Data Types

In [1]:

```
s = "Hello World!"
```

```
stdout   -->  /dev/stdout
stdin    --> /dev/stdin
stderror --> /dev/stderr
```

In [6]:

```
import sys
sys.stdout.write('Hello World')
```

```
Hello World
```

In [4]:

```
print("Hello World")
# print --> sys.stdout --> jupyter --> shell
```

```
Hello World
```

```
print, input --> files
```

In [7]:

```
'Hello World' # output on python shell
```

Out[7]:

```
'Hello World'
```

In [2]:

```
print(s[-1:3:-2])
# this plain output is known as standard output
```

```
!lo
```

In [3]:

```
s[-1:3:-2]
# red colors show it's a shell output, it will not be visible if you
# run your program as script first.py
```

Out[3]:

```
'!lo '
```

In [8]:

```
print(repr(s[-1:3:-2]))
```

```
'!lo '
```

repr --> raw representation of string (non-printable)
str ---> string representation (printable)

In [10]:

```
s = "Hello World"
print(s)
sys.stdout.write(str.__str__(s)) # way complication so forget it
```

```
Hello World
Hello World
```

In [11]:

```
s = "Hello \n\tWorld"
```

In [12]:

```
print(s)
```

```
Hello
        World
```

In [13]:

```
print(repr(s))
print(str(s))
```

```
'Hello \n\tWorld'
Hello
        World
```

In [15]:

```
print(s[-1:3:-2])
```

```
drW
o
```

In [17]:

```
s = "Hello World!"
print(repr(s[-1:3:-2]))
```

```
'!lo '
```

directory

available function to process your data

dunder methods or magic methods

double under score methods

__magic_method__

Everything is object in python

In [27]:

```
r = 50 + 45
print(r)
print(int.__add__(50, 40))
```

```
95
90
```

In [28]:

```
s = "Hello "  + "World" # concatenation
print(s)
print(str.__add__("Hello ", "World"))
```

```
Hello World
Hello World
```

In [29]:

```
lang = [ 'java', 'c', 'c++'] + [ 'python', 'R', 'scala']
print(lang)
print(list.__add__(['java'], ['python']))
```

```
['java', 'c', 'c++', 'python', 'R', 'scala']
['java', 'python']
```

In [30]:

```
print(dir(str))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__
getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ite
r__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new_
_', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__se
tattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefo
ld', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'forma
t', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'is
digit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'i
stitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partit
ion', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstri
p', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'tran
slate', 'upper', 'zfill']
```

```
+ --> __add__

- --> __sub__

* --> __mul__
```

In [31]:

```
s = "HellO WoRlD!"
```

. --> access specifier

In [32]:

```
new_str = s.swapcase()
```

In [33]:

```
print(s) # immutable
print(new_str)
```

```
HellO WoRlD!
hELLo wOrLd!
```

In [36]:

```
print("Original: ",s)
print("Lower case: ",s.lower())
print("Upper case: ",s.upper())
print("Swap case: ",s.swapcase())
print("title case: ",s.title())
print("capitalize case: ", s.capitalize())
```

```
Original:  HellO WoRlD!
Lower case:  hello world!
Upper case:  HELLO WORLD!
Swap case:  hELLo wOrLd!
title case:  Hello World!
capitalize case:  Hello world!
```

In [38]:

```
help(s.lower)
```

```
Help on built-in function lower:

lower() method of builtins.str instance
    Return a copy of the string converted to lowercase.
```

In [40]:

```
help(s.title)
```

```
Help on built-in function title:

title() method of builtins.str instance
    Return a version of the string where each word is titlecased.

    More specifically, words start with uppercased characters and all remain
ing
    cased characters have lower case.
```

In [48]:

```
print("Sachin Yadav".center(1050, '_'))
```

```
_____Sachin Yadav_____
```

In [54]:

```
s = "     \n\tHello World!\n\n\t\t    "
```

In [55]:

```
print(s)
print(repr(s))
```

```
        Hello World!


'     \n\tHello World!\n\n\t\t    '
```

In [56]:

```
process_str = s.strip()
```

In [57]:

```
print(repr(s))
print(repr(process_str))
```

```
'     \n\tHello World!\n\n\t\t    '
'Hello World!'
```

In [59]:

```
s = "    hello world   "
print(repr(s))
print(repr(s.strip()))
```

```
'    hello world   '
'hello world'
```

In [62]:

```
s = "\n\n\thello\t\n\n"
s1 = s.strip()
print("before: ",repr(s))
print("after: ",repr(s1))
```

```
before:  '\n\n\thello\t\n\n'
after:  'hello'
```

In [68]:

```
s = "   hello        world   "
s1 = s.strip()
print(s)
print(s1)
```

```
   hello        world
hello        world
```

In [ ]:

```

```

In [67]:

```
s = "    \n   \t    hello world   \n  \t  "
print(repr(s))
s1 = s.strip() # by default " ", "\n\t"
print(repr(s1))

print(s)
print(s1)
```

```
'    \n   \t    hello world   \n  \t  '
'hello world'

            hello world

hello world
```

In [70]:

```
s = "----------@@@@@@sachin yaadav@@@@@--------------"
print(s)
s1 = s.strip('-@')
print(s1)
```

```
----------@@@@@@sachin yaadav@@@@@--------------
sachin yaadav
```

In [73]:

```
s = "    \n\thello world\n   "
s1 = s.strip(' ')
print(s)
print(s1)
```

```
    hello world


hello world
```

In [74]:

```
print(repr(s))
print(repr(s1))
```

```
'    \n\thello world\n   '
'\n\thello world\n'
```

Immutable

which does not change, or we can not modify them

mutable

we can add remove delete update elements to them

Strings --> Ordered sequential immutable data type

List

ordered, sequential, mutable data type

also known as array (if list is homogenous)

List is collection of elements(objects) (homogenous or hetrogenous)

In [76]:

```
lang = [ 'python', 'java','c', 'c++', 'ruby', 'perl']

# homogenous list (string array)
```

In [77]:

```
num_list = [ 10, 20, 50, 100, 200, 590]

# homogenous list (number array)
```

In [78]:

```
print(lang)
print(num_list)
```

```
['python', 'java', 'c', 'c++', 'ruby', 'perl']
[10, 20, 50, 100, 200, 590]
```

In [81]:

```
unique = [ 'java', 1, ['ello', 'bye'], 3.15, 45+6j ]
# hetrogenous list
```

In [80]:

```
print(unique)
```

```
['java', 1, ['ello', 'bye'], 3.15, (45+6j)]
```

In [82]:

```
lang
```

Out[82]:

```
['python', 'java', 'c', 'c++', 'ruby', 'perl']
```

muttable

func -->

    which returns value of specific type input --> str

        which returns None print --> None

In [85]:

```
name = input("name: ") # ---> value
print(name) # --> None
```

```
name: sachin
sachin
```

In [86]:

```
s = print("hello " )
print(s)
```

```
hello
None
```

In [ ]:

```

```

In [87]:

```
s = "hello world" # strings are immutable
# immutable type always return value on operations
s1 = s.upper()
print(s)
print(s1)
```

```
hello world
HELLO WORLD
```

In [88]:

```
lang = [ 'java', 'c', 'c++', 'ruby', 'perl', 'python', 'swift']
```

In [ ]:

```

```

In [90]:

```
lang.append('bash') # mutable type
```

In [91]:

```
print(lang)
```

```
['java', 'c', 'c++', 'ruby', 'perl', 'python', 'swift', 'bash']
```

In [89]:

```
print(dir(lang))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__di
r__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__ge
titem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_
subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmu
l__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook
__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']
```

In [92]:

```
d = { 'name':'sachin', 'age': 24} # dictionary { key-value pair}
# map type object
print(d)
```

```
{'name': 'sachin', 'age': 24}
```

Everythin is Object in Python

object is a run time instance of class

class is a collection of methods and attribute

In [93]:

```
1 int
```

Out[93]:

```
int
```

In [94]:

```
1 type(int)
```

Out[94]:

```
type
```

In [95]:

```
1 float
```

Out[95]:

```
float
```

In [96]:

```
1 complex
```

Out[96]:

```
complex
```

In [98]:

```
1 str
```

Out[98]:

```
str
```

In [99]:

```
1 list
```

Out[99]:

```
list
```

In [100]:

```
x = 5
print(type(x), x, id(x))
y = "Hello World"
print(type(y), y, id(y))

z = [ 1, 2, 3, 4]
print(type(z), z, id(z))
```

```
<class 'int'> 5 140719517377040
<class 'str'> Hello World 1655075031472
<class 'list'> [1, 2, 3, 4] 1655075146632
```

In [102]:

```
# abstraction
x  = 5 + 9
s = "Hello " + "World"
l = [ 1, 2,] + [ 3, 4]

print(x) # int.__add__
print(s) # str.__add__
print(l) # list.__add__
```

```
14
Hello World
[1, 2, 3, 4]
```

In [ ]:

```
9782131159, 7042397420

sachinyadv3496@gmail.com


```