STRINGS

COMPOUND DATA TYPE

- Strings are qualitatively different from Integer and Float type.
- Characters which collectively form a String is a Compound Data Type.

For Eg.

fruit = “apple”

letter = fruit[1]

print (letter)

index

0	1	2	3	4
a	p	p	l	e
-5	-4	-3	-2	-1

Output : p // (index value starts from 0 as in C & C++)

LENGTH OF STRINGS

- The inbuilt function to find the length of a string is **‘len()’.**

  E.g..

  fruit = “banana”

  len(fruit)

  Output : 6

0	1	2	3	4	5
b	a	n	a	n	a
-6	-5	-4	-3	-2	-1

- To get the last letter we might try

  length = len(fruit)

  last = fruit[length]   #ERROR

  (because there is no character at 6th place)

LENGTH OF STRINGS

(to be continued…..)

- Right Method to do this is :

  length = **len(fruit)**

  last = fruit[length-1]

0	1	2	3	4	5
b	**a**	**n**	**a**	**n**	**a**
-6	**-5**	**-4**	**-3**	**-2**	**-1**

- Another way to get the elements from last is :

  fruit[-1]   # yields the last letter

  fruit[-2]   # yields the second last letter

TRAVERSAL USING WHILE LOOP

- Processing one character at one time.

For Eg.

```
index = 0
while    index < len(fruit):
    letter = fruit[index]
    print (letter)
    index = index + 1
```

(Take care of the indentation)

0	1	2	3	4	5
b	a	n	a	n	a
-6	-5	-4	-3	-2	-1

TRAVERSAL USING FOR LOOP

- For loop provides us a privilege to access the characters without using index.

For Eg.

```
fruit="apple"
for  ch  in  fruit:
        print (ch)
```

(Each time through the loop a character is assigned to the variable char)

STRING SLICES

- A segment of a string is called a slice.
- The syntax to select a slice from a string is **a[n : m]**, where **a** contains strings, **n** is the starting index and **m** is the end index.
- Includes the first index and **excluding** the last index.

Eg:

s = “Peter, Paul, and Mary”

s[0 : 5] **# Peter** **(**index from 0 to 4**)**

s[7 : 11] **# Paul**

s[17 : 21] **# Mary**

STRING SLICES

(to be continued…..)

```
fruit = “banana”
fruit[ : 3]        #ban
fruit[ 3 :]        #ana
fruit[:]           ?
```

0	1	2	3	4	5
b	a	n	a	n	a
-6	-5	-4	-3	-2	-1

Slicing with steps

String[m : n : step]

By default

m=0

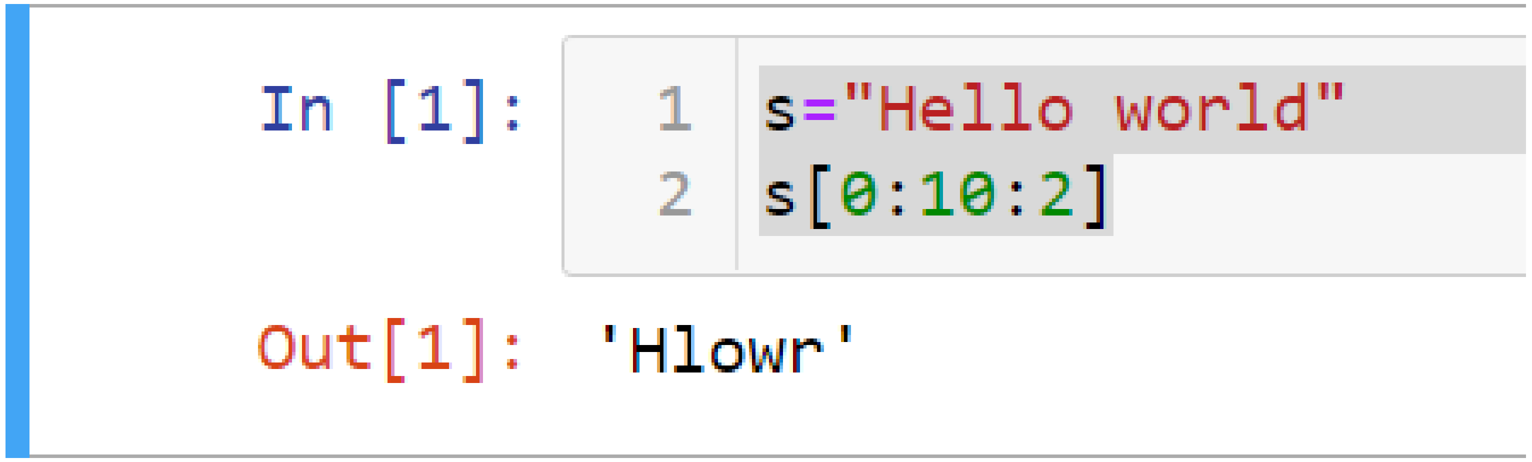
n=len() – 1

step = 1

```
s="Hello world"
s[0:10:2]
```

?

Slicing with steps



```
In [1]:  s="Hello world"
         s[0:10:2]

Out[1]: 'Hlowr'
```

STRING COMPARISON

- Equality Comparison

  “banana” == “banana”

  True
- Other Comparisons

```
if    “apple”  <  “banana”:
        print ”apple comes before banana.”
else:
        print ”apple comes after banana.”
```

Note: Comparison of UNICODE of first mismatched character

“India”>=“india”

?

STRING COMPARISON

(to be continued…..)

- > and < comparison operations are useful for putting words in alphabetical order:
- Uppercase letters ,numerals and special symbol comes before Lowercase letters in Python.
- Need to maintain a standard format of the strings.

STRINGS ARE IMMUTABLE

- An existing string cannot be modified

  For Eg :

  greeting = "Hello, world!“

  greeting[0] = ’J’ **# ERROR!**

  For Eg :

  print (greeting)

  Output : Hello, world!

STRINGS ARE IMMUTABLE

(to be continued….)

- The Solution of the problem is

```
greeting = "Hello, world!“
newGreeting = ’J’ + greeting[1:]
print newGreeting
```

Output : Jello, World

- The original string remains intact.

Creation of Find Function in Strings

- The find() method finds the first occurrence of the specified value.
- The find() method returns -1 if the value is not found.
- *string*.find(*value, start, end*)
- start and end are optional index values
- Ex.

    S="I am Indian"

    S.find("di")

    Output

    7

- Python string rfind() method is similar to find(), except that search is performed from right to left.

Creation of Find Function in Strings

```
def    myfind(st, ch):
    index = 0
    while   index <= len(st) - 1:
        if st[index] == ch:
            return  index
        index = index + 1
    return -1
```

LOOPING AND COUNTING

- The following code counts the number of times a appears in the string.

```
fruit = “grapes"
count = 0
for   ch   in   fruit:
          if ch == ’a’:
                    count = count + 1
print (count)
```

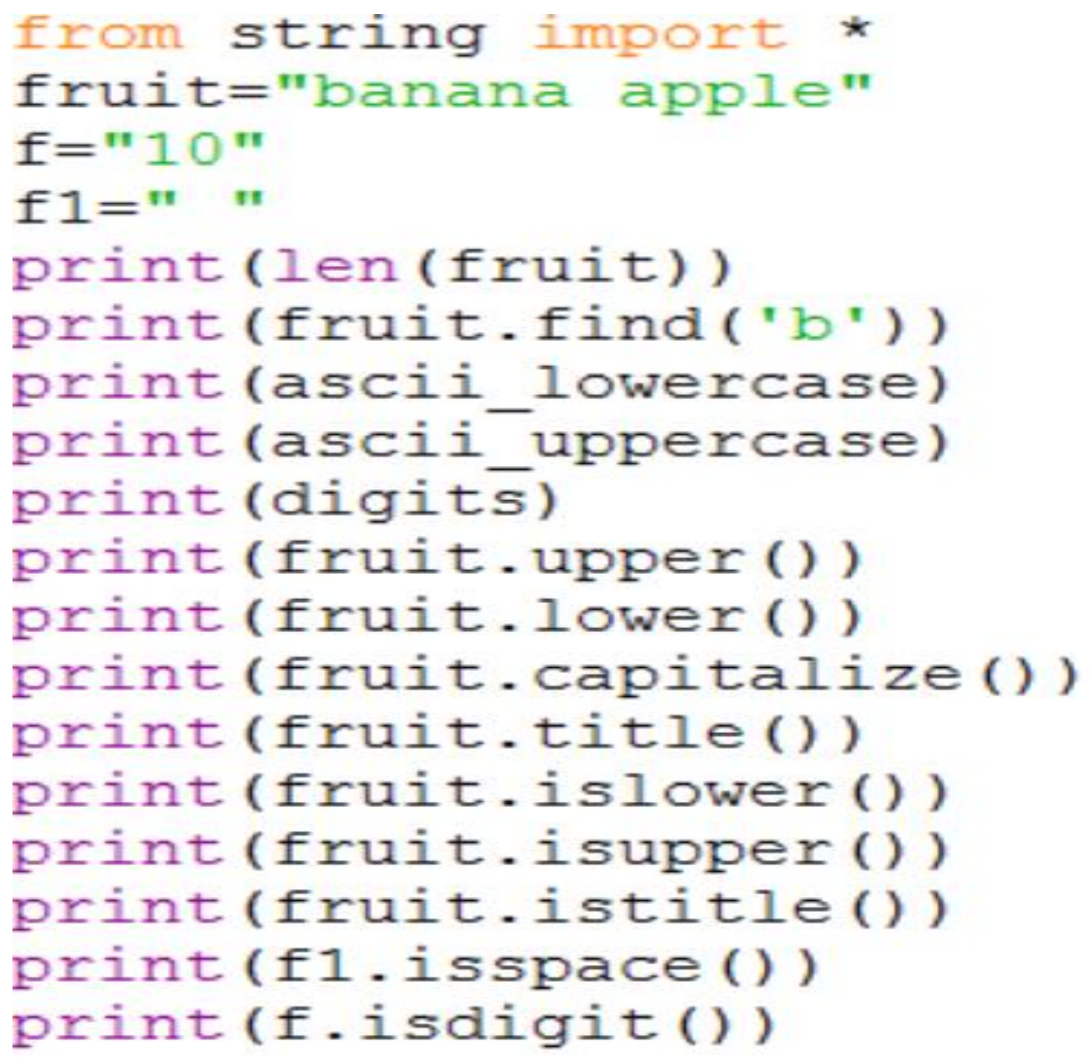
Output : 1

CHARACTER CLASSIFICATION

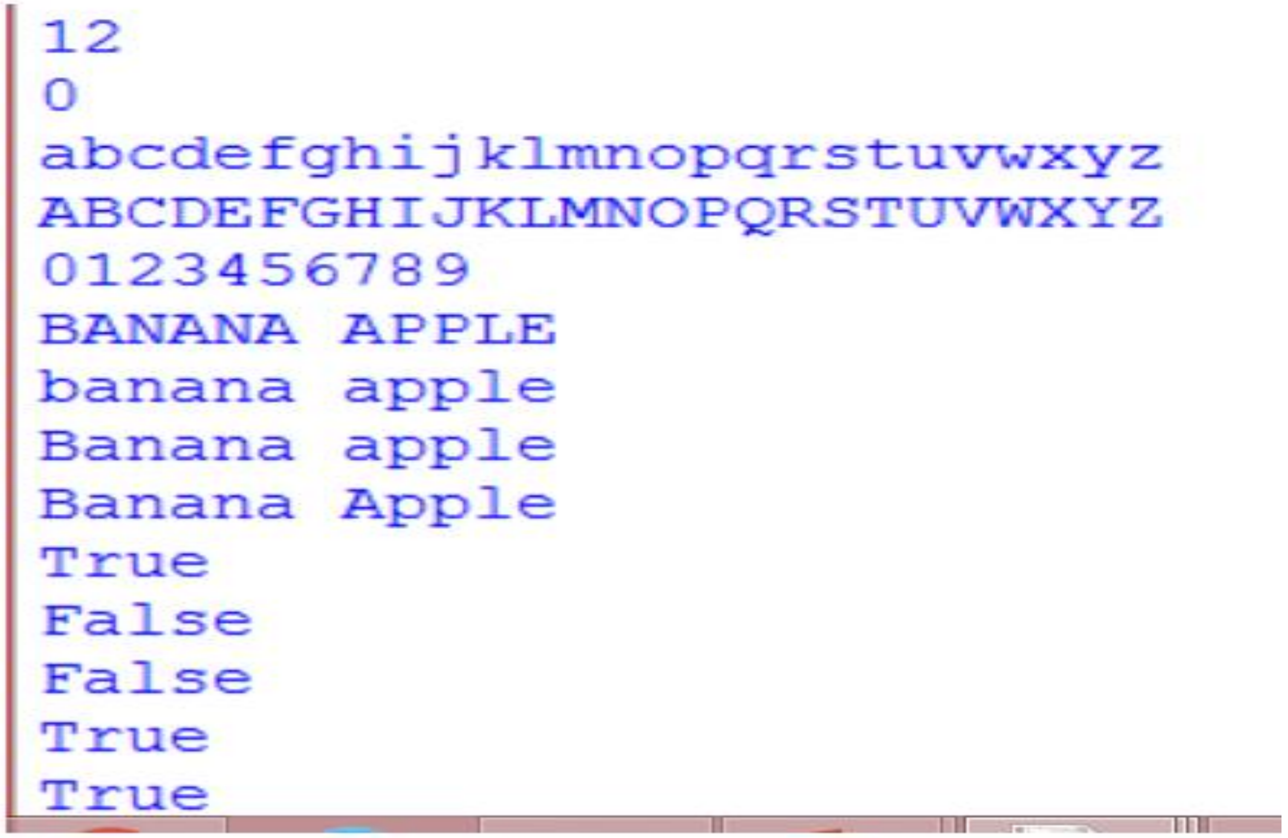
- **string** module provides several constants that are useful for these purposes.
- **string.lowercase** contains all the letters that the system considers to be lowercase.
- **string.uppercase** contains all the letters that the system considers to be uppercase.

```
import    string
print  (string.ascii_lowercase)
print  (string.ascii_uppercase)
print  (string.digits)
```

String Operations



```
from string import *
fruit="banana apple"
f="10"
f1=" "
print(len(fruit))
print(fruit.find('b'))
print(ascii_lowercase)
print(ascii_uppercase)
print(digits)
print(fruit.upper())
print(fruit.lower())
print(fruit.capitalize())
print(fruit.title())
print(fruit.islower())
print(fruit.isupper())
print(fruit.istitle())
print(f1.isspace())
print(f.isdigit())
```



```
12
0
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
BANANA APPLE
banana apple
Banana apple
Banana Apple
True
False
False
True
True
```

UNICODE

❖Unicode is **a universal character encoding standard that assigns a code to every character and symbol** in every language in the world. Since no other encoding standard supports all languages, Unicode is the only encoding standard that ensures that you can retrieve or combine data using any combination of languages.

❖Unicode can be implemented by different character encodings. The Unicode standard defines Unicode Transformation Formats (UTF): UTF-8, UTF-16, and UTF-32, and several other encodings.

How to find UNICODE

```
>>>ord(‘a’)

Output  97

>>>char(97)

Output  ‘a’
```

Assignment Questions:

Q1: Write a program in python to reverse a string.

Q2: Write a program in python to check whether a given string is palindrome or not.

Q3:Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters.
Sample String : 'The quick Brow Fox'
Expected Output :
No. of Upper case characters : 3
No. of Lower case Characters : 12

Q4:Write a Python function to check whether a string is pangram or not.
Note : Pangrams are words or sentences containing every letter of the alphabet at least once.
For example : "The quick brown fox jumps over the lazy dog"
Q5: Write a python program to count the occurrence(frequency) of a particular character in a string.

Q6:Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string. Sample String : 'wakawama'
Expected Result : 'wama'
Sample String : 'wa'
Expected Result : 'wawa'
Sample String : ' w'
Expected Result : Empty String

Q7:Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add ‘EE' instead. If the string length of the given string is less than 3, leave it unchanged.

Sample String : 'abc'
Expected Result : 'abcing'
Sample String : 'string'
Expected Result : 'stringEE‘

Q8: Write a Python program to find the first appearance of the substring 'not' and 'poor' from a given string, if 'bad' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.

Sample String : 'The lyrics is not that poor!'
Expected Result : 'The lyrics is good!'

Q10: Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.

Sample String : 'restart'
Expected Result : 'resta$t’