Out[70]:
```
str
```

In [71]:
```
n2=int(input("enter your age:"))
n2
```

```
enter your age:24
```

Out[71]:
```
24
```

In [72]:
```
type(n2)
```

Out[72]:
```
int
```

In [73]:
```
# Let's have one more example
name = input("What is your name? ")
print ("It was nice talking you " +name + "!")
```

```
What is your name? SRK
It was nice talking you SRK!
```

In [74]:
```
age = input("Enter your age? ")
print("Hey, you are already " + age + " years old, " + name + "!")
```

```
Enter your age? 24
Hey, you are already 24 years old, SRK!
```

In [75]:
```
# multiple inputs in a single entry
var_1, var_2 = input("Enter your name: ").split(" ")
print(var_1)
print(var_2)
```

```
Enter your name: siva rama
siva
rama
```