19th-nov

String

- How to intialize the strings
- in built functions
  - type
  - print
  - len
  - min
  - max
  - reversed
  - sorted
- index operation
- concatenation
- mutable vs immutbale
- slicing

Initialization

- string represent with single quotes
- strings represent with double quotes
- string represent with triple quotes
- when you print a string, the output shows as without quotes
- A string represent with triple quotes: **Doc String**
- If you want highlight any word in entire string
  - provide the entire string in double quotes and highlight the word with singel quotes vice versa

In [1]:
```
str1 = 'python'
print(str1),type(str1)
```

```
python
```

Out[1]: (None, str)

In [3]:
```
str2 = "python"
print(str2),type(str2)
```

```
python
```

Out[3]: (None, str)

Doc-String

In [4]:
```
str3= """Hello
Python
Can we create a roburt using Python"""
print(str3)
```

```
Hello
Python
Can we create a roburt using Python
```

In [5]:
```
str4 = 'I like "Python"'
print(str4)
```

```
I like "Python"
```

In [5]:
```
str5 = "I like 'python'"
print(str5)
```

```
I like 'python'
```

some inbuilt functions

- max()
- min()
- type()
- print()
- len()
- reversed()
- sorted()

In [8]:
```
str1 = 'python'
max(str1)
# Because of ASCII
```

Out[8]: 'y'

In [9]:
```
for i in str1:
    print(i,ord(i))
```

```
p 112
y 121
t 116
h 104
o 111
n 110
```

In [ ]:
```
max == 121 === 'y'
min == 104 === 'h'
```

In [6]:
```
max('python')
```

Out[6]: 'y'

In [7]:
```
min('python')
```

Out[7]: 'h'

In [1]:
```
str1 = 'python for coding'
min(str1)
```

Out[1]: ' '

- space is a character so don't get panic
- We can pass direct string inside the fuction

In [8]:
```
type('python')
```

Out[8]: str

In [9]:
```
len('pyhton')
```

Out[9]: 6

In [11]:
```
n1 = reversed('Python')
print(n1)
```

```
<reversed object at 0x00000203DB73E500>
```

SORTED

- Sorted means sorting the letters based on ascii number
- there are two possible sorting available
  - ascending : small to high
  - decending : high to small

In [14]:
```
sorted(str1) # by default ascendeing order
```

Out[14]: ['h', 'n', 'o', 'p', 't', 'y']

- when you apply the shfit + tab

- there is some arguments will be available
- Focus on two arguments
  - iterable
  - reverse = False
- because reverse is a default argument by default ascending order is coming
- if you want to change the order then change the default parameter value

In [15]:
```
sorted(str1, reverse=True)
```

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

In [ ]:
```
sorted(iterable=str1,reverse=False)  # Fails
sorted(iterable=str1,False)  # Fails
sorted(str1,False)  # Fails

sorted(str1, reverse=True) # Works
```

- we should not allowed to use iterabe arguments name while providing the vaule
- we should use the argument names before '/'
  - iterable argument name is there before '/'
  - so do not use iterable name

- any function indicates * means
- you can use any variable after *
- after * there are two arguments is there
  - key
  - reverse
- you can use both
- you can use anyone
- you no need to use anthing
- **What is the meaning of star**
  - you can provide one variable
  - you can provide two variable
  - or you cant proivde any variable it will works

In [10]:
```
sorted(iterable = str1, reverse=False) # fail
sorted(iterable = str1, False) # fail
sorted(str1, False) # fail

sorted(str1, reverse=False) # Works
```

```
  Cell In[10], line 2
    sorted(iterable = str1, False) # fail
                                   ^
SyntaxError: positional argument follows keyword argument
```

Why 3 are fails

- iterable name or iterable variable name we cant use

In [ ]:
```
sorted("hello") # Works
sorted(iterable='Hello') # Fails
```

In [18]:
```
sorted("hello") # no need to use any variable
```

Out[18]:
```
['e', 'h', 'l', 'l', 'o']
```

In [19]:
```
sorted("hello",reverse=True) # one variable use
```

Out[19]:
```
['o', 'l', 'l', 'h', 'e']
```

- based on ascii value it is giving the order

In [2]:
```
sorted('hello', key=len) # one variable use

# You will not understand Now when Dictionary will start then you know
```

Out[2]:
```
['h', 'e', 'l', 'l', 'o']
```

In [11]:
```
sorted('hello', reverse=True, key=len) # two variable use
```

Out[11]:
```
['h', 'e', 'l', 'l', 'o']
```

reversed

In [13]:
```
str1 = 'python'
reversed(str1)
```

Out[13]:
```
<reversed at 0x1d6e39545e0>
```

what is the meaning of these

- <reversed at 0x1d6e39545e0>
- The output is stored a memory location
- whenever you want to see the answer or output
- use a list or for loop

In [15]:
```
str1 = 'python'
ans = reversed(str1)
```

In [16]:
```
type(ans)
```

Out[16]:
```
reversed
```

In [28]:
```
reversed(str1)

# the output is stored a memory location
# whenever you want to see the output
# use a list or for loop
```

Out[28]:
```
<reversed at 0x203db723bb0>
```

To see the output using for loop

In [29]:
```
for i in ans:
    print(i)
```

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

To see the output using a list

In [31]:
```
str1 = 'python'
ans = reversed(str1)
list(ans)
```

Out[31]:
```
['n', 'o', 'h', 't', 'y', 'p']
```

concatenation

- String concatenation in Python refers to the process of joining two more strings into a new string. This can be done using operators (+ or +=), join() method, formatting techniques, etc. We can also concatenate strings with numbers using the str() method.

In [32]:
```
str1 = 'hello'
str2='python'
str1+str2
```

Out[32]:
```
'hellopython'
```

In [33]:
```
str1 = 'hello'
str2='python'
str1/str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 3
      1 str1 = 'hello'
      2 str2='python'
----> 3 str1/str2

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

In [34]:
```
str1 = 'hello'
str2='python'
str1-str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 3
      1 str1 = 'hello'
      2 str2='python'
----> 3 str1-str2

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

In [35]:
```
str1 = 'hello'
str2='python'
str1*str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 3
      1 str1 = 'hello'
      2 str2='python'
----> 3 str1*str2

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

In [37]:
```
str1*5
```

Out[37]: 'hellohellohellohellohello'

In [38]:
```
'python'*3
```

Out[38]: 'pythonpythonpython'

20th-nov

index

- The index() method in Python finds the first occurrence of a specific character or element in a string or list.

In [4]:
```
str1 = 'python'
#    -6  -5   -4  -3   -2    -1
 #   'p'  'y'  't'  'h'  'o'  'n'
#    0    1    2    3    4    5
```

In [5]:
```
str1[-3]
```

Out[5]: 'h'

In [6]:
```
str1[-33]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[6], line 1
----> 1 str1[-33]

IndexError: string index out of range
```

In [8]:
```
str1[0]
str1[1]
str1[2]
str1[3]
str1[4]
str1[5]

# what is common
#str1[]
# what is changing i
#i

#=> str1[i] i is a number i=0 and ends=5
```

Out[8]: 'n'

In [18]:
```
for i in range (6):
    print(str1[i])

# i=0 str1[0]
# i=1 str1[1]
# i=2 str1[2]
#... i=5 str1[5]
```

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

- In for loop there are two option is there
  - **in** means direct access
  - **range** operator required numbers

In [12]:
```
for i in str1: # here direct access
    print(i)
```

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

In [11]:
```
for i in range(6): # here we are giving range/number=6
    print(str1[i])
```

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

In [16]:
```
str1 = 'python  ' # here space that's why we are write len(str1)
for i in range(len(str1)): # here we are giving range
    print(i, str1[i])
```

```
0 p
1 y
2 t
3 h
4 o
5 n
6
7
```

In [18]:
```
str1 = 'python'
for i in range(len(str1)):
    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 [23]:
```
str1 = 'python'
for i in range(len(str1)):
    print(f'The negative index of {str1[i]} is: {i-6}')
```

```
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 [24]:
```
n = len(str1)
str1 = 'python'
for i in range(n):
    print(f'The negative index of {str1[i]} is: {i-n}')
```

```
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 [25]:
```
n = len(str1)
str1 = 'python'
for i in range(n):
    print(f'{i-n} is the negative index, {i} is positive index for a letter {str
```

```
-6 is the negative index, 0 is positive index for a letter p
-5 is the negative index, 1 is positive index for a letter y
-4 is the negative index, 2 is positive index for a letter t
-3 is the negative index, 3 is positive index for a letter h
-2 is the negative index, 4 is positive index for a letter o
-1 is the negative index, 5 is positive index for a letter n
```

Use case of following inside the print statement

- str1[i] for alphabetes like p,y,t,h,o,n
- i for numbers like 0,1,2,3,4,5
- i-n for negative numbers like -6,-5,-4,-3,-2,-1

When to use range - when to use in operator

- any problem if we work with **index use range operator**
- range operator will give **index** as well as **character means alphabets**
- any problem if index is not useful, only character is useful then go with **in**
- with **in** operator we can not get **index or number**

In [33]:
```
# QUE:-1
count=0
str1 = 'ola ola ola ola'
for i in range(len(str1)): # here i = 1,2,3,4.....
    if str1[i] == 'a':
        count=count+1
print(count)
```

```
4
```

In [34]:
```
count = 0
for i in str1:
    if i == 'a': # here i = ola ola... means alphabets
        count = count+1
print(count)
```

```
4
```

In [41]:
```
count = 0
str1='ola ola ola ola'
for i in str1:
    if i =='a':
        count=count+1
print(f'the total "a" are:- {count}')
```

```
the total "a" are:- 4
```

In [42]:
```
count = 0
str1='ola ola ola ola'
for i in range(len(str1)):
    if str1[i] =='a':
        count=count+1
print(f'the total "a" are:- {count}')
```

```
the total "a" are:- 4
```

In [41]:
```
# QUE:-2

str1 = 'ola ola oaal ola'
for i in range (len(str1)):
    if str1[i] == 'a':
        print(i,end=' ')
```

```
2 6 9 10 15
```

In [45]:
```
str1='oal oala oalaa'
for i in range(len(str1)):
    if str1[i] == 'a':
        print(f'{str1[i]} has index at {i}')
```

```
a has index at 1
a has index at 5
a has index at 7
a has index at 10
a has index at 12
a has index at 13
```

In [46]:
```
#QUE:-3

summ = 0
str1='ola ola ola lao'
for i in range(len(str1)):
    if str1[i] == 'a':
        summ = summ+i
print(summ)
```

```
31
```

In [47]:
```
summ = 0
str1='olaa laala'
for i in range(len(str1)):
    if str1[i] =='a':
        summ = summ+i
print(f'the total sum of "a" is:- {summ}')
```

```
the total sum of "a" is:- 27
```

- when we need index str1[i]

In [58]:
```
# QUe:-4
str1 = 'hello how are you python'
count = 0
for i in str1:
    if i in 'aeiou': # in means direct access
        print(i)
        count = count+1
print(f'The vowel count is {count}')
```

```
e
o
o
a
e
o
u
o
The vowel count is 8
```

In [61]:
```
str1 ='hello how are you'
vowel = 'aeiou'
count = 0
for i in str1:
    if i in vowel:
        print(i)
        count+=1
print(count)
```

```
e
o
o
a
e
o
u
7
```

In [49]:
```
str1 ='hello how are you'
vowel = 'aeiou'

for i in str1:
    print(i)
```

```
h
e
l
l
o

h
o
w

a
r
e

y
o
u
```

In [52]:
```
count = 0
str1 ='hello how are you'
vowel = 'aeiou'

for i in str1:
    if i in 'aeiou':
        print(i)
        count=count+1
print(f"The number of vowels are: {count}")
```

```
e
o
o
a
e
o
u
The number of vowels are: 7
```

In [53]:
```
count = 0
str1 ='hello how are you'
vowel = 'aeiou'

for i in str1:
    if i in 'aeiou':
        #print(i)
        count=count+1
print(f"The number of vowels are: {count}")
```

```
The number of vowels are: 7
```

In [65]:
```
count = 0
str1 ='hello how are you'
vowel = 'aeiou'
s1=''
for i in str1:
    if i in vowel and i not in s1:
        s1=s1+i
        count=count+1
print(count)
print(s1)
```

```
4
eoau
```

In [68]:
```
s1=''
str1 ='hello how are you'
vowel = 'aeiou'
count=0
for i in str1:
    if i in vowel and i not in s1:
        print(i)
        s1 = s1+i
        count = count+1
print(f"The number of vowels are: {s1}")
print(f"The number of vowels count are: {count}")
```

```
e
o
a
u
The number of vowels are: eoau
The number of vowels count are: 4
```

IMP

- slicing
- Methods string
- functions
- string

21-Nov

mutable-Immutable

- mutable == change
- immutable == No change

In [69]:
```
str1='welcome'

str1[1]='w'

# I want to change 'w' with 'l'
# can we change 'l' with 'L'?
# ans. not possible
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[69], line 3
      1 str1='welcome'
----> 3 str1[1]='w'

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

In [62]:
```
list = [11,22,131,1,245,1,456,785]
list[3] = 300
list
```

Out[62]:
```
[11, 22, 131, 300, 245, 1, 456, 785]
```

- if the elements will change using indexing is called as mutable
- otherwise immutable

Slicing

- slice : piece of the string
- It is similar to for loop start:stop:step

In [2]:
```
str1='hello how are you'
```

In [77]:
```
len(str1)
```

Out[77]: 17

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
```

- when you see normal parenthisis **() means** a function or method
  - ex:- range(), print(),etc.....
-
- When you see square brackets **[] means** you are **accessing "प्रवेश करणे"** elements

Note

- in slicing we are going only front side means (-->)this direction
- we can go digonally also
  - for ex:- 2 to -5 (Valid)
  - for ex:- -16 to 11 (Valid)
  - **not valid:- 11 to -16**
  - **It is not valid but not showing any error**

case-1:str1[start:]

- start = start value only
- last not mentioned means it will go to till last

In [79]:
```
str1[5:]
```

Out[79]: ' how are you'

In [81]:
```
str1[-5:]
```

Out[81]: 'e you'

case-2:- str1[start:stop]

- start = start value

- step is not mentioned means it will go till the end
    - positive direction
- last = stop-1

In [66]:
```
str1[2:14]

# start = 2
# stop = 14-1 = 13
```

Out[66]: 'llo how are '

In [67]:
```
str1[-16:8]
```

Out[67]: 'ello ho'

case-3: str1[start:stop:step]

In [84]:
```
str1[2:15:2]
```

Out[84]: 'lohwaey'

In [86]:
```
str1[2:-15:2]
```

Out[86]: ''

In [4]:
```
str1[2:25]

# in slicing not error come if not possibel then it is show '' this
```

Out[4]: 'llo how are you'

In [5]:
```
str1[25]

# meaning accessing the 25 element that is not available thats why error will co
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[5], line 1
----> 1 str1[25]
      3 # meaning accessing the 25 element that is not available thats why error
will come

IndexError: string index out of range
```

In [68]:
```
str1[:] # start to end
str1[::] # start to end
str1[::-1] # reversed the string
```

Out[68]: 'uoy era woh olleh'

- intialization
- print

- type
- min/max
- len
- reversed/sorted
- index
  - for loop with range
  - for loop with in
- mutable vs immutable
- slicing
- concatination

In [89]:
```
min("hello")
min([10,20,30])
```

Out[89]:
```
10
```

Methods

In [90]:
```
len('')
```

Out[90]:
```
0
```

In [91]:
```
len(' ')
```

Out[91]:
```
1
```

In [ ]:
```
dir(random)
        randint

random.randint()
```

In [94]:
```
name = 'python'
dir(name)
```

Out[94]:
```
['__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']
```

upper

In [6]:
```
name='python'

name.upper()
```

Out[6]: 'PYTHON'

In [8]:
```
name # here the output will not change
```

Out[8]: 'python'

output will not overwrite

- whenever we apply the function
- always apply shift+tab
- then if any arguments are there try to understand those
- in that one argument name is called as **inplace**
- **inplace = True** the result will be overwrite in original variable
- if inplace not available or inplace = False
- then the output will not overwrite
- in the upper case there is no arguments
- so when we apply upper the output will not overwrite

In [3]:
```
name='python'
name_upper = name.upper()
name_upper # now it will change
```

Out[3]: 'PYTHON'

In [11]:
```
print(name) # here the output not overwrite
print(name_upper)
```

```
python
PYTHON
```

In [96]:
```
name.lower()
```

Out[96]: 'python'

- lower and casefold is same work (convert into lower case)
- title means 1st letter of word turns into upper case
- upper means convert all letters into upper case
- capitalize means convert 1st word 1st letter into upper case

In [4]:
```
name.casefold()
```

Out[4]: 'python'

In [2]:
```
'Python'.casefold()
```

Out[2]: 'python'

In [15]:
```
'PYTHON'.casefold()
```

Out[15]: 'python'

In [16]:
```
'python'.casefold()
```

Out[16]: 'python'

In [102…
```
name.center(10)
```

Out[102… ' python '

In [103…
```
name.capitalize()
```

Out[103… 'Python'

In [17]:
```
'hello how are you'.capitalize()
```

Out[17]: 'Hello how are you'

In [106…
```
n1 = 'python how are you'
n1.title()
```

Out[106… 'Python How Are You'

In [109…
```
name = 'prashant'
name.upper()
```

Out[109… 'PRASHANT'

In [110…
```
name
```

Out[110… 'prashant'

upper dosent have inplace

In [111…
```
name_upper = name.upper()
name_upper
```

Out[111… 'PRASHANT'

In [12]:
```
PYTHON.lower() # error
lower("PYTHON") # error
# Why error

min("Python") # ans
max("Python") # ans
# Why answer
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 PYTHON.lower() # error
      2 lower("PYTHON") # error
      3 # Why error

NameError: name 'PYTHON' is not defined
```

Methods vs inbuilt functions

- inbuilt functions applicable for all python data types and it is generic
- ex: print,type,min,max....
- access: inbuilt_function()
  - for ex:- max("PYTHON")
  - for ex:- max(str1)

  .....VS.....
- method is a specific use for different data types
- access:package.method_name()
  - for ex:- str1.upper()
- access: element.method_name()
  - for ex:- 'Python'.upper()

In [112…
```
str1 = 'hello how are you'
str1.capitalize()
```

Out[112… 'Hello how are you'

In [113…
```
str1.title()
```

Out[113… 'Hello How Are You'

In [114…
```
'welcome'.title()
```

Out[114… 'Welcome'

In [115…
```
'python'.center(20,'*')
```

Out[115… '*******python*******'

replace

In [19]:
```
str1='welcome'
# replace 'l' with 'L'
str1.replace('l','L') # by default
```

Out[19]: 'weLcome'

In [25]:
```
str1='wellllcome'
str1.replace('l','L') # by default
```

Out[25]: 'weLLLLcome'

In [22]:
```
str1='wellllcome'
str1.replace('l','L',1)
```

Out[22]: 'weLllllcome'

In [6]:
```
'welllcome'.replace('l','L')
```

Out[6]: 'weLLLcome'

In [119…
```
'welllcome'.replace('l','L',1)
```

Out[119… 'weLlllcome'

In [120…
```
'welllcome'.replace('l','L',2)
```

Out[120… 'weLLlcome'

In [121…
```
'wellcolome'.replace('l','L',3)
```

Out[121… 'weLLcoLome'

In [122…
```
str1.replace('z','Z')
```

Out[122… 'welcome'

- **in above code no error will come**
- After entering to shift + tab

- In all doc-string no error written that's why using replace we dont get error

Center

In [2]:
```
str1='python'
str1.center(12)
```

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

In [3]:
```
str1.center(20,'+')
```

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

In [5]:
```
str1.center(10,'*')
```

Out[5]:
```
'**python**'
```

23-Nov

- upper
- lower
- capitalize
- title
- casefold
- center
- replace

replace

In [7]:
```
str1 = 'hello'
str1.replace('l','L') # count=-1
```

Out[7]:
```
'heLLo'
```

In [8]:
```
str1.replace('l','L',1)
```

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

How to replace the only second letter

In [14]:
```
str1 = 'restart'
# ans: 'resta$t'

s1 = str1[:5]
s2 = str1[5:]
s3 = s2.replace('r','$')
```

```
new = s1+s3
print(new)
```

```
resta$t
```

- till first 'r' is the one string
- after first 'r' is the another string

In [17]:
```
str1='restart'
s1 = str1[0] #indexing
s2 = str1[1:] # slicing
s3 = s2.replace('r','$') # method replace
s1+s3 # concatenation
```

Out[17]: 'resta$t'

In [22]:
```
str1='restart'

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

Out[22]: 'resta$t'

In [25]:
```
str1='restart'
str1[::-1].replace('r','$',1)[::-1]
```

Out[25]: 'resta$t'

Drowback in above code

- assume str1 = 'restart restart'
- ans question is apply on third letter then can't work using above reverse method

In [26]:
```
#que:-
str1 = 'restart restart'
# ans:- 'restart $estart'

# I want to replace the third 'r'
# third 'r' means we have already 'r's
# we need to slice the sentence
# till second 'r' : one string
# after second 'r' : one string
```

- in above code the drawback is we need to find the index of third 'r'
- we are manually counting
- QUe:
  - is there any method to give automatically index of letter

index

In [28]:
```
str1= 'hello hai how are you'

str1.index('h')

# when we passes 'h' then it will take lowest index of the string

# thats why output is 0
```

Out[28]:
```
0
```

In [30]:
```
str1.index('w'),str1.index('o')
```

Out[30]:
```
(12, 4)
```

This is very very important

- hello hai
- 012345678
- str1 = 'hello hai'
- str1.index('h',0+1)

In [ ]:
```
# hello hai
# 012345678

str1 = 'hello hai'
str1.index('h',0+1) # here previous ans + 1 so the next h index will check
```

In [26]:
```
str1 = 'welho hai how are you'
str1.index('h',3+1)
```

Out[26]:
```
6
```

In [44]:
```
str1 = 'hai hai hai hai'
i1 = str1.index('h')
i2 = str1.index('h',i1+1) # we are searhing for 'h' from i1+1(0+1 = 1) index onw
i3 = str1.index('h',i2+1) # we are searhing for 'h' from i2+1(4+1 = 5) index onw
i4 = str1.index('h',i3+1) # we are searhing for 'h' from i1+1(8+1 = 9) index onw
i1,i2,i3,i4
```

Out[44]:
```
(0, 4, 8, 12)
```

In [43]:
```
str1 = 'hai hai hai hai'
i1 = str1.index('h')
i2 = str1.index('h',i1+1)
i3 = str1.index('h',i2+1)
i4 = str1.index('h',i3+1)
i5 = str1.index('h',i4+1)
i1,i2,i3,i4,i5

# there is no 'h' present in string thats why error will occur
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[43], line 6
      4 i3 = str1.index('h',i2+1)
      5 i4 = str1.index('h',i3+1)
----> 6 i5 = str1.index('h',i4+1)
      7 i1,i2,i3,i4,i5
      9 # there is no 'h' present in string thats why error will occur

ValueError: substring not found
```

- in above code the error comes
- so we will know that another 'h' not present in the string

In [46]:
```
str1.index('h',str1.index('h',str1.index('h')+1)+1)
```

Out[46]: 10

- in above code we are giving substing and start
- we can give end also

In [48]:
```
str1 = 'hai hai hai hai'
str1.index('a',2,8)

# it's means is str1[2:8] searcing form 2 to 8
```

Out[48]: 5

In [49]:
```
str1 = 'hai hai hai hai'
str1.index('j',2,8)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[49], line 2
      1 str1 = 'hai hai hai hai'
----> 2 str1.index('j',2,8)

ValueError: substring not found
```

In [56]:
```
str1 = 'restart restart'
i1 = str1.index('r')
i2 = str1.index('r',i1+1)
s1 = str1[:i2+1]
s2 = str1[i2+1:]
s3 = s2.replace('r','$',1)
s1+s3
```

Out[56]: 'restart $estart'

In [65]:
```
str1 = 'restart restart'
i1 = str1.index('r')
i2 = str1.index('r',i1+1)
s1 = str1[:i2+1] # [starting: 2nd_r_index+1]
s2 = str1[i2+1:] # [2nd_r_index+1: last]
```

```
s3 = s2.replace('r','$',1)
s1+s3
```

Out[65]: 'restart $estart'

find

- just replace index with find

In [20]:
```
str1 = 'hai hai hai hai'
i1 = str1.find('h')
i2 = str1.find('h',i1+1)
i3 = str1.find('h',i2+1)
i4 = str1.find('h',i3+1)
i5 = str1.find('h',i4+1)
i1,i2,i3,i4,i5
```

Out[20]: (0, 4, 8, 12, -1)

In [69]:
```
str1 = 'hello how are you'
str1.find('z',3)
```

Out[69]: -1

IMP QUE

- What is the difference between index and find
  - ans: in index method error will come when we are passing wrong letter that was not present in given string
  - but in find there is no error comes
  - **otherwise both work is same**
    - both are finding the index of a letter
- index and find both are working for same to find the index of a letter
- if any substring not found index will **throw error**
- but find method will **give -1**

count

In [73]:
```
count = 0
str1 = 'hai hai hai hai'
for i in str1:
    if i == 'a':
        count = count+1

print(count)

# it is important for interview purpose
```

4

In [74]:
```
str1 = 'hai hai hai hai'
str1.count('a')

# here we directly using method for counting
```

Out[74]: 4

In [75]:
```
str1.count('a',2,7)

# count the number of 'a' between 2 to (7-1) = 6 index
```

Out[75]: 1

In [76]:
```
str1 = 'hai hai hai hai'
str1.count('ha')
```

Out[76]: 4

In [77]:
```
str1 = 'hai hai hai hai'
str1.count('hai')
```

Out[77]: 4

In [78]:
```
str1 = 'hai hai hai hai'
str1.count('hai hai')
```

Out[78]: 2

In [79]:
```
str1 = 'hai hai hai hai'
str1.count('hai hai ')
```

Out[79]: 1

In [81]:
```
str1 = 'hai hai hai hai'
str1.count(str1)

# it will ask in interview also for confusing
```

Out[81]: 1

In [ ]:
```
str = 'hello pyhton how are you I want to build a one project so you can help me
str.count('o')
# when we press(Shift+Tab)
# Only Sub-string we are use

######################################################################

str.count('o',5)
# when we press(Shift+Tab)
# now Sub-string and Start we are use

#########################################################################

str.count('o',5,12)
# when we press(Shift+Tab)
# now Sub-string and Start and end we are use
```

In [83]:
```
str1 = 'hai hai hai'
str1.count('hai')
```

Out[83]: 3

- QUE:- ans should come like above => 3
- by hand written code

In [85]:
```
str1 ='hai hai hai'
count = 0
for i in str1:
    if i =='hai':
        count = count+1

print(count)

# Here the answer not coming so we want to ans 3
# then we can use
# sliding windows Approach
```

0

Sliding windows approach

In [87]:
```
str1 ='hai hai hai'
count = 0
for i in range(len(str1)):
    if str1[i:i+3] =='hai':
        count = count+1

print(count)

# step-1: i=0 if str1[0:3] : 'hai'=='hai' True
# step-2: i=1 if str1[1:4] : 'ai '=='hai' False
# step-3: i=2 if str1[2:5] : 'i h'=='hai' Flase
# step-4: i=3 if str1[3:6] : ' ha'=='hai' false
# step-5: i=4 if str1[4:7] : 'hai'=='hai' True
```

3

In [1]:
```
str1 = 'hai hai hai hai ai aih'
count=0
for i in range(len(str1)):
    if str1[i:i+2] == 'ai':
        count=count+1

print(count)

# [i:i+2]
# [0:0+2] = 0:1 'ha' == 'ai' False
# [1:1+2] = 1:2 'ai' == 'ai' True count = 1
# [2:2+2] = 2:3 'i ' == 'ai' False
# [3:3+2] = 3:4 ' h' == 'ai' False
# [4:4+2] = 4:5 'ha' == 'ai' False
# [5:5+2] = 2:6 'ai' == 'ai' True  count=2
```

```
6
```

Important Interview Quetion

In [13]:
```
str1 = 'virat.kohli@rcb.com'
str2 = 'rohit.sharma@mi.com'
str3 = 'ms.dhoni@chennai.com'

first_dot = str1.index('.')
f_name = str1[0:first_dot]
symbol_at = str1.index('@')
s_name = str1[first_dot+1:symbol_at]
s_name
second_dot = str1.index('.',first_dot+1)
c_name = str1[symbol_at+1:second_dot]
c_name

print(f_name,s_name,c_name)
```

```
virat kohli rcb
```

In [23]:
```
str3 = 'ms.dhoni@chennai.com'
def extraction(str1):
    first_dot = str1.index('.')
    f_name = str1[:first_dot]
    symbol_at = str1.index('@')
    s_name = str1[first_dot+1:symbol_at]
    s_name
    second_dot = str1.index('.',first_dot+1)
    c_name = str1[symbol_at+1:second_dot]
    c_name

    print(f_name,s_name,c_name)
extraction('ms.dhoni@chennai.com')
```

```
ms dhoni chennai
```

In [24]:
```
def extraction(str1):
    first_dot = str1.index('.')
    f_name = str1[:first_dot]
    symbol_at = str1.index('@')
    s_name = str1[first_dot+1:symbol_at]
    s_name
    second_dot = str1.index('.',first_dot+1)
```

```
    c_name = str1[symbol_at+1:second_dot]
    c_name

    print(f_name,s_name,c_name)
extraction('rohit.sharma@mi.com')
```

```
rohit sharma mi
```

In [1]:
```
def xyz(str1):
    first_dot = str1.index('.')
    s_dot = str1.index('.',first_dot+1)
    at_symbol = str1.index('@')
    f_name = str1[:first_dot]
    s_name = str1[first_dot+1:at_symbol]
    c_name = str1[at_symbol+1:s_dot]
    print(f_name,s_name,c_name)

xyz('Hello.good morning@mr.com')
```

```
Hello good morning mr
```

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

The Above methods are very very Important other are normal so focus on above only

In [27]:
```
s1 = ''
```

In [28]:
```
dir(s1)
```

```
Out[28]: ['__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 [3]:
```
str1 = 'pythpn'
str1.isalnum()
str1.isalpha()
```

Out[3]:
```
True
```

lstrip-strip-rstrip

- The strip() method removes any leading, and trailing whitespaces.
- strip is use for remove space or any unwanted character
- string with leading whitespace removed.
- remove characters in chars instead.

- lstrip : we can remove some letters from left side of string
- rstrip : we can remove some letters from right side of string
- strip : we can remove some letters from both side of string

In [8]:
```
s1 = ' hai how are you'

s2 = 'hai how are you '

s3 = ' hai how are you '
```

In [19]:
```
s1.lstrip(), s1.rstrip(),s1.strip()
# by default removes space
# remove   , not remove,  remove
```

Out[19]:
```
('hai how are you', ' hai how are you', 'hai how are you')
```

In [9]:
```
s2.rsplit(),s2.lstrip(),s2.strip()

# remove, not removed, remove
```

Out[9]:
```
(['hai', 'how', 'are', 'you'], 'hai how are you ', 'hai how are you')
```

In [10]:
```
s3.strip()
# remove
```

Out[10]:
```
'hai how are you'
```

In [7]:
```
s4 = '****how are you'
s4.lstrip('*')

# Here the (*) character will be removed
```

Out[7]:
```
'how are you'
```

In [16]:
```
s4 = 'how are you'
s4.lstrip('h')

# Here the (h) character will be removed
```

Out[16]:
```
'ow are you'
```

In [18]:
```
s4 = 'how are yoh'
s4.strip('h')

# Here the (*) character will be removed
```

Out[18]:
```
'ow are yo'
```

startswith-endswith

In [36]:
```
str1 = 'hai how are you'
str1.startswith('hai how are you')
```

Out[36]:
```
True
```

In [37]:
```
str1 = 'hai how are you'
str1.startswith(str1)
```

Out[37]:
```
True
```

In [38]:
```
str1.endswith(str1)
```

Out[38]: True

In [41]:
```
str1.endswith('hai how are')
```

Out[41]: False

split विभाजन

- The split() method is the most common way to split a string into a list in Python.

In [12]:
```
str1 = 'hello how are you'
str1.split() #by default
```

Out[12]: ['hello', 'how', 'are', 'you']

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

Out[13]: ['', 'ello ', 'ow are you']

- In split what character you will provided is really important.
- When set to None (the default value), will split on any whitespace
- Splitting starts at the front of the string and works to the end.

Most ask interview questions

- in a given sentence find the most repeated word
- in a given sentence find the min and max len of word
- In a given sentence most occured second max length of word
- sliding window
- strings ===== list vise versa
- hai how are you: Hai How Are You (without methods)

- String is very imp the interviewer will ask 1st on string, List Dictionary

In [ ]:
```
```