INDEXUNG

- INDEXING MEANS ACCESSING individual characters (elements) of a string by its position within the string.

- NOTE: Positive indexing starts with 0
- LEFT T0 RIGHT

- NOTE: Negative indexing starts with -1
- RIGHT TO LEFT

```
example = "PYTHON IS EASY "
print(example[10])
```

```
E
```

```
x = "my name is rohit"
print(x[13])
```

```
h
```

```
# negative indexing
```

```
example[-12]
```

```
't'
```

```
example[-2]
```

```
's'
```

```
x[-5]
```

```
'r'
```

```
example = "rohit"
print(example[3])
```

```
i
```

```
number="12345678"
print (number[-5])
```

```
4
```

```
x = "hello googal"
print(x[4])
```

```
o
```

slicing

- slicing means accessing a subset of a string based on its indices.
- positive slicing
- negative slicing
- sreing[starts index: stop index]

```
example ="PYTHON IS EASY"
example[0:6]
```

```
'PYTHON'
```

```
example = "123456789"
example[1:6]
```

```
'23456'
```

```
example = "PYTHON IS EASY "
example[10:15]
```

```
'EASY '
```

```
example[0:200]
```

```
'PYTHON IS EASY'
```

```
A = "hello sir"
A[0:5]
```

```
'hello'
```

```
example[ : ]
```

```
'PYTHON IS EASY'
```

```
example[7:]
```

```
'IS EASY'
```

Negative slicing .

```
example[-1:-10] # python is moving forward direction.output is ''
because always in neg slicing first argument shut bhi small
```

```
''
```

```
example[-6:]
```

```
'S EASY'
```

```
example[-5:]
```

```
' EASY'
```

```
example[-15:-9]
```

```
'PYTHON'
```

```
example = "python is easy"
```

```
example[-15:15]
```

```
'python is easy'
```

```
example[8:-15]
```

```
''
```

```
example[-7:9]
```

```
'is'
```

```
example=" rohit "
example[-6:-1]
```

```
'rohit'
```

```
example[:]
```

```
'PYTHON IS EASY '
```

```
example = "my name is mohit "
example[0:8]
```

```
'my name '
```

```
example[-9:-7]
```

```
'is'
```

```
name = "python"
name[:]
```

```
'python'
```

```
name = "mohit"
name[0:3]
```

```
'moh'
```

```
print('gaurav\rrohit')
```

```
gaurav rohit
```

Espace sequence

```
print( '\'rohit\'') # single code \'
```

```
'rohit'
```

```
print('rohitt\b') # backspace \b
```

```
rohitt
```

```
print('\\rishi\\') # backslash \\
```

```
\rishi\
```

```
print('my name\nmohit') # new line corrector \n
```

```
my name
mohit
```

```
print('name\tage\thobby') # tab (gives you 4 step gap)
```

```
name  age   hobby
```

```
print('how are you today\r where do you live') # carriage return (correction) \r
```

```
how are you today  where do you live
```

```
print('gaurav\rrohit') # \r
```

```
gaurav rohit
```

```
print('name\t\teniglish\tmaths\t\tscience')
print('rohit\t\t89\t\t90\t\t80')
print('mohit\t\t99\t\t99\t\t99')
```

```
name        eniglish    maths       science
rohit       89          90          80
mohit       99          99          99
```

```
print('Name\t|\tEnglish\t|\tMaths\t|\tscience')
print('_____________________________________________________')
print('Mohit\t|\t99\t|\t90\t|\t80')
```

```
print('Rohit\t|\t99\t|\t97\t|\t99')
print('Rahul\t|\t89\t|\t90\t|\t98')
```

```
Name  |     English    |     Maths |     science
_____________________________________________________
Mohit |     99    |     90    |     80
Rohit |     99    |     97    |     99
Rahul |     89    |     90    |     98
```

```
print('subject\t:-\trohit\t-\tmohit\t-\trahul')
print('-------------------------------------------------------')
print('english\t:-\t65%\t-\t70%\t-\t90%')
```

```
subject    :-    rohit -     mohit -     rahul
-------------------------------------------------------
english    :-    65%   -     70%   -     90%
```

```
Name = input('\'What is your name\' :- ')

Age = input('\'what is your age\' :- ')

Hobby = input('\'what is your hobby\' :- ')


print('\n')
print('Name - ',Name)
print('Age - ',Age)
print('Hobby - ',Hobby)
```

```
'What is your name' :- Rohit
'what is your age' :- 26
'what is your hobby' :- Snooker


Name -  Rohit
Age -  26
Hobby -  Snooker
```

slicing

```
example = " iam learning python "
example[-100:-1]
```

```
' iam learning python'
```

```
example = " iam learning python "
example[9:-11]
```

```
'n'
```

```
example = " i am learning python "# negative and positive slicing exm
example[-21:21]
```

```
'i am learning python'
```

stride [step]

- Steing silcing can also accept a third parameter ,the stride ,which refers to how many steps you want to take from the first charactor ofthe stride.
- striding alway starts with 1

```
example = "python is easy" # positive stride step
example[0:15:1]
```

```
'python is easy'
```

```
example = "python is easy"
example[0:15:2]
```

```
'pto ses'
```

```
example = "python is easy"
example[0:15:3]
```

```
'ph  s'
```

```
example = "python is easy"
example[2:13:1]
```

```
'thon is eas'
```

```
example = "python is easy"
example[:15:4]
```

```
'poss'
```

```
example = "python is easy"
example[0::6]
```

```
'p s'
```

```
example = "python is easy"
example[0::1]
```

```
'python is easy'
```

```
example[-9::-1] # Negative stride
```

```
'nohtyp'
```

```
example[-6:-8:-1]
```

```
'si'
```

```
example[-1:-5:-1]
```

```
'ysae'
```

```
example[-1:-8:-2]
```

```
'ya i'
```

```
example[::-2]
```

```
'YA INHY'
```

```
example[::-5]
```

```
'YSH'
```

Strings methods and function

- concatination of string

```
# .upper() is a methods.
x = x.upper()
print(x.upper())
```

```
HELLO WORD
```

```
x ="hello word "
```

```
x.upper()
```

```
'HELLO WORD '
```

```
x.upper()
```

```
'HELLO WORD'
```

```
Z = "rohit"
Z.upper()
```

```
'ROHIT'
```

```
x.lower() #lower()
```

```
'hello word'
```

```
x.lower()
```

```
'hello word '
```

```
name = "RoHiT"
```

```
print(name.lower())
print(name)
```

```
rohit
RoHiT
```

```
z = " hello world"
print(z.upper())
print(z)
```

```
 HELLO WORLD
 hello world
```

.replace()

```
x = "hello world"
x =x.replace("l","Z")
x
```

```
'heZZo word'
```

```
x = "python is easy"
x.replace("y","t")
```

```
'ptthon is east'
```

```
x = x.replace("e",'y')
x
```

```
'wyZZo word'
```

```
x = "hello world"
x.replace(" ","-")
```

```
'hello-world'
```

```
x = x.replace("h",'my')
x
```

```
'wyZZo word'
```

```
x ="data"
x.replace("","=")
```

```
'=d=a=t=a='
```

```
x.replace("","hello")
```

```
'hellodhelloahellothelloahello'
```

```
print(x.replace("","h").upper())
print(x.replace("","h").upper().lower())
```

```
HDHAHTHAH
hdhahthah
```

```
y = "stride"

y = y.replace("t",'e')
y
```

```
'seride'
```

```
y = y.replace("e",'t')
y
```

```
'stridt'
```

.count()

```
xy ="rohit,rohit"
xy = xy.count("rohit")
xy
```

```
2
```

```
x = " notebook "
x.count('o')
```

```
3
```

```
x.count('n')
```

```
1
```

.Endswith()

The endswith() method is a string method that allows you to check if string ending with the letter or not

```
x = "hello world"
x.endswith("d")
```

```
True
```

```
x = "work"
x.endswith('rk')
```

```
True
```

```
k = "123456"
k.endswith("56")
```

```
True
```

```
x.endswith("wor")
```

```
False
```

```
y = "hello"
y.endswith()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[61], line 2
      1 y = "hello"
----> 2 y.endswith()

TypeError: endswith() takes at least 1 argument (0 given)
```

.split() method.

```
x = "Hello World"
x.split(" ")
```

```
['Hello', 'World']
```

```
x.split("z") # not able to split
```

```
['Hello World']
```

```
x.split("l") # '' is the space of ll
```

```
['He', '', 'o Wor', 'd']
```

```
a = "mohit is a boy"
a.split('  ')
```

```
['mohit', 'is', 'a', 'boy']
```

```
a.split("  ") # not able to split
```

```
['rohit']
```

```
a = "rohit mohit"
a.split("  ")
```

```
['rohit mohit']
```

```
w = "hello mohit"          # bydefault takes space.
w.split()
```

```
['hello', 'mohit']
```

.find() method

```
a = "gauravrohit"
a.find(" ")     # 5 is index value.
```

```
-1
```

```
a.find("t")
```

```
4
```

```
a.find("bo")
```

```
11
```

```
a = "rohit"
a.find("ho")
```

```
-1
```

```
n = "hello world"
n.find("w")
```

```
6
```

Function.

```
# len()
# print()
# type()
x = "python"
len(x)
```

```
6
```

```
len("123")
```

```
3
```

```
e = "rohit"
len(e)
printtype(e))
```

```
  Cell In[6], line 3
    print(len(e)type(e))
          ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

```
x = "python is easy"
len(x)
```

```
14
```

Tokens

A token is the smallest individual unit in a python program. All statements and instruction in a program are built with tokens.

- Keywords
- ldentifiers
- literals
- operators
- punctuators

Keywords

- Keywords are special words witch have a unique meaning and purpose in python.
- Keywords cannot be used as variable name,function names etc.

Identifires

- identifiers are names given to any variables,class, method, function etc.

```
help("keywords")
```

```
Here is a list of the Python keywords.  Enter any keyword to get more
help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
```

```
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

Literals

literals are simply the raw data thats is assigned into the variables.

- Literals.

  1- string literals.

  2- numeric literls.

```
  A-integer
  B-float
  C-Complex
```

  3- boolean literls,

  4- special literls.(none)

  5- literal collections. (Non primivite)

```
  A-list literal
  b-tuple literal
  c- dictionary literal
  d-set literal
```

Punctuators

- punctuators are symbols that are used to structure the statements.Some commonly used punctuators are :'',,@.{},[],etc

Operators

```
  Operators are symbol which are used to perform a specific operation
between values
1- arithmetic operators
2- comparion operators
3- assingment operators (=)
4- logical operators
5- bitwise operators
6- membership operators
7- identity operators
```

```
# Arithmetic Operators + , - , * , / , // , ** , %
5-6-7
```

```
-8
```

```
56//45
```

```
1
```

```
10//5
```

```
2
```

Comparison Operators == , != , > , < , >= , <=

```
x = 8
y = 10
print(x==y)
```

```
False
```

```
d = 44
t = 44
print(d==t)
```

```
True
```

```
w = 7
a = 9
print(w!=9),(w>a),(w>=a)
```

```
True
```

```
(None, False, False)
```

Precedences (BODMAS)

```
- ()
- **
- * , / , // , %
- + , -
```

logical operators

- PYTHON WILL ALWAYS COUNT FIRST AND THEN OR THEN NOT

and

- true and true - true
- true and false- false
- false and false- false
- false and true- false

or

- true or true - true
- true or false- true
- false or false- false
- false or true- true

not

- not True = false
- not false = true

```
z = 20
o = 40
z>=30 and o==z ,z<o or z!=o ,  not z<=o
```

```
(False, True, False)
```

```
True and  False or True and True or False
```

```
True
```

```
True and  False or True and True or True or True and True and True
```

```
True
```

```
not z<o and z!=o
```

```
False
```

```
z==o or o!=z
```

```
True
```

```
a = 10
b = 30
not a==b and b==a
```

```
False
```

```
not z<o
```

```
False
```

identity operaters . {is,is not}

- check whether two variables or objects share the same memory location or not

```
x = 12
y = 22
print(id(x))
print(id(y))
print(x is y)
print(x is not y)
```

```
1991212466768
1991212467088
False
True
```

```
a = [33]
b = [33]
print(id(a))
print(id(b))
a is b
a is b
b is not a
```

```
1429060390656
1429060390080

True
```

```
print(x is y)
```

```
True
```

```
print(x is not y)
```

```
False
```

```
a = [12]
b = [12]
print(a is b)                # is chackint the location
print(a==b)                 # == is chacking the value is same or not
```

```
False
True
```

membership operaters {in, not in}

- membership operaters check whether a value is present in asequence of vales
- sequence - strings, lists, tuples, sets, dictionaries

```
a = "data science"
b = " science"
b in a
a in b
```

```
False
```

```
a = [1,2,3,4,5,6]
b = 3

b in a
```

```
True
```

```
c = [1,2,3,4,5,6,7]
d = 6

d in c
```

```
True
```

Bitwise operaters

- & - bitwise AND
- | - bitwise OR
- ^ - bitwise XOR
- ~ - bitwise NOT
- << - bitwise LEFT SHIFT
- {>> - bitwise RIGHT SHIFT}

assignments operators

= , += , -= , *= , /= , //= , **= , %= , |= , &= , ^= , >>= , <<=

```
z = 8
z += 7
z
```

```
15
```

```
x = 66
x |=78
x
```

```
78
```

```
a = 5
b = 2
a <<= b
a
```

```
20
```

```
a = 6
b = 1
a >>= b
a
```

```
3
```

```
a = 6
b = 1
a /= b
a
```

```
6.0
```

```
a = 8
b = 2
a //= b
a
```

```
4
```

```
100%95
```

```
5
```

```
40%25
```

```
15
```

```
20%41
```

```
20
```

```
c = [1,[2],3]
x = [2]
x in c
```

```
True
```

```
7 and 5
5
0 and True
0
5 and 7
7
```