In [ ]:

statements: Vaishnavi -

Try-Exception

- Functions: Banishree
 - Functions with out arguments - Functions with argumenrs =======4 Ravi===========
 - Default paramters
 - return
 - global vs local
 - functions in functiom

- For loop:sohel

- While loop:suman

==========
Offline================
==== 1)Vaishanvi:
if-else
2) Banishree:
Functions(2)
3) Ravi: Functions(4)
4) Sohel: For loop
5) Suman: while loop

========== Online
================== 1)
Aysuh: if-else
2) Reshma patil:
Functions(2)
3) Sathwik:
Functions(4)
4) Raghusai: For loop
5) Sarita: while loop

In [1]: In [2]:

open jupyter notebook
execute the code at home
explain the same

Python Part-1

- Basic python codes

- Packages

- Conditional

```
name='python'

# strings are in red colour: quotes # variable are in black colour # keywords are in green colour

type(name)
```

Out[2]: str

```
In [3]:    on"
name1="pyth type(name1)
```

Out[3]: str

In [6]:
```
print(name2)
# entire string in single quotes
# python word is in double quotes
```

In [7]:
```
i like "python"
```

In [11]:
```
name3="i like 'python'"
print(name3)
```

```
i like 'python'
```

In [12]:

❓❓❓❓❓❓❓❓❓❓❓❓ – ❓❓❓❓❓❓❓❓❓❓❓❓

triple quotes meaning is doc string
if you want to convey the information to user
In jupyter notebook: markdown
but in another platforms like Vscode, pycharm dont have markdown at that place to convey the information we use triple quotes that is called docstring

In [13]:
```
name4="""python"""
```

In [ ]:
```
print(name) # careful with print statement
```

```
python
```

```
import random
random.randint()
```

type
max
min
len
reversed
sorted

```
print(name1) # careful with print statement
```

```
python
```

```
name2='i like "python"'
```

In [14]:
```
name='python'
max(name)
```

```
keyword(<variable_name>) #
print(name)
# type(name)
#
```

Out[14]: 'y'

❓❓❓❓❓❓❓❓❓❓

American Standard Code Information Interchange
A: 65
a: 97

❓❓❓❓❓❓-❓❓h❓❓

In [19]:
```
ord('p'),ord('y'),ord('t')
,ord('h'),ord('n')
```

```
Out[19]: (112, 121, 116, 104, 110)
           max('python
           ')
In [20]:

Out[20]: 'y'
           min('python
           ')
In [21]:

Out[21]: 'h'
            chr(112),ord(
            'p')
In [24]:

Out[24]: ('p', 112)
           00'
           max(str1)
In [27]:
str1='123-1

Out[27]: '3'
            ord('-'),ord
            ('3')
In [29]:

Out[29]: (45, 51)
In [35]: In [36]:
```

```
&,39 = ',40 = (,41 = ),42 = *,43 = +,44 =
,,45 = -,46 = .,47 = /,48 = 0,49 = 1,50 =
2,51 = 3,52 = 4,53 = 5,54 = 6,55 = 7,56 =
8,57 = 9,58 = :,59 = ;,60 = <,61 = =,62 =
>,63 = ?,64 = @, 65 = A,66 = B,67 = C,68
= D,69 = E,70 = F,71 = G,72 = H,73 = I,74
= J,75 = K,76 = L,77 = M,78 = N,79 = O,80
= P,81 = Q,82 = R,83 = S,84 = T,85 = U,86
= V,87 = W,88 = X,89 = Y,90 = Z,91 = [,92
= \,93 = ],94 = ^,95 = _,96 = `, 97 =
a,98 = b,99 = c,100 = d,101 = e,102 =
f,103 = g,104 = h,105 = i,106 = j,107 =
k,108 = l,109 = m,110 = n,111 = o,112 =
p,113 = q,114 = r,115 = s, 116 = t,117 =
u,118 = v,119 = w,120 = x,121 = y,122 =
z,123 = {,124 = |,12 5 = },126 = ~,127 =
,
```

```
# How many ascii values are existing
for i in range(33,128):
 print(i,'=',chr(i),end=',')
```

```
33 = !,34 = ",35 = #,36 = $,37 = %,38 =
```

```
'Banana'>'banana' # 'B': 66 'b':99 66>99
```

```
Out[36]: False
                    'Banana'>'BAnana' #
                    a='97' 'A':65
In [37]:

Out[37]: True
```

```
        ??

        ??

        ??
```

```
In [39]:        python'
str1='i like    len(str1)
```

