STRINGS

In [ ]:
```
string is a sequence of characters
```

1) Reading strings

In [2]:
```
# how to read a string in single quotes
name='python'
name
```

Out[2]:
```
'python'
```

In [3]:
```
# how to read a string in double quotes
name1="python"
name1
```

Out[3]:
```
'python'
```

In [ ]:
```
# whatever we mentioned in single quotes or double quotes, we can get output s
```

In [5]:
```
# how to read a string in triple quotes
# triple quotes are called as doc string
# whatever we mentioned in the doc string, it is known as information
string1="""hai how are you
 i am good
 i am learning python"""
string1
```

Out[5]:
```
'hai how are you\n i am good\n i am learning python'
```

In [ ]:
```

```

In [6]:
```
print("hello 'python'")
```

```
hello 'python'
```

In [7]:
```
print('hello,"python"')
```

```
hello,"python"
```

- entire string will be in double quotes, the highlited string in single quotes
- entire string will be in single quotes, the highlited string in double quotes

In [ ]:

2) Type,len,max,min

type

In [9]:
```
# type is used to define the type of datatype
string1='python'
type(string1)
```

Out[9]:
```
str
```

len

In [10]:
```
# len is used to tell the length of a string
string1='python'
len(string1)
```

Out[10]:
```
6
```

max

In [11]:
```
# max is used to find the maximum character of a string, based on ASCII values
string1='python'
max(string1)
```

Out[11]:
```
'y'
```

min

In [12]:
```
# min is used to find the minimum character of a string, based on ASCII values
string1='python'
min(string1)
```

Out[12]:
```
'h'
```

In [ ]:

3)Concatenation

In [13]:
```
# concantenation is used to combine two or more strings
string1='hai'
string2='python'
string1+string2
```

Out[13]: 'haipython'

In [ ]:
```

```

4) Mutability concept

In [15]:
```
# strings are immutable

string1='python'
string1[0]='P'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[15], line 2
      1 string1='python'
----> 2 string1[0]='P'

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

In [ ]:
```
# i want to replace 'p' with 'P'
# based on index operation if you change it or not
# if you change the value by using index operation : mutable
# if you could not change the value by using index operation : immutable  (str
```

In [ ]:
```

```

5)String index

In [16]:
```
# Indexing allows you to access individual characters in a string directly by
name='python'
```

In [ ]:
```
# how many letters are there: 6
# python index starts with : 0

p    y    t    h    o    n
0    1    2    3    4    5
```

In [19]:
```
name[0]
```

Out[19]: 'p'

In [ ]:
```
name[0]   #  p
name[1]   #  y
name[2]   #  t
name[3]   #  h
name[4]   #  o
name[5]   #  n
```

6)String slice

In [ ]:
```
# this concept is same as range() in for loop
# range(start,stop,step)
# string1[start:stop:step]
```

In [24]:
```
string1="hello how are you"
string1[2:10]

# string1[start:stop]
# start = 2
# stop = 10-1 = 9
```

Out[24]:
```
'llo how '
```

In [ ]:
```
h  e  l  l  o     h  o  w     a   r   e       y   o   u
0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16
```

In [25]:
```
string1[1:15:2]

# start=1
# stop=15-1 = 14
# step = 2
```

Out[25]:
```
'el o r '
```

- nothing mentioned at start position : simply starting of letter
- nothing mentioned at stop position : simply last letter
- nothing mentioned at step size : it is positive direction with step value 1

In [26]:
```
print(string1[0:])
print(string1[:len(string1)])
print(string1[:])
print(string1[::])
```

```
hello how are you
hello how are you
hello how are you
hello how are you
```

In [ ]:
```
-17 -16 -15  -14 -13 -12  -11  -10  -9  -8  -7  -6  -5   -4  -3  -2  -1
 h   e   l    l   o       h    o    w      a   r   e       y   o   u
0    1   2    3   4   5   6    7    8   9  10  11  12  13  14  15  16
```

In [27]:
```
string1[-2:-15:-1]
# start= -2
# stop= -15+1 =-14
# step is : -ve
```

Out[27]:
```
'oy era woh ol'
```

In [ ]:

7)String methods

In [29]:
```
dir('string')
```

Out[29]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
```

```
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'removeprefix',
 'removesuffix',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

Capitalize/Casefold/upper/lower

- capitalize

In [31]:
```
name1='venkatesh'
```

In [30]:
```
help(name.capitalize)                # help is used to explain how to represe
```

```
Help on built-in function capitalize:

capitalize() method of builtins.str instance
    Return a capitalized version of the string.

    More specifically, make the first character have upper case and the rest
lower
    case.
```

In [32]:
```
# capitalize will make first letter as upper case letter in a given string
name1.capitalize()
```

Out[32]:
```
'Venkatesh'
```

- Casefold

In [36]:
```
name1='VENKATESH'
```

In [39]:
```
help(name1.casefold)
```

```
Help on built-in function casefold:

casefold() method of builtins.str instance
    Return a version of the string suitable for caseless comparisons.
```

In [37]:
```
# casefold will convert a given string into lower case letters
name1.casefold()
```

Out[37]:
```
'venkatesh'
```

- upper

In [40]:
```
name1='venkatesh'
```

In [41]:
```
help(name1.upper)
```

```
Help on built-in function upper:

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

In [42]:
```
# upper will convert a given string into upper case letters
name1.upper()
```

Out[42]:
```
'VENKATESH'
```

- lower

In [44]:
```
name1='VENKATESH'
```

In [45]:
```
help(name1.lower)
```

```
Help on built-in function lower:

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

In [46]:
```
# lower will convert a given string into lower case letters
name1.lower()
```

Out[46]:
```
'venkatesh'
```

In [ ]:
```

```

Index/find

- index

In [47]:
```
# The process of accessing a specific element in a sequence
# if i want to know index of 't'
string1='python'
string1.index('t')
```

Out[47]:
```
2
```

In [49]:
```
'python'.index('t')
```

Out[49]:
```
2
```

In [50]:
```
# suppose if i want to extract index number of 't' in a given string
string1='restart'
f_o=string1.index('t')
s_o=string1.index('t',f_o+1)
print(f_o,s_o)
```

```
3 6
```

In [54]:
```
string1='python'
string1.index('T')
# whenever given input is not found in a given string, then it will print erro
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[54], line 2
      1 string1='python'
----> 2 string1.index('T')

ValueError: substring not found
```

- find

- both index and find function are same
- difference between index and find
- when it is error in index , it will show error

- when it is error in find , it will return -1 value

In [56]:
```
string1='python'
string1.find('t')
```

Out[56]: 2

In [52]:
```
'python'.find('t')
```

Out[52]: 2

In [53]:
```
# suppose if i want to extract index number of 't' in a given string, we can a
string1='restart'
f_o=string1.index('t')
s_o=string1.index('t',f_o+1)
print(f_o,s_o)
```

```
3 6
```

In [55]:
```
string1='python'
string1.find('T')
# whenever given input is not found in a given string, then it will return -1
```

Out[55]: -1

In [ ]:
```

```

Strip-rstrip-lstrip

In [60]:
```
string1=' hai how are you '
```

In [61]:
```
string1.strip(),string1.lstrip(),string1.rstrip()
```

Out[61]: ('hai how are you', 'hai how are you ', ' hai how are you')

- if you want to remove the spaces we use method strip
- strip: it will remove spaces both sides
- lstrip: it will remove left side space
- rstrip: it will remove right side space

In [ ]:
```

```

Startswith/endswith

In [62]:
```
string1='hai how are you'
string1.startswith('hai')
```

Out[62]: True

In [65]:
```
string1.startswith('h')
```

Out[65]: True

In [66]:
```
string1.startswith('how')
```

Out[66]: False

In [64]:
```
string1.endswith('you')
```

Out[64]: True

In [68]:
```
string1.endswith('u')
```

Out[68]: True

In [ ]:
```

```

Count

In [69]:
```
string1="HAI HAI hai hai"
string1.count('h')
# here it is counting the total number of 'h' in a given string
```

Out[69]: 2

In [71]:
```
string2="ola ola ola ola"
print(string2.count('a',4))
# here 4 means we are counting 'a' from 4th index

# string.count(<char>,start_index)

print(string2.count('a',4,9))
# how many 'a' are there in between 4 and 8th(9-1) index
```

```
3
1
```

In [72]:
```
string2.count('ola')
```

Out[72]: 4

In [ ]:

Replace

In [ ]:
```
The replace() method replaces a specified phrase with another specified phrase
```

In [77]:
```
string1='venkatesh'
# i want to replace 't' with 'T'
string1.replace('t','T')
```

Out[77]:
```
'venkaTesh'
```

In [ ]:
```
# string1.replace(old,new)
```

In [81]:
```
print(string1.replace('e','E',1))
print(string1[:2]+string1[2:].replace('e','E',1))
```

```
vEnkatesh
venkatEsh
```

In [82]:
```
string2='restart'
print(string2.replace('r','$'))
print(string2.replace('r','$',1))
```

```
$esta$t
$estart
```

In [ ]:

In [ ]:

In [ ]: