Strings

- Anything within the quotes is called a string
- Collection of characters

In [2]:

```
s = "jupyter"
print(s)
print(type(s))
```

```
jupyter
<class 'str'>
```

In [14]:

```
print(s[0],s[1],s[2],s[3]) # forward index/positive index
print(s[-1],s[-2],s[-3],s[-4])
print(len(s))
print(min(s),max(s),sorted(s))
```

```
j u p y
r e t y
7
e y ['e', 'j', 'p', 'r', 't', 'u', 'y']
```

In [ ]:

```
# slicing
### cutting into pieces (or) extracting sub strings from
        ##  the original string
```

In [20]:

```
## [start,end,stepcount]
print(s)
print(s[0:4])
print(s[3:7])
print(s[3:])

```

...

In [24]:

```
## jptr
print(s[0:7:2])
print(s[0:7:3])
print(s[0::3])
print(s[::-1])
```

...

In [25]:

```
print(dir(str),end=' ')
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul_
_', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitaliz
e', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'fi
nd', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'is
decimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintabl
e', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartiti
on', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'sw
apcase', 'title', 'translate', 'upper', 'zfill']
```

In [26]:

```
# capitalize()
g = "data analysis workshop"
g.capitalize()
```

Out[26]:

```
'Data analysis workshop'
```

In [28]:

```
# casefold()
g1,g2 = "python","WORKSHOP"
print(g1.casefold())
print(g2.casefold())
```

```
python
workshop
```

In [29]:

```
# title()
print(g.title())
```

```
Data Analysis Workshop
```

In [31]:

```
# upper(),lower()
h1,h2 = "BOOK","pencil"
print(h1.lower())
print(h2.upper())
```

```
book
PENCIL
```

In [33]:

```
# center()
n = "anaconda"
n.center(50)
```

Out[33]:

```
'                    anaconda                    '
```

In [36]:

```
# count()
print(n.count("a"))
print(n.count("n"))
```

```
3
2
```

In [41]:

```
# index(), rindex()
# anaconda
print(n.index('a'))
print(n.rindex('a'))
print(n.index('n'))
print(n.rindex('n'))
```

...

In [45]:

```
# find, rfind()
print(n.find('a'))
print(n.find('h'))
print(n.rfind('a'))
print(n.rfind('z'))
```

...

In [49]:

```
# startswith(), endswith()
a = "vanitha"
b = "apssdc"
print(a.startswith('v'))
print(a.startswith('i'))
print(b.endswith('c'))
print(b.endswith('a'))
```

...

In [53]:

```
# isupper(), islower()
d = "INBOX"
e = "outbox"
print(d.isupper())
print(d.islower())
print(e.islower())
print(e.isupper())
```

```
True
False
True
False
```

In [56]:

```
## isalpha(),isalnum()
b = "WORKSHOP"
print(b.isalpha())
b1 = "workshop123"
print(b1.isalnum())
```

...

In [60]:

```
# isdigit(),isspace()
j = "3874"
print(j.isdigit())
j1 = "  "
print(j1.isspace())
```

```
True
True
```

In [63]:

```
# split()
s = "Welcome To Da Workshop"
print(s.split())
print(s.istitle())
```

```
['Welcome', 'To', 'Da', 'Workshop']
True
```

In [65]:

```
# join()
f = "workshop"
z = "#".join(f)
print(z)
```

```
w#o#r#k#s#h#o#p
```

In [67]:

```
s1 = "W@el@come T#o Da Wor@ksho#p"
print(s1.split("@"))
print(s1.split("#"))
```

```
['W', 'el', 'come T#o Da Wor', 'ksho#p']
['W@el@come T', 'o Da Wor@ksho', 'p']
```

In [70]:

```
# strip(), lstrip(), rstrip()
z = "      screen          "
print(z.strip())
z1 = "       work"
print(z1.lstrip())
z2 = "work1      "
print(z2.rstrip())
```

```
screen
work
work1
```

In [71]:

```
# swapcase()
f1 = "PYThonWorksHOP"
f1.swapcase()
```

Out[71]:

```
'pytHONwORKShop'
```

In [75]:

```
# zfill()
k = "save"
k.zfill(40)
```

Out[75]:

```
'0000000000000000000000000000000000save'
```

In [78]:

```
# madam -- madam
# level --- level
n = input()
if(n==n[::-1]):
    print("Palindrome")
else:
    print("Not palindorme")
```

```
mom
Palindrome
```

1. write a python program to get a string made of the first 2 and last 2 characters of a given string.If the string length is less than2, return the empty string instead

In [83]:

```
n = input()
le=len(n)
if(le<2):
    print("Empty string")
else:
    print(n[0:2]+n[-2:])
```

...

In [ ]:

```
# ApsSDc#6238@($^)
st = input()
up=lw=dg=sp=""
for i in st:
    if(i.isupper()):
        up=up+i
    elif(i.islower()):
        lw=lw+i
    elif(i.isdigit()):
        dg=dg+i
    else:
        sp=sp+i
print("Uppercase Letters are : ",up)
print("Lowercase Letters are: ",lw)
print("Digits are: ",dg)
print("Special Characters are: ",sp)
```

functions

- It is a block of code which runs only when it is called

- Functions are divided into 2 types
  - 1.builtin functions
    - These funtions are developed by the developers
    - ex:- min(),max(),print(),input(),.....
  - 2.user defined functions
    - These functions are created by the users

  - Again UDF are divided into 4 types

  1.with arguments with return value
  2.with arguments without return value
  3.without arguments with return value
  4.without arguments without return value

  Syntax for functions

```
-----------------------------------------
def functionname(arguments/parameters):  # function definition
    statements
functionname(arguments/parameters)   # function calling
```

In [85]:

```
#  1.with arguments with return value
# addition of 2 numbers
a,b = 7,5
def add1(a,b):
    return a+b
add1(a,b)
```

...

In [86]:

```
x,y=int(input()),int(input())
def add2(x,y):
    return x+y
add2(x,y)
```

...

In [87]:

```
# 2.with arguments without return value\
s,e = int(input()),int(input())
def add3(s,e):
    print(s+e)
add3(s,e)
```

...

In [88]:

```
# 3.without arguments with return value
d1,d2 = int(input()),int(input())
def add4():
    return d1+d2
add4()
```

```
847
40958
```

Out[88]:

```
41805
```

In [90]:

```
# i/p: 7
# o/p: odd
n = int(input())
def evenodd(n):
    if(n%2==0):
        print("Even")
    else:
        print("odd")
evenodd(n)
```

```
9
odd
```

Data Structure

```
- A way of organizing the data is called a ds

- tuple
- list
- dictionary
- set
```

tuple()

- It is used to store multiple items in a single variable
- It is immutable (Immutable means we can't change the data once we assigned)
- We can store any type of data
- It can allow the duplicates
- Represented by (), values can be separted by ,

In [94]:

```
t = ()
print(t,type(t))
```

```
() <class 'tuple'>
```

In [95]:

```
t1 = (3,4,"ab","v",6.8,3.7)
print(t1)
```

```
(3, 4, 'ab', 'v', 6.8, 3.7)
```

In [98]:

```
t1 = (3,4,6,8,6.8,3.7,4,3)
print(min(t1),max(t1),sum(t1))
print(sorted(t1),len(t1))
```

```
3 8 38.5
[3, 3, 3.7, 4, 4, 6, 6.8, 8] 8
```

In [99]:

```
print(dir(tuple),end=' ')
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__', '__s
izeof__', '__str__', '__subclasshook__', 'count', 'index']
```

In [101]:

```
g = (3, 3, 3.7, 4, 4, 6, 6.8, 8)
print(g.count(3))
print(g.index(3))
```

```
2
0
```

In [102]:

```
g[2]=4
print(g)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-102-8481b3184c66> in <module>
----> 1 g[2]=4
      2 print(g)

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

In [ ]:

```

```