Out[39]: 13

```
str2='' # empty
string len(str2)
```

In [45]:

Out[45]: 0

```
str3=' ' # it has
one space len(str3)
```

In [46]:

Out[46]: 1

```
inherently> # we use the
for loop to get the
output
```

In [47]:

```
reversed('python')

# <output is stored
```

Out[47]: <reversed at 0x19be6fc48b0>

```
'y' in 'python'
't' in 'python'
'h' in 'python'
'o' in 'python'
'n' in 'python'

for i in 'python':
 print(i)
```

In [52]:

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

```
for i in
reversed('python')
:  print(i)
```

In [54]: In [53]:

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

```
'p' in 'python'    reversed('python')
```

Out[53]: <reversed at 0x19be6fc5360>

In [56]:

p
p
l
e

```
In [ ]: In [57]:
```

```
sorted('python')
# by default
reverse=False: acending
order # sorted keyword
sort the letters #
[104,110,111,112,116,121]
```

```
s1='apple'
for i in s1:
 print(i)
```

```
a
```

```
Out[57]: ['h', 'n', 'o', 'p', 't', 'y']
```

```
sorted('python',rev
erse=True)
```

```
In [61]:
```

```
Out[61]: ['y', 't', 'p', 'o', 'n', 'h']
In [60]: In [
```

y 121
t 116
h 104
o 111
n 110

- type

- max

```
]:
```

- min

- ord

- chr

- len

- reversed

- in

- sorted

```
str1='hello'
str2='python'

#
Concatenation
str1+' '+str2
```

```
In [65]:
for i in
'python':
print(i,ord(i)
)
```

```
'helo'+'python
'
```

```
p 112
```

```
Out[65]: 'helopython'
```

```
In [66]:
str1-str2
```

```
---------------------------------------
-------------------------------- -
TypeError Traceback (most recent call las
t)
Cell In[66], line 1
----> 1 str1-str2

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

```
In [67]: In [68]:
```

```
str1*str2
# str and str
# non int type and str
```

```
---------------------------------------
-------------------------------- -
TypeError Traceback (most recent call las
t)
Cell In[67], line 1
----> 1 str1*str2

TypeError: can't multiply sequence by


non-int of type 'str' 3*str2
```

```
Out[68]: 'pythonpythonpython'
```

```
In [69]: In [ ]:
```

```
str1/str2
```

```
---------------------------------------
-------------------------------- -
TypeError Traceback (most recent call las
t)
Cell In[69], line 1
----> 1 str1/str2

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

- type

```
In [71]:          #p y t h o n #0 1
str1='python'     2 3 4 5

#-6 -5 -4 -3 -2 -1str1

Out[71]: 'python'
```

- max
- min
- ord
- chr
- len
- reversed
- in
- sorted
- concatenation

In python index start with zero

In [81]:

```
t
h
o
n
```

```
str1='python'
# I want to
iterate this  # in
: i means direct
letter # range : i
means number
```

```
str1='python'
for i in str1:
 print(i,end=' ')
```

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

In [82]: In [83]:

```
str1='python'
for i in range(6):
```

```
print(str1[i],end=
```

```
' ') p y t h o n
```

In [87]: In [89]:

```
str1='python'
for i in str1:
 print(i,end=' ')
```

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

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

```
p
y
```

```
str1='python'
for i in range(6):
```

```
print(str1[i],end=
```

```
' ') p y t h o n p
```

```
y t h o n
```

[illegible]

In [ ]: In [4]:

In [8]:

```
in=====i== > direct char
range === i number


string1='python'
n=len(string1)
for i in range(n):
 print(i,string1[i])

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


#WAP print the index of a given
string
# str1='python'
# o/p: the index of p is 0
# the index of y is 1

string1='python'
n=len(string1)
for i in range(n):
 #print(i,string1[i])
 print(f"the postive index of
{string1[i]} is {i}")

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

In [14]:

```
# o/p: the negative index of p
is -6
# the negative index of y is -5
# try to print -6 to -1
string1='python'
n=len(string1)
#print(n) # 6 -6 n=== > -n
for i in range(-n,0):
 #print(i,string1[i])
 print(f"the negative index of
{string1[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 [ ]: In [15]:

```
string1='python'
n=len(string1)
for i in range(n): # 0 to
len(string)-1
 print(f"the postive index of
{string1[i]} is {i}")
for i in range(-n,0): #
-len(string) to -1  print(f"the
negative index of {string1[i]}
is {i}")
```

Postive: 0 to len(string1)-1
range(0,len(string1))
start=0 end= last-1 : len(string1)-1
Negative: -len(string1) to -1
range(-len(string1),0)
start= -len(string1), end= last-1 0-1=-1

```
s1="hello how are you"
n=len(s1)
for i in range(n):
 print(f"the postive index of
{s1[i]} is {i}")
```

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

```
the negative index of is -12
the negative index of h is -11
the negative index of o is -10
the negative index of w is -9
the negative index of is -8
the negative index of a is -7
the negative index of r is -6
the negative index of e is -5
the negative index of is -4
the negative index of y is -3
the negative index of o is -2
the negative index of u is -1
```

```
# the postive index is 0 and
negative index is -6 for p # the
postive index is 1 and negative
index is -5 for y s1='python'
n=len(s1) # 6
for i in range(n):
 print(i,i-n)
```

```
0 -6
1 -5
2 -4
3 -3
4 -2
5 -1
```

```
# Task is implemenent the same
# q1) Positive index
# q2) negative index
# q3) posand neg together
# using while loop
# screenshot in whatsapp group
In [23]:
```

```
In [21]: In [ ]:
```

```
for i in range(-n,0): #
-len(string) to -1  print(f"the
negative index of {s1[i]} is
{i}")
```

```
the negative index of h is -17
the negative index of e is -16
the negative index of l is -15
the negative index of l is -14
the negative index of o is -13
```

```
s1='hello how are you i am good'
count=0
for i in s1:
 if i=='a': # a==a
 count=count+1

print("the total number of 'a'

are:",count) the total number of

'a' are: 2
```

```
# WAP find the indexes of 'a'
present in a given string #
s1='hello how are you i am good'
# ans:10,20

# idea: iterate the loop using
range
# automaticall i becomes number
# condition
# print only i

s1='hello how are you i am good'
for i in range(len(s1)):
 if s1[i]=='a':
 print(i)

10
20
```

In [26]: In [ ]:

```
s1='hello how are you i am good'
count=0
for i in s1:
 if i=='a': # a==a
 count=count+1
print("the total number of 'a'
are:",count)
```

```
# WAP find the number of 'a'
present in a given string #
s1='hello how are you i am good'
# ans: 2

# count=0
# Idea: iterat using for loop
# get each letter
# whenever that letter equal to
a
# count= count+1
```

```
s1='hello how are you i am good'
for i in range(len(s1)):
 if s1[i]=='a':
 print(i)
```

In [28]:

```
 if s1[i] in 'aeiou':
 count=count+1
print("the number of vowels are:",count)
```

```
the number of vowels are: 11
```

Postive index
Negative index
Postive and Negative
Number of 'a'
Index of 'a'
Vowels in a given string

```
# In the above problem vowels are
repating
# you need to print only single vowel
s1='abca'
s2=''
for i in s1:
 if i in 'aeiou':
 if i not in s2:
 s2=s2+i

print(len(s2))
```

```
1
```

In [29]: In [31]:

```
#WAP ask the user find the number vowels
in a given string # s1='hello how are
you'

s1='hello how are you i am good'
for i in range(len(s1)):
 if s1[i]=='a' or s1[i]=='e' or
s1[i]=='i' or s1[i]=='o' or s1[i]=='u':
print(s1[i])
```

```
e
o
o
a
e
o
u
i
a
o
o
```

```
s1='hello how are you i am good'
count=0
for i in range(len(s1)):
```

Mutable:Change the element by using index operation
In [ ]: In [35]:

In [37]: In [38]:

```
s1='hello'

# can we change 'l' to 'L'
```

```
#s1='heLlo'
# will you achieve this by using index
operation
```

item assignment

Note

Strings are immutable

```
s1='hello'
s1[2]='L'
```

```
----------------------------------------
--------------------------------- -
TypeError Traceback (most recent call las
t)
Cell In[35], line 2
 1 s1='hello'
----> 2 s1[2]='L'

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

```
l1=[1,2,3] # 2 to 200
l1[1]=200
```

```
l1
```

```
Out[38]: [1, 200, 3]
```

In [ ]: In [1]:

- Postive index
- Negative index
- Postive and Negative - Number of 'a'
- Index of 'a'
- Vowels in a given string - unique vowels

[illegible]:

```
s1='hello how are you'
```

In [ ]:

```
                                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 [11]:                        s1[11:-11:-1] # ans there
-17 -16 -15 -14 -13 -12 -11 -10 -9s1[11:-3:-1] # no anser
-8 -7 -6 -5 -4 -3 -2 -1 h e l l o s1[11:-11:1] # no answer
```

```
s1[11:-3:1] # answer start=11      last=-3-1=-4

Out[11]: 're '
                                   s1[3:14:-2] # NP
                                   s1[3:-14] # satrt=3 last=
In [12]:                           -14-1=-15
-17 -16 -15 -14 -13 -12 -11 -10 -9 s1[3:-14:-1] # start=3 dire=-ve
-8 -7 -6 -5 -4 -3 -2 -1 h e l l o  last = -14+1=-13
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           s1[:]
s1[3:-14:2] # Np start=3 step=2 +  s1[::1]
last= -14-1=-15 s1[3:14:2] # P     s1[::]
s1[-3:-14:-2] # P                  s1[::-1]
s1[-3:14:-2] # Np start=-3 step=-2
-ve stop=14+1=15 s1[-3:-14:2] # NP

Out[12]: ''
           s1[3:-14:-
           1]
In [15]:

Out[15]: ''
                                   range(Start,stop,step)
 In [ ]: In [5]:
                                   s1[2:10] # start=2 dire =+ 1
                                   stop=10-1=9 s1[2:10:2] #
                                   start=2 dire= + 2
 s1[<start>:<stop>] ----           stop=10-1=9 # 2 4 6 8
 range(start,stop)
 s1[start:stop:step] ----

 Out[5]: 'lohw'
                                # dire = -ve 1
                                # last= -3+1=-2
 In [8]: In [9]:                for i in

                                range(11,-3,-1):print(i,

                                end=' ') 11 10 9 8 7 6 5

                                4 3 2 1 0 -1 -2

 s1[11:-3:-1]
 # start=11                     s1[11:-3:-1]

 Out[9]: ''
 In [3]: In [ ]:
                                        - reversed

 for i in
 range(20,2,3):print(i,end=' ')         - sorted

                                        - ord

 - How to read strings
                                        - chr

 - Different ways single-quotes
                                        - in

 doule-quotes triple-quotes -
                                        - in vs range

 type
                                        - index

 - min
                                        - mutable immutable

 - max
                                        - concatenation

 - len
```

??????h??????

- slice

??????????????

```
In [16]: s1='python'
         # we stored 'python' in a varaiable s1
         # s1 act as package
         # every pcakages has some methods
         # dir(package)
         dir(s1) # dir('')
Out[16]: ['__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',
'rfind',
'rindex',
'rjust',
'rpartition',
'rsplit',
'rstrip',
'split',
'splitlines',
'startswith',
'strip',
'swapcase',
'title',
'translate',
'upper',
'zfill']
```

[illegible]

```
s1='python'
# s1 act as package
# Capitalize act as
method #
help(<package>.<meth
od_name>)
#help(s1.capitalize)
s1.capitalize()
```

In [23]:

```
'maketrans',
'partition',
'removeprefix',
'removesuffix',
'replace',
```

Out[23]: 'Python'

In [ ]: In [ ]:

```
len(s1) capitalize(s1)
(wrong) max(s1)
min(s1)

s1.capitalize()
# Keywords belongs to
entire python # methods
are related to only that
data type
```

[illegible]

In [24]:

```
s1='python'
s1.capitalize()
```

```
s1='python'
s1.upper()
```

```
# keywords vs methods
```

Out[24]: 'PYTHON'

```
          '
          s1.lower()
In [26]:
s1='PyThon

Out[26]: 'python'
In [28]:   s1.casefold
s1='PyThon' ()

Out[28]: 'python'
```

????????

??

```
In [29]:                if i=='a':
s1='hai hai hai' #      count=count+1
how many a count=0      count
for i in s1:

Out[29]: 3
           s1.count('a
           ')
In [30]:

Out[30]: 3
                        s1.count('ola ')
                        s1.count(s1) #
In [35]:                s1.count('ola ola ola')
s1='ola ola ola'
s1.count('ola')

Out[35]: 1
                    s1.count('o',2) #
                    'a ola ola'
In [39]:            s1.count('o',2,5) #
s1.count('o') #     'a ol'
'ola ola ola'

Out[39]: 1
            s1.count('o
            a')
In [40]:

Out[40]: 0
                        #'ol a ol a ola' # 012
                        s1.count('o',2,5)
In [43]:

Out[43]: 1
```

????????????????

```
          s1.replace('c','
          C')
In [45]:
s1='wellcome'   s1.replace('l','
# replace 'l'   L')
with 'L'

Out[45]: 'weLLcome'
In [55]:
s1='wellllcome'       # whenever slash
s1.replace('l','L',2) symbol is there #
# shift+tab           dont give the
                      argument name
```

```
Out[55]: 'weLLllcome'
                    # no error
                    s1.count('z') # 0
In [56]:            no error
s1='wellllcome'
s1.replace('z','L')

Out[56]: 'wellllcome'
                              ') # slice replace
                              print(str1+str2) #
In [65]:                      Concatenation
#inuput: 'restart'
#output: 'resta$t'

                              s1[::-1].replace('r','$',1
                              )[::-1]
s1='restart'
str1=s1[0] # Index
str2=s1[1:].replace('r','$    resta$t

Out[65]: 'resta$t'
                  random.randint(a=
                  20,b=30)
In [54]:
import random

Out[54]: 24
```

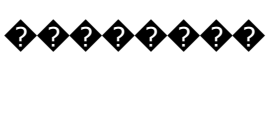
Capitalize/ upper /lower/casefold

count

replace

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

Out[2]: 'weLcome'
```



```
In [3]:      str1.index(
str1='pytho  'y')
n'

Out[3]: 1
In [8]:                                  i3=str2.index('a',6)
# Return the lowest index in S where
substring sub is found str2='hai hai i1,i2,i3
hai'
i1=str2.index('a') # first               # Q1) what is the meaning of index:
postition: 1                             0 h
i2=str2.index('a',2) # second-a:         # Q2) what is the index method will
already 1 a is over,                     do : it will find the postion

Out[8]: (1, 5, 9)


In [ ]: In [ ]:
```

In [ ]: In [9]:

```
# i1=str2.index('a')
# i2=str2.index('a',str2.index('a')+1)
#
i3=str2.index('a',str2.index('a',str2.inde
x('a')+1) +1) #
i4=str2.index('a',str2.index('a',str2.inde
x('a',str2.index('a')+1) +1)+1)


i1=str2.index('z')
```

```
---------------------------------------
------------------------------- -
ValueError Traceback (most recent call las
t)
Cell In[9], line 1
----> 1 i1=str2.index('z')

ValueError: substring not found
```

In [10]:

```
# when I, asking you second 'a'
# we started seeing after first 'a'
# first 'a' index =1
# which index will start to see= 1+1=2


str2='hai hai hai hai'
i1=str2.index('a') # 1
i2=str2.index('a',i1+1) # 5
i3=str2.index('a',i2+1) # 9
i4=str2.index('a',i3+1)

i1,i2,i3
```

```
for i in range(len(str2)):
 if str2[i]=='a':
 print(i)
```

```
1
5
9
```

if substring not found

replace : same string

count : 0

index : error

In [19]:


```
str2='hai hai hai
hai'
i1=str2.find('a')
# 1
i2=str2.find('a',i
1+1) # 5
i3=str2.find('a',i
2+1) # 9
i4=str2.find('a',i
3+1)
i5=str2.find('a',i
4+1)

i1,i2,i3,i4,i5
```

Out[19]: (1, 5, 9, 13, -1)

If substring not found find method will return -1

if substring not found

replace : same string

count : 0

index : error

find : -1

```
In [ ]: In [21]:
```

```
s3='rohit.sharma@mi.com'
s4='lokesh.rahul@lsg.com'
s5='a.b@c'

# idea-1 extrat . index
s1[0:<.index>] # ida-2
extract @ index s1[.:@] #
idea=3 extract second .
index s1[@:.]


s3="a.b@c.com"
dot=s3.find('.')
atrate=s3.find('@')
sec_dot=s3.find('.',dot+1
)
print(s3[:dot])
print(s3[dot+1:atrate])
print(s3[atrate+1:sec_dot
])

a
b
c
```

```
s1='omkar.nallagoni@cogni
zant.com' # first
name='omkar'
# sur name='nallagoni'
# cname='cognizant'

s2='virat.kohli@rcb.com'
```

```
In [32]:
s1='virat.kohli@cog
nizant.com' # 5
should come
automatically # 5
is an index of '.'
i1=s1.index('.')
```

```
i2=s1.index('@')
i3=s1.index('.',i1+
1)
fname=s1[:i1]
sname=s1[i1+1:i2]
cname=s1[i2+1:i3]
fname,sname,cname
```

```
Out[32]: ('virat', 'kohli', 'cognizant')
```

```
In [33]: In [38]:
```

```
str3='python '

# str1 has space in both
sides
str1.strip(),str1.lstrip
(),str1.rstrip()
```

```
str1=' python '
str2=' python'
```

```
Out[38]: ('python', 'python ', ' python')
```

```
(),str2.lstrip() #strip
is boss
```

```
In [42]:
str2.strip(),str2.rstrip
```

```
Out[42]: ('python', ' python', 'python')
```

```
n@@@@'
str2.strip('@')
```

```
In [43]:
str2='@@@@pytho
```

```
Out[43]: 'python'
```

```
str1.strip() # nothing
mentioned empty space
```

```
In [44]:
str1=' python '
```

```
Out[44]: 'python'
                    $$$'
                    str2.lstrip('@').r
In [46]:            strip('$')
str2='@@@@python$$

Out[46]: 'python'
               str2.strip('
               @$')
In [47]:

Out[47]: 'python'
                         how are you' #op
                         'hello how are you'
 In [ ]:
 str1='hello /r/r/r/n/


          ℎ-ℎ
In [52]:       s1.startswith(
s1='hai how    'hai')
are you'

Out[52]: True
              s1.startswith
              (s1)
In [53]:

Out[53]: True
                    s1.endswith('y
                    ou')
In [58]:
s1='hai how         s1.endswith(s1
are you'            )

Out[58]: True

          ??

          ??

          ??

          ??

          ??


                         will be in list  s1='hai
In [59]:                 how are you'
# It will split the      s1.split()
words
# and the words output

Out[59]: ['hai', 'how', 'are', 'you']
                    how,are you'
                    s2.split(',')
In [60]:
s2='hai

Out[60]: ['hai how', 'are you']
                    are you'
                    s3.split('a')
In [61]:
s3='hai how
```

```
                you
# h, i how ,re
```

```
Out[61]: ['h', 'i how ', 're you']
```

capitalize/upper/lower/casefold
replace
count
index/find
rstrip/strip/lstrip
startwith/endswith
split

```
In [62]: dir('')
Out[62]: ['__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',
 'removeprefi
 x',
 'removesuffi
 x',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition'
 , 'rsplit',
 'rstrip',
 'split',
 'splitlines'
 ,
 'startswith'
 , 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

In [ ]: In

```
'isalnum',
'isalpha',
'isascii',
'isdecimal',
'isdigit',
'isidentifie
r',
'islower',
'isnumeric',
'isprintable
',
'isspace',
'istitle',
'isupper',
```

[66]:

```
s1='123'
s1.isalpha()
```

```
'maketrans',
'partition',
```

Out[66]: False

In [ ]:

In [ ]:

In [ ]: