In [ ]:
```
###################### part-1#########################
- Dat types

- type casting

- Basic codes using eval and input

- Conditional statements

- Try-except

- Functions

- For and while
###################### part-2###################

- strings

- list

- dictionary

- tuple and set

- lambda functions

- file handling

############# Part-3 OOPS ####################
```

strings

In [ ]:
```
- int

- float

- str

- boolean

- complex

- list

- tuple

- dic

- set

- frozen set

- range

- byte array
```

- intilaiztion

- type
- len
- min
- max
- sorted
- reveresd
- in
- for loop with in
- index
- for loop with index
- mutable immutable
- concatenation
- methods
- conditions

In [1]:
```
str1='python'
str1
```

Out[1]: 'python'

In [2]:
```
str2="python"
str2
```

Out[2]: 'python'

In [3]:
```
print(str1)
```

```
python
```

- strings defined by using single quotes or double quotes
- but python by default will give as single quotes only
- When you print the string, the answer will display without quotes

doc string

- we can write the strings using triple quotes
- This method called as doc string

- It is the way of communicating information to the user
- Do not apply triple quotes when you do the code

In [4]:
```
import random
random.randint()
```

type

In [5]:
```
type(str1)
```

Out[5]: str

In [6]:
```
len(str1)
```

Out[6]: 6

In [ ]:
```
'python'  has 6 charcters
```

In [7]:
```
max(str1)
```

Out[7]: 'y'

- ascii number of 'y' is greater than compare to others

In [8]:
```
len('abc123')
```

Out[8]: 6

In [9]:
```
max('abc123')
```

Out[9]: 'c'

In [10]:
```
max('321')
```

Out[10]: '3'

In [11]:
```
min('abc123')
```

Out[11]: '1'

In [ ]:
```
type(<>)
print(<>)
len(<>)
max(<>)
min(<>)
```

- if we observe the above pattern
- its like keyword()
- these all are inbuilt function

- inbulit means these function can use any data type
- means we can apply for strings,list,tuple,dictionay etc

difference between methods and inbuilt functions

- methods comes from packages
- inorder to use any method we need import the package
- for in-built functions no need of any packages
- import random === random.randint
- import math ==== math.sqrt
- import time === time.sleep
- type() print() min() max() len()

reveresd

In [12]:
```
reversed('python')
```

Out[12]:
```
<reversed at 0x18829b734f0>
```

- when we apply reveresed in built function
- the output will store at memory
- we can see the output by iterate a loop
- we can see the output by applying a list type cast

In [13]:
```
out=reversed('python')
for i in out:
    print(i)
```

```
n
o
h
t
y
p
```

In [14]:
```
int('10')
```

Out[14]:
```
10
```

In [15]:
```
list(reversed('python'))

# eval()
# int()
# float()
```

Out[15]: ['n', 'o', 'h', 't', 'y', 'p']

In [16]:
```
tuple(reversed('python'))
```

Out[16]: ('n', 'o', 'h', 't', 'y', 'p')

In [17]:
```
reversed(sequence='python')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 reversed(sequence='python')

TypeError: reversed() takes no keyword arguments
```

In [ ]:
```
import random
random.randint(a=10,b=20)
```

In [ ]:
```
(a,/)
a=10 ==== error
10   ==== no error

(a,/,b)
a=10,b=20 error
10,20 no error
10,b=20 no error
```

- / means position only papatemers
- before / dont give the variable name
- after / give varibale name

sorted

In [18]:
```
sorted('python')
```

Out[18]: ['h', 'n', 'o', 'p', 't', 'y']

In [ ]:
```
sorted('python',True) # work error
sorted(iterable='python',True) # error
sorted(iterable='python',reverse=True) # error
sorted('python',reverse=True) # works
```

In [19]:
```
sorted('python',reverse=True)
```

Out[19]: ['y', 't', 'p', 'o', 'n', 'h']

- type
- len
- max

- min
- reveresd
- sorted

In [ ]:
```
What is the actual use of '/'?
how can you wh
# Default arguments

def add(a,b,c=10)
    add(a=10,b,c)
```

in

In [20]:
```
'p' in str1
```

Out[20]: True

In [21]:
```
for i in str1:
    print(i,end=' ')
```

```
p y t h o n
```

In [24]:
```
#Q1wap ask the user how many times 'a' repaeted in a given strin
# string='hai hai hai'

# idea
# iterate each letter using in
# apply the conditon if that letter equal to 'a'
# then count it
count=0
string='hai hai hai'
for i in string:
    if i=='a':
        count=count+1

print("the number of a's are:",count)
```

```
the number of a's are: 3
```

In [ ]:
```
# WAP to count the number of vowels in a given string
# string= 'hai how are'
# Vowels= a,i,o,a,e: 5

# non repetaed Vowels= a,i,o,e: 4
```

In [ ]:
```
# Wap Q2)
# WAP to count the number of vowels in a given string
# string= 'hai how are'
# Vowels= a,i,o,a,e: 5
```

In [2]:
```
string1='hai hello how'
for i in string1:
    if i=='a' or i=='e' or i=='i' or i=='o' or i=='u':
        print(i)
```

```
a
i
e
o
o
```

In [3]:
```
string1='hai hello how'
vowels='aeiou'
for i in string1:
    if i in vowels:
        print(i)
# step-1:  i='h'  === > 'h' in 'aeiou' F
# step-2:  i='a'  === > 'a' in 'aeiou' T
# step-3:  i='i'  ==== > 'i' in 'aeiou' T
```

```
a
i
e
o
o
```

In [8]:
```
# we are creating a empty room
# we are calling each person
# we are checking that person is available in the room or not
# if he is not avilable
#       #we are checking the age
#            he is our person
s1='hai how are'
s2='aeiou'
s3=''
for i in s1:
    if i not in s3 and i in s2:
        print(i)
        s3=s3+i
len(s3)

# step-1: i='h' == > 'h' not in '' and 'h' in 'aeiou' : Not
# step-2: i='a' === > 'a' not in '' and 'a' in 'aeiou': C
#              ''+'a'='a'   s3='a'
```

```
a
i
o
e
```

Out[8]: 4

index

In [ ]:
```
-6   -5   -4    -3   -2   -1 # Negative index
p    y     t     h    o    n
0    1     2     3    4     5  # Postive index
```

In [14]:
```
str1='python'
str1[0]   # Access
str1[1]
str1[2]
str1[3]
str1[4]
str1[5]
```

```
str1[i]
```

Out[14]: 'n'

In [ ]:
```
len()
type()
eval()
input()
int()
float()
sorted()
reversed()
```

In [17]:
```
str1='python python'
n=len(str1)
for i in range(n):
    print(i,str1[i])
```

```
0 p
1 y
2 t
3 h
4 o
5 n
6
7 p
8 y
9 t
10 h
11 o
12 n
```

In [18]:
```
str1='python'
n=len(str1)
for i in range(n):
    print(f"the positive index of {str1[i]} is {i}")
```

```
the positive index of p is 0
the positive index of y is 1
the positive index of t is 2
the positive index of h is 3
the positive index of o is 4
the positive index of n is 5
```

In [ ]:
```
str1='python'
str1[0]   # p
str1[1]   # y
str1[2]   # t
str1[3]   # h
str1[4]   # o
str1[5]   # n

str1='python'
n=len(str1)
for i in range(n):
    print(f"the positive index of {str1[i]} is {i}")
```

In [24]:
```
# Negative index means negative numbers
# -6   -5   -4   -3   -2   -1
```

```
#  p    y   t    h    o    n
#  0    1   2    3    4    5
str1='python'
n=len(str1)
for i in range(-n,0):
    print(f"the negative index of {str1[i]} is {i}")
```

```
the negative index of p is -6
the negative index of y is -5
the negative index of t is -4
the negative index of h is -3
the negative index of o is -2
the negative index of n is -1
```

In [ ]:
```
# Negative index means negative numbers
# -6   -5   -4   -3   -2   -1
#  p    y   t    h    o    n
#  0    1   2    3    4    5
# the postive index of p is 0 and negative index is -6
# the postive index of y is 1 and negative index is -5

# Get the above answer using single for loop only
```

In [28]:
```
str1='python'
n=len(str1)
for i in range(n):
    print(f"the postive index of {str1[i]} is {i} and negative index is {i-n}")
```

```
the postive index of p is 0 and negative index is -6
the postive index of y is 1 and negative index is -5
the postive index of t is 2 and negative index is -4
the postive index of h is 3 and negative index is -3
the postive index of o is 4 and negative index is -2
the postive index of n is 5 and negative index is -1
```

Mutable-immutable

- Mutable means change
- Immutable means not able to change
- Example: 'welcome'
- Output : 'weLcome'
- is this possible using **index opeartions**? NO
- Strings are **immutable**
- We can not change the letters using index

In [30]:
```
str1='welcome'
str1[2]='L'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[30], line 2
      1 str1='welcome'
----> 2 str1[2]='L'

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

In [32]:
```
l=['w','e','l','c','o','m','e']
l[2]='L'
l
```

Out[32]:
```
['w', 'e', 'L', 'c', 'o', 'm', 'e']
```

when to use range and when to use in

- in operators access the elements directly
- range operator access the elements by using index
- For example : str='welcome'
- If you want to access each letter from str
- we can use in and range
- in: for i in str: i becomes w,e,l,c,o,m,e
- range: for i in range(len(str)): i becomes 0,1,2,3,4,5
    - str[i]

Use the range operator when we you required a index use

In [35]:
```
str1='welcome'
for i in str1:
    print(i)

for i in range(len(str1)):
    print(str1[i])
```

```
w
e
l
c
o
m
e
w
e
l
c
o
m
e
```

In [ ]:
```
sir while index v shud represent as len(str)
```

```
index==== number
in or range === range
how to provid the number in range using == len(str)
```

In [ ]:
```
# Q1) count the number of 'a' in a given string: in
# Q2) count the number of 'a' in a given string: range
# Q3) print the indexed of 'a' in a given string
# Q4) sum the indexed of 'a' in a given string
# string: 'hello how are you'
# Q5) count the all the vowels in a given string: in
# Q6) count the all the vowels in a given string: range
# Q7) count the non repated vowels in a given string: in
# Q8) count the all the vowels in a given string: range
# Q9) Postive index negative index using for loop
# Q10) Postive index negative index using While loop
```

Slice

- Slice means part of string
- It is exatcly same like range operation start,stop,step
- str[start:stop:step]

In [ ]:
```
s='hai how are you'
#-15  -14  -13  -12  -11  -10  -9  -8  -7  -6  -5  -4  -3  -2  -1
#h     a    i         h    o    w       a   r   e       y   o   u
#0     1    2   3     4    5    6   7   8   9   10  11  12  13  14
```

$pattern - 1$

string[start:]

- start means start value only
- Increment by one only
- Nothing mentioned at the stop position means till last

In [2]:
```
#-15  -14  -13  -12  -11  -10  -9  -8  -7  -6  -5  -4  -3  -2  -1
#h     a    i         h    o    w       a   r   e       y   o   u
#0     1    2   3     4    5    6   7   8   9   10  11  12  13  14
s='hai how are you'
s[2:]
```

Out[2]:
```
'i how are you'
```

$pattern - 2$

string[start:stop]

- start means start value only
- Increment by one only
- last= stop-1

- string[2:10]
  - start =2
  - last=10-1=9

In [3]:
```
#-15  -14  -13  -12  -11  -10  -9  -8  -7  -6  -5  -4  -3  -2  -1
#h     a    i         h    o    w       a   r   e       y   o   u
#0     1    2   3     4    5    6   7   8   9   10  11  12  13  14
s='hai how are you'
s[2:10]
```

Out[3]:
```
'i how ar'
```

In [4]:
```
s[-14:-4]
# start=-14
# last = -4-1=-5
```

Out[4]:
```
'ai how are'
```

In [5]:
```
s[2:-2]
# start=2
# stop=-2-1=-3
```

Out[5]:
```
'i how are y'
```

In [6]:
```
s[10:2]
```

Out[6]:
```
''
```

In [7]:
```
for i in range(10,2):
    print(i)
```

In [8]:
```
s[-4:-14]
```

Out[8]:
```
''
```

In [ ]:
```
s[2:-2] # ans

for i in range(2,-2): # No ans
    print(i)
```

$pattern - 3$

string[start:stop:step]

- start means start value only
- step means how much gap
  - positive value indicates positive direction
    - last=stop-1
  - negative value indicates negative direction

- last= stop+1

In [9]:
```
#-15  -14  -13  -12  -11  -10  -9  -8  -7  -6  -5  -4  -3  -2  -1
#h     a    i         h    o    w       a   r   e       y  o   u
#0     1    2   3     4    5    6  7    8   9   10  11  12 13  14
s='hai how are you'
s[2:12:2]
# start=2  step=2 postive  last=12-1=11
```

Out[9]:
```
'ihwae'
```

In [10]:
```
s[2:-2:2]
# start=2  step=2 postive  last =-2-1=-3
```

Out[10]:
```
'ihwaey'
```

In [11]:
```
s[2:-2:-2]
```

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

In [15]:
```
#-15  -14  -13  -12  -11  -10  -9  -8  -7  -6  -5  -4  -3  -2  -1
#h     a    i         h    o    w       a   r   e       y  o   u
#0     1    2   3     4    5    6  7    8   9   10  11  12 13  14
s='hai how are you'

print('1:',s[3:13:3])  # w
print('2:',s[3:13:-3]) # F
print('3:',s[4:-13:3]) # ===== F
print('4:',s[4:-13:-3]) #  === h
print('5:',s[-3:13:3])  # w
print('6:',s[-3:13:-3]) # f
print('7:',s[-3:-13:-3]) #  w
print('8:',s[13:3:3]) #  F
print('9:',s[13:3:-3]) # W
print('10:',s[13:-3:3]) #  F
print('11:',s[13:-3:-3]) # W
print('12:',s[-13:3:3]) #  w
print('13:',s[-13:3:-3]) # f
print('14:',s[-13:-3:3]) # w
print('15:',s[-13:-3:-3]) # f
```

```
1:  wry
2:
3:
4: h
5: y
6:
7: yrw
8:
9: oe h
10:
11: o
12: i
13:
14: ioa
15:
```

In [ ]:
```
string[start:stop:step]
step ==== how much gap
```

```
1  2   3  4  5
step=3
1  4
```

Concatenation

- Take two strings
- Apply + - / *
- Tell me which works and which not works

In [16]:
```
str1='hello'
str2='hai'
str1+str2
```

Out[16]:
```
'hellohai'
```

In [17]:
```
str1*str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 str1*str2

TypeError: can't multiply sequence by non-int of type 'str'
```

In [18]:
```
str1-str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 str1-str2

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

In [19]:
```
str1/str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[19], line 1
----> 1 str1/str2

TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

In [ ]:
```
* :TypeError: can't multiply sequence by non-int of type 'str'
- :TypeError: unsupported operand type(s) for -: 'str' and 'str'
/ :TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

- subtraction and division operands never use
- but we can use multiplication by using a str with any integer

In [20]:
```
str1='hai'
str1*3
```

Out[20]: 'haihaihai'

In [ ]:
```
#str1+str1+str1 ==== str1*3
```

In [ ]:
```
- intialiization

- len

- type

- max

- min

- reversed

- sorted

- in

- index

- mutable immutable

- slice

- concatenation
```

string methods

In [21]:
```
# Import the pcakge
# dir(package)
str1='abc'
dir('')  # dir(str1)
```

Out[21]:
```
['__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']
```

In [ ]:
```
# import package
# pcakge_name.method_name()

# len()
# type()
```

upper

- upper is a string method
- will convert string into upper case

In [22]:
```
str1='hello'
str1.upper()
#upper(str1) # It is not a inbuilt
```

Out[22]: 'HELLO'

In [24]:
```
str1='welcome'
str1.upper()
```

Out[24]: 'WELCOME'

- lower
- casefold
- capitalize
- center

In [26]:
```
s1='hello'
s1.capitalize()
```

Out[26]: 'Hello'

In [28]:
```
s1='HELLO'
s1.lower()
```

Out[28]: 'hello'

In [29]:
```
s1='HELLO'
s1.casefold()
```

Out[29]: 'hello'

- lower and casefold will give caseless means lower case

In [37]:
```
s1='hello'  # 5
s2=s1.center(10,'-') # 10
len(s2)
```

Out[37]: 10

In [38]:
```
s2
```

Out[38]: '--hello---'

In [ ]:
```
'hello' ===== '   hello  '
```

In [39]:
```
s1='gmail'
s1.center(10,'*')
```

Out[39]: '**gmail***'

Title

In [40]:
```
s='hello how are you'
s.title()
# Understand the difference with
# Capitalize and Upper
```

Out[40]: 'Hello How Are You'

Count

In [41]:
```
str1='hello hai how are you'
# I want to know how many 'a' are avialable
count=0
for i in str1:
    if i=='a':
        count=count+1

print(count)
```

2

In [42]:
```
str1='hello hai how are you'
str1.count('a')
```

```
# with out using any method
```

Out[42]: 2

In [ ]:
```
# Q1) str='hello hai how are you'
#     str='Hello Hai How Aare You'  with out using method
```

In [49]:
```
str1='ola ola ola'
str1.count('ola ola ola')
str1.count(str1)
```

Out[49]: 1

In [50]:
```
str1.count('uber')
```

Out[50]: 0

In [ ]:
```
# Q1) str='hello hai how are you'
#     str='Hello Hai How Aare You'  with out using method

# Q2) str='ola ola ola'
#     count=3   with out using method
```

In [ ]:
```
# Idea is you need to divide the sentence into words
# you need to iterate each word and apply .title
# again you need to concatenate with all the words
```

In [6]:
```
str1='ola ola ola'
str1.count('o',8) # we are searching the number of 'o' from 8th index
```

Out[6]: 1

In [8]:
```
str1='ola ola ola'
str1.count('o',2,4) # we are searching the 'o' from 2 to 3
```

Out[8]: 0

In [10]:
```
# o l a   o l a   o l  a
# 0 1 2 3  4 5 6 7 8 9 10
str1[2:4]  # what you are seeing output
```

Out[10]: 'a '

replace

In [11]:
```
str1='welcome'
# o/p: 'weLcome'
```

In [13]:
```
str1[2]='L'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 str1[2]='L'

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

In [14]:
```
str1.replace('l','L')
```

Out[14]: 'weLcome'

In [15]:
```
str1='wlllcome'
str1.replace('l','L') # old='l'  new='L'
# by default it will replace all occurences
```

Out[15]: 'wLLLcome'

In [16]:
```
str1='wlllcome'
str1.replace('l','L',1)
# old='l'  new='L'
# count=1
# replace first occurence
```

Out[16]: 'wLllcome'

In [17]:
```
str1='wlllcome'
str1.replace('l','L',2)
# old='l'  new='L'
# count=2
# replace two occurence
```

Out[17]: 'wLLlcome'

In [18]:
```
str1='wlllcome'
str1.replace('l','L',3)
```

Out[18]: 'wLLLcome'

In [22]:
```
str1='wlllcome'
str1.replace('l','L',2)
```

Out[22]: 'wLLlcome'

In [23]:
```
# str='restart'
# op='resta$t'
str1='restart'
str1.replace('ar','a$')
```

Out[23]: 'resta$t'

In [29]:
```
str1='restart'
s1=str1[:1]
s1 # 'r'
s2=str1[1:]
s2 # 'estart'
s3=s2.replace('r','$') # 'esta$t'
s3
s4=s1+s3
s4
# slice
# replace
# concate
```

Out[29]: 'resta$t'

In [30]:
```
str1,s1,s2,s3,s4
```

Out[30]:
```
('restart', 'r', 'estart', 'esta$t', 'resta$t')
```

In [35]:
```
str1[::-1].replace('r','$',1)[::-1]
```

Out[35]:
```
'resta$t'
```

In [43]:
```
s1=str1[::-1]
s2=s1.replace('r','$',1)
s3=s2[::-1]
s1,s2,s3

str1[::-1].replace('r','$',1)
str1[::-1].replace('r','$',1)[::-1]
```

Out[43]:
```
('tratser', 't$atser', 'resta$t')
```

In [38]:
```
s1='hello python hello restart'
s1[:6]
```

Out[38]:
```
'hello '
```

Index-find

- Index
  - in above example we are counting the 'r' by own
  - here 'r' is first letter then we are able to do
  - imagine that 'r' is after 100 charcters , we are not able to do
  - the index of the 'r' should come automatically

In [39]:
```
s='hello hai how are you'
#s[start:stop:step]
#s[start:stop]
s[2:]  # start=2  stop=
```

Out[39]:
```
'llo hai how are you'
```

In [40]:
```
s[:10] # start=0 stop=10-1=9
```

Out[40]:
```
'hello hai '
```

In [41]:
```
s[::]# start=0 stop=till last
```

Out[41]:
```
'hello hai how are you'
```

In [42]:
```
s[::-1]
```

Out[42]:
```
'uoy era woh iah olleh'
```

In [48]:
```
str1='ola ola ola'
#str1.count()
#str1.replace()
str1.index('l')
str1.index('l',6) # we are searching the 'l' index from 5th
```

Out[48]: 9

In [49]:
```
str1.index('l',2,4)
# If substring not found it will raise value error
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[49], line 1
----> 1 str1.index('l',2,4)

ValueError: substring not found
```

In [50]:
```
str1='ola ola ola'
str1.replace('z','l')
```

Out[50]: 'ola ola ola'

In [51]:
```
str1.count('z')
```

Out[51]: 0

In [54]:
```
str1='ola ola ola'
# I want all 'l' indexes
str1.index('l')
# If i want to get 2nd index our eyes start look from where
# after first l
# after 1 ==== 2
str1.index('l',2)
str1.index('l',6)
```

Out[54]: 9

In [60]:
```
str1='ola ola ola ola ola'
str1.index('l',1+0)
str1.index('l',1+1)
str1.index('l',1+5)
str1.index('l',1+9)
str1.index('l',1+13)
```

Out[60]: 17

In [62]:
```
str1='ola ola ola ola ola'
i1=str1.index('a',1+0)
i2=str1.index('a',1+i1)
i3=str1.index('a',1+i2)
i4=str1.index('a',1+i3)
i5=str1.index('a',1+i4)
i1,i2,i3,i4,i5
```

Out[62]: (2, 6, 10, 14, 18)

In [68]:
```
str1='ola ola ola ola ola'
i1=str1.index('l')
i2=str1.index('l',1+i1)
i3=str1.index('l',1+i2)
i4=str1.index('l',1+i3)
i5=str1.index('l',1+i4)
```

In [72]:
```
#how do we come up with numbers of 5,9 and 13?
str1='ola ola ola ola ola'
# 2nd index of 'l'
str1.index('l',2)
```

Out[72]: 5

Find

- Tell me the difference between index and find

In [73]:
```
str1='ola ola ola ola ola'
i1=str1.find('l')
i2=str1.find('l',1+i1)
i3=str1.find('l',1+i2)
i4=str1.find('l',1+i3)
i5=str1.find('l',1+i4)
i6=str1.find('l',1+i5)
i1,i2,i3,i4,i5,i6
```

Out[73]: (1, 5, 9, 13, 17, -1)

- upper/lower/capitalize/casefold/title
- count
- replace
- index/find

In [74]:
```
dir('')
```

Out[74]:
```
['__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']
```

In [ ]:
```
'isalnum',
'isalpha',
'isascii',
'isdecimal',
'isdigit',
'isidentifier',
'islower',
'isnumeric',
'isprintable',
'isspace',
'istitle',
'isupper',
```

In [75]:
```
'hello'.isupper()
```

Out[75]:
```
False
```

In [76]:
```
'HELLO'.isupper()
```

Out[76]:
```
True
```

In [ ]:
```
str1='ola ola ola'
str1.count('z')  # 0
str1.replace('z','a') # same string
str1.index('z') # Value error
str1.find('z')  # -1
```

lstrip-rstrip-strip

In [1]:
```
str1=' hai how are you ' # strip
str2='hai how are you '  # right side rstrip
str3=' hai how are you'  # left side means left strip
```

In [5]:
```
str1.strip(),str1.rstrip(),str1.lstrip()
```

Out[5]:
```
('hai how are you', ' hai how are you', 'hai how are you ')
```

In [6]:
```
str2.strip(),str2.rstrip(),str2.lstrip()
```

Out[6]:
```
('hai how are you', 'hai how are you', 'hai how are you ')
```

In [ ]:
```
str2.ltrip(),str2.lstrip(),str2.lstrip()
```

startswith-endswith

In [9]:
```
str1='hello hai how are you'
str1.startswith('hello')
```

Out[9]:
```
True
```

In [13]:
```
str1='hello hai how are you'
str1.endswith('you')
```

Out[13]:
```
True
```

In [14]:
```
str1.startswith(str1)
str1.endswith(str1)
```

Out[14]:
```
True
```

In [ ]:
```
'omkar.nallagoni@cognizant.com'
'virat.kohli@rcb.com'
'rohit.sharma@mi.com'
'a.b@c.com'

# 3 indexes
# first dot index
# @ index
# second dot index
# apply the slice
```

In [17]:
```
str1='omkar.nallagoni@cognizant.com'
i1_dot=str1.index('.')
i2_dot=str1.index('.',1+i1_dot)
i3=str1.index('@')
```

In [20]:
```
# First name between start and first dot
fname=str1[:i1_dot]
# second name between first dot and @
sname=str1[i1_dot+1:i3]
# Cname between @ to second dot
cname=str1[i3+1:i2_dot]
cname
```

Out[20]:
```
'cognizant'
```

In [23]:
```
str1='rohith.kohli@india.com'
i1_dot=str1.index('.')
i2_dot=str1.index('.',1+i1_dot)
i3=str1.index('@')
fname=str1[:i1_dot]
sname=str1[i1_dot+1:i3]
```

```
cname=str1[i3+1:i2_dot]
print(fname,sname,cname)
```

```
rohith kohli india
```

split

In [24]:
```
str1='hello hai how are you'
str1.split()
```

Out[24]:
```
['hello', 'hai', 'how', 'are', 'you']
```

In [25]:
```
str1='hello hai, how are you'
str1.split(',')
```

Out[25]:
```
['hello hai', ' how are you']
```

In [26]:
```
str1='hello hai, how are you'
str1.split('h')
#
# ello
# ai,
# ow are you
```

Out[26]:
```
['', 'ello ', 'ai, ', 'ow are you']
```

- upper/lower/casefold
- capitalize/title
- center
- count
- replace
- index/find
- startwith/endswith
- lstrip/rstrip/strip
- split
- is_

In [27]:
```
dir('')
```

Out[27]:
```
['__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']
```

In [ ]: