Unit IV Strings

* Introduction to Strings :-

- Strings are collection (groups) of characters contained within quote marks
- strings are immutable in nature. [not be updated]
- Strings can be contained within single, double, triple quote marks [' " """ ''']
- Each character has a numeric code and belongs to set of characters known as the Unicode character set.
- Unicode ⇒ character set includes the character A-Z, 0-9, punctuation marks, the space character etc.
- String Examples :-

1.) "Anne was here"
2.) "Anne is now 18 years old"
3.) "9860076475"
4.) "A"
5.) "7"

} Single line string

6.) Writing Multiline code string

Input ⇒

```
my_string = """ Hello welcome to world of Python
            welcome to mmcoe. """
print (my_string)
```

Output ⇒ Hello welcome to world of python
welcome to mmcoe.

* Indexing :-

- Individual characters in a string are accessed using subscript ([ ]) operator.
- The expression in brackets [ ] is called index
- from left it starts from '0' and ends 'n-1' ∴ n is no of characters
- from right it starts from '-1' ends '-n' ∴ n is no of characters

0	1	2	3	4	5
P	Y	T	H	O	N
-6	-5	-4	-3	-2	-1

* Traversing a String :-
- A string can be traversed by accessing character from one index to another.
- Visiting each character specifically.

Example ⇒ Input ⇒

```
message = "Hello"
index = 0
for i in message:
    print ("Message of:", [index], i)
    index = index + 1
```

Output ⇒ Message of : H
Message of : e
Message of : l
Message of : l
Message of : o

* Iterate over string with index using range()
- range (len (String Obj)) function will generate the sequence from 0 to n-1 (n is size of String)
- Now iterate over this sequence and for each index access the character from string using operator [].

Example: Input ⇒

```
sampleStr = "Hello!!"
for i in rang (len (sampleStr)):
    print (sampleStr[i])
```

Output ⇒ H
e
l
l
o
!
!

Iterate ⇒ To do something again

* Iterating strings using for loop:

Example: Input ⇒
```
sample = "Hello"
for elem in sample:
    print (elem)
```
Output ⇒ H
e
l
l
o

* Iterate over a portion of string only: Slicing
- To generate slice we use [] operator.
- string [start : stop : step size]

Example: Input ⇒
```
sample = "Hello"
for elem in sample [0:100:2]:
    print (elem)
```
Output ⇒ H
l
o

- start index is included and end is excluded

1] What is string? How we can create string variable in python?

→ String is a data type of python, it is set of character where characters could be letter, digit, whitespace or any other symbol.

a). Creating String ⇒ strings can be created using single, double, triple quotes [' ' " " ''' ''']

Example:
```
string 1 = 'Welcome'     # single quotes
string 2 = "Welcome"     # double quotes
string 3 = '''Welcome'''  # triple quotes
```

b). Accessing string ⇒ Individual characters of a string can be accessed by using the method of Indexing or range slice method [:].

string	W	E	L	C	O	M	E
Index	0	1	2	3	4	5	6
'-' Index	-7	-6	-5	-4	-3	-2	-1

Example:
```
string = 'Welcome'
print (string [0])   index  W
print (string [0:2]) slice  W E L
```

c). Deleting / Updating from a String :

- updating or deletion of characters from a string is not allowed as strings are immutable
- Although deletion of entire string is possible with the use of a built-in del keyword.

Example: string = 'welcome'
del string

2] Operations on string :

① Concatenation (+) :- It joins two strings and return new list.

Example: Input ⇒
x = "Good"
y = "Morning"
z = x + y
print (z)

Output ⇒
Good Morning

② Append (+ =) :- add one string at end of another string.

Example: Input ⇒
x = "Good"
y = "Morning"
x += y
print (x)

Output ⇒
Good Morning

③ Repetition (*) :- repeats elements from the strings n number of times

Example: Input ⇒
x = "Hello"
y = x * 2
print(y)

Output ⇒
HelloHello

④ Slice [ ] :- give you character from specified index.

Example: Input ⇒
```
x="Hello"
print(x[1])
```
Output ⇒
```
e
```

⑤ Range slice [:] :- characters from specified range slice

Example: Input ⇒
```
x="Hello"
print(x[0:5])
```
Output ⇒
```
Hello
```

* Strings are immutable :-

- Python strings are immutable
- once it is created it cannot be changed.
- Whenever you try to change / modify / update an existing string, a new string is created.
- As every object (variable) is stored at some address in computer memory.
- The id() function returns the address of object in memory. String get new address in memory.

Example : Input ⇒
```
String 1 = "Good"
print ("String 1 value is:", string 1)
print ("Add. of string 1 is:", id(string1))
```

Output ⇒
```
string 1 value is: Good
Add. of string 1 is : 1000
```

if any operation is done then Adderess changes and becomes new address.

* Strings formatting operator :-
- % sign is a string formating operator
- % operator takes a format string on the left and the corresponding values in a tuple on the right.
- format operator % allows users to replace parts of string with the data stored in variables.
- syntax :- "<format>" % (<values>)

%c	Character
%d or %i	Signed decimal integer
%s	string
%u	Unsigned decimal integer
%o	Octal integer
%x or %X	Hexadecimal integer
%e or %E	Exponential notation
%f	Floating point number
%g or %G	short no. in floating point or exponential notation

Example: name = "Aryan"
age = 19
print ("Name = %s and Age = %d % (name, age))
print ("Name = %s and Age = %d % ("Ajit", 16))

Output →
Name = Aryan and Age = 19
Name = Aryan and Age = 16

%s → replaced by string
%d → replaced by integer value

* ord() and chr() functions :-

• The ord() function return the ASCII code of character

```
x = 'R'
print (ord (x))
```

O/P

82

```
y = 'p'
print (ord (y))
```

O/P

112

• The chr() function returns character represented by ASCII number.

```
print(chr(82))
```

O/P

R

```
print(chr(112))
```

O/P

p

* Built-in String methods and functions

① capitalize()

This letter is used to capitalize first letter of string

Example:

```
str = "hello"
print(str.capitalized())
```

O/P ⇒ Hello

② isalnum()

Returns true if string has at least 1 character and every character is either a number or an alphabet and False otherwise

Example: I/P

```
str = "JamesBond007"
print(str.isalnum())
```

O/P ⇒ True

③ isalpha()

Returns true if string has at least 1 character is an alphabet and False other wise.

Example: I/p

```
str = "James Bond"
print(str.isalpha())
```

O/P ⇒ True

```
lstr = "James 007"
print(lstr.isalpha())
```

False

④ isdigit()
Returns true if string has at least 1 character and every character is a digit and false otherwise.
Example: I/P str = "007"
print (str.isdigit())
O/P ⇒
True

⑤ islower()
Returns ture if string has at least 1 character and every character is a lower case alphabet and false otherwise
Example: I/P str = "aryan"
print(str.islower())
O/P ⇒
True

⑥ isspace()
Returns ture if string contains only white space character and false otherwise
Example: I/P str = " "
print (str.isspace())
O/P ⇒
True

⑦ isupper()
Returns true if string has at least 1 character and every character is an uppercase alphabet and false otherwise.
Example: I/P str="PYTHON"
print (str.isupper())
O/P ⇒
True

⑧ len (string)
Returns length of the string
Example: I/P str = "Hello"
print (len(str))
O/P ⇒
5

⑨ lower()
Converts all characters in the string into lower case
Example: I/P str="HELLO"
print (str.lower())
O/P ⇒
hello

(10) zfill(width)

Returns string left padded with zeros to a total of width characters.

It is used with numbers and also retain its sign (+ or -)

Example: str = "1234" O/P →

print(str.zfill(10)) 0000001234

10

(11) upper()

Converts all characters in the string into uppercase

Example: str = "Hello" O/P →

print(str.upper()) HELLO

(12) lstrip()

Removes all leading white space in string.

Example: str = " Hello" O/P →

print(str.lstrip()) Hello

(13) rstrip()

Removes all trailing white space in string of right side

Example: I/P str = " Hello " O/P →

print(str.rstrip()) Hello

(14) max(str)

Returns the highest alphabetical character (having highest ASCII value) from the string str.

Example: I/P str = "hello friendz" O/P →

print(max(str)) z

(15) strip()

Removes all leading white space and trailing white space in string

Example: I/P str = " Hello " O/P

print(str.strip()) Hello

(15) center() :- will centre align (place) the string using spaces.

ex. s=Hello World
news = s.center(width, '*')
O/P **** Hello world ****

Date : / /
Page

(16) min(str)

Returns the lowest alphabetical character (having lowest ASCII value) from the string str.

Example: I/P str="hellofriendz" O/P ⇒
print(min(str)) d

(17) replace(old,new[,max])

Replaces all or max (if given) occurrences of old in string with new.

Example: I/P str="hello hello hello" O/P ⇒
print(str.replace("he","fo")) follo follo follo

(18) title()

Returns string in title case

Example: str="The world is beautiful" O/P ⇒
print(str.title()) The World Is Beautiful

(19) swapcase()

Uppercase character becomes lowercase and vice versa
Lowercas character becomes uppercase and vice versa

Example: str="The World Is Beautiful
print(str.swapcase())

O/P ⇒

tHE wORLD iS bEAUTIFUL

(20) split(delim)

Returns a list of substrings

Example: str="abc,def,ghi,jkl"
print(str.split(','))

O/P ⇒

['abc','def','ghi','jkl']

(24) ljust() :- will left (plate) align the string, using a specified
(25) rjust() character (space) as fill character

Date : / /

(26) rfind / lfind() :- finds last occurence of specified value
returns -1 if not found

(21) join (list)

It is just the opposite of split.

Example : I/P print ('-'.join(['abc', 'def', 'ghi', 'jkl']))

O/P ⇒

abc-def-ghi-jkl

(22) isidentifier()

Returns true if the string is a valid identifier.

Example : I/P str = "Hello"
print (str.isidentifier())

O/P ⇒

True

(23) enumerate (str)

Returns an enumerate object that lists the index and value of all the characters in the string as pairs.

Example : I/P str = "Hello World"
print (list (enumerate (str))

O/P ⇒

[(0,'H'), (1,'e'), (2,'l'), (3,'l'), (4,'o'), (5,' '), (6,'W'), (7,'o'), (8,'r'), (9,'l'), (10,'d')]

* Slice operation :-

Slice ⇒ A substring of a string is called a slice.

Slicing operator : A subset of a string from the original string by using [] operator known as Slicing Operator.

• Indices

P	Y	T	H	O	N
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

str = "PYTHON"

print(str[:6]) → PYTHON
print(str[0:]) → PYTHON
print(str[:]) → PYTHON
print(str[-6:]) → PYTHON
print(str[:-1]) → PYTHON
print(str[-1]) → N
print(str[0:6:2]) → PTO

Date : / /
Page

* Membership operators : in and not in operators can be used with strings to determine whether a string is present in another string. Therefore the in and not in operator is known as membership operators.

Example:

①
```
str1="Welcome to the world of Python!!!"
str2=" the"
if str2 in str1:
    print ("found")
else:
    print ("not found")
```

O/P ⇒ found

②
```
str1="Welcome to the world of Python !!!"
str2=" best"
if str2 in str1:
    print (" found")
else:
    print ("not found")
```

O/P ⇒ not found

• You can also use in and not in operators to check whether a character is present in a word.

Example:

① 'u' in "starts"

O/P ⇒ False

② 'v' not in "success"

O/P ⇒ True

Date : / /
Page

* Comparison Operators

① ==

If two strings are equal, it returns true

"AbC" = "AbC"

⇒ True

② != or <>

If two strings are not equal, it returns True

~~"AbC" != "AbC"~~ "AbC" != "abc"

⇒ True

③ >

If the 1st string is greater than the second, it returns true

"abc" = "Abc"

⇒ True

④ <

If the 2nd string is greater than the 1st, it returns true

"abC" = "abc"

⇒ True

⑤ >=

If the 1st string is greater than or equal to the 2nd. returns true.

"aBC" = "ABC"

⇒ True

⑥ <=

If the 2nd string is greater than or equal to the 1st, it returns true.

"ABc" <= "ABc"

⇒ True

Date : / /
Page

- These operators compare the strings by using ASCII value of characters.
- The ASCII values of A-Z are 65-90 and ASCII code for a-z is 97-122
- Example: book is greater than Book because the ASCII value of 'b' is 98 and 'B' is 66

* Iterating strings
- Using for loop

Example:

```
str = "Welcome to python"
for i in str:
print (i,end=' ')
```

O/P⇒ Welcome to Python

- Using while loop

Example:

```
message = "Welcome to python"
index = 0
while index < len (message):
        letter = message [index]
        print (letter, end = ' ')
        index = index + 1
```

O/P ⇒ Welcometopython

* String Module:
- The string module consists of of a number of useful constant, classes and functions.
- These functions are used to manipulate strings

- string.printable :

String of printable character which includes digits, letters, punctuation, whitespaces

Date : / /
Page

- string.ascii_letters :
  Combination of ascii-~~letters~~lowercase and ascii-uppercase constants.

- string.ascii_lowercase :
  Refers to all lowercase letters from a-z.

- string.ascii_uppercase :
  Refers to all uppercase letters from A-Z.

- string.lowercase :
  A string that has all the characters that are considered lowercase letters.

- string.uppercase :
  A string that has all the characters that are considered uppercase letters.

- string.digits :
  Refers to digits from 0-9.

- string.hexdigits :
  Refers to hexadecimal digits, 0-9, a-f, and A-F.

- string.octdigits :
  Refers to octal digits from 0-7.

- string.punctuation :
  string of ASCII characters that are considered to be punctuation characters (all symbols who has ascii valu

- string.whitespace : A string that has all characters that are considered whitespaces like space, tab, return, vertical tab.

Date : / /
Page

```
import.
Str = "Welcome to the world of Python"
print ("Uppercase-", str.upper())
print("Lowercase-", str.lower())
print("Split-", str.split())
print("Join-", '-'.join (str.split()))
print ("Replace-", str.replace ("Python", "Java"))
print ("Count of o-", str.count('o'))
print ("find of -", str.find ("of"))
```

example

```
import.string
print(string.digit())
```

OP →

Unit

* Classes :-

example.

```
Class student:
    def __init__(self.name, roll-no)
        sef.name =name
        self.roll no. = roll.no
    def show (self)
        print (self.name)
        print (self.roll no.)
stud =student ("Ram", 20)
stud.show()
stud 2 = student ("seeta", 30)
stud2.show ()
```

O/P → Ram
20
Seeta
30