String Concept-

1. How to read string

There are three type of quotes-

1. Single quote
2. Double quote
3. Triple quote

$Single - quote$

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

Out[1]:
```
'aviansh'
```

$Double - quote$

In [2]:
```
str1="aviansh"
str1
```

Out[2]:
```
'aviansh'
```

$Triple - quote\ (Docstring)$

- Triple quote is also known as doc string.
- It is used to say some information about your python code.

In [4]:
```
"""In this code we take two variable that is 'a' and 'b' and
the value asign 'a' is 10 and 'b' is 20  now add both and store in
third variable that is 'c' now print 'c' and get the answer."""

a=10
b=20
c=a+b
print(c)
```

```
30
```

How to hightlight the words

In [7]:
```
str2='"aviansh" kurrey'
print(str2)

# hear we try to hightlight 'avinash' using double quote
```

```
"aviansh" kurrey
```

In [8]:
```
str2="avinash 'kurrey'"
print(str2)

# hear we try to hightlight 'kurrey' using single quote
```

```
avinash 'kurrey'
```

2. Type-

Python type() is a built-in function that is used to return the type of data stored in the objects or variables in the program.

how to use this function ==> type(object), object=name,number etc

In [9]:
```
str3='apple'
type(str3)                        # here the 'apple' is string data type
```

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

In [11]:
```
str3=10
type(str3)                        # here the '10' is integer data type
```

Out[11]:
```
int
```

3. len

The len() function returns the number of items in an object. When the object is a string.

The len() function returns the number of characters in the string.

how to use this function ==> len(object), object should be in string.

In [12]:
```
str4='python'
len(str4)                # here the 6 is alphabetical count of 'python'
                         # which is start from zero
```

Out[12]:
```
6
```

ASCII

- ASCII= American standard code for information interchange.
- ASCII assigns standard numeric values to letters, numerals,punctuation marks, and other characters used in computers.
- ASCII code for the character A is 65, and 90 is for Z.

- ASCII code 97 is for a, and 122 is for z.
- ASCII printable characters (32-126) (most commonly referred to)

$chr - ord$

Python's built-in function chr() is used for converting an Integer to a Character,

while the function ord() is used for converting character to a Integer.

In [13]:
```
str1='A'
ord(str1)
```

Out[13]:
```
65
```

In [14]:
```
str2=70
chr(str2)
```

Out[14]:
```
'F'
```

In [15]:
```
str2=100
chr(str2)
```

Out[15]:
```
'd'
```

In [19]:
```
user2='python'

ord('p'),ord('y'),ord('t'),ord('h'),ord('o'),ord('n')
```

Out[19]:
```
(112, 121, 116, 104, 111, 110)
```

In [18]:
```
chr(112),chr(121),chr(116),chr(104),chr(111),chr(110)
```

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

4. max-min

- Finding the largest and smallest strings in the iterable.
- max() and min() will return the result based on the ASCII value of each character.

In [20]:
```
user1='python'
max(user1),min(user1)

# here the 'y' is maximum value and 'h' is minimum value accroding to ASCII
# 'y'= 121  and 'h'= 104
```

Out[20]:
```
('y', 'h')
```

5. Arithmetic operation of two strings

Concatenation

Addition

- String concatenation means add strings together.
- Use the + character to add a variable to another variable.

In [21]:
```
str1='python'
str2='code'

str1+str2
```

Out[21]:
```
'pythoncode'
```

Subtraction

In [22]:
```
str1='python'
str2='code'
str1-str2

"""The Python "TypeError: unsupported operand type(s)
for -: 'str' and 'str'"occurs when we try to
use the subtraction - operator with two strings.
To solve the error, convert the strings to integers
before subtracting the two numbers."""
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[22], line 4
      1 str1='python'
      2 str2='code'
----> 4 str1-str2

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

Multiplication

In [23]:
```
str1='python'
str2='code'
str1*str2

"""The Python "TypeError: can't multiply sequence by non-int
of type 'str'"occurs when we try to multiply a
sequence (e.g. a string or a list) by a string.To solve
the error, convert the string to a float or an integer"""
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[23], line 3
      1 str1='python'
      2 str2='code'
----> 3 str1*str2

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

Multiplication is possible when we write like this

In [24]:
```
str1='python'
3*str1

"""when we try to multiply to string with integer
than they will result like this"""
```

Out[24]:
```
'pythonpythonpython'
```

Division

In [25]:
```
str1='python'
str2='code'
str1/str2

"""The Python "TypeError: unsupported operand type(s)
for -: 'str' and 'str'"occurs when we try to
use the divide / operator with two strings.
To solve the error, convert the strings to integers
before division the two numbers."""
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[25], line 3
      1 str1='python'
      2 str2='code'
----> 3 str1/str2

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

6. In-operation

- In Python, the in operator determines whether a given value is a constituent element of a sequence such as a string, array, list, or tuple. When used in a condition, the statement returns a Boolean result of True or False. The statement returns True if the specified value is found within the sequence.
- It used only for strings

In [26]:
```
string1='python'

'p' in string1
'y' in string1
't' in string1
'h' in string1
'o' in string1
'n' in string1
```

Out[26]:
```
True
```

In [28]:
```
# using in operator in for loop to print 'python'

for i in string1:
    print(i,end="")
```

```
python
```

6. Index

- The index() method finds the first occurrence of the specified value.
- The index() method raises an exception if the value is not found.

In [30]:
```
string1='code'
```

In [ ]:
```
# position of letters

c  o  d  e
1  2  3  4
```

In [31]:
```
# represent the index operation and get output

string1[0],string1[1],string1[2],string1[3]
```

Out[31]:
```
('c', 'o', 'd', 'e')
```

In [38]:
```
# Using for loop

for i in range(4):
    print(string1[i],end=' ')

#i=0 ====== string1[0]===='c'
#i=1======= string1[1]===='o'


# i == will give index
# string[i] ==== letter
```

```
c o d e
```

In [37]:
```
# I want to print the letters using for loop
# in
# range

string1='code'
for i in string1:
    print(i)

for i in range(len(string1)):
    print(i,string1[i])
```

```
c
o
d
e
0 c
1 o
2 d
3 e
```

Negative Index

In [ ]:
```
# position of letters

c  o  d  e
-4 -3 -2 -1
```

In [36]:
```
# represent the index operation and get output


string1='code'
string1[-4],string1[-3],string1[-2],string1[-1]
```

Out[36]:
```
('c', 'o', 'd', 'e')
```

In [41]:
```
# Using for Loop

for i in range(-4,0):
    print(string1[i],end=' ')
```

```
c o d e
```

In [39]:
```
# I want to print the letters using for Loop
# in
# range


string1='code'
for i in range(-len(string1),0):
    print("the negative index of {} is {}".format(string1[i],i))
```

```
the negative index of c is -4
the negative index of o is -3
the negative index of d is -2
the negative index of e is -1
```

In [ ]:
```
# Using While Loop in posetive index and negative index
```

In [44]:
```
# posetive index

i=0
string1='code'
while True:
    print("the posetive index of {} is {}".format(i,string1[i]))
    i=i+1
    if i==4:
        break
```

```
the posetive index of 0 is c
the posetive index of 1 is o
the posetive index of 2 is d
the posetive index of 3 is e
```

In [58]:
```
# Negative index

i=0
string1='code'
while i>-4:
    i=i-1
    print("the negative index of {} is {}".format(i,string1[i]))
```

```
the negative index of 0 is c
the negative index of -1 is e
the negative index of -2 is d
the negative index of -3 is o
```

7. Mutable and Immutable concept

- Mutable objects are those that allow you to change their value or data in place without affecting the object's identity.
- Immutable objects are those that not allow you to change their value or data.
- Strings are immutable

In [59]:
```
# Immutable

# i want to change 'c' to 'C'
# output should be = 'Code'

str1='code'
str1[0]='C'

"""The reason why you get that error message is
because strings are immutable i.e. you can't modify them."""
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[59], line 7
      1 # Mutable
      2
      3 # i want to change 'c' to 'C'
      4 # output should be = 'Code'
      6 str1='code'
----> 7 str1[0]='C'

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

In [ ]:
```
# Mutable

"""A string is one of those data types which are immutable
(in other words cannot be changed once created). Your only
option is to create a new string with the relevant changes - normally
that would be using one of the string methods or by slicing"""
```

8. Slice

- The slice operator in Python is used to extract a specific portion of a sequence, such as a string, list, or tuple.

In [61]:
```
# how to slice?

str5='python code'
```

In [ ]:
```
0  1  2  3  4  5  6  7  8  9  10
p  y  t  h  o  n     c  o  d  e
```

In [69]:
```
str5='python code'       # simple argument
str5[0:6]
```

Out[69]:
```
'python'
```

In [68]:
```
str5[7:11]               # simple argument
```

Out[68]:
```
'code'
```

In [72]:
```
str5[1:8:2]             # argument with steps
```

Out[72]:
```
'yhnc'
```

In [70]:
```
str5[:]                # no agrument means all includes
```

Out[70]:
```
'python code'
```

In [71]:
```
str5[::]              # no agrument means all includes
```

Out[71]:
```
'python code'
```

In [ ]:
```
# Trying to negative way
```

In [ ]:
```
 0   1   2   3   4   5    6  7   8  9   10
 p   y   t   h   o   n       c   o  d   e
-11 -10 -9  -8  -7  -6  -5  -4  -3  -2 -1
```

In [2]:
```
str5='python code'
str5[2:-3:2]
```

Out[2]:
```
'to '
```

In [7]:
```
str5='python code'
str5[-2:10:2]
```

Out[7]:
```
'd'
```

In [8]:
```
str5='python code'
str5[-2:-10:-2]

# start=-2
# stop=-10+1=-9
# step= 2 (negative direction)
```

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

9. String Methods-

Capitalize-

- Python String capitalize() method returns a copy of the original string and converts the first character of the string to a capital (uppercase) letter, while making all other characters in the string lowercase letters.

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

Out[2]: 'Python'
```

Upper case-

- The Python upper() method is used to convert lowercase letters in a string to uppercase. The isupper() method, on the other hand, returns True if all the letters in a string are uppercase.

```
In [3]: str1='python'
        str1.upper()

Out[3]: 'PYTHON'
```

Lower case-

- The LowerCase method converts a string to lowercase letters.

```
In [4]: str1='PYTHON'
        str1.lower()

Out[4]: 'python'
```

Casefold-

- The casefold() method returns a string where all the characters are in lower case. It is similar to the lower() method, but the casefold() method converts more characters into lower case.

```
In [5]: str1='PyThon'
        str1.casefold()

Out[5]: 'python'
```

Count-

- The count() method returns the number of times a specified value appears in the string.

In [6]:
```
# how many 'a' are there

str1='hai how are you'
str1.count('a')
```

Out[6]:
```
2
```

In [8]:
```
str1='avi avi avi'
str1.count('avi')
```

Out[8]:
```
3
```

Replace-

- The replace() method replaces a specified phrase with another specified phrase.All occurrences of the specified phrase will be replaced, if nothing else is specified.

In [9]:
```
# Replace 'l' with 'L

string1='welcome'
string1.replace('l','L')
```

Out[9]:
```
'weLcome'
```

In [10]:
```
# This is all the occrences

string1='restart rrr'
string1.replace('r','$',-1)
```

Out[10]:
```
'$esta$t $$$'
```

Index-

- The index() method finds the first occurrence of the specified value. The index() method raises an exception if the value is not found. The index() method is almost the same as the find() method, the only difference is that the find() method returns -1 if the value is not found.

In [11]:
```
string1='welcome python'
string1.index('c')
```

Out[11]:
```
3
```

In [12]:
```
string1='welcome python'
string1.index('z')

""" When index() function fails to find a
given substring within a string,"""
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[12], line 2
      1 string1='welcome python'
----> 2 string1.index('z')
      4 """ When index() function fails to find a
      5 given substring within a string,"""

ValueError: substring not found
```

In [14]:
```
string1='hai hai hai hai hai'
# i want to first occurence of 'a'

i1=string1.index('a')
# second occurence of 'a'

i2=string1.index('a',i1+1)
string1.index('a',i2+1)

string1.index('a',string1.index('a')+1)
string1.index('a',string1.index('a',string1.index('a')+1)+1)
```

Out[14]:
```
9
```

In [15]:
```
string1='welcome helo hello'
string1.index('l')
string1.index('l',string1.index('l')+1)
string1.index('l',string1.index('l',string1.index('l')+1)+1)
```

Out[15]:
```
15
```

Find-

- Python find() function is used to return the lowest index value of the first occurrence of the substring from the input string; else it returns -1. The Python find() is an in-built string method that returns the index position of the character if found; else it returns value -1

In [16]:
```
string1='aviansh'
string1.find('s')
```

Out[16]:
```
5
```

In [19]:
```
string1='aviansh'
string1.find('z')      # if substring not found it return
```

Out[19]:
```
-1
```

In [18]:
```
string1='aviansh'
string1.find('y')
```

Out[18]:
```
-1
```

In [ ]:
```
# importent expression if substring not found.

str1='aviansh avinash'

string1.find('z')  # No error
# if substring not found it returns -1

string1.index('z')
# ValueError: substring not found

string1.count('z')  # No error
# returns zero
```

Strip-

- The strip() method removes any leading, and trailing whitespaces. Leading means at the beginning of the string, trailing means at the end. You can specify which character(s) to remove, if not, any whitespaces will be removed.

In [21]:
```
str1=' hello how are you '
print(str1.strip())
```

```
hello how are you
```

lstrip-

- If you want to remove the spaces only left side then use lstrip: Left strip.

In [22]:
```
str2=' hello how are you'
print(str1.lstrip())
```

```
hello how are you
```

rstrip-

- If you want to remove the spaces only right side then use rstrip: Right strip

In [23]:
```
str3='hello how are you '
print(str1.rstrip())
```

```
 hello how are you
```

Startwith-

- The startswith() method returns True if a string starts with the specified prefix(string). If not, it returns False .

In [29]:
```
str1='hay how are you'
str1.startswith('hai')
str1.startswith('h')
```

Out[29]:
```
True
```

Endwith-

- Python endswith() is a string method that returns True if the input string ends with the specified suffix(string); else it returns False.

In [28]:
```
str1='hay how are you'
str1.endswith('you')
```

Out[28]:
```
True
```

split-

- The split() method splits a string into a list

In [27]:
```
str1='hay how are you'
str1.split()
```

Out[27]:
```
['hay', 'how', 'are', 'you']
```

isalpha-

- The isalpha() function is a built-in function used for string handling in python, which checks if the single input character is an alphabet or if all the characters in the input string are alphabets.

In [30]:
```
s1='avinash'
s1.isalpha()
```

Out[30]:
```
True
```

isnumeric-

- The isnumeric() method returns True if all the characters are numeric (0-9), otherwise False. Exponents, like ² and ¾ are also considered to be numeric values.

In [32]:
```
string = "123456789"
result = string.isnumeric()
print(result)
```

```
True
```

In [31]:
```
s1='avinash'
s1.isnumeric()
```

Out[31]:
```
False
```

isalnum-

- The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).

In [33]:
```
s1='avinash'
s1.isalnum()
```

Out[33]:
```
True
```

Thankyou