CHAPTER 7

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)

  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()'.**

  For Eg.

  fruit = "banana"

  len(fruit)

  Output : 6

- To get the last letter we might try

  length = len(fruit)

  last = fruit[length] #ERROR

  (because there is no character at 6^{th} place)

LENGTH OF STRINGS

(to be continued…..)

- Right Method to do this is :

  length = **len(fruit)**

  last = fruit[length-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)

TRAVERSAL USING FOR LOOP

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

  For Eg.

  fruit=“apple”

  for char in fruit:

  print (char)

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

TRAVERSAL USING FOR LOOP

(to be continued…..)

ABECEDARIAN SERIES – Print using for loop

- A series or list in which the elements appear in alphabetical order.

  For Eg. In Robert McCloskey's book Make way for Duckings the names of the ducklings were Jack, Kack, Lack, Mack, Nack, Ouack, Pack, Quack

- To print them in order the code is :

  prefixes = "JKLMNOPQ"

  suffix = "ack"

STRING SLICES

- A segment of a string is called a slice, i.e. a character.
- 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.

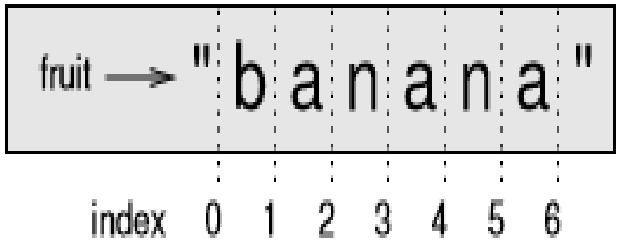
Eg:

```
s= “Peter, Paul, and Mary”
print s =[0:5]          # Peter
print s =[7:11]         # Paul
print s =[17:21]        # Mary
```

STRING SLICES

(to be continued…..)

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



`[m:n]`, `[m:]`, `[:n]`, `[m:n:step]`	Slicing to get a substring.
	From index `m` (included) to `n` (excluded) with an optional `step` size.
	The default `m` is `0`, `n` is `len()-1`, `step` is `1`.
		`s[1:3]` ⇒ `'el'`
		`s[1:-2]` ⇒ `'el'`
		`s[3:]` ⇒ `'lo'`
		`s[:-2]` ⇒ `'Hel'`
		`s[:]` ⇒ `'Hello'`
		`s[0:5:2]` ⇒ `'Hlo'`

```
# Slicing
>>> s[1:3]     # Substring from index 1 (included) to 3 (excluded)
'el'
>>> s[1:-1]
'ello, worl'
>>> s[:4]      # Same as s[0:4], from the beginning
'Hell'
>>> s[4:]      # Same as s[4:-1], till the end
'o, world'
>>> s[:]       # Entire string; same as s[0:len(s)]
'Hello, world'
```

STRING COMPARISON

- Equality Comparison

```
if word ==“banana!”
```

- Other Comparisons

```
if word < “banana”:
        print ”Your word,”+ word + “,comes before banana.”
elif word > “banana”:
        print ”Your word,”+ word + “,comes after banana.”
```

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!
  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

- Find function is used to find the index of a particular character inside the string, else it returns void.
- It is opposite to that of the [] operator.

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

LOOPING AND COUNTING

- For and while loops can be used for looping and counting.
- The following code counts the no of times a appears in the string.

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

Output : 1

STRING MODULE

- String module is a package which contains useful functions for manipulating strings.
- To use the string module we need to import it first by using the following line of code i.e.

  from string import*

- Find Function : This inbuilt function finds the index of the character in the string.

  fruit = “guava”

  index = fruit.find(”a”)

  Output : 2

STRING MODULE

(to be continued….)

- Find Function :

Try out

A=“Banana”

A.find(”na”) **#2**

A.find(”na”,3) **#4, (starts from index 3)**

A.find(”b”,1,3) #-1, (checks between 1 to 3 excluding 3 index)

CHARACTER CLASSIFICATION

- Character Classification is a recognition of character (lowercase or uppercase) or it’s a digit.
- 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.

```
From string import*
print ascii_lowercase
print ascii_uppercase
print digits
```

Three ways to recognize lowercase

Method 1 :

```
def isLower(ch):
    return string.ascii_lowercase.find(ch) != -1
```

Method 2 :

```
def isLower(ch):
    return ch in string.ascii_lowercase
```

Method 3 :

```
def isLower(ch):
    return 'a' <= ch <= 'z'
```

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
```

Assignment Questions:

- WAP to find a sub string and capitalize it
- WAP to validate mobile no given by user
- WAP to validate Email id given by